

# UC San Diego

## UC San Diego Electronic Theses and Dissertations

**Title**

Error Floors of LDPC Codes and Related Topics

**Permalink**

<https://escholarship.org/uc/item/1gr3m0wm>

**Author**

Butler, Brian K.

**Publication Date**

2013

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA, SAN DIEGO

**Error Floors of LDPC Codes and Related Topics**

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

in

Electrical Engineering  
(Communication Theory and Systems)

by

Brian K. Butler

Committee in charge:

Professor Paul H. Siegel, Chair  
Professor Tara Javidi  
Professor Daniele Micciancio  
Professor Laurence B. Milstein  
Professor Alexander Vardy

2013

© Copyright  
Brian K. Butler, 2013  
All rights reserved.

The dissertation of Brian K. Butler is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

---

---

---

---

---

Chair

University of California, San Diego

2013

## DEDICATION

*To my fantastic children.*

## EPIGRAPH

*It is not a good habit to stuff one's text with quotations from prominent thinkers, except to make fun of them or provide a historical reference.*

—Nicholas Taleb, *The Black Swan*

## TABLE OF CONTENTS

|                                                                   |     |
|-------------------------------------------------------------------|-----|
| Signature Page .....                                              | iii |
| Dedication .....                                                  | iv  |
| Epigraph .....                                                    | v   |
| Table of Contents .....                                           | vi  |
| List of Figures .....                                             | ix  |
| List of Tables .....                                              | xi  |
| Acknowledgements .....                                            | xii |
| Vita .....                                                        | xv  |
| Abstract of the Dissertation .....                                | xix |
| Chapter 1      Introduction .....                                 | 1   |
| 1.1    Background .....                                           | 1   |
| 1.2    A Very Brief History of Coding .....                       | 2   |
| 1.3    (7,4) Hamming Code .....                                   | 3   |
| 1.4    Dissertation Overview .....                                | 5   |
| Bibliography .....                                                | 6   |
| Chapter 2      Introduction to LDPC Codes .....                   | 8   |
| 2.1    Binary Linear Block Codes .....                            | 8   |
| 2.2    LDPC Codes .....                                           | 9   |
| 2.3    Memoryless Channels .....                                  | 11  |
| 2.4    Ideal Decoding Bounds .....                                | 13  |
| 2.5    Sum-Product Algorithm .....                                | 14  |
| 2.6    Error Floor .....                                          | 16  |
| Bibliography .....                                                | 17  |
| Chapter 3      Numerical Issues Affecting LDPC Error Floors ..... | 18  |
| 3.1    Introduction .....                                         | 18  |
| 3.2    Background .....                                           | 19  |
| 3.2.1    Floating-Point Formats .....                             | 19  |
| 3.2.2    Numerical Problem in FP .....                            | 21  |
| 3.3    Preferred SPA Solution .....                               | 22  |
| 3.4    Additional SPA Formulations .....                          | 23  |
| 3.4.1    Min-Sum Algorithm (MSA) .....                            | 23  |
| 3.4.2    Gallager's Involution Transform (GIT) .....              | 24  |
| 3.4.3    Gallager's Involution Transform, Amended (GIT2) .....    | 24  |
| 3.4.4    Likelihood Ratio (LR) .....                              | 26  |

|           |       |                                                                                                 |     |
|-----------|-------|-------------------------------------------------------------------------------------------------|-----|
|           | 3.4.5 | Analysis of Extended Likelihood Ratio (ELR) . . . . .                                           | 27  |
|           | 3.4.6 | Likelihood Difference (LD) or Tanh Domain . . . . .                                             | 28  |
|           | 3.4.7 | Offset-Likelihood Difference (OLD) . . . . .                                                    | 29  |
|           | 3.4.8 | Summary of SPA Formulations . . . . .                                                           | 30  |
|           | 3.4.9 | Hybrid SPAs . . . . .                                                                           | 31  |
|           | 3.5   | Floating-Point Resolution . . . . .                                                             | 32  |
|           | 3.6   | Simulation . . . . .                                                                            | 34  |
|           | 3.7   | Conclusion . . . . .                                                                            | 36  |
|           |       | Bibliography . . . . .                                                                          | 38  |
| Chapter 4 |       | Error Floor Approximation for LDPC Codes in the AWGN Channel . . . . .                          | 40  |
|           | 4.1   | Introduction . . . . .                                                                          | 40  |
|           | 4.2   | Preliminaries . . . . .                                                                         | 45  |
|           | 4.2.1 | Nonnegative Matrices . . . . .                                                                  | 45  |
|           | 4.2.2 | General Graph Theory . . . . .                                                                  | 46  |
|           | 4.2.3 | LDPC Codes, the AWGN Channel, and SPA Decoding . . . . .                                        | 50  |
|           | 4.2.4 | Trapping Sets and Absorbing Sets . . . . .                                                      | 50  |
|           | 4.3   | State-Space Model . . . . .                                                                     | 52  |
|           | 4.3.1 | Check Node Gain Model . . . . .                                                                 | 53  |
|           | 4.3.2 | Linear State-Space Model . . . . .                                                              | 55  |
|           | 4.3.3 | Graphical Assumptions and Interrelationships . . . . .                                          | 59  |
|           | 4.4   | Dominant Eigenvalues and Probability Model . . . . .                                            | 64  |
|           | 4.4.1 | Utilizing Dominant Eigenvalues . . . . .                                                        | 64  |
|           | 4.4.2 | Probability of Error Model . . . . .                                                            | 66  |
|           | 4.4.3 | Codewords . . . . .                                                                             | 67  |
|           | 4.4.4 | Non-codewords . . . . .                                                                         | 67  |
|           | 4.4.5 | Bounds on Spectral Radius of the Matrix $\mathbf{A}$ . . . . .                                  | 69  |
|           | 4.5   | Analysis Using DE without Saturation . . . . .                                                  | 70  |
|           | 4.6   | Modifying and Applying Richardson's Semi-Analytical Tech-<br>nique without Saturation . . . . . | 73  |
|           | 4.7   | Numerical Results with Saturation . . . . .                                                     | 77  |
|           | 4.8   | Conclusion . . . . .                                                                            | 85  |
|           | 4.A   | Appendix: Reducible $\mathbf{A}$ Matrices . . . . .                                             | 87  |
|           | 4.B   | Appendix: Adding Leaves and Branches . . . . .                                                  | 89  |
|           | 4.C   | Appendix: Tables of Absorbing Set Structure . . . . .                                           | 91  |
|           | 4.D   | Appendix: Density Evolution in the Error Floor Region . . . . .                                 | 96  |
|           |       | Bibliography . . . . .                                                                          | 103 |
| Chapter 5 |       | Bounds on the Minimum Distance of Punctured Quasi-Cyclic LDPC Codes                             | 108 |
|           | 5.1   | Introduction . . . . .                                                                          | 108 |
|           | 5.2   | Protographs and the AR4JA Family of Codes . . . . .                                             | 110 |
|           | 5.3   | QC Expansion and Polynomial Representation . . . . .                                            | 114 |
|           | 5.4   | Minimum Distance Bounds for QC Codes . . . . .                                                  | 116 |
|           | 5.4.1 | Upper Bounds for QC Codes . . . . .                                                             | 117 |
|           | 5.4.2 | Upper Bounds for Punctured QC Codes . . . . .                                                   | 121 |
|           | 5.5   | Tighter Bounds on the Minimum Distance . . . . .                                                | 125 |

|           |                                                                                   |     |
|-----------|-----------------------------------------------------------------------------------|-----|
| 5.6       | QC Expansion of the AR4JA Family of Codes .....                                   | 129 |
| 5.7       | Numerical Results Obtained by Search .....                                        | 131 |
| 5.7.1     | Distance Bounds from Codeword Search for AR4JA Codes ..                           | 132 |
| 5.7.2     | Bounds on the Girth of AR4JA Codes .....                                          | 135 |
| 5.8       | Conclusion .....                                                                  | 135 |
| 5.A       | Appendix: Proof of Lemma 5.2 .....                                                | 136 |
| 5.A.1     | Matrices Over Commutative Rings .....                                             | 136 |
| 5.A.2     | Finite Commutative Rings .....                                                    | 138 |
|           | Bibliography .....                                                                | 140 |
| Chapter 6 | Sharp Bounds on the Spectral Radius of Nonnegative Matrices and<br>Digraphs ..... | 142 |
| 6.1       | Introduction .....                                                                | 142 |
| 6.2       | Bounds on the Spectral Radius of Nonnegative Matrices .....                       | 144 |
| 6.3       | Further Equality Conditions on the Spectral Radius Bounds .....                   | 148 |
| 6.4       | Bounds Applied to Digraphs .....                                                  | 151 |
| 6.5       | Conclusions .....                                                                 | 158 |
|           | Bibliography .....                                                                | 158 |

## LIST OF FIGURES

|             |                                                                                                                                                                                                                                                                                                                     |    |
|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| Figure 1.1: | A simplified communication or storage system block diagram. . . . .                                                                                                                                                                                                                                                 | 2  |
| Figure 2.1: | The Tanner graphs of two codes. . . . .                                                                                                                                                                                                                                                                             | 10 |
| Figure 2.2: | Three memoryless channels of interest. . . . .                                                                                                                                                                                                                                                                      | 12 |
| Figure 2.3: | The channel capacity of the AWGN channel with real and binary input. . . . .                                                                                                                                                                                                                                        | 13 |
| Figure 2.4: | The two types of message updates during iteration $l$ of the SPA. . . . .                                                                                                                                                                                                                                           | 15 |
| Figure 2.5: | The general appearance of the waterfall and error floor regions in BP decoding. (High output error rates are undesirable.) . . . . .                                                                                                                                                                                | 16 |
| Figure 3.1: | Numerical accuracy of several versions of $-\ln \tanh(x/2)$ plotted as significant bits vs. $x$ (LLR value) using DP-FP computations. . . . .                                                                                                                                                                       | 25 |
| Figure 3.2: | RMS LLR error versus mean input LLR value, $m_\lambda$ , for check node calculation with 4 input LLRs having i.i.d. Gaussian distribution with variance $\sigma_\lambda^2 = 2m_\lambda$ . . . . .                                                                                                                   | 32 |
| Figure 3.3: | 64-bit FP resolution in LLR units versus equivalent LLR $\lambda$ . . . . .                                                                                                                                                                                                                                         | 33 |
| Figure 3.4: | 64-bit FP resolution in LLR units (in magnified view) versus equivalent LLR $\lambda$ . . . . .                                                                                                                                                                                                                     | 34 |
| Figure 3.5: | FER vs. $E_b/N_0$ in dB for the Margulis LDPC code in AWGN. . . . .                                                                                                                                                                                                                                                 | 35 |
| Figure 3.6: | FER vs. $E_b/N_0$ in dB for $n = 1057$ LDPC code in AWGN. . . . .                                                                                                                                                                                                                                                   | 36 |
| Figure 3.7: | Tail distribution of the maximum LLR magnitude upon successful decoding of the $n = 1057$ LDPC code conditioned on decoding success occurring when the iteration count is in the 20–27 range. . . . .                                                                                                               | 37 |
| Figure 3.8: | The same as Fig. 3.7, but conditioned on decoding success occurring when the iteration count is in the 20–100 000 range. . . . .                                                                                                                                                                                    | 37 |
| Figure 4.1: | FER vs. $E_b/N_0$ for (2640,1320) Margulis LDPC code in AWGN. . . . .                                                                                                                                                                                                                                               | 44 |
| Figure 4.2: | Four sample elementary Tanner subgraphs of a code with $d_v = 3$ . Odd-degree check nodes are shown shaded. . . . .                                                                                                                                                                                                 | 51 |
| Figure 4.3: | Illustration of a check node of degree two in the trapping set and degree $d_c$ in the Tanner graph. . . . .                                                                                                                                                                                                        | 53 |
| Figure 4.4: | Several graphical descriptions of the $(4, 2)$ trapping set with model's states numbered. Arrows have been added to the Tanner subgraph in (a) only to emphasize the direction of state variable flow. The spectral radii of $G$ , $\mathcal{L}(G)$ , and $D$ are 2.5616, 3.2316, and 1.5214, respectively. . . . . | 57 |
| Figure 4.5: | Three sample diagrams of undirected multigraphs. . . . .                                                                                                                                                                                                                                                            | 62 |
| Figure 4.6: | Check node output LLR mean and variance versus iteration number for the Margulis code at $E_b/N_0$ of 2.8 dB. . . . .                                                                                                                                                                                               | 71 |
| Figure 4.7: | Correlation coefficient among the four LLR check node output metrics used as linear model inputs versus iteration number for $(12, 4)$ trapping sets of the Margulis code as measured in LLR-domain SPA simulation. . . . .                                                                                         | 72 |

|              |                                                                                                                                                                                                                                                                                                                   |     |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| Figure 4.8:  | Three elementary TSs of a $d_v = 3$ code shown with degree-one check nodes shaded and with degree-two check nodes omitted. The two on the left are absorbing sets and dominate the error floor of the saturated SPA decoder for the Margulis code, while the one on the right fails is not an absorbing set. .... | 74  |
| Figure 4.9:  | FER vs. $E_b/N_0$ in dB for Margulis LDPC Code in AWGN. The semi-analytical (SA) technique only estimates errors due to the (12,4) and (14,4) trapping sets at $E_b/N_0 = 2.8$ dB. ....                                                                                                                           | 75  |
| Figure 4.10: | FER vs. $E_b/N_0$ in dB for the (640, 192) QC code with simulation and the TS model for predicting error floors using unit gains in the model and no check node inversions. ....                                                                                                                                  | 79  |
| Figure 4.11: | FER vs. $E_b/N_0$ in dB for the (640, 192) QC code with simulation and the TS model for predicting error floors using mean gains in the model and no check node inversions. ....                                                                                                                                  | 79  |
| Figure 4.12: | FER vs. $E_b/N_0$ in dB for the (640, 192) QC code by simulation and the TS model for predicting error floors using the modified mean gains of (4.11) in the model which takes into account check node inversions. ....                                                                                           | 82  |
| Figure 4.13: | FER vs. $E_b/N_0$ in dB for (2640, 1320) Margulis code with simulation and TS model predicted floor using modified mean gains. ....                                                                                                                                                                               | 82  |
| Figure 4.14: | BER vs. $E_b/N_0$ in dB for (2048, 1723) LDPC code from 802.3an with simulation, Richardson's semi-analytical (8, 8) floor estimation, and TS model predicted (8, 8) floor using modified mean gains. ....                                                                                                        | 83  |
| Figure 4.15: | FER vs. $E_b/N_0$ in dB for (155, 64) QC-LDPC code with simulation and TS model predicted floor using modified mean gains. ....                                                                                                                                                                                   | 84  |
| Figure 4.16: | Threshold $E_b/N_0$ in dB vs. $m_{\lambda(\text{ex})}^{(l-1)}$ to achieve the LLR growth needed to overcome a trapping set characterized by $r$ as specified in the legend for (3, 6)-regular codes such as Margulis. ....                                                                                        | 101 |
| Figure 5.1:  | The protograph $G$ and the corresponding protomatrix $\mathbf{A}$ . ....                                                                                                                                                                                                                                          | 110 |
| Figure 5.2:  | To obtain the derived graph, the protograph $G$ is replicated $N = 3$ times and the edge connections within each edge set are permuted. ....                                                                                                                                                                      | 111 |
| Figure 5.3:  | AR4JA protograph, rate-1/2. ....                                                                                                                                                                                                                                                                                  | 112 |
| Figure 5.4:  | AR4JA protograph, rate-2/3. ....                                                                                                                                                                                                                                                                                  | 112 |
| Figure 5.5:  | Minimum distance vs. block length (in bits) for rate-1/2 AR4JA. ....                                                                                                                                                                                                                                              | 134 |
| Figure 6.1:  | The digraph $G_1$ and its $5 \times 5$ adjacency matrix $\mathbf{A}(G_1)$ . ....                                                                                                                                                                                                                                  | 155 |

## LIST OF TABLES

|            |                                                                                                                                                                                                                                                        |     |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| Table 3.1: | Floating-point basic binary formats of IEEE Standard 754 [9]. . . . .                                                                                                                                                                                  | 20  |
| Table 3.2: | Floating-point extended formats of IEEE Standard 754. . . . .                                                                                                                                                                                          | 20  |
| Table 3.3: | Upper LLR limits of SPA formulations with respect to check node inputs. . . .                                                                                                                                                                          | 31  |
| Table 4.1: | Submatrix Correspondence Rules of the Directed Lifting of the Line Graph (for $i = j$ , only the all-zero matrix is allowed) . . . . .                                                                                                                 | 62  |
| Table 4.2: | Ratio of (12, 4) trapping set error floor contribution to (14, 4) for the Margulis code at $E_b/N_0 = 2.8$ dB using Richardson's semi-analytical technique. . . . .                                                                                    | 77  |
| Table 4.3: | DDE numerical results by iteration for (3, 6)-regular code at an $E_b/N_0$ of 2.8 dB with LLR saturation set to $\pm 25$ at check node output. LLR pdfs discretized to pmfs with a resolution of 50/2047. In this case, $m_\lambda = 3.8109$ . . . . . | 80  |
| Table 4.4: | SPA simulation results by iteration for the (2640, 1320) Margulis code which is (3, 6)-regular at an $E_b/N_0$ of 2.8 dB with LLR saturation set to $\pm 25$ at check node output. In this case, $m_\lambda = 3.8109$ . . . . .                        | 81  |
| Table 4.5: | Connected, elementary absorbing sets from the set of 4-cycle-free, degree-3 variable-regular codes, with $r_{\max} > 1.3$ . . . . .                                                                                                                    | 92  |
| Table 4.6: | Connected, elementary absorbing sets from the set of 4-cycle-free, degree-4 variable-regular codes . . . . .                                                                                                                                           | 93  |
| Table 4.7: | Connected, elementary absorbing sets from the set of 4-cycle-free, degree-5 variable-regular codes, with $r_{\max} > 2.6$ . . . . .                                                                                                                    | 94  |
| Table 4.8: | Extreme Relative Estimation Errors for the Spectral Radius . . . . .                                                                                                                                                                                   | 96  |
| Table 5.1: | Minimum distance of AR4JA protomatrices after first QC expansion (independent of block length). . . . .                                                                                                                                                | 131 |
| Table 5.2: | Distance of CCSDS AR4JA parity-check matrices. . . . .                                                                                                                                                                                                 | 132 |
| Table 5.3: | Estimated weight spectrum for CCSDS AR4JA rate-4/5, $k = 1024$ using search parameters: $I = 150$ and $T = 10$ . . . . .                                                                                                                               | 132 |
| Table 5.4: | Estimated weight spectrum for CCSDS AR4JA rate-4/5, $k = 4096$ using search parameters: $I = 300$ and $T = 19$ . . . . .                                                                                                                               | 132 |
| Table 5.5: | Estimated weight spectrum for CCSDS AR4JA rate-2/3, $k = 1024$ using search parameters: $I = 300$ and $T = 19$ . . . . .                                                                                                                               | 133 |
| Table 5.6: | Girth of the CCSDS AR4JA codes. . . . .                                                                                                                                                                                                                | 136 |
| Table 6.1: | Intermediate computations and bounds on $\rho(G_1)$ from Corollary 6.12 for all $(k, L)$ , such that $k + L \leq 4$ . . . . .                                                                                                                          | 156 |
| Table 6.2: | Lower and upper bounds on $\rho(G_1)$ from Theorem 6.11 with $M = 1$ . . . . .                                                                                                                                                                         | 157 |
| Table 6.3: | Lower and upper bounds on $\rho(G_1)$ from Theorem 6.7 with $L = 1$ . . . . .                                                                                                                                                                          | 157 |

## ACKNOWLEDGEMENTS

The past several years at UCSD have been an endeavor that I will never forget.

First of all, I would like to extend my sincere gratitude to my advisor, Professor Paul H. Siegel, for his invaluable advice, unreserved support, and for spending so much valuable time in guiding my doctoral research. His enthusiasm for research and mathematical precision has made a strong impression on me. From him, I not only learned how to structure mathematical proofs, but how to improve my writing more generally. His attention to detail is truly impressive. I feel very fortunate to have had him as my advisor.

I wish I could thank the late Prof. Jack K. Wolf. Jack was my favorite instructor ever. He brought a true love of teaching and an enthusiasm for the subject to the classroom. I was very fortunate to work with Jack at QUALCOMM, attend his courses, have him on my doctoral committee for a time, and have him as a friend and colleague for years. Additionally, I'm grateful to Toby Wolf and the Wolf family for their friendship and enthusiastic support.

I thank Profs. Tara Javidi, Daniele Micciancio, Laurence B. Milstein, and Alexander Vardy for their time and effort in serving on my committee and for their valuable comments on my research. I thank Prof. Alon Orlitsky, my initial advisor in the UCSD graduate program, for his cheerfulness and guidance. I thank Profs. Neal Bertram, Kenneth Kreutz-Delgado, Lance Small, and Bhaskar Rao, all of UCSD, for their wisdom and insightful discussions.

I am also very grateful to several more mentors. In particular, I thank Dr. Roberto Padovani of QUALCOMM and Prof. John I. Molinder of Harvey Mudd College for their encouragement of my work and for inspiring a deep interest in communications and coding. I also thank Drs. Kenneth Andrews and Dariush Divsalar of the Jet Propulsion Laboratory, Prof. Richard Brualdi of the University of Wisconsin-Madison, Prof. Christian Schlegel of Halhousie University, Prof. Roxana Smarandache of the University of Notre Dame, and Dr. Pascal Vontobel of HP Labs for their guidance and encouragement. I especially thank Pascal for carefully reviewing two of my papers in his role as associate editor of the *IEEE Transactions on Information Theory*.

I wish to thank many of my former colleagues at QUALCOMM for creating an energetic work place with very high standards. The following individuals, who I came to know at QUALCOMM, were particularly helpful in guiding me through the world of coding during my studies: Thomas Richardson, Andrew J. Viterbi, and Ephriam Zehavi, now at Bar Ilan University in Israel.

I would like to thank the Signal Transmission and Recording (STAR) group members

whose tenure overlapped with mine, for providing a fascinating research environment and making the STAR group such an enjoyable place for research. I am grateful to my office mates Junsheng Han and Xiaojie (Eric) Zhang, as well as Sharon Aviran, Aman Bhatia, Ismail Demirkan, Federica Garin, Amir Hadi Djahanshahi, Bing Fan, Aravind Iyengar, Seyhan Karakulak, Scott Kayser, Marco Papaleo, Minghai Qin, Mohammad Hossein Taghavi, Zeinab Taghavi, Ido Tal, Saeed Sharifi Tehrani, Han Wang, Zheng (Jane) Wu, and Eitan Yaakobi for the camaraderie and insightful discussions. I would like to especially thank Eitan and Xiaojie for providing the  $\text{\LaTeX}$  formatting template of this dissertation. I am thankful to the Siegel/Wolf alumni Panu Chaichanavong, Michael Cheng, Jilei Hou, Brian Kurkoski, Henry Pfister, and Joseph Soriaga for their support.

I thank Dr. Yang Han and Prof. William E. Ryan for providing the parity-check matrix of the  $(640, 192)$  QC code used in Section 4.7. Also, I thank the anonymous reviewer for greatly simplifying the proof of Theorem 6.5 and providing other insights.

I wish to thank the staff of the Center for Magnetic Recording Research, particularly Ray Descoteaux, Betty Manoulian, Julie Matsuda, Iris Villanueva, and Kevin Wong for making CMRR such a wonderful place. I thank the staff of the department of Electrical and Computer Engineering, especially Samantha Garcia, Laura Hurt, Gennie Miranda, Karol Previte, Shana Slebiada, and Bernadette Villaluz for their excellent guidance and administrative support.

Last, but most importantly, I would also like to extend my deepest gratitude to my family.

This research was supported in part by the Center for Magnetic Recording Research at University of California, San Diego and the National Science Foundation under Grants CCF-0829865 and Grant CCF-1116739.

Chapter 3 is in part a reprint of the material in the paper: Brian K. Butler and Paul H. Siegel, “Numerical Issues Affecting LDPC Error Floors,” in *Proceedings of the IEEE Global Communication Conference (Globecom 2012)*, Anaheim, CA, pp. 3225–3231, December 9–13, 2012. Chapter 4 is in part a reprint of the material in the papers: Brian K. Butler and Paul H. Siegel, “Error Floor Approximation for LDPC Codes in the AWGN Channel,” submitted to *IEEE Transactions on Information Theory*, 2012 and Brian K. Butler and Paul H. Siegel, “Error Floor Approximation for LDPC Codes in the AWGN Channel,” in *Proceedings of the 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton 2011)*, Monticello, IL, pp. 203–211, Sept. 28–30, 2011. Chapter 5 is in part a reprint of the material in the papers: Brian K. Butler and Paul H. Siegel, “Bounds on the Minimum Distance of Punctured Quasi-Cyclic LDPC Codes,” to appear in *IEEE Transactions on Information Theory*, vol. 59, 2013 and

Brian K. Butler and Paul H. Siegel, “On Distance Properties of Quasi-Cyclic Protograph-Based LDPC Codes,” in *Proceedings of the IEEE International Symposium on Information Theory (ISIT 2010)*, Austin, TX, pp. 809–813, June 13–18, 2010. Chapter 6 is in part a reprint of the material in the paper: Brian K. Butler and Paul H. Siegel, “Sharp Bounds on the Spectral Radius of Nonnegative Matrices and Digraphs,” to appear in *Linear Algebra and its Applications*, vol. 439, pp. 1468–1478, 2013. The dissertation author was the primary investigator and author of all these papers.

## VITA

|           |                                                                                                          |
|-----------|----------------------------------------------------------------------------------------------------------|
| 1989      | B.S. in Engineering, Harvey Mudd College                                                                 |
| 1990      | M.S. in Electrical Engineering, Stanford University                                                      |
| 1990–2008 | QUALCOMM Incorporated, San Diego, California                                                             |
| 2013      | Ph.D. in Electrical Engineering (Communication Theory & Systems),<br>University of California, San Diego |

## PUBLICATIONS

Brian K. Butler and Paul H. Siegel, “Bounds on the Minimum Distance of Punctured Quasi-Cyclic LDPC Codes,” *IEEE Transactions on Information Theory*, to appear, vol. 59, 2013.

Brian K. Butler and Paul H. Siegel, “Sharp Bounds on the Spectral Radius of Nonnegative Matrices and Digraphs,” *Linear Algebra and its Applications*, to appear, vol. 439, pp. 1468–1478, 2013.

Brian K. Butler and Paul H. Siegel, “Error Floor Approximation for LDPC Codes in the AWGN Channel,” submitted to *IEEE Transactions on Information Theory*, 2012.

Brian K. Butler and Paul H. Siegel, “Numerical Issues Affecting LDPC Error Floors,” in *Proceedings of the IEEE Global Communication Conference (Globecom 2012)*, Anaheim, CA, pp. 3225–3231, December 9–13, 2012.

Brian K. Butler and Paul H. Siegel, “Error Floor Approximation for LDPC Codes in the AWGN Channel,” in *Proceedings of the 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton 2011)*, Monticello, IL, pp. 203–211, Sept. 28–30, 2011.

Brian K. Butler and Paul H. Siegel, “On Distance Properties of Quasi-Cyclic Protograph-Based LDPC Codes,” in *Proceedings of the IEEE International Symposium on Information Theory (ISIT 2010)*, Austin, TX, pp. 809–813, June 13–18, 2010.

Brian K. Butler, “AWGN Performance of Superorthogonal Convolutional Codes,” in *Proceedings of the IEEE Military Communication Conference (MILCOM 2006)*, Washington DC, pp. 1–5, October 23–25, 2006.

Brian K. Butler, King-Chung Lai, Keith Saints, and Bruce Meagher, “The MSM5100<sup>TM</sup>cdma-2000 + AMPS + gpsOne<sup>TM</sup>+ Bluetooth Multimode ASIC for 3G Handsets,” in *Digest of the IEEE Radio Frequency Integrated Circuits Symposium (RFIC 2002)*, Seattle, WA, pp. 186A–186F, June 2–4, 2002.

Sandip Sarkar, Brian K. Butler, and Edward G. Tiedemann, Jr., “Phone Standby Time in cdma-2000: The Quick Paging Channel in Soft Handoff,” *IEEE Transactions on Vehicular Technology*, vol. 50, pp. 1240–1249, September, 2001.

Vladimir Aparin, Brian K. Butler, and Paul J. Draxler, "Cross Modulation Distortion in CDMA Receivers," in *Digest of the IEEE MTT-S International Microwave Symposium (IMS 2000)*, Boston, MA, vol. 3, pp. 1953 - 1956, June 11–16, 2000.

Sandip Sarkar, Brian K. Butler, and Edward G. Tiedemann, Jr., "Soft Handoff on the Quick Paging Channel," in *Proceedings of the IEEE Global Communication Conference (Globecom 1999)*, Rio de Janeiro, Brazil, vol. 5, pp. 2794–2798, December 5–9, 1999.

Sandip Sarkar and Brian K. Butler, "Phone Standby Time and the Quick Paging Channel," in *Proceedings of the 10th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 1999)*, Osaka, Japan, September 12–15, 1999.

Brian K. Butler and Nicholas Yu, "The MSM3000<sup>TM</sup> dual-mode CDMA+AMPS ASIC: for increased functionality and standby time of handsets," in *Digest of the IEEE Radio Frequency Integrated Circuits Symposium (RFIC 1999)*, Anaheim, CA, pp. 69–72, June 13–15, 1999.

Roberto Padovani, Brian K. Butler, and Robert Boesel, "CDMA Digital Cellular: Field Test Results," in *Proceedings of the 44th IEEE Vehicular Technology Conference (VTC 1994)*, Stockholm, Sweden, vol. 1, pp. 11-15, June 8-10, 1994.

#### U.S. PATENTS

7,787,389, Chen, Wolf, Odenwalder, Tiedemann, Butler, Wei, "Method and system for utilization of an outer decoder in a broadcast services communication system."

7,734,257, Chen, Achour, Butler, Tiedemann, Zhang, Saints, Blessent, Wei, "Dynamic shared forward link channel for a wireless communication system."

7,190,749 & 7,903,770, Levin, Wilborn, Butler, Bender, "Method and apparatus for canceling pilot interference in a wireless communication system."

7,062,224, Baker, Butler, Wheatley, "Method and system for identifying and monitoring repeater traffic in a code division multiple access system."

7,046,701, Mohseni, Butler, John, Zhang, "System, method, and apparatus for fractional delay."

6,999,500, Mohseni, Butler, John, Zhang, "System for direct sequence spreading."

6,986,092, Butler, Wolf, Milne, "Erasure-and-single-error correction decoder for linear block codes."

6,820,103, Butler, John, Zhang, Mohseni, "Digital filter with state storage."

6,771,692 & 7,702,001, Rick, Subrahmanya, Butler, Patel, Ekvetchavit, "Time tracking in a non-negligible multipath spacing environment."

6,748,010 & 7,113,537, Butler, Zhang, Tiedemann, "Combined searching and page monitoring using offline sample storage."

6,744,749, Abrishamkar, Butler, Lin, "Method and apparatus for pilot estimation using a Wiener filter."

6,731,936 & 7,254,394 & 7,937,086, Chen, Wang, Sinnarajah, Butler, Tiedemann, "Method and system for a handoff in a broadcast communication system."

6,714,780, Antonio, Marsh, Stewart, Bjerede, Chockalingam, Kerns, Butler, Grob, Determan, Gover, Schiff, Ames, "System and method for reducing call dropping rates in a multi-beam communication system."

6,687,870, Mohseni, Butler, Zhang, "Method and apparatus for interleaving for information transmission or storage applications."

6,680,727, Bender, Butler, Jou, Zhang, "Method and apparatus for canceling pilot interference in a CDMA communication system."

6,608,858, Sih, Agrawal, Roh, Butler, Challa, "Multipath Doppler adjusted frequency tracking loop."

6,594,303, Agrawal, Bender, Blessent, Butler, Motiwala, "Coherent demodulator for use in the presence of phase discontinuities."

6,545,989 & 6,771,620, Butler, "Transmit gating in a wireless communication system."

6,535,562, Mohseni, Butler, John, Zhang, "Method and apparatus for rotating the phase of a complex signal."

6,516,360, Mohseni, Butler, John, "Method and apparatus for buffering data transmission between producer and consumer."

6,496,706, Jou, Saifuddin, Tiedemann, Butler, "Method and system for transmit gating in a wireless communication system."

6,493,376, Harms, Butler, Skinner, "Multi-layered PN code spreading in a multi-user communications system."

6,421,540, Gilhousen, Butler, Bender, Agrawal, "Method and apparatus for maximizing standby time using a quick paging channel."

6,393,295 & 6,832,094 & 7,555,302 & 7,970,420 & 7,983,695 & 8,046,005 & 8,068,859 & 8,224,356, Butler, Gilhousen, "Dual event slotted paging."

6,333,939, Butler, Yu, Easton, "Synchronization of a low power oscillator with a reference oscillator in a wireless communication device utilizing slotted paging."

6,249,539 & 6,831,940, Harms, Gregory, Butler, Agrawal, "System and method for narrowing the range of frequency uncertainty of a Doppler shifted signal."

6,243,561 & 6,584,313, Butler, Zhang, Tiedemann, Zou, Sih, Agrawal, "Offline page monitoring."

6,208,858 Antonio, Marsh, Stewart, Chockalingam, Kerns, Butler, Grob, Determan, Grover, Schiff, Ames, "System and method for reducing call dropping rates in a multi-beam communication system."

6,205,186 & 6,519,297, Butler, Bayley, Hansquine, Tiedemann, “Decoding with partial state information on a convolutionally encoded channel.”

6,134,215, Agrawal, Butler, “Using orthogonal waveforms to enable multiple transmitters to share a single CDM channel.”

6,111,865, Butler, Gilhousen, “Dual channel slotted paging.”

6,028,884, Silberger, Butler, Schiff, “Method and apparatus for measuring nonlinear effects in a communication system.”

5,909,434, Odenwalder, Butler, Padovani, Zehavi, “Bright and burst mode signaling data transmission in an adjustable rate wireless communication system.”

5,710,784, Kindred, Butler, Zehavi, Wolf “Multirate serial Viterbi decoder for code division multiple access system applications.”

5,710,521, Butler, “Out-of-band compensation for non-linear device.”

5,566,206 & 5,774,496, Butler, Padovani, Zehavi, “Method and apparatus for determining data rate of transmitted variable rate data in a communications receiver.”

5,504,773 & 5,511,073, Padovani, Tiedemann, Weaver, Butler, “Method and Apparatus for the formatting of data for transmission.”

## ABSTRACT OF THE DISSERTATION

### **Error Floors of LDPC Codes and Related Topics**

by

Brian K. Butler

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

University of California, San Diego, 2013

Professor Paul H. Siegel, Chair

Since their rediscovery in the late 1990s, low-density parity-check (LDPC) codes have rapidly become one of the most important classes of error correcting codes. LDPC codes have a graphical description and are generally decoded iteratively. The graph of an iteratively decoded code's error rate performance versus the channel quality is typically divided into two regions. The error floor region appears at higher channel quality and is characterized by a more gradual decrease in error rate as channel quality improves. For iterative decoders the error floor is apparently determined by small structures within the code that are specific to the selected graphical description of the code and subtle details of the decoding algorithm.

This thesis addresses several issues of LDPC error floors in the additive white Gaussian noise channel. First, we find the source of several well publicized error floors to be numerical

issues within the message-passing decoder. We propose numerical improvements to several decoding formulations which substantially lower error floors.

Next, we examine and refine error-floor prediction models. A time-variant linear state-space model is presented and analyzed. Graph theory is applied to analyze the small error-prone structures within the code. While we find our techniques encounter considerable problems in determining the error floors for iterative decoders with unrestricted dynamic range, their predictions are quite accurate when significant range restrictions are present.

The final issue of LDPC error floors which we address relates to the minimum distance of quasi-cyclic (QC) LDPC codes. We derive improved minimum distance upper bounds for codes which are punctured and codes which contain a significant number of zeros in their protomatrices. These bound are then applied to a family of protograph-based LDPC codes standardized for use in deep-space communications. In the case of the rate-1/2 code, our upper bounds demonstrate that the previously promulgated ensemble lower bounds do not apply to QC constructions at block lengths beyond 4400 bits.

The last part of this dissertation is a fairly separate piece of work on bounding the spectral radius of a large class of nonnegative matrices. We apply these bounds to digraphs and derive several associated equality conditions.

# Chapter 1

## Introduction

### 1.1 Background

The reliable transmission and storage of information is essential in our society—or any society which values recordable “works.” Readily encodable observations and human outputs include books, journals, drawings, pictures, music, movies, conversations, financial data, and measurements. Once encoded, storage allows this material to be preserved for future use, while transmission allows these works to be accessed and disseminated. Additionally, two-way communication allows for the possibility of interaction and commerce. The non-volatile digital storage on which these may now be recorded has exploded over the last several decades to include optical discs, magnetic tapes, hard disk drives, semiconductor flash memory, and several more in the development stages. Likewise, the means of transmission has undergone a revolution to digital. For example, most analog television broadcasts in the United States were required by the FCC to terminate in 2009 to complete the transition to the digital ATSC standard. Other countries may employ digital television broadcasting according to other standards, such as DVB-T (Europe), ISDB-T (Japan), and DTMB (China). Of course, digital transmission is utilized for many other services including broadcast music, cable television, two-way telephony, space exploration, and the very general set of services available over the Internet (*i.e.*, email, information retrieval, social networking, and commerce) whether over wires or wirelessly.

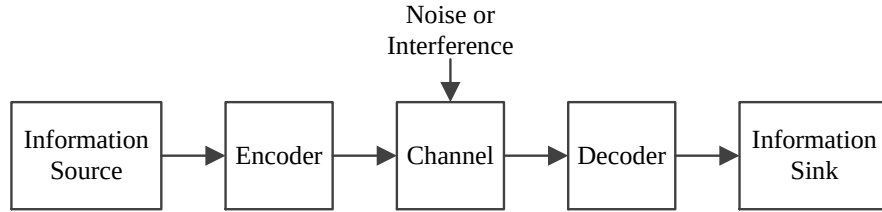


**Figure 1.1:** A simplified communication or storage system block diagram.

## 1.2 A Very Brief History of Coding

The great variety of digital storage and communication systems has a common theoretical foundation published by Claude Shannon in his 1948 paper “A Mathematical Theory of Communication” [1]. In this paper, Shannon established that all information sources have a “source rate” associated with them which can be measured in bits per second, and all channels have a “capacity” measured in the same units. Then, he showed that the source information can be transmitted over the channel, with a probability of output error as low as desired, if and only if the source rate does not exceed the channel capacity. Fig. 1.1 shows a simplified version of such digital systems. While Shannon mathematically formulated the system’s limits, his proofs were not constructive. Thus, a separate effort was needed to develop the coding techniques to approach the “Shannon capacity” of the channel.

Error correction coding techniques may be divided into three major categories. The first category is one that employs *automatic repeat requests* (ARQ). When errors are detected by the decoder, the retransmission request is sent to the encoder. The second category is one that only employs error-correcting redundancy built into the transmission and no ARQ. Since this technique does not use feedback, it is termed *forward error correction* (FEC). Thirdly, when a combination of the two techniques are used, it is called *hybrid ARQ*.

Shortly after Shannon’s landmark paper, the first nontrivial codes were published by Hamming<sup>1</sup> and Golay [2, 3]. (The trivial FEC schemes are symbol repetition and an overall parity check.) Both the Hamming and Golay codes are examples of binary linear block codes, a broad category central to this dissertation. The Hamming (7,4) code will be introduced in the next subsection. A Golay code was used on the Voyager 1 & 2 spacecraft launched in 1977 to probe the outer planets and interstellar space. Other early block codes, including Reed-Muller [4, 5], Bose-Chaudhuri-Hocquenghem (BCH) [6, 7], and Reed-Solomon [8] codes, are

---

<sup>1</sup>In fact, Hamming’s (7,4) code appears in Shannon’s paper [1] as they worked together at Bell Labs.

significant members of the family of algebraic codes.

The block codes introduced above work on single blocks of data at a time. The history of coding includes the important alternative strategy employed by “convolutional codes” or “trellis codes” which works on a continuous stream of data [9]. They encode the data much like the convolution operation of a digital filter, and their operation is easily described by a state transition diagram. Convolutional codes saw heavy use from the 1970s through the late 1990s.

Finally, our brief history of coding enters the era of modern coding with the publication of turbo codes in 1993 [10,11]. Coding gains, which had not increased in years, rose significantly and block lengths could be large if desired. Surprisingly, the decoders employed for turbo codes are iterative, but suboptimal. These codes, while encoded using concatenated convolutional encoders separated by a random interleaver, were actually block codes due to the termination of the encoders. Turbo codes have been widely used in 3G and 4G digital cellular standards and the Mars Reconnaissance Orbiter [12], for example.

It was during this period of interest in iteratively-decoded, random-like codes of the late 1990s that low-density parity-check (LDPC) codes were rediscovered. LDPC codes, an interesting class of linear block codes, were first published by Gallager in 1962 [13, 14]. The study of these codes remained largely dormant for decades, with the important exception of Tanner’s work on graph-based code constructions [15]. The renaissance of interest in these codes began with work by MacKay, Neal, and Wiberg in the late 1990s [16–18]. Progress has been rapid, with theoretic channel capacity essentially reached in some examples and standardization complete for commercial use in others, *e.g.*, DVB-S2 for satellite broadcast and IEEE 802.3an for 10 Gbit/s Ethernet. These codes will be the focus of this dissertation and introduced in detail in the following chapter.

Nearly all of the pre-1970 references may be found in [19].

### 1.3 (7,4) Hamming Code

As a motivating example of error correction and binary linear block codes we present the (7, 4) Hamming code. With a block length of 7 bits, each block protects a payload of 4 information bits. Since this code is a *linear* block code, we may describe the encoding of codewords as an operation on vectors. Consider the payload to be represented by the column vector<sup>2</sup>  $\mathbf{u}$  of 4 binary elements. Then its associated codeword is formed by the linear operation  $\mathbf{G}^T \mathbf{u}$ , where  $\mathbf{G}$

---

<sup>2</sup>Column vectors will be the notational standard for this dissertation except for Chapter 5.

is the generator matrix of the code. The specific generator matrix for the (7, 4) Hamming code is

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}. \quad (1.1)$$

The  $2^4 = 16$  possible codewords consist of *all* linear combinations of the rows of  $\mathbf{G}$ , which are the following [20]:

|                   |                |
|-------------------|----------------|
| no row            | 0 0 0 0 0 0 0  |
| row #1            | 1 0 0 0 1 1 1  |
| row #2            | 0 1 0 0 1 1 0  |
| row #3            | 0 0 1 0 0 1 1  |
| row #4            | 0 0 0 1 1 0 1  |
| #1 + #2           | 1 1 0 0 0 0 1  |
| #1 + #3           | 1 0 1 0 1 0 0  |
| #1 + #4           | 1 0 0 1 0 1 0  |
| #1 + #2 + #3      | 1 1 1 0 0 1 0  |
| #1 + #2 + #4      | 1 1 0 1 1 0 0  |
| #2 + #3 + #4      | 0 1 1 1 0 0 0  |
| #2 + #3           | 0 1 1 0 1 0 1  |
| #2 + #4           | 0 1 0 1 0 1 1  |
| #3 + #4           | 0 0 1 1 1 1 0  |
| #1 + #3 + #4      | 1 0 1 1 0 0 1  |
| #1 + #2 + #3 + #4 | 1 1 1 1 1 1 1. |

Since every codeword differs in at least 3 positions from every other codeword, we say it has minimum (Hamming) distance  $d_{\min}$  is equal to 3. One finds that the decoder for this code can employ either a single-error correction or a double-error detection strategy (but not both), for each block. If extended to  $d_{\min} = 4$  by the addition of an overall parity check, it is capable of simultaneous single-error correction *and* double-error detection.

## 1.4 Dissertation Overview

In this dissertation, we focus on several issues of LDPC error floors in the additive white Gaussian noise (AWGN) channel. The presence of these error floors has limited the adoption of LDPC codes in certain applications, such as broadcast and storage. These applications have low error-rate requirements while operating in an environment in which retransmissions are expensive or impractical. In the case of the DVB-S2 standard for satellite broadcast, an outer algebraic code was added to suppress the LDPC code's error floor.

Chapter 2 provides an introduction to LDPC codes and the most frequently encountered memoryless channels. It then introduces the most broadly used decoding techniques for LDPC codes with soft-information—those based on iterative message-passing (MP) decoding.

In Chapter 3, we begin our study of error floors with MP decoding with a focus on the numerical issues of floating-point computation. In contrast to prior observations, we show that the source of the error floors observed in the literature is often due to numerical implementation issues of the iterative decoder used. In particular, these problems arise when MP is implemented in specific domains, such as the log-likelihood ratio (LLR) domain. We examine the numerical limits and available resolution of the commonly used domains of floating-point computation. We propose numerical improvements to several domains which substantially extend the dynamic range of MP decoding and lower error floors. With these improved decoders we are able to identify the extent of LLR growth in the error floor region for two LDPC codes commonly found to have high error floors.

In Chapter 4 we examine and refine prediction models of the LDPC error floors applied to the AWGN channel. A time-variant linear state-space model is presented and analyzed. Graph theory techniques are applied to the error-prone small structures within the graphical description of the code. These techniques include bounds on the spectra, *i.e.*, the eigenvalue of maximum modulus, of graphs and digraphs. While we find these techniques encounter significant problems when applied to decoders without dynamic range limits, they perform quite well at predicting error floor levels for decoders with dynamic range limits.

We specialize to quasi-cyclic (QC) LDPC codes in Chapter 5. This class of LDPC codes has proven to be very important in recent years due to its ordered internal structure and low-complexity encoding. First, we review recent improvements to the upper bounds on the minimum distance of QC-LDPC codes by Smarandache and Vontobel. Knowing the minimum distance is significant to estimating error floors, because in practice it tends to upper bound the size of the error-prone structures within the code. We derive improved minimum distance

upper bounds for codes which are punctured and for codes which contain a significant number of zeros in their protomatrices. These bound are then applied to the AR4JA family of LDPC codes standardized for use in deep-space communications. In the case of the rate-1/2 AR4JA code, we show that our upper bounds dominate the previously promulgated ensemble lower bounds at block lengths larger than 4400 bits.

Chapter 6 is a self-contained study which develops new sharp bounds on the spectral radius of nonnegative matrices and digraphs. Our interest in this subject area derives from the linear modeling of trapping sets which involves the dominant eigenvalues of nonnegative matrices as described in Chapter 4. The spectral radius  $\rho(\mathbf{A})$  of a matrix  $\mathbf{A}$  is the maximum modulus of the eigenvalues. We present bounds on  $\rho(\mathbf{A})$  that are often tighter and are applicable to a larger class of nonnegative matrices than previously reported. The bounds are particularly suited to matrices which are sparse. We then apply these bounds to digraphs, deriving the associated equality conditions which relate to the outdegree regularity of the digraph. Finally, we show that the equality conditions may be achieved only for very specific values of the digraph's spectral radius.

## Bibliography

- [1] C. E. Shannon, "A mathematical theory of communication," *Bell Systems Tech. J.*, vol. 27, pp. 379–423, 623–656, Jul., Oct. 1948.
- [2] R. W. Hamming, "Error detecting and error correcting codes," *Bell Systems Tech. J.*, vol. 29, pp. 147–160, Apr. 1950.
- [3] M. J. E. Golay, "Notes on digital coding," *Proc. IRE (Corresp.)*, vol. 37, p. 657, Jun. 1949.
- [4] D. E. Muller, "Application of Boolean algebra to switching circuit design and to error detection," *IRE Trans. Electron. Comput.*, vol. 3, pp. 6–12, Sep. 1954.
- [5] I. S. Reed, "A class of multiple-error-correcting codes and the decoding scheme," *IRE Trans. Inform. Theory*, vol. 4, pp. 38–49, Sep. 1954.
- [6] A. Hocquenghem, "Codes correcteurs d'erreurs (Error correcting codes)," *Chiffres*, vol. 2, pp. 147–156, 1959.
- [7] R. C. Bose and D. K. Ray-Chaudhuri, "On a class of error correcting binary group codes," *Inform. Control*, vol. 2, pp. 68–79, Mar. 1960.
- [8] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *J. Soc. Indust. Appl. Math.*, vol. 8, pp. 300–304, Jun. 1960.
- [9] P. Elias, "Coding for noisy channels," *IRE Conv. Rec.*, vol. 4, pp. 37–46, 1955.

- [10] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near shannon limit error-correcting coding and decoding: Turbo codes," in *Proc. IEEE Int. Conf. Commun.*, Geneva, Switzerland, May 1993, pp. 1064–1070.
- [11] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: turbo-codes," *IEEE Trans. Commun.*, vol. 44, no. 10, pp. 1261–1271, Oct. 1996.
- [12] R. J. McEliece, "Shannon lecture XXVI," Jun. 2004, Chicago, IL.
- [13] R. G. Gallager, "Low-density parity-check codes," *IRE Trans. Inform. Theory*, pp. 21–28, Jan. 1962.
- [14] —, *Low-Density Parity-Check Codes*. Cambridge, MA: M.I.T. Press, 1963.
- [15] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inf. Theory*, vol. 27, no. 5, pp. 533–547, Sep. 1981.
- [16] N. Wiberg, H.-A. Loeliger, and R. Kötter, "Codes and iterative decoding on general graphs," *Eur. Trans. on Telecomm.*, vol. 6, pp. 513–525, Sep./Oct. 1995.
- [17] N. Wiberg, "Codes and decoding on general graphs," Ph.D. dissertation, Dept. Elect. Eng., Linköping Univ., Linköping, Sweden, Apr. 1996.
- [18] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low density parity check codes," *Electronics Letters*, vol. 32, no. 18, pp. 1645–1646, Aug. 1996.
- [19] E. R. Berlekamp, Ed., *Key Papers in the Development of Coding Theory*, ser. IEEE Press Selected Reprint Series. New York, NY: IEEE Press, 1974.
- [20] J. K. Wolf, "Algebraic coding," UCSD, 1992, unpublished, course notes.

# Chapter 2

## Introduction to LDPC Codes

### 2.1 Binary Linear Block Codes

This section provides a short description of linear block codes. The interested reader is referred to [1–3] for a more complete treatment. Before proceeding further into the realm of linear block codes, it is useful to be aware of Elias’s findings of 1955 [4]. At the time, the search for useful codes at short block lengths was dominated by linear block codes. Interestingly, Elias showed, at long block lengths on the binary symmetric channel, that the best codes possible were not substantially better than the average linear block codes. This implied that good codes with reasonably simple descriptions could be easily found, but decoding them accurately might be still be very complex.

As mentioned in Section 1.2, block codes work on a single block of data at a time. The number of information symbols in the block is commonly denoted by  $k$  and may be called its *payload*. The number of symbols in the block that are sent over the channel is commonly denoted by  $n$  and is called the *block length*. Thus, the shorthand notation of  $(n, k)$  is commonly employed in describing a code such as the  $(n, k) = (7, 4)$  Hamming code of Section 1.3. Lastly, the *code rate*, which is the ratio  $k/n$ , is one measure of efficiency of a coding scheme. That is, a code with a rate close to one introduces very little added redundancy while a rate near zero introduces lots of redundancy.

The codes throughout this dissertation are binary. This means that the “symbols” mentioned above simply belong to the set  $\{0,1\}$ . There is only one finite field of size two, which we denote by  $\mathbb{F}_2$ . Addition and multiplication over  $\mathbb{F}_2$  are defined below.

$$\begin{array}{c|cc} + & 0 & 1 \\ \hline 0 & 0 & 1 \\ 1 & 1 & 0 \end{array} \qquad \begin{array}{c|cc} \times & 0 & 1 \\ \hline 0 & 0 & 0 \\ 1 & 0 & 1 \end{array} \tag{2.1}$$

Let  $\mathbf{u} \in \mathbb{F}_2^k$  and  $\mathbf{c} \in \mathbb{F}_2^n$  be column vectors representing the information payload and codeword, respectively. Using the *generator matrix*  $\mathbf{G}$  to describe the binary code  $\mathcal{C}$  we take  $\mathcal{C} = \{\mathbf{G}^T \mathbf{u} : \mathbf{u} \in \mathbb{F}_2^k\}$  as suggested in Section 1.3. Thus, the set of all codewords form a  $k$ -dimensional subspace in the vector space  $\mathbb{F}_2^n$ . Any linear code may alternatively be described according to the null space of another matrix, the parity-check matrix  $\mathbf{H}$ , that is  $\mathcal{C} = \{\mathbf{c} \in \mathbb{F}_2^n : \mathbf{H}\mathbf{c} = \mathbf{0}\}$ . Of note, the all-zero vector of length  $n$  must be a codeword in every linear block code.

The matrix  $\mathbf{H}$  always has  $n$  columns, and, assuming it is full rank, it has  $n - k$  rows. We note that, frequently, the parity check matrices used to describe LDPC codes are not full rank. Thus, some care must be taken in deducing the exact value of  $k$  from the dimensions of  $\mathbf{H}$ .

**Example 2.1** We return to the  $(7, 4)$  Hamming code of Section 1.3. For this example the parity-check matrix

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}. \tag{2.2}$$

We may verify that this is a valid  $\mathbf{H}$  to associate with  $\mathbf{G}$  of (1.1) by checking that  $\mathbf{H}\mathbf{G}^T = \mathbf{0}$  and that  $\text{rank}(\mathbf{H})$  is  $n - k = 3$ . Note that other values of  $\mathbf{H}$  may also satisfy these conditions, implying that a given code can be described by many different  $\mathbf{H}$  matrices.

Since  $\mathbf{H}$  defines the null space of the code, we can think of it describing the constraints that must be satisfied for an arbitrary binary vector  $\mathbf{x} \in \mathbb{F}_2^n$  to be a codeword. We may examine  $\mathbf{H}$  row-wise, where each row defines a (usually independent) parity check. Denote the  $j$ th row of  $\mathbf{H}$  by the vector  $\mathbf{h}_j$ . If  $\mathbf{h}_j \mathbf{x} = 0$ , then that particular constraint, known as a *parity check*, is satisfied. Of course, if  $\mathbf{h}_j \mathbf{x} = 1$ , then it is unsatisfied. Referring back to the math tables of (2.1), the parity check calculation  $\mathbf{h}_j \mathbf{x}$  is simply the modulo-2 sum of select entries of  $\mathbf{x}$ .

## 2.2 LDPC Codes

As mentioned in Section 1.2, low-density parity-check (LDPC) codes were invented by Gallager in 1962 [5, 6]. LDPC codes are linear block codes defined by the null space of a sparse parity-check matrix  $\mathbf{H}$ . Decoding algorithms for LDPC codes are generally iterative.

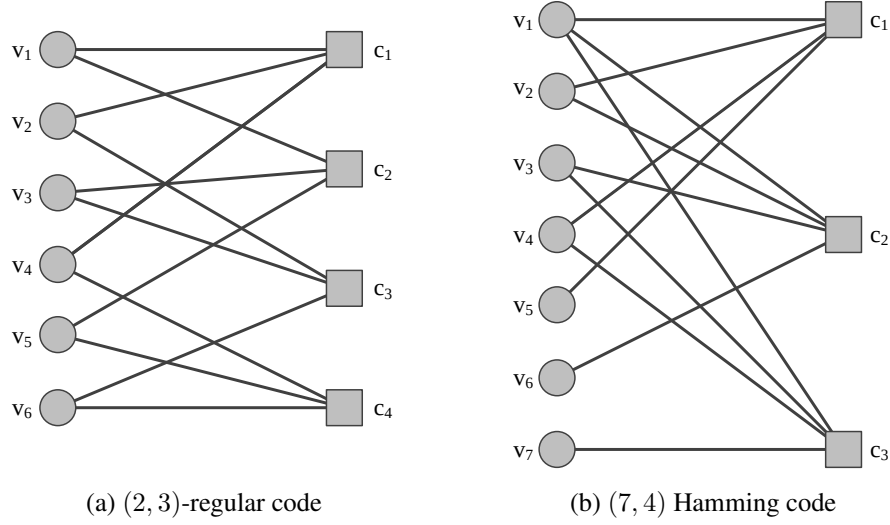


**Figure 2.1:** The Tanner graphs of two codes.

The  $\mathbf{H}$  matrix over  $\mathbb{F}_2$  may be associated with a bipartite graph  $B = (V, C, E)$  termed a *Tanner graph* [7, 8]. The vertices of any Tanner graph are divided into two types: variable nodes and check nodes. The Tanner graph  $B = (V, C, E)$  consists of a set of variable nodes  $V$ , a set of check nodes  $C$ , and a set of edges  $E$ . Each edge,  $e \in E$ , connects a variable node in  $V$  to a check node in  $C$ . The set of variable nodes  $V$  represent the symbols of the codeword that are sent over the channel and correspond to the columns of the parity-check matrix. The set of check nodes  $C$  enforce the parity-check equations represented by the rows of  $\mathbf{H}$ . Thus, there is an edge in  $B$  between  $v_i$  and  $c_j$  if and only if the entry in the  $i$ th column and  $j$ th row of  $\mathbf{H}$  is 1. For example, the Tanner graph of (2.2) is shown in Fig. 2.1b, where the numbered variables nodes correspond to the columns of (2.2) from left-to-right.

The *degree* of any node on a graph is the number of edges *incident* with it. For example, in Fig. 2.1a, every variable node is of degree 2 and every check node is of degree 3. We call such a code a  $(2, 3)$ -regular LDPC code (particularly, if it were sparser). More generally, we refer to LDPC codes with uniform variable degree  $d_v$  and uniform check degree  $d_c$  as *regular* LDPC codes, which we compactly state as  $(d_v, d_c)$ -regular. We define other LDPC codes as *irregular*, such as the Hamming code in Fig. 2.1b.

Irregular LDPC codes are characterized by their node degree distributions. We denote by  $d_l$  and  $d_r$  the maximum variable and check node degrees, respectively. We define the variable

and check *degree distributions from a node perspective* as the polynomials

$$\Lambda(x) = \sum_{i=1}^{d_l} \Lambda_i x^i \quad \text{and} \quad P(x) = \sum_{i=1}^{d_r} P_i x^i, \quad (2.3)$$

where  $\Lambda_i$  ( $P_i$ ) is the number of variable (resp., check) nodes of degree  $i$  [9, § 3.4]. We also define the normalized variable and check *degree distributions from an edge perspective* as the polynomials

$$\lambda(x) = \sum_{i=1}^{d_l} \lambda_i x^{i-1} = \frac{\Lambda'(x)}{\Lambda'(1)} \quad \text{and} \quad \rho(x) = \sum_{i=1}^{d_r} \rho_i x^{i-1} = \frac{P'(x)}{P'(1)}, \quad (2.4)$$

where  $\lambda_i$  ( $\rho_i$ ) is the the fraction of edges that connect to variable (resp., check) nodes of degree  $i$ . Note that  $\Lambda'(x)$  represents the derivative of the polynomial  $\Lambda(x)$  with respect to  $x$  and that  $\Lambda'(1) = \sum_i i \Lambda_i = \sum_i i P_i = P'(1)$  is simply the total number of edges in the graph.

Finally, we introduce the concept of neighborhoods. Two vertices on a graph are called *adjacent* if an edge exists between them. The set of *neighbors* of a vertex  $x$  are all its adjacent vertices, which we denote by  $\mathcal{N}(x)$ . We also consider the neighbors of a set of vertices to be the union of the individual neighborhoods; that is  $\mathcal{N}(\{x_1, x_2\}) = \mathcal{N}(x_1) \cup \mathcal{N}(x_2)$ . We use the backslash notation to exclude an element (or even a set of elements) from a set. Thus,  $\mathcal{N}(x) \setminus y$  are all the neighbors of  $x$ , except  $y$ .

**Example 2.2** In the Tanner graph of Fig. 2.1a, the neighbors of  $v_2$  are  $\mathcal{N}(v_2) = \{c_1, c_3\}$  and the neighbors of  $c_3$  excluding  $v_2$  are  $\mathcal{N}(c_3) \setminus v_2 = \{v_3, v_6\}$ .

## 2.3 Memoryless Channels

In this section we will describe the three memoryless channels discussed in this work. We will denote the transmitted binary code symbol as  $c_i \in \{0, 1\}$  and the channel output as  $r_i$  at each symbol time  $i$ .

First, the *binary symmetric channel* (BSC) is shown in Fig. 2.2a. The channel inverts the transmitted symbol with probability  $p$ ; that is  $p = \Pr\{r_i = 0 | c_i = 1\} = \Pr\{r_i = 1 | c_i = 0\}$ . This channel may be described quite simply using vectors of length- $n$  with entries in the field  $\mathbb{F}_2$  as  $\mathbf{r} = \mathbf{c} + \mathbf{e}$ , where  $n$  is the block length of the code. Thus, the  $i$ th entry of the error vector  $\mathbf{e}$  is 1, which forces an inversion, with probability  $p$ . The channel capacity is maximized when the transmit probabilities are symmetric, i.e.,  $\Pr\{c_i = 0\} = \Pr\{c_i = 1\} = 0.5$ . Under this condition, the channel capacity of the BSC is  $C = 1 - p \log_2(p) - q \log_2(q)$  in bits per channel use, where  $q = 1 - p$ .

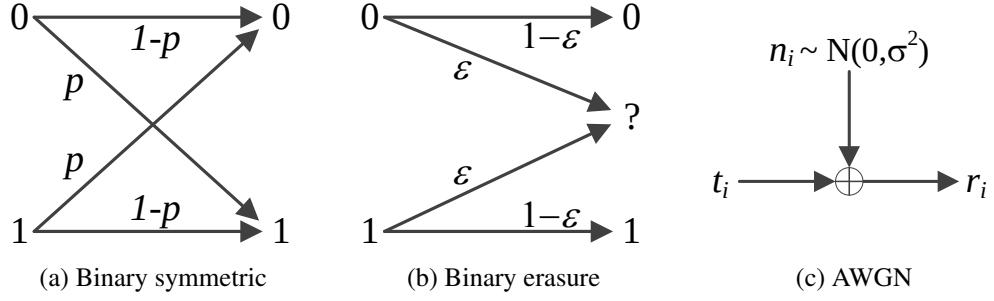


**Figure 2.2:** Three memoryless channels of interest.

Secondly, the *binary erasure channel* (BEC) is shown in Fig. 2.2b. With probability  $\epsilon$  the input binary symbols are *erased*, indicated by a “?” symbol. During erased symbol times, the channel provides no information as to whether the transmitted symbol was a 1 or a 0. The capacity is again maximized with equal transmit probabilities, which results in a channel capacity of  $C = 1 - \epsilon$  in bits per channel use.

Thirdly, the *additive white Gaussian noise* (AWGN) is shown in Fig. 2.2c. Throughout this thesis, we restrict the inputs to the AWGN channel to binary inputs. After encoding, each binary symbol  $c_i \in \{0, 1\}$  is transmitted over the AWGN channel as a binary antipodal symbol  $t_i \in \{+1, -1\}$ . Every received symbol  $r_i$  is simply the sum  $t_i + n_i$  of the transmitted symbol plus independent and identically-distributed (i.i.d.) Gaussian noise of mean zero and variance  $\sigma^2$ . This yields a channel SNR of  $1/\sigma^2$ . Equating this measure to traditional bandpass communication figures of merit for coherent detection of binary phase-shift keying, we have  $1/\sigma^2 = 2E_s/N_0 = 2RE_b/N_0$ , where  $E_s$  is the received energy per symbol-time,  $E_b$  is the received energy per information bit,  $R$  is the rate of the code in information bits per binary symbol, and  $N_0$  is the one-sided power spectral density of the noise. The capacity is now maximized when the transmit symbols are from a Gaussian distribution, which results in a channel capacity of  $C = \frac{1}{2} \log_2(1 + \frac{2E_s}{N_0})$  in bits per channel use. The prior result only restricted the signal and noise to be real values. Restricting the channel input to be binary-valued, which is more applicable to our problem, is a significantly more complex result [3, §1.5.1.3]. The channel capacity with each restriction is illustrated in Fig. 2.3.

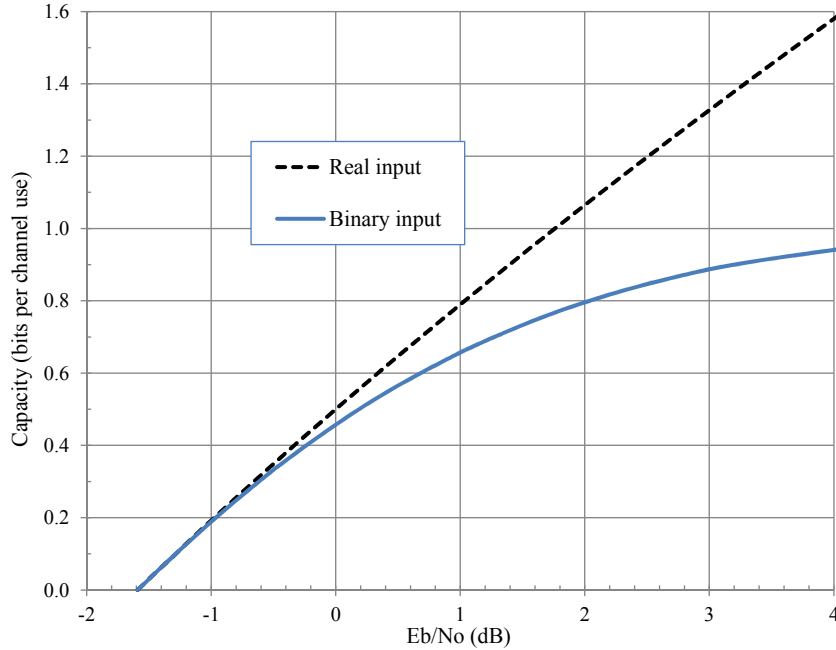


**Figure 2.3:** The channel capacity of the AWGN channel with real and binary input.

## 2.4 Ideal Decoding Bounds

The *Hamming weight* (or simply weight)  $w_H(\mathbf{a})$  of the binary vector  $\mathbf{a}$  is defined to be the number of nonzero entries in  $\mathbf{a}$ . The *Hamming distance* (or simply distance)  $d_H(\mathbf{a}, \mathbf{b})$  of the vectors  $\mathbf{a}$  and  $\mathbf{b}$  with entries in  $\mathbb{F}_2$  equals  $w_H(\mathbf{a} + \mathbf{b})$ . In effect, the Hamming distance counts the number of positions in which two binary vectors disagree. Given a block code  $\mathcal{C}$ , we are frequently interested in the *minimum distance* of  $\mathcal{C}$ , denoted by  $d_{\min}$ , which is defined as

$$d_{\min} \triangleq \min \{d_H(\mathbf{a}, \mathbf{b}) : \mathbf{a}, \mathbf{b} \in \mathcal{C}, \mathbf{a} \neq \mathbf{b}\}. \quad (2.5)$$

If  $\mathcal{C}$  is a linear block code, then the sum of two codewords is also a codeword and (2.5) simplifies to

$$d_{\min} \triangleq \min \{w_H(\mathbf{a}) : \mathbf{a} \in \mathcal{C}, \mathbf{a} \neq \mathbf{0}\}.$$

The minimum distance is a metric that readily describes the guaranteed correction capabilities of the block code. For example, the random-error correcting capability of the code is  $t = \lfloor (d_{\min} - 1)/2 \rfloor$  symbols per block. It may be possible to sometimes correct more than  $t$  errors, depending on the exact error pattern. One may envision a bounded radius decoder which corrects up to  $t$  errors over the BSC by comparing the received syndrome, which is  $\mathbf{s} = \mathbf{H}\mathbf{r} = \mathbf{H}\mathbf{e}$ , to all possible error patterns with  $t$  or fewer errors. The number of patterns

$\sum_{i=1}^t \binom{n}{i}$  may be quite large. Given such a  $t$ -error correcting decoder, the probability that it fails to decode correctly on the BSC channel is then upper bounded by [2, §3.4]

$$P_f \leq \sum_{i=t+1}^n \binom{n}{i} p^i (1-p)^{n-i}.$$

Now, we briefly address decoding for the binary-input AWGN channel. The numbers  $A_1, A_2, \dots, A_n$  are the *weight distribution* of the linear block code  $\mathcal{C}$ , where  $A_i$  is the number of codewords of weight  $i$  in  $\mathcal{C}$ . Consider a soft-decision maximum likelihood decoder (MLD) which seeks to maximize  $\Pr\{\mathbf{r}|\mathbf{c}\}$  over all possible codewords  $\mathbf{c} \in \mathcal{C}$ . By using Bayes' rule, one finds that the MLD decision  $\hat{\mathbf{c}}$ ,

$$\hat{\mathbf{c}} = \operatorname{argmax}_{\mathbf{c} \in \mathcal{C}} \Pr\{\mathbf{r}|\mathbf{c}\} = \operatorname{argmax}_{\mathbf{c} \in \mathcal{C}} \frac{\Pr\{\mathbf{c}|\mathbf{r}\} \Pr\{\mathbf{r}\}}{\Pr\{\mathbf{c}\}} = \operatorname{argmax}_{\mathbf{c} \in \mathcal{C}} \frac{\Pr\{\mathbf{c}|\mathbf{r}\}}{\Pr\{\mathbf{c}\}},$$

is equivalent to the maximizing  $\Pr\{\mathbf{c}|\mathbf{r}\}$  if every codeword is equally likely. The probability that the MLD fails to decode the block of data is upper bounded as follows [2, §10.1]

$$P_f \leq \sum_{i=d_{\min}}^n A_i Q\left(\sqrt{2iRE_b/N_0}\right), \quad (2.6)$$

where

$$Q(\alpha) \triangleq \frac{1}{\sqrt{2\pi}} \int_{\alpha}^{\infty} e^{-x^2/2} dx$$

is called the Q-function. This upper bound is known as the *union bound*. It is derived by considering decision boundaries to be midway between the points associated with codewords in the high-dimensional Euclidean space  $\mathbb{R}^n$ . In effect, maximum-likelihood (ML) decodes to the nearest codeword in terms of Euclidean distance. Since the first term of (2.6) often dominates,  $P_f$  is sometimes approximated by

$$P_f \approx A_{d_{\min}} Q\left(\sqrt{2d_{\min}RE_b/N_0}\right).$$

The idealized decoders described in this section can be constructed for convolutional codes and for algebraic codes in some cases. However, they are generally not feasible for LDPC codes of even moderate block length (*e.g.*,  $n \geq 500$  bits).

## 2.5 Sum-Product Algorithm

We next describe the sum-product algorithm (SPA) that is one of the most widely used forms of belief propagation (BP) decoding for LDPC codes. We assume that the code to be

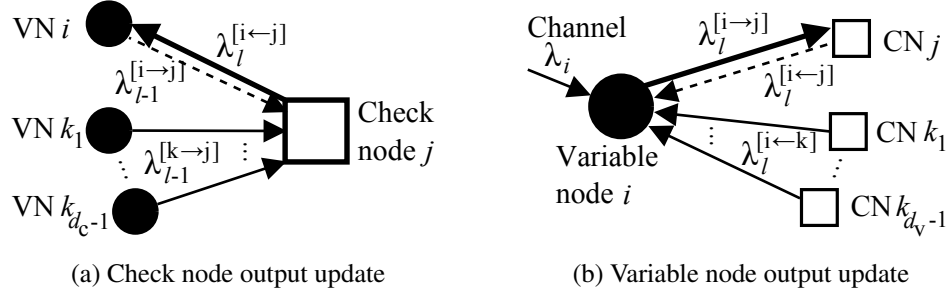


**Figure 2.4:** The two types of message updates during iteration  $l$  of the SPA.

decoded is represented by the Tanner graph  $B = (V, C, E)$ . For codes with a cycle-free Tanner graph representation, SPA decoding on such a graph is optimal, in the sense that it is equivalent to symbol-wise, maximum a-posteriori (MAP) decoding. Cycle-free codes are generally not attractive [10]. Although no longer optimal when used to decode LDPC codes whose Tanner graph representations have cycles, SPA decoding has nevertheless been found to provide near-optimal performance, at least in certain ranges of signal-to-noise ratio. Like many, we prefer to implement our decoder simulation in the log-likelihood-ratio (LLR) domain, as it avoids normalization and is closer to the approximations used in building hardware.

First, the received symbols  $r_i$  are converted to log-likelihood ratios (LLRs) defined by

$$\lambda_i = \ln \frac{P(r_i | t_i = +1)}{P(r_i | t_i = -1)}. \quad (2.7)$$

For the AWGN channel, (2.7) is the received symbol scaled by twice the SNR, *i.e.*,  $\lambda_i = 2r_i/\sigma^2$ .

The LLR  $\lambda_i$  is the initial message passed from variable node  $v_i$  to each of its adjacent check nodes in the Tanner graph. That is,  $\lambda_0^{[i \rightarrow j]} = \lambda_i$  for all  $i, j$  such that  $(v_i, c_j) \in E$ . The iteration counter  $l$  is now initialized to 1.

On the first half-iteration of iteration  $l$ , the message sent from check node  $c_j$  to the adjacent variable node  $v_i$  is given by

$$\lambda_l^{[i \leftarrow j]} = 2 \tanh^{-1} \left[ \prod_{k \in \mathcal{N}(j) \setminus i} \tanh \frac{\lambda_{l-1}^{[k \rightarrow j]}}{2} \right], \quad (2.8)$$

for each edge in the graph. During the second half-iteration, the return message sent from variable node  $v_i$  to adjacent check node  $c_j$  is given by

$$\lambda_l^{[i \rightarrow j]} = \lambda_i + \sum_{k \in \mathcal{N}(i) \setminus j} \lambda_l^{[i \leftarrow k]}. \quad (2.9)$$

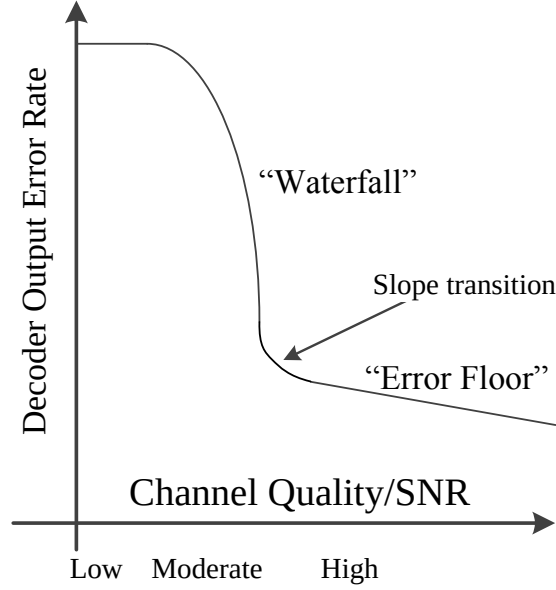


**Figure 2.5:** The general appearance of the waterfall and error floor regions in BP decoding.  
(High output error rates are undesirable.)

If the early termination logic detects a codeword or the iteration counter  $l$  exceeds the maximum allowed count, we stop the iteration loop. Otherwise, to continue iterating, we increment  $l$  and proceed to again update all the messages from the check nodes (2.8), then all the messages from the variable nodes (2.9), and so on. The message passing of this BP decoder is diagrammed in Fig. 2.4. Upon the completion of the iterations, the soft decision of the  $i$ th symbol, denoted by  $\tilde{\lambda}_l^{[i]}$ , is the incoming message sum at variable node  $v_i$ , that is,

$$\tilde{\lambda}_l^{[i]} = \lambda_i + \sum_{k \in \mathcal{N}(i)} \lambda_l^{[i \leftarrow k]} \quad (2.10)$$

and the hard decision of the  $i$ th symbol is simply  $\text{sign}(\tilde{\lambda}_l^{[i]})$ . The decoding result at iteration  $l$  is a valid codeword if the set of hard decisions satisfies all the parity check equations.

## 2.6 Error Floor

The graph of an iteratively decoded code's error rate performance versus the channel quality is typically divided into two regions as shown in Fig. 2.5. The first region, termed the *waterfall* region, occurs at poorer channel quality, close to the decoding threshold, and is characterized by a rapid drop in error rate as channel quality improves. The second region, called the

*error floor* region, is the focus of this thesis. The error floor appears at higher channel quality and is characterized by a more gradual decrease in error rate as channel quality improves.

## Bibliography

- [1] R. E. Blahut, *Algebraic Codes for Data Transmission*. Cambridge, UK: Cambridge Univ. Press, 2003.
- [2] S. Lin and D. J. Costello, Jr., *Error Control Coding*, 2nd ed. Upper Saddle River, NJ: Pearson Prentice Hall, 2004.
- [3] W. E. Ryan and S. Lin, *Channel Codes: Classical and Modern*. Cambridge, UK: Cambridge Univ. Press, 2009.
- [4] P. Elias, "Coding for noisy channels," *IRE Conv. Rec.*, vol. 4, pp. 37–46, 1955.
- [5] R. G. Gallager, "Low-density parity-check codes," *IRE Trans. Inform. Theory*, pp. 21–28, Jan. 1962.
- [6] ———, *Low-Density Parity-Check Codes*. Cambridge, MA: M.I.T. Press, 1963.
- [7] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inf. Theory*, vol. 27, no. 5, pp. 533–547, Sep. 1981.
- [8] N. Wiberg, "Codes and decoding on general graphs," Ph.D. dissertation, Dept. Elect. Eng., Linköping Univ., Linköping, Sweden, Apr. 1996.
- [9] T. J. Richardson and R. L. Urbanke, *Modern Coding Theory*. Cambridge, UK: Cambridge Univ. Press, 2008.
- [10] T. Etzion, A. Trachtenberg, and A. Vardy, "Which codes have cycle-free Tanner graphs?" *IEEE Trans. Inf. Theory*, vol. 45, no. 6, pp. 2173–2181, Sep. 1999.

## Chapter 3

# Numerical Issues Affecting LDPC Error Floors

Numerical issues related to the occurrence of error floors in floating-point simulations of belief propagation decoders are examined in this chapter. We find that carefully processing messages corresponding to highly-certain bit values can sometimes reduce error floors by several orders of magnitude. Computational solutions for properly handling such messages are provided for several formulations of the sum-product algorithm decoder. Finally, several simulation findings are briefly presented.

### 3.1 Introduction

Belief propagation (BP) decoders based upon the sum-product algorithm (SPA) are widely used to decode error-correcting codes that have a sparse graphical representation, including, most notably, low-density parity-check (LDPC) codes. This chapter addresses numerical issues arising in the implementation of SPA decoding and several of its variants that have an impact on the occurrence and severity of frequently observed error floors in the decoder performance curves. (For background on BP and LDPC coding, the reader is referred to Chapter 2.)

In [1], MacKay and Postol examined *near codewords* associated with the error floors that they observed in BP decoding of Margulis-type codes over the additive white Gaussian noise (AWGN) channel. Shortly thereafter, Richardson [2] wrote a seminal paper on the error floors of LDPC codes in the setting of the AWGN channel and the binary symmetric channel, identifying decoder-dependent, error-prone substructures in the Tanner graph, dubbed *trapping*

*sets*, that were responsible for the observed floors. Later, Han and Ryan also studied error floors and their properties for LDPC codes used on the AWGN channel [3]. With the exception of [2], which used a 5-bit hardware simulation to generate error-rate curves, it appears that the error floors reported in these prior studies were found with floating-point (FP) simulations of the BP decoder.

A quantization scheme, such as FP, imposes a limited range and finite resolution on the values to be represented. Additionally, the use of non-linear functions in a quantized environment may dramatically further limit the domains and/or images of said functions. This chapter addresses these issues of range and resolution of the messages in several SPA variants implemented in FP. Recent work has shown that the error floors of variable-regular LDPC codes, such as the Margulis code considered by the authors of [1–4], are largely the result of numerical problems associated with the processing of highly certain messages in the BP decoder implementation (see [5] and Chapter 4).

This work improves the range of several known SPA variants, introduces a new SPA formulation, and presents simulation results for two variable-regular LDPC codes whose error floors are reduced by orders of magnitude by addressing these numerical range issues. Motivated by these observations, Zhang and Siegel [6] have devised several small-bit-width decoders that substantially lower the observed error floors of several LDPC codes. These decoders use non-uniform quantization techniques that accept reduced message resolution for highly certain messages for the sake of increased message range.

## 3.2 Background

**Assumptions.** We are only concerned with codes over the binary field  $\mathbb{F}_2$ . Also, we assume binary antipodal signaling over the AWGN channel for the purposes of illustration, but the fundamental numerical issues are independent of these assumptions.

### 3.2.1 Floating-Point Formats

Floating-point (FP) formatting offers an economical computer representation (*i.e.*, quantization) of real numbers covering a wide dynamic range with a significant level of relative accuracy [7, 8]. We use *accuracy* as a general term to describe the overall degree of correctness in a real value after quantization or a set of computations. *Resolution* is the local quantization step-size. Specifically, for floating-point it represents the value of a unit in the least-significant

**Table 3.1:** Floating-point basic binary formats of IEEE Standard 754 [9].

| Name                     | Bits stored | $p$ bits | $emax$  |
|--------------------------|-------------|----------|---------|
| Single-precision (SP)    | 32          | 24       | +127    |
| Double-precision (DP)    | 64          | 53       | +1023   |
| Quadruple-precision (QP) | 128         | 113      | +16 383 |

**Table 3.2:** Floating-point extended formats of IEEE Standard 754.

| Name               | Bits stored | $p$ bits   | $emax \geq$ |
|--------------------|-------------|------------|-------------|
| Extended binary32  | $\geq 43$   | $\geq 32$  | +1023       |
| Extended binary64  | $\geq 79$   | $\geq 64$  | +16 383     |
| Extended binary128 | $\geq 145$  | $\geq 128$ | +65 535     |

position of the significand. The *precision* of an FP quantity is the number of digits in the significand, counting any implicit bits, but not counting the sign bit.

The elements of an FP number are stored separately: the sign bit, the exponent, and the significand. Table 3.1 shows the basic binary formats included in IEEE Standard 754-2008, which uses base-2 representation [9]. The parameters  $p$  and  $emax$  denote the precision and the maximum exponent, respectively. The standard requires the minimum exponent be  $emin = 1 - emax$ . Since normalized FP numbers are expressed with the radix point after the first binary digit of the significand, the maximum supported FP value may be computed to be

$$\overbrace{(1.111 \dots 111)_2}^{p \text{ ones}} \cdot 2^{emax} = (2 - 2^{1-p}) 2^{emax}.$$

Normalization of binary FP numbers is required by the standard when possible. These *normalized* numbers must have 1 as the most significant bit of the significand, which need not be stored. Very small FP numbers may become subnormal (or “denormalized”) when they are too small to be normalized. Subnormal numbers are supported by the standard and do not use the full precision available. The smallest normalized positive value is  $2^{emin}$  and the smallest subnormal positive value is

$$\overbrace{(.000 \dots 001)_2}^{p-1 \text{ bits}} \cdot 2^{emin} = 2^{emin+1-p}$$

per [9, §3.3]. The available binary FP values in the immediate vicinity of 1 are the following:

$$1 - 3 \cdot 2^{-p}, 1 - 2 \cdot 2^{-p}, 1 - 2^{-p}, 1, \text{ and } 1 + 2^{1-p}. \quad (3.1)$$

Table 3.2 shows the extended precision (binary) formats included in the standard [9, §3.7]. Implementation of the extended precision formats are not required by the standard, but “are recommended for extending the precisions used for arithmetic beyond the basic formats.” The extended precision formats are more loosely defined than the basic formats, described above. For example, the  $p$  and  $emax$  values shown are minimum values and the specific encoding details are “implementation-defined.”

The primary hardware support of extended precision formats is the x87 math unit in the Intel IA-32 architecture (and its extension, the x86-64 architecture) [10]. In the x87 unit, by default, intermediate FP computations are kept in 80-bit registers supporting extended precision ( $p = 64$ ). When the value in a register is moved to memory, the value is rounded to double-precision FP (DP-FP) unless the programmer uses the `long double` data type. Starting with the Pentium 4, which was introduced by Intel in 2001, and the Athlon 64, which was introduced by AMD in 2003, DP-FP computations may take place in the x87 unit or with the Streaming SIMD Extensions 2 (SSE2) instruction set. (SIMD stands for “single instruction, multiple data,” a parallel computer architecture with an emphasis on vectors.) The SSE2 instructions offer a choice between standard-compliant SP- and DP-FP, and it does not support extended formats. Thus, it is currently possible to achieve (perhaps inadvertently) extended precision in critical computation by leveraging the legacy math unit. The resulting program would not be very portable [10].

### 3.2.2 Numerical Problem in FP

Direct implementation of (2.8) yields numerical problems at high LLRs. Letting  $y = \tanh(\lambda/2) < 1$ , we may express  $y$  as

$$y = \frac{1 - e^{-\lambda}}{1 + e^{-\lambda}} = 1 - \frac{2e^{-\lambda}}{1 + e^{-\lambda}}$$

in order to find when FP quantizes  $y$  to 1. Ideally, the values of  $y$  near 1 are rounded to the closest quantized level listed in (3.1). So to be rounded-up to 1,  $y \geq 1 - 2^{-p}/2$  must be satisfied, which is equivalent to

$$\frac{2e^{-\lambda}}{1 + e^{-\lambda}} \leq 2^{-(p+1)} \text{ and} \quad (3.2)$$

$$\lambda + \ln(1 + e^{-\lambda}) \geq (p + 2) \ln 2. \quad (3.3)$$

We may very accurately approximate (3.3) by  $\lambda \geq (p + 2) \ln 2$  or 38.1230949 in DP-FP (64-bit IEEE 754) with its  $p = 53$  bits of precision.

As an argument of  $\pm 1$  will cause the  $\tanh^{-1}$  function to overflow, protection from high magnitude LLRs must be added to (2.8) or (2.9) to ensure numerical integrity or an alternative solution not using  $\tanh^{-1}$  must be found. Thus, preventing  $\tanh^{-1}$  overflow by limiting LLRs will result in a maximum producible LLR magnitude. Our examination of published error-floor results suggests that such LLR limiting (or “saturating” or “clipping”) is commonly employed.

### 3.3 Preferred SPA Solution

In this section we describe our preferred SPA solution, its numerical limits, and speed issues. The relationship known as the Jacobian logarithm is

$$\begin{aligned} \ln(e^x + e^y) &= \ln(e^x) + \ln(1 + e^{y-x}) \\ &= \ln(e^y) + \ln(1 + e^{x-y}) \\ &= \max(x, y) + \ln(1 + \exp(-|x - y|)). \end{aligned} \quad (3.4)$$

Using (3.4) and other identities, an alternative exact pairwise check node reduction may be derived [11, 12]. For example, if we let  $\mathcal{N}(j) \setminus i = \{k_1, k_2\}$ , the check-node message in (2.8) can be computed as follows:

$$\begin{aligned} \lambda_l^{[i \leftarrow j]} &= \text{sign } \lambda_{l-1}^{[k_1 \rightarrow j]} \cdot \text{sign } \lambda_{l-1}^{[k_2 \rightarrow j]} \cdot \\ &\quad \min \left( \left| \lambda_{l-1}^{[k_1 \rightarrow j]} \right|, \left| \lambda_{l-1}^{[k_2 \rightarrow j]} \right| \right) \\ &\quad + \ln \left( 1 + \exp \left[ - \left| \lambda_{l-1}^{[k_1 \rightarrow j]} + \lambda_{l-1}^{[k_2 \rightarrow j]} \right| \right] \right) \\ &\quad - \ln \left( 1 + \exp \left[ - \left| \lambda_{l-1}^{[k_1 \rightarrow j]} - \lambda_{l-1}^{[k_2 \rightarrow j]} \right| \right] \right) \end{aligned} \quad (3.5)$$

The check-node-message re-formulation in (3.5) contains no possibility of overflow, regardless of the LLR magnitude, which for DP-FP extends to approximately  $1.798 \cdot 10^{308}$ . The only potential for overflow in the SPA is now just the addition operation within (2.9) at extremely high LLRs.

The computer implementation of (3.5) does not necessarily have a large impact on simulation speed. A single hyperbolic tangent evaluation consumes nearly the same number of CPU cycles as four exponential or logarithmic evaluations on a modern processor in our experiments. The most significant impact is that (3.5) forces us to organize computations pairwise. Hu *et al.* present a substantial speed improvement by organizing computation pairs onto a single-state

trellis by a forward-backward algorithm, which computes *all* the output messages of the check node [11].

The next level of potential speed optimization is to approximate the  $\ln(1 + \exp(-|x|))$  operations, for which results lie in the interval  $(0, \ln 2]$ . There has been significant work in this area, much of it focused toward hardware implementation [11, 13]. For simulation, we have often adopted the following two-piece linear approximation of Richter *et al.* [14].

$$\ln(1 + e^{-|x|}) \approx \begin{cases} 0.6 - 0.24|x|, & \text{if } |x| < 2.5 \\ 0, & \text{otherwise.} \end{cases}$$

We have noted losses of about 0.02 dB in the AWGN waterfall region of the FER curve for the (2640, 1320) Margulis code with this approximation, while the simulation runs nearly 4 times faster. This is an acceptable trade-off for a decoder simulation tool used to study the error floor.

### 3.4 Additional SPA Formulations

In this section we address the numerical issues of other versions of the SPA implemented using FP computer processing. As explained in Section 3.2.2, the  $\tanh(\lambda/2)$  function loses accuracy and rounds to 1 for  $\lambda \geq (p+2)\ln 2$ . Thus, changing to larger FP formats for added precision increases the LLR limits only linearly. This section explores alternative formulations of SPA, some of which increase LLR dynamic range.

#### 3.4.1 Min-Sum Algorithm (MSA)

The min-sum algorithm (MSA) uses the following approximation to the check-node-update expression (2.8):

$$\lambda_l^{[i \leftarrow j]} = \min_{k \in \mathcal{N}(j) \setminus i} |\lambda_{l-1}^{[k \rightarrow j]}| \cdot \prod_{k \in \mathcal{N}(j) \setminus i} \text{sign } \lambda_{l-1}^{[k \rightarrow j]}. \quad (3.6)$$

Since this expression is equivalent to (3.5) with the two logarithmic terms assumed to be zero, (3.6) also will not overflow. However, the decoder performance losses that have been observed by using MSA have ranged from 0.60 to 1.22 dB [13] in AWGN, depending on the code. To reduce these losses, variations on (3.6) known as normalized-BP and offset-BP have been used successfully [4, 13]. Note that as the magnitudes of the LLRs get extremely large, the MSA closely approximates the SPA as the two logarithmic terms of (3.5) become relatively small.

### 3.4.2 Gallager's Involution Transform (GIT)

In [15], Gallager proposed another version of the check-node-update expression (2.8), using logarithms to replace the multiplications with additions. The resulting check-node update becomes

$$\lambda_l^{[i \leftarrow j]} = \prod_{k \in \mathcal{N}(j) \setminus i} \text{sign}(\lambda_{l-1}^{[k \rightarrow j]}) \cdot \Phi \left( \sum_{k \in \mathcal{N}(j) \setminus i} \Phi(|\lambda_{l-1}^{[k \rightarrow j]}|) \right),$$

where we define

$$\Phi(x) \triangleq -\ln \tanh\left(\frac{x}{2}\right) = \ln\left(\frac{1 + e^{-x}}{1 - e^{-x}}\right), \quad (3.7)$$

for  $x > 0$ . Since the function  $\Phi(x)$  is its own inverse, *i.e.*,  $\Phi(\Phi(x)) = x$ , for all  $x > 0$ , this technique is sometimes called Gallager's involution transform (GIT). Because the function  $\Phi(x)$  transforms values between domains in which addition is the primary means of computing, this technique was originally proposed for low-complexity hardware implementation of the SPA. We are also aware of its appearance in SPA simulation code, in spite of the fact that addition is no faster than multiplication on a modern FP processor.

Both expressions in (3.7) suffer finite-precision problems prior to taking the logarithm. As explained in Section 3.2.2,  $\tanh(\lambda/2)$  is rounded to 1 for  $\lambda \geq (p+2) \ln 2$ , which would yield  $\Phi(\lambda) = -\ln 1 = 0$ . Similarly,  $1 - e^{-x}$  and  $1 + e^{-x}$  are rounded to 1 for  $x \geq (p+1) \ln 2$  and  $x > p \ln 2$ , respectively. Thus, the computational limit of LLR magnitude using (3.7) is at best  $\lambda \geq (p+2) \ln 2$ .

### 3.4.3 Gallager's Involution Transform, Amended (GIT2)

Given the following series expansion in the range  $x > 0$ :

$$\ln(1 + e^{-x}) = e^{-x} - \frac{e^{-2x}}{2} + \frac{e^{-3x}}{3} - \dots,$$

the series expansion of (3.7) is readily found to be

$$\ln\left(\frac{1 + e^{-x}}{1 - e^{-x}}\right) = 2 \left[ e^{-x} + \frac{e^{-3x}}{3} + \frac{e^{-5x}}{5} + \dots \right], \quad (3.8)$$

for  $x > 0$ . This can clearly be approximated by a small number of terms as  $x$  grows large. What has been gained is that (3.8) will not round to zero until  $x > (emax + p) \ln 2$ , thus providing a substantial increase in LLR dynamic range. Also, the value  $e^{-x + \ln 2}$  will not begin to lose accuracy as it “denormalizes” until  $x > emax \ln 2$ .

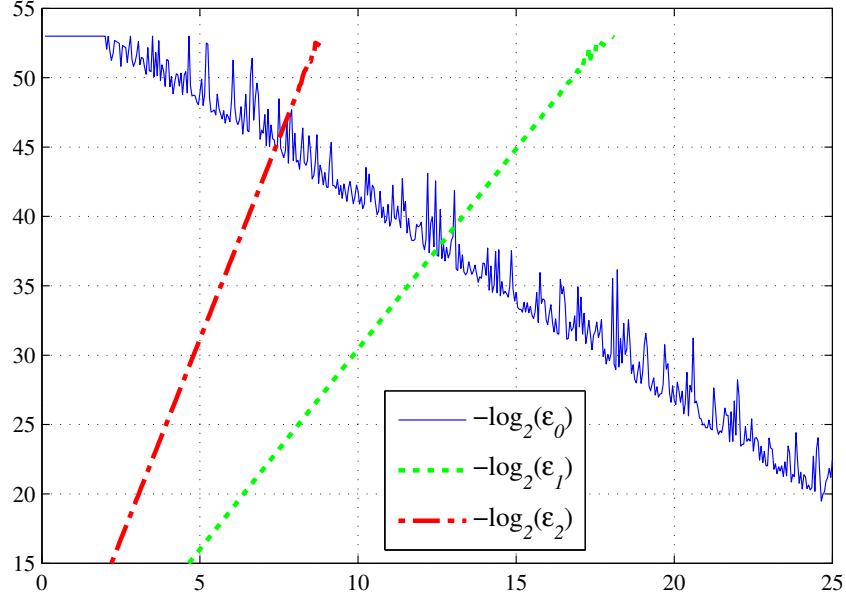


**Figure 3.1:** Numerical accuracy of several versions of  $-\ln \tanh(x/2)$  plotted as significant bits vs.  $x$  (LLR value) using DP-FP computations.

Our next step is to find the cross-over in accuracy between (3.7) and (3.8) with a limited number of terms. We propose the following computations:

$$\begin{aligned}\Phi_0(x) &= -\ln \tanh\left(\frac{x}{2}\right), \\ \Phi_1(x) &= 2e^{-x}, \\ \Phi_2(x) &= 2e^{-x} + \frac{2e^{-3x}}{3}, \quad \text{and} \\ \epsilon_i(x) &= |\Phi_i(x) - \Phi(x)| / \Phi(x),\end{aligned}$$

where we compute  $\Phi(x)$  using our best approximation at each  $x$  and not (3.7) explicitly. The expression for  $\epsilon_i(x)$  computes the relative error of the corresponding computation  $\Phi_i(x)$ .

In Fig. 3.1 we plot the relative error of each computation as  $-\log_2(\epsilon_i)$  versus  $x$  to show the accuracy in bits of the several computational versions of  $\Phi(x)$  in DP-FP. We observe a cross-over of the lower bound of the accuracy of  $\Phi_0(x)$  and the accuracy of the single-term power series  $\Phi_1(x)$  at  $x \approx 12.4$  and 37.2 significant bits of accuracy, which is still quite good. Thus, computationally, the following approximation to (3.7) has much greater dynamic range than directly implementing (3.7) on a computer, at acceptable accuracy for many applications:

$$\Phi(x) \approx \begin{cases} -\ln \tanh\left(\frac{x}{2}\right), & 0 < x < 12.4 \\ e^{-x+\ln 2}, & x \geq 12.4. \end{cases} \quad (3.9)$$

The value returned by (3.9) will not round to zero until  $x > (emax + p) \ln 2$  or 745.8 in DP-FP. If even more accuracy is desired, additional terms may be employed at an earlier cross-over. For instance, Fig. 3.1 shows that the cross-over in the accuracy of  $\Phi_0(x)$  and the two-term power series  $\Phi_2(x)$  occurs at  $x \approx 7.35$  with over 44 significant bits of accuracy. However, no larger dynamic range is achievable while employing Gallager's involution transform.

Our proposed computation (3.9) achieves a factor of  $(emax + p)/(p + 2)$  increase in the LLR dynamic range compared with (3.7), which is approximately a 20-fold improvement for DP-FP. Also, at LLR magnitudes larger than 12.4, this simple modification achieves higher accuracy with reduced computational complexity than (3.7).

### 3.4.4 Likelihood Ratio (LR)

An alternative to using  $\tanh$  in the SPA is to perform the required computations in the likelihood ratio (LR) domain [16, 17]. Interestingly, the non-uniform FP quantization of LRs maps to nearly uniform quantization in the LLR domain. If we let  $L_i \triangleq \exp \lambda_i$ ,  $L_l^{[i \rightarrow j]} \triangleq \exp \lambda_l^{[i \rightarrow j]}$ , and so on, the variable-node update becomes

$$L_l^{[i \rightarrow j]} = L_i \prod_{k \in \mathcal{N}(i) \setminus j} L_l^{[i \leftarrow k]} \quad \text{or} \quad (3.10)$$

$$L_l^{[i \rightarrow j]} = \exp \left( \ln L_i + \sum_{k \in \mathcal{N}(i) \setminus j} \ln L_l^{[i \leftarrow k]} \right). \quad (3.11)$$

The check-node update may be computed by converting (2.8) to the LR domain by noting

$$\ln L_l^{[i \leftarrow j]} = \lambda_l^{[i \leftarrow j]} = 2 \tanh^{-1}(y) = \ln \frac{1 + y}{1 - y},$$

where

$$y = \prod_{k \in \mathcal{N}(j) \setminus i} \tanh \frac{\lambda_{l-1}^{[k \rightarrow j]}}{2} = \prod_{k \in \mathcal{N}(j) \setminus i} \frac{L_{l-1}^{[k \rightarrow j]} - 1}{L_{l-1}^{[k \rightarrow j]} + 1}.$$

The check-node update simplifies rapidly by treating the inputs two-at-time. Letting  $\mathcal{N}(j) \setminus i = \{k_1, k_2\}$  and noting that

$$y = \frac{(L_{l-1}^{[k_1 \rightarrow j]} - 1)(L_{l-1}^{[k_2 \rightarrow j]} - 1)}{(L_{l-1}^{[k_1 \rightarrow j]} + 1)(L_{l-1}^{[k_2 \rightarrow j]} + 1)},$$

the pairwise check-node update reduces to

$$L_l^{[i \leftarrow j]} = \frac{1 + y}{1 - y} = \frac{1 + L_{l-1}^{[k_1 \rightarrow j]} L_{l-1}^{[k_2 \rightarrow j]}}{L_{l-1}^{[k_1 \rightarrow j]} + L_{l-1}^{[k_2 \rightarrow j]}}. \quad (3.12)$$

Andrews [17] notes that the intermediate multiplicative overflow within (3.12) is avoided if the input LRs are limited to  $L_{l-1}^{[k \rightarrow j]} < \sqrt{2^{emax}(2 - 2^{1-p})}$ , which corresponds to an LLR limit of about 354.89 in DP-FP. Alternatively, this multiplicative overflow problem may be addressed numerically, to very high accuracy, by breaking the computation into two cases according to

$$L_l^{[i \leftarrow j]} \approx \begin{cases} \frac{L_{l-1}^{[k_1 \rightarrow j]}}{1 + L_{l-1}^{[k_1 \rightarrow j]}/L_{l-1}^{[k_2 \rightarrow j]}}, & \text{if } L_{l-1}^{[k_1 \rightarrow j]} L_{l-1}^{[k_2 \rightarrow j]} \text{ overflows} \\ \frac{1 + L_{l-1}^{[k_1 \rightarrow j]} L_{l-1}^{[k_2 \rightarrow j]}}{L_{l-1}^{[k_1 \rightarrow j]} + L_{l-1}^{[k_2 \rightarrow j]}}, & \text{otherwise.} \end{cases} \quad (3.13)$$

We designate the reformulated check-node update of (3.13) the extended likelihood ratio (ELR) formulation. It may process all input LRs in the symmetric interval  $L_{max}^{-1} \leq L_l^{[i \rightarrow j]} \leq L_{max}$ , where  $L_{max} = 2^{emax}(2 - 2^{1-p})$ , without numerical issues, as explained in the following subsection. This reformulation represents an increase in equivalent LLR range by a factor of two compared with (3.12) as presented in [17].

Note that the variable-node update (3.10) is more sensitive to multiplication overflow than (3.12) for  $d_v \geq 3$ . Also, (3.10) is vulnerable to intermediate *underflow* (i.e., rounding to zero). However, (3.10) may be re-stated as (3.11) which has no intermediate overflows or underflows. We may gracefully limit the argument of the exponential in (3.11) to restrict the output LR range. Of course, (3.11) is more complex to evaluate and any resolution advantages of LR over LLR are reduced.

### 3.4.5 Analysis of Extended Likelihood Ratio (ELR)

Now we are ready to examine the numerical issues within (3.13) and its development from (3.12). Note the range of allowed input values is  $0 < L_{l-1}^{[i \rightarrow j]} < \infty$ . The operation shown in (3.12) tends to compress the LRs toward 1. This can be shown by rewriting (3.12) as

$$\ell = \frac{1 + ab}{a + b} = \min(a, b) \left( \frac{1/\min(a, b) + \max(a, b)}{a + b} \right) \quad (3.14)$$

and noting that when  $a$  and  $b$  are greater than 1, then the right side of (3.14) is less than 1. This forces the output  $\ell$  to be  $1 < \ell < \min(a, b)$ , in this case. Similarly, for two inputs less than 1, we can show that  $1 < \ell < \min(a^{-1}, b^{-1})$ . Finally, if  $a > 1$  and  $b < 1$ , a related argument shows that  $\max(a^{-1}, b) < \ell < 1$ . So, input LRs or their reciprocals are compressed toward 1.

If the multiplication in the numerator of (3.12) overflows several conditions must hold. These are  $L_{l-1}^{[k_1 \rightarrow j]} > 1$ ,  $L_{l-1}^{[k_2 \rightarrow j]} > 1$ , and  $1 + L_{l-1}^{[k_1 \rightarrow j]} L_{l-1}^{[k_2 \rightarrow j]}$  is equivalent to  $L_{l-1}^{[k_1 \rightarrow j]} L_{l-1}^{[k_2 \rightarrow j]}$  to

well within the available precision of FP. Thus, we may rewrite (3.12) as

$$L_l^{[i \leftarrow j]} = \frac{L_{l-1}^{[k_1 \rightarrow j]}}{1 + L_{l-1}^{[k_1 \rightarrow j]} / L_{l-1}^{[k_2 \rightarrow j]}}, \quad (3.15)$$

for this case. Since all LRs in (3.15) are greater than 1 for the case of interest, the expression shows no possibility of overflow or underflow. The drawback to computing (3.15) is an extra division operation, but this will typically only be employed when trying to overcome a trapping set. This completes our justification of (3.13).

We have two last issues in the second case of (3.13) to check. In FP, for positive values  $a$  and  $b$ , if  $a \cdot b$  does not overflow, we are guaranteed that  $a + b$  also will not overflow. This is not the case for integer arithmetic. For example, let  $a$  be the maximum positive value and  $b = 1$ . The calculation of  $a + b$  overflows for integers, but rounds to  $a$  for FP, since  $emax$  is larger than  $p$ . Also of concern, the division in (3.12) may overflow when the denominator is a very small positive (denormalized) FP number. This issue may be eliminated if the CN inputs are symmetrically saturated, such that  $L_{max}^{-1} \leq L_{l-1}^{[i \rightarrow j]} \leq L_{max}$ , where  $L_{max}$  may be as large as  $(2 - 2^{1-p}) 2^{emax}$ . Symmetric saturation is typically desirable in simulation studies to remove bias.

### 3.4.6 Likelihood Difference (LD) or Tanh Domain

Another alternative is to perform the computations in the likelihood difference (LD) domain, on the interval  $(-1, +1)$  [16]. This is variously known as the tanh or soft-bit domain [18]. From the point of view of quantization efficiency for hardware implementation, Howard *et al.* proposed performing SPA computation in this domain [18]. They showed uniformly quantizing in this domain to have advantages over other schemes.

If we let  $\delta_i \triangleq P(r_i | t_i = -1) - P(r_i | t_i = +1)$  and so on, the pairwise variable-node update becomes

$$\delta_l^{[i \rightarrow j]} = \frac{\delta_l^{[i \leftarrow k_1]} + \delta_l^{[i \leftarrow k_2]}}{1 + \delta_l^{[i \leftarrow k_1]} \delta_l^{[i \leftarrow k_2]}}, \quad (3.16)$$

while the check-node update is simply

$$\delta_l^{[i \leftarrow j]} = \prod_{k \in \mathcal{N}(j) \setminus i} \delta_{l-1}^{[k \rightarrow j]}. \quad (3.17)$$

The dominant numerical issue for the check-node update is the resolution of  $\delta$  in the neighborhood of  $\delta = \pm 1$ . The LD messages closest to absolute certainty in FP are  $\pm(1 - 2^{-p})$ . Since

LLRs are related to LDs by  $\lambda = \ln(1 + \delta) - \ln(1 - \delta)$ , the greatest nearly certain message available in LD is equivalent to an LLR magnitude of

$$\begin{aligned}\lambda &= \ln(2 - 2^{-p}) - \ln(2^{-p}) \\ &= (p + 1) \ln 2 + \ln(1 - 2^{-p-1}) \approx (p + 1) \ln 2,\end{aligned}\tag{3.18}$$

where our approximation is accurate to more than  $p$  significant bits. Note that the variable-node update (3.16) may suffer from rounding to  $\pm 1$  issues and divide by 0 errors for highly certain messages, but our focus in this section has been on the limitations of the check-node updates. Thus, for comparative purposes, we use (3.18) to represent the equivalent LLR limits of this formulation.

### 3.4.7 Offset-Likelihood Difference (OLD)

The magnitude of a quantization error in (normalized) FP is roughly proportional to the amplitude of the represented value [7, p. 257]. Thus, FP quantizes with greater absolute accuracy close to 0 than close to 1. This leads us to propose that LDs be offset such that check-node computations are in the form  $f = 1 - |\delta| = 1 - \tanh |\lambda/2|$ , so that highly certain messages are near  $f = 0$ . Since the check-node update for LD (3.17) is odd in every input and  $\text{sign}(\delta) = \text{sign}(\lambda)$ , we may simply handle the sign at the conclusion of the check-node-message calculation.

To begin the derivation of the check-node computation for the offset-likelihood difference (OLD) algorithm we note that

$$f_l^{[i \leftarrow j]} = 1 - \prod_{k \in \mathcal{N}(j) \setminus i} \left| \delta_{l-1}^{[k \rightarrow j]} \right|$$

simplifies significantly when performed pairwise. Letting  $\mathcal{N}(j) \setminus i = \{k_1, k_2\}$ , the check-node computation becomes

$$f_l^{[i \leftarrow j]} = f_{l-1}^{[k_1 \rightarrow j]} + f_{l-1}^{[k_2 \rightarrow j]} - f_{l-1}^{[k_1 \rightarrow j]} \cdot f_{l-1}^{[k_2 \rightarrow j]}.\tag{3.19}$$

If we wish to perform the variable-node update in the LLR domain, then we note that the transformations between domains are relatively simple, since

$$f_i = \frac{2e^{-|\lambda_i|}}{1 + e^{-|\lambda_i|}} = \frac{2}{1 + e^{|\lambda_i|}}\tag{3.20}$$

and

$$|\lambda_i| = \ln \left( \frac{2 - f_i}{f_i} \right).\tag{3.21}$$

---

**Algorithm 3.1** Offset-LD Check-Node Update for One Edge

---

```

1: procedure OFFSET_LD_CN( $n, \lambda$ )
2:    $f \leftarrow 0$   $\triangleright f$  holds the magnitude,  $0 < f \leq 1$ 
3:    $s \leftarrow +1$   $\triangleright s$  holds the sign,  $s \in \{+1, -1\}$ 
4:   for  $i \leftarrow 1, n$  do
5:      $g \leftarrow 2 * \exp(-|\lambda[i]|) / [1 + \exp(-|\lambda[i]|)]$ 
6:      $f \leftarrow f + g - f * g$ 
7:      $s \leftarrow s * \text{sign}(\lambda[i])$ 
8:   end for
9:    $\lambda_{out} \leftarrow s * \ln\left(\frac{2-f}{f}\right)$ 
10:  return  $\lambda_{out}$ 
11: end procedure

```

---

The pairwise calculation of (3.19) generalizes easily to the recursion of Algorithm 3.1. By definition, the range of  $f$  is the interval  $(0, 1]$ . For simplicity Algorithm 3.1 is written to compute a single output message; however, a check node must compute an output for each edge. Thus, the transformed LLR values from (3.20) on line 5 of Algorithm 3.1 may be pre-computed (just once) for all output messages. Furthermore, we may organize the pairwise computations on a single-state trellis as in [11].

The computational complexity of Algorithm 3.1 is less than that of (3.5); however the dynamic range is also less. The numerical dynamic range is limited due to underflowing  $2e^{-|\lambda|}$  in (3.20). This was shown in Section 3.4.3 to occur as LLRs exceed  $(emax + p) \ln 2$ .

The other numerical concern is overflowing  $2/f$  in (3.21) when  $f$  is so small that it is significantly denormalized. However, for small  $f$  (e.g.,  $f_i < 2^{-p}$ ), we may accurately restate (3.21) as  $|\lambda_i| = \ln 2 - \ln f_i$ , which has no major numerical concerns and fewer operations.

### 3.4.8 Summary of SPA Formulations

Table 3.3 summarizes the findings on LLR limits with respect to check-node inputs for a variety of FP formats. The traditional formulations based on tanh operations in the LLR-domain, Gallager's involution transform (GIT), and likelihood difference (LD) are all very limited in their usable LLR range. Our preferred approach, using the Jacobian, has a range which is greater than the others by 305 orders of magnitude or more, depending on the FP format.

Fig. 3.2 shows the output accuracy of a CN using 64-bit FP calculations for several

**Table 3.3:** Upper LLR limits of SPA formulations with respect to check node inputs.

| Domain &<br>Formulation | Equivalent LLR Upper Limit for ... |                        |                         |                         |
|-------------------------|------------------------------------|------------------------|-------------------------|-------------------------|
|                         | General Case<br>(approx.)          | binary64<br>(DP-FP)    | Extended<br>binary64    | binary128<br>(QP-FP)    |
| LLR, tanh               | $(p + 2) \ln 2$                    | 38.12                  | 45.75                   | 79.72                   |
| LLR, Jacobian           | $2^{emax+1}$                       | $1.798 \cdot 10^{308}$ | $1.190 \cdot 10^{4932}$ | $1.190 \cdot 10^{4932}$ |
| LLR, MSA <sup>a</sup>   | $2^{emax+1}$                       | $1.798 \cdot 10^{308}$ | $1.190 \cdot 10^{4932}$ | $1.190 \cdot 10^{4932}$ |
| GIT                     | $(p + 2) \ln 2$                    | 38.12                  | 45.75                   | 79.72                   |
| GIT2                    | $(emax + p) \ln 2$                 | 745.8                  | 11400                   | 11434                   |
| LR                      | $(emax + 1) \ln 2/2$               | 354.9                  | 5678                    | 5678                    |
| ELR                     | $(emax + 1) \ln 2$                 | 709.8                  | 11357                   | 11357                   |
| LD                      | $(p + 1) \ln 2$                    | 37.43                  | 45.05                   | 79.02                   |
| OLD                     | $(emax + p) \ln 2$                 | 745.8                  | 11400                   | 11434                   |

<sup>a</sup>MSA decoder approximates SPA with a performance loss of 0.6 to 1.22 dB.

SPA formulations. The MSA approximation peaks at low LLRs and then improves substantially, while offset-BP is more accurate than MSA at low LLR and less accurate at high LLR. The approximation by Richter *et al.* [14] has substantially less error than the other approximations throughout the LLR range. Finally, each of the two vertical lines shown are indicative an exact formulation reaching its upper LLR limit and its error beginning to grow linearly with respect to the input LLR value.

### 3.4.9 Hybrid SPAs

Fig. 3.2 also suggests that hybrid SPA solutions—that is, switching from a precise formulation to an approximation as LLRs grow—may be acceptable. In fact, one of our SPA decoder implementations in DP-FP switches from the Richter *et al.* approximation to MSA once a very large LLR, say  $2^{p+3}$ , is first seen. At such a point in decoding, the approximation error of the less-complex MSA is less than the resolution of the values for most of the messages in the graph.

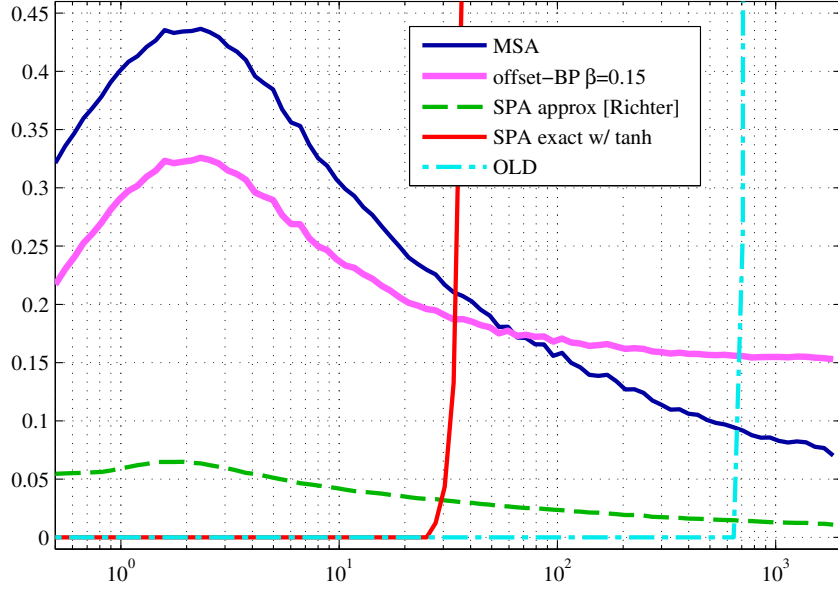


**Figure 3.2:** RMS LLR error versus mean input LLR value,  $m_\lambda$ , for check node calculation with 4 input LLRs having i.i.d. Gaussian distribution with variance  $\sigma_\lambda^2 = 2m_\lambda$ .

### 3.5 Floating-Point Resolution

Since range is not the only issue in quantization, we briefly examine floating-point (FP) resolution for the several domains covered. Fig. 3.3 shows the DP-FP resolution (or quantization step-size) of several domains plotted in terms of LLR resolution versus equivalent LLR. Since the curve for “LLR domain” is in its native domain, the resolution takes on discrete values. It is approximately proportional to the value represented, with a proportionality constant of  $\alpha = 2^{-52.5} = 1.6 \cdot 10^{-16}$ . Fig. 3.3 plots the true LLR-domain resolution  $\Delta\lambda$  as a function of the DP-FP value  $\lambda$  using the following non-linear expression

$$\Delta\lambda = \max\{2^{\lfloor \log_2 |\lambda| \rfloor - 52}, 2^{-1074}\}, \quad (3.22)$$

provided  $|\lambda|$  is a supported FP value. The value  $2^{-1074} = 2^{emin+1-p}$  dominates as  $|\lambda|$  values become small enough to denormalize.

We will plot the other domains using a simple transformation. We may start with the resolution  $\Delta z$  as a function of the FP value  $z$  as being approximately

$$\Delta z \approx \alpha |z| \text{ in native units.}$$

We need only multiply this by the magnitude of the derivative  $|d\lambda/dz|$  to transform the resolution to the LLR domain.

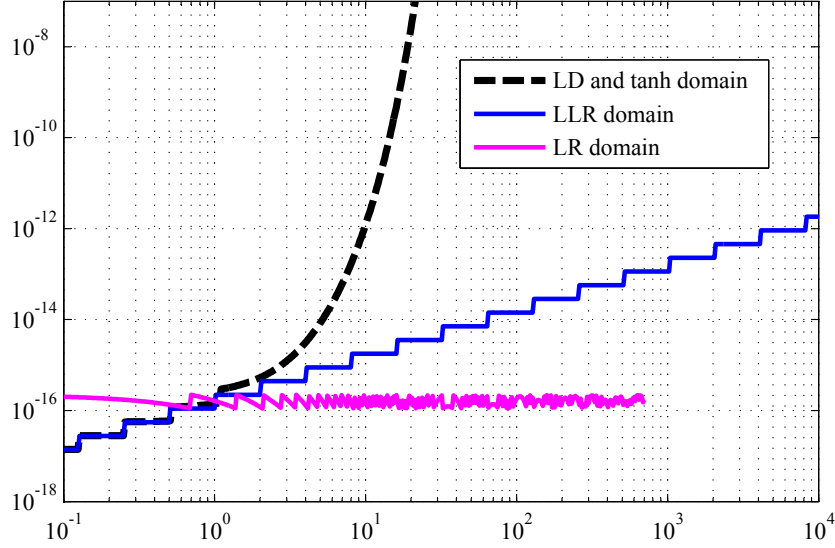


**Figure 3.3:** 64-bit FP resolution in LLR units versus equivalent LLR  $\lambda$ .

First, we examine the LR domain ( $L = e^\lambda$ ) of Section 3.4.4. Since  $|d\lambda/dL| = 1/L$ , the LR domain resolution expressed in the LLR units is simply

$$\left| \frac{d\lambda}{dL} \right| \Delta L \approx \alpha L / L = \alpha \text{ in LLR units,}$$

approximately a constant value. As shown in Fig. 3.3 the true LR resolution in LLR units dithers about our approximate result until it runs out of range.

Similarly, in the LD domain ( $\delta = \tanh(\lambda/2)$ ) of Section 3.4.6, we use the derivative  $|d\lambda/d\delta| = 2/(1 - \delta^2)$  to derive

$$\left| \frac{d\lambda}{d\delta} \right| \Delta \delta \approx \frac{2\alpha|\delta|}{1 - \delta^2} = \frac{\alpha}{2} |e^\lambda - e^{-\lambda}| \text{ in LLR units.}$$

This exponential behavior of the LD resolution is plotted in the top curve of Fig. 3.3.

Since we find that the GIT2 and OLD domains yield a resolution overlapping the LR result, we omit them in Fig. 3.3 and focus on these in Fig. 3.4. For the GIT2 domain ( $x = -\ln \tanh(|\lambda/2|)$ ) of Section 3.4.3, we use the derivative  $|d\lambda/dx| = 1/\sinh(|x|)$  to derive

$$\left| \frac{d\lambda}{dx} \right| \Delta x \approx \frac{\alpha|x|}{\sinh(|x|)} \text{ in LLR units.} \quad (3.23)$$

Since, by (3.8),  $x$  gets small exponentially fast as the LLR  $\lambda$  increases and the Taylor expansion of  $\sinh(x)$  is  $x + x^3/6 + x^5/120 + \dots$ , then (3.23) rapidly approaches the constant value  $\alpha$  as the LLR  $\lambda$  grows. Also, in Fig. 3.4, one will note that the step-size of GIT2 is computed to be less than  $\alpha$  at LLRs less than 0.5.

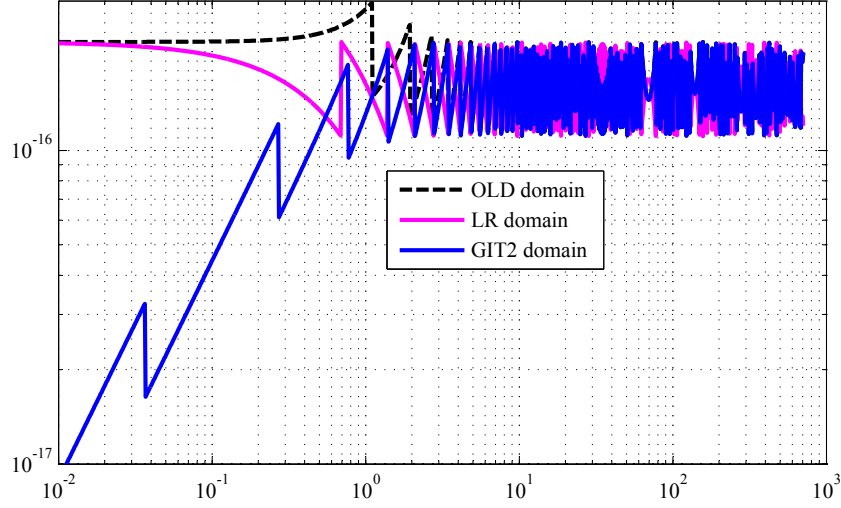


**Figure 3.4:** 64-bit FP resolution in LLR units (in magnified view) versus equivalent LLR  $\lambda$ .

For the OLD domain ( $f = 1 - \tanh(|\lambda/2|)$ ) of Section 3.4.7, we use the derivative  $|d\lambda/df| = 2/[f(f-2)]$  to derive

$$\left| \frac{d\lambda}{df} \right| \Delta f \approx \frac{2\alpha}{2-f} = \alpha \left( 1 + e^{-|\lambda|} \right) \text{ in LLR units.} \quad (3.24)$$

As the LLR  $\lambda$  gets large, (3.24) approaches the constant value  $\alpha$  exponentially fast. Even for small LLRs the approximation in (3.24) is never more than  $2\alpha$ , as can be seen in Fig. 3.4.

Note that the resolutions shown in Fig. 3.4 may be misleading in that they primarily apply to the CN update equations. In the case of the GIT2 and OLD algorithms, as described, the VN update is performed in the LLR domain and would adopt its resolution when performing the addition operations of (2.9). Alternatively, the GIT2 and OLD algorithms may perform their VN update in the LR domain. However, the LR domain has subtle VN update issues as described at the end of Section 3.4.4. That is, if the extended range of LR is desired, a solution to the intermediate overflow (and underflow) of the VN update must be found, such as transforming it to the LLR domain.

### 3.6 Simulation

We present performance simulation results for two codes to demonstrate the techniques of this chapter. Fig. 3.5 shows the frame error rate (FER) of the (2640, 1320) Margulis code, which is a (3, 6)-regular LDPC code. We show MacKay and Postol's [1] results in addition to our own. While they show the start of an error floor at  $10^{-6}$  at  $E_b/N_0 = 2.4$  dB, other studies

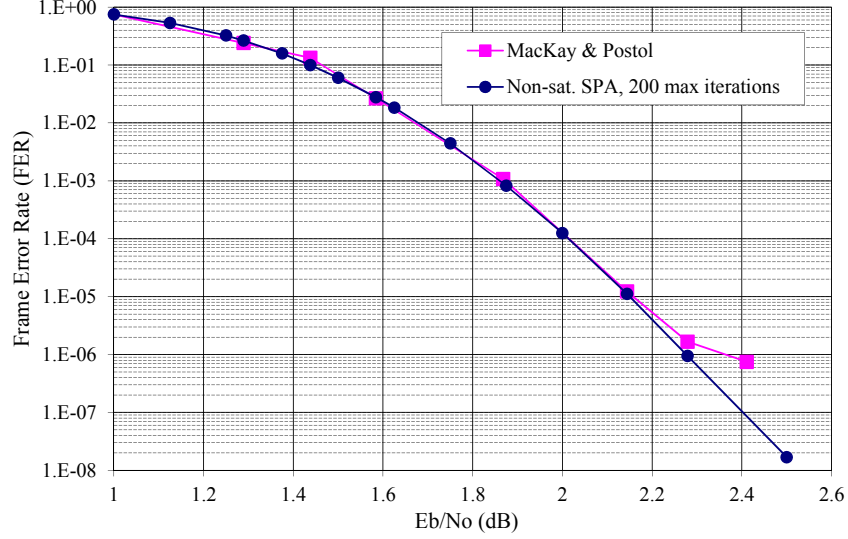


**Figure 3.5:** FER vs.  $E_b/N_0$  in dB for the Margulis LDPC code in AWGN.

have found the floor to be substantially higher [2–4]. MacKay and Postol note that this error floor is caused by certain near codewords [1]. Our own simulation, using a non-saturating SPA decoder running for a maximum of 200 iterations, showed no sign of a floor down to  $10^{-8}$  and significantly lower. In fact, resorting to techniques similar to importance sampling we found an error-rate contribution due to the supposedly dominant near codewords of just  $2 \cdot 10^{-11}$  at  $E_b/N_0 = 2.8$  dB for this decoder (see Section 4.6).

Fig. 3.6 shows the FER results for an LDPC code listed as 1057.244.3.457 in [19] and studied in [20]. It is a  $(3, 13)$ -regular code, with block length  $n = 1057$ , rate  $R \approx 0.77$ , and  $d_{\min} = 8$ . Again, we compare our results to those of MacKay. We used our non-saturating SPA decoder with three settings for the maximum number of iterations:  $10^3$ ,  $10^5$ , and  $10^7$ . The average number of decoder iterations performed at  $E_b/N_0 = 4.574$  dB was 2.290, 2.345, and 4.0, respectively. Despite the small increase in average iterations required, significant performance improvements were obtained by using the larger maximum number of iterations.

These performance results inspire us to directly investigate the limits of LLR growth. The effort can help us understand which decoding formulations of Table 3.3 are acceptable. Fig. 3.7 shows the complementary cumulative distribution function (*i.e.*,  $\text{ccdf} = 1 - \text{CDF}$ ) of the maximum LLR magnitude upon successful decoding for this code at three  $E_b/N_0$  values. The focus of this figure is on the relatively early iterations; we defined success to be a correct decoding of the block when the iteration count is in the 20–27 range. Since LLR magnitudes are typically highly correlated when the iteration count is this large (when decoding on a graph

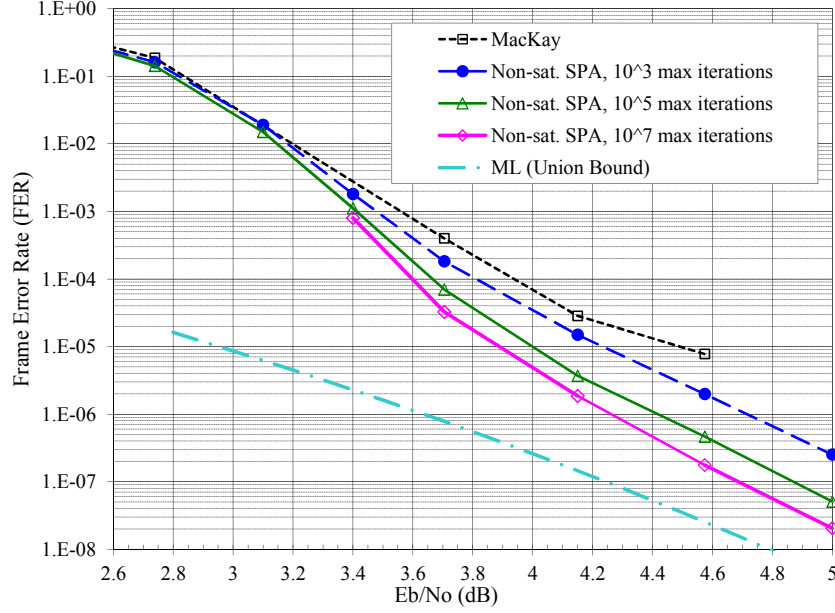


**Figure 3.6:** FER vs.  $E_b/N_0$  in dB for  $n = 1057$  LDPC code in AWGN.

with cycles), we assume that the maximum LLR magnitude is representative of many of the LLR magnitudes propagating within the graph. Under the conditions described, this figure shows a significant likelihood of LLR magnitudes exceeding several of the limits presented in Table 3.3. If we extend the iteration range of the analysis, the tail probabilities diverge from the previous case at probabilities less than  $10^{-2}$ , as shown in Fig. 3.8. At  $E_b/N_0 = 4.574$  dB, the maximum-magnitude LLR at an exceedance rate of  $10^{-4}$  is about 130 and 250 for the small and large iteration cases, respectively.

### 3.7 Conclusion

We have addressed numerical limitations on the allowable size of log-likelihood ratios (LLRs) in floating-point (FP) simulations of several formulations of the sum-product algorithm (SPA) decoder of LDPC codes. We have described preferred techniques to accommodate very large LLR values. Additionally, we have proposed simple numerical improvements to Gallager's involution transform that extend its dynamic range by a factor of about 20 for double-precision FP (DP-FP) computations and to the likelihood-ratio domain that extend its dynamic range by a factor of 2 (in LLR). Finally, we introduced a new exact SPA formulation, "offset-likelihood difference," which supports about a 20-fold improvement in the dynamic range (in LLR for DP-FP) compared to the likelihood difference formulation and has low computational complexity.

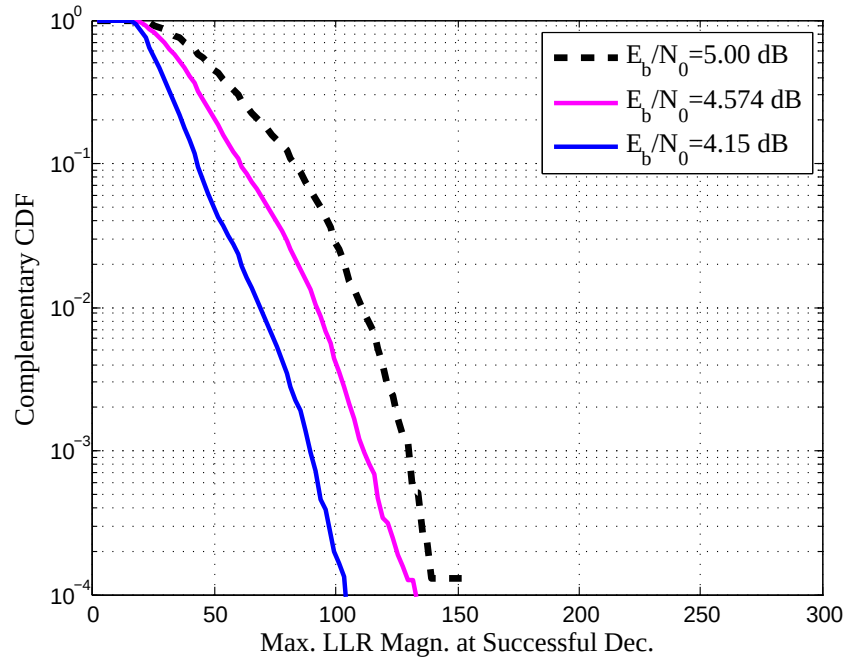


**Figure 3.7:** Tail distribution of the maximum LLR magnitude upon successful decoding of the  $n = 1057$  LDPC code conditioned on decoding success occurring when the iteration count is in the 20–27 range.

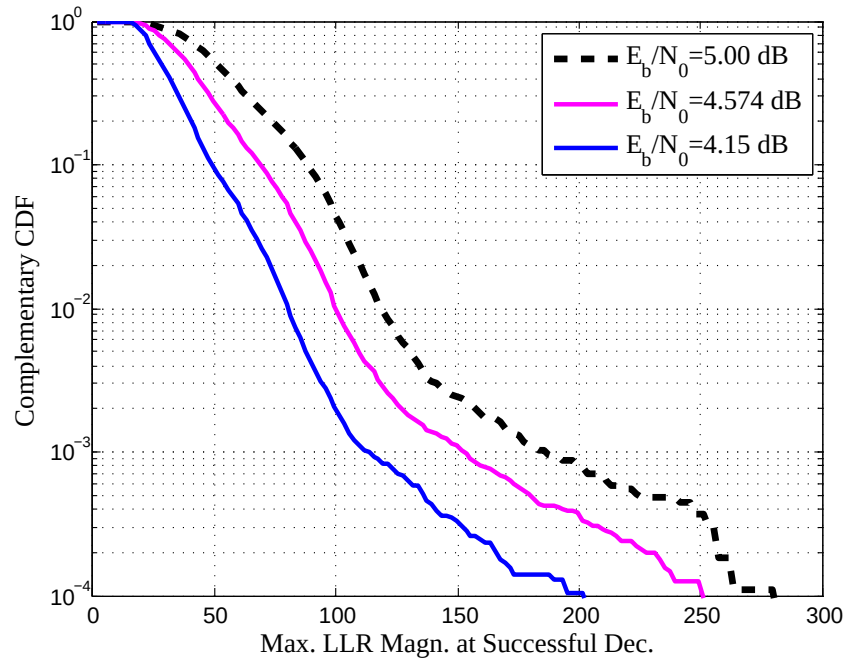


**Figure 3.8:** The same as Fig. 3.7, but conditioned on decoding success occurring when the iteration count is in the 20–100 000 range.

## Acknowledgement

This chapter is in part a reprint of the paper: Brian K. Butler and Paul H. Siegel, “Numerical Issues Affecting LDPC Error Floors,” in *Proceedings of the IEEE Global Communication Conference (GlobeCom 2012)*, Anaheim, CA, pp. 3225–3231, December 9–13, 2012. The dissertation author was the primary author of this paper.

## Bibliography

- [1] D. J. C. MacKay and M. S. Postol, “Weaknesses of Margulis and Ramanujan–Margulis low-density parity-check codes,” in *Proc. of the 2nd Irish Conf. on the Math. Foundations of Comput. Sci. and Inf. Tech. (MFCSIT)*, ser. Electronic Notes in Theoretical Comput. Sci., vol. 74, Galway, Ireland, 2003.
- [2] T. J. Richardson, “Error-floors of LDPC codes,” in *Proc. of the 41st Annu. Allerton Conf.*, Monticello, IL, Oct. 2003, pp. 1426–1435.
- [3] Y. Han and W. E. Ryan, “Low-floor decoders for LDPC codes,” *IEEE Trans. Commun.*, vol. 57, no. 6, pp. 1663–1673, Jun. 2009.
- [4] W. E. Ryan and S. Lin, *Channel Codes: Classical and Modern*. Cambridge, UK: Cambridge Univ. Press, 2009.
- [5] J. Sun, “Studies on graph-based coding systems,” Ph.D. dissertation, Dept. Elect. Eng., Ohio State Univ., Columbus, 2004.
- [6] X. Zhang and P. H. Siegel, “Quantized min-sum decoders with low error floor for LDPC codes,” in *Proc. IEEE Int. Symp. on Inform. Theory*, Cambridge, MA, Jul. 2012, pp. 2881–2885.
- [7] B. Widrow and I. Kollár, *Quantization Noise: Roundoff Error in Digital Computation, Signal Processing, Control, and Communications*. Cambridge, UK: Cambridge Univ. Press, 2008.
- [8] D. E. Knuth, *The Art of Computer Programming, Vol. 2: Seminumerical Algorithms*, 2nd ed. Reading, MA: Addison-Wesley, 1981.
- [9] *IEEE Standard for Floating-Point Arithmetic*, IEEE Std. 754-2008, Aug 29, 2008.
- [10] D. Monniaux, “The pitfalls of verifying floating-point computations,” *ACM Trans. Program. Lang. Syst.*, vol. 30, no. 3, pp. 12:1–12:41, May 2008.
- [11] X.-Y. Hu, E. Eleftheriou, D. M. Arnold, and A. Dholakia, “Efficient implementations of the sum-product algorithm for decoding LDPC codes,” in *Proc. IEEE Global Telecommun. Conf.*, vol. 2, San Antonio, TX, Nov. 2001, pp. 1036–1036E.

- [12] A. Anastasopoulos, "A comparison between the sum-product and the min-sum iterative detection algorithms based on density evolution," in *Proc. IEEE Global Telecommun. Conf.*, vol. 2, San Antonio, TX, Nov. 2001, pp. 1021–1025.
- [13] J. Chen and M. P. C. Fossorier, "Density evolution for BP-based decoding algorithms of LDPC codes and their quantized versions," in *Proc. IEEE Global Telecommun. Conf.*, vol. 2, Taipei, Taiwan, Nov. 2002, pp. 1378–1382.
- [14] G. Richter, G. Schmidt, M. Bossert, and E. Costa, "Optimization of a reduced-complexity decoding algorithm for LDPC codes by density evolution," in *Proc. IEEE Int. Conf. on Commun.*, vol. 1, Seoul, South Korea, May 2005, pp. 642–646.
- [15] R. G. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA: M.I.T. Press, 1963.
- [16] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001.
- [17] K. Andrews, "LDPC codes: Design and shortcomings," Mar. 2001, unpublished.
- [18] S. L. Howard, V. C. Gaudet, and C. Schlegel, "Soft-bit decoding of regular low-density parity-check codes," *IEEE Trans. Circuits Syst. II*, vol. 52, no. 10, pp. 646–650, Oct. 2005.
- [19] D. J. C. MacKay. Encyclopedia of sparse graph codes. [Online]. Available: <http://www.inference.phy.cam.ac.uk/mackay/codes/data.html>
- [20] S. Barman, X. Liu, S. C. Draper, and B. Recht, "Decomposition methods for large scale LP decoding," Mar. 2012. [Online]. Available: <http://arxiv.org/abs/1204.0556>

## Chapter 4

# Error Floor Approximation for LDPC Codes in the AWGN Channel

This chapter addresses the prediction of error floors of low-density parity-check (LDPC) codes with variable nodes of constant degree in the additive white Gaussian noise (AWGN) channel. Specifically, we focus on the performance of the sum-product algorithm (SPA) decoder formulated in the log-likelihood ratio (LLR) domain. We hypothesize that several published error floor levels are due to the manner in which decoder implementations handled the LLRs at high SNRs. We employ an LLR-domain SPA decoder that does not saturate near-certain messages and find the error rates of our decoder to be lower by at least several orders of magnitude. We study the behavior of trapping sets (or near-codewords) that are the dominant cause of the reported error floors.

We develop a refined linear model, based on the work of Sun and others, that accurately predicts error floors caused by elementary trapping sets for saturating decoders. Performance results of several codes at several levels of decoder saturation are presented.

### 4.1 Introduction

Iterative decoders based upon the sum-product algorithm (SPA) are widely used to decode error-correcting codes that have a sparse graphical representation, including, most notably, low-density parity-check (LDPC) codes. This chapter refines prediction models for an LDPC code's error floor level and tools to measure the error floors when they are not observable by traditional Monte-Carlo simulation. (For background on BP and LDPC coding, the reader is

referred to Chapter 2.)

The graph of an iteratively decoded code's error rate performance versus the channel quality is typically divided into two regions. The first region, termed the *waterfall* region, occurs at poorer channel quality, close to the decoding threshold, and is characterized by a rapid drop in error rate as channel quality improves. The second region, called the *error floor* region, is the focus of the present chapter. The error floor appears at higher channel quality and is characterized by a more gradual decrease in error rate as channel quality improves. For message-passing iterative decoders the error floor is apparently determined by small structures within the code that are specific to the selected graphical description of the code.

The understanding of the LDPC code error floor has progressed significantly, but issues remain. For the binary erasure channel (BEC), the structures that limit the performance of message-passing iterative decoders as channel conditions improve are known as *stopping sets* [1]. These sets have a combinatorial description and can be enumerated to accurately predict the error floor for the BEC.

Other memoryless channels are more difficult to characterize and the presence of error floors has limited the adoption of LDPC codes in some applications. For example, in the DVB-S2 standard, an outer algebraic code is concatenated with an LDPC code to ensure a low error floor. For Margulis-type codes, MacKay and Postol found that the error floors seen in the additive white Gaussian noise (AWGN) channel with sum-product algorithm (SPA) decoding were caused by *near-codewords* in the Tanner graph [2]. Richardson wrote a seminal paper on the error floors of memoryless channels shortly afterward [3]. In it, he called near-codewords *trapping sets* and defined them with respect to the decoder in use as the error-causing structures. The parameters  $(a, b)$  are used to characterize both near-codewords and trapping sets, where  $a$  is the number of variable nodes in the set and  $b$  is the number of unsatisfied check nodes when just those variable nodes are in error. The  $(a, b)$  parameters of error-floor-causing structures are typically small.

In [3], Richardson emphasized error-floor analysis techniques for the AWGN channel. He detailed a methodology to estimate a trapping set's impact on the error rate by simulating the AWGN channel in the neighborhood of the set. His semi-analytical technique was shown to be accurate at predicting the error floors given that the trapping sets were known. The method involved significant simulation, but much less than standard Monte Carlo simulation. Roughly speaking, error floors can be measured down to frame error rates of about  $10^{-8}$  in standard computer simulation and about  $10^{-10}$  in hardware simulation, depending on code complexity and the computational resources available. Richardson's method allows us to reach orders of

magnitude further in characterizing the error floor.

Further work on trapping sets includes algorithms to enumerate the occurrence of small trapping sets in specific codes [4–8] and the characterization of trapping set parameters in random ensembles of LDPC codes [9, 10].

Several other significant works on error floors in the AWGN channel exist. For example, Dolecek *et al.* noted empirically that the trapping sets dominating the error floor, termed *absorbing sets*, had a combinatorial description [11]. Their study made use of hardware-oriented SPA decoders. A variety of works describe techniques to overcome error floors. With respect to these, we note that [12] proposes several different techniques, [13] slows decoder convergence using averaged decoding, [14] selectively biases the messages from check nodes, [15] employs informed scheduling of nodes for updating, [16] adds check equations to the parity-check matrix, and [17] replaces traditional BP with a “difference-map” message passing algorithm. In contrast to these, we present in this chapter an error floor reduction technique that makes no changes to the standard SPA decoding aside from fixing common numerical problems.

Following the pioneering work of Richardson, several authors have proposed analytical techniques to predict error floors in AWGN. In his Ph.D. thesis [18], Sun developed a model of elementary trapping set behavior based on a linear state-space model of decoder dynamics on the trapping set, using density evolution (DE) to approximate decoder behavior outside of the trapping set [18, 19]. This work has gained little attention, even though it reaches a conclusion contrary to prior results. Specifically, Sun’s model shows no error floor for elementary trapping sets (excluding codewords) in regular, infinite-length LDPC codes if the variable-degree of the code is at least three and the decoder’s metrics are allowed to grow very large. Sun is able to make similar claims for irregular LDPC codes under certain conditions. In these cases, Sun shows that the graph outside of the trapping set will eventually correct the trapping set errors; in his example, using a  $(3, 1)$  trapping set, this occurred at a mean log-likelihood ratio (LLR) of about  $10^{10}$  after about 40 iterations.

Sun’s analytical development was asymptotic in the number of iterations and the block length of the code. When applied to finite-length codes with cycles, we show that the model breaks down due to strongly correlated LLR messages when the decoder is non-saturating.

Schlegel and Zhang [5] extended Sun’s work by adding time-variant gains and channel errors from outside of the trapping set to the model. For the 802.11n code, using a model specific to the dominant  $(8, 8)$  trapping set, they then compared their analytically predicted error floors to error floors found by FPGA-based simulation and importance sampling simulation. Their results

show that the external errors have very little impact and that the error floor can be predicted quite accurately.

We generalize their model and compare predicted error floors to simulations for four different codes with several LLR limits. We find the model estimates the error floor to within 0.15 dB for three of the four codes that we examine. We also show how a model that breaks down for finite-length codes in a non-saturating scenario may work fairly well when saturated LLRs are introduced.

Brevik and O’Sullivan developed a new mathematical description of decoder convergence on elementary trapping sets using a model of the probability-domain SPA decoder [20]. While considering just the channel inputs, they are able to find regions of convergence to the correct codeword, regions of convergence to the trapping set, and regions of non-convergence typified by an oscillatory behavior.

Xiao *et al.* [21] have developed a technique to estimate the error floor in the quantized decoder case using a binary input AWGN channel. In contrast, the present chapter focuses on unquantized decoders and different floor-estimation techniques.

Several authors have found empirically that increasing LLR limits in the SPA decoder lowers the error floor [22–25]. Our findings concur with theirs. Furthermore, our results show that when care is taken to implement the SPA decoder without saturation, error floors can be lowered quite dramatically, often by as much as several orders of magnitude. In the cases we examine, the error floors are reduced by many orders of magnitude. Recent independent work by Zhang and Schlegel confirms this [26, 27].

Fig. 4.1 illustrates the improvement in performance that non-saturating decoders can provide. The curves represent the frame error rate (FER) reported in [2] for the (2640, 1320) Margulis code and the corresponding results obtained from our simulation of a non-saturating SPA decoder. The error floor found in [2] starts at an FER of about  $10^{-6}$  with an  $E_b/N_0$  of 2.4 dB. The dominant errors corresponded to near-codewords. (It should be noted that others have reported even higher error floors for this code, with an FER of  $10^{-6}$  appearing at an  $E_b/N_0$  of 2.8 dB [3, 12].) Our simulation, on the other hand, shows no evidence of an error floor. The lowest simulated point at an FER of  $1.8 \cdot 10^{-8}$  represents 154 error events observed in 8600 hours of floating-point simulation, with a maximum of 200 iterations per decoded frame. Just one of the error events was a (14, 4) near-codeword, which is one of the two dominant errors floor causing structures in the previously reported results. Thus, we see very little of the usual early signs that an error floor is developing, suggesting that any error floor would manifest itself

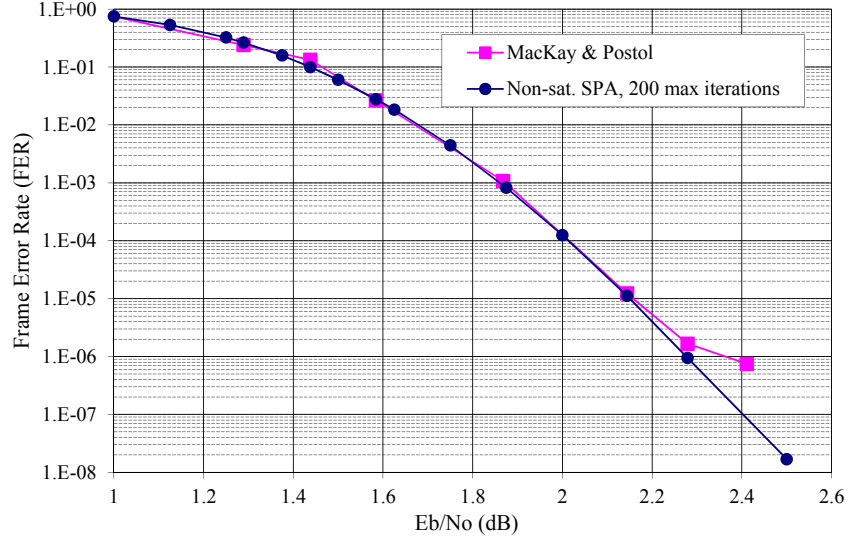


**Figure 4.1:** FER vs.  $E_b/N_0$  for (2640,1320) Margulis LDPC code in AWGN.

only at a substantially lower FER.

The framework we develop in this chapter will help to explain the empirically observed error floor properties described above. Our focus will be on the prediction of error floors for binary LDPC codes with constant variable node degree when used on the AWGN channel. The results presented here are an extension of those discussed in [3, 5, 18, 26]. Section 4.2 introduces the general terminology related to nonnegative matrices, graphs, trapping sets, and SPA decoders. We introduce message saturation as a technique to avoid numerical overflow in the SPA decoder and then describe the non-saturating decoding algorithm that will provide our performance benchmarks. Section 4.3 develops the state-space model of decoder dynamics on the trapping set we will be using for the analytical results of this chapter. It also develops approximations to the dominant eigenvalue required to characterize the system, using connections to graph theory. Section 4.4 develops the probability that the state-space model fails to converge to the correct decoding solution and the related error-rate estimates for the error floor. Section 4.5 discusses the use of the Gaussian approximation for DE to model the unsatisfied-check LLR values to determine the ultimate error floor bounds of non-saturating decoders. Section 4.6 applies a modified version of Richardson’s semi-analytical technique to estimate the error floor of a non-saturating SPA decoder. Section 4.7 demonstrates the accuracy of the state-space model’s predictions of saturated decoder performance by comparing them to simulation results for four LDPC codes. Finally, we draw our conclusions. Several additional related topics are covered in the appendices.

## 4.2 Preliminaries

### 4.2.1 Nonnegative Matrices

In this subsection we introduce the terminology of Perron-Frobenius theory of nonnegative matrices [28–31].

A vector  $\mathbf{v} \in \mathbb{R}^n$  is said to be *positive* if every element is strictly greater than zero. Likewise, the vector  $\mathbf{v}$  is said to be *nonnegative* if every element is greater than or equal to zero. In the present chapter, we use column vectors by default.

A matrix  $\mathbf{M} \in \mathbb{R}^{m \times n}$  is said to be a *positive matrix* if every entry is strictly greater than zero. Likewise, the matrix  $\mathbf{M}$  is said to be a *nonnegative matrix* if every entry is greater than or equal to zero. These conditions are denoted  $\mathbf{M} > \mathbf{0}$  and  $\mathbf{M} \geq \mathbf{0}$ , respectively. Moreover, the nonnegative matrix  $\mathbf{M}$  is said to be a  $(0, 1)$ -matrix if every entry belongs to the set  $\{0, 1\}$ .

A *permutation matrix*  $\mathbf{P}$  is a square  $(0, 1)$ -matrix, in which each row and each column contains a single 1 entry. The *symmetric permutation* of the square matrix  $\mathbf{M}$  by  $\mathbf{P}$  is  $\mathbf{PMP}^T$ . The nonnegative square matrix  $\mathbf{M}$  is said to be a *reducible matrix* if there exists a permutation matrix  $\mathbf{P}$  such that  $\mathbf{PMP}^T = \begin{bmatrix} \mathbf{X} & \mathbf{Y} \\ \mathbf{0} & \mathbf{Z} \end{bmatrix}$  where  $\mathbf{X}$  and  $\mathbf{Z}$  are square submatrices. Otherwise, the matrix  $\mathbf{M}$  is said to be an *irreducible matrix*.

Given  $\mathbf{M} \in \mathbb{R}^{m \times m}$ , the set of distinct eigenvalues is called the *spectrum* of  $\mathbf{M}$ , denoted  $\sigma(\mathbf{M}) = \{\mu_1, \dots, \mu_n\}$ , where  $\mu_k \in \mathbb{C}$ . The *spectral radius* of  $\mathbf{M}$  is the real number  $\rho(\mathbf{M}) = \max \{|\mu| : \mu \in \sigma(\mathbf{M})\}$ . The *spectral circle* is a circle in the complex plane about the origin with radius  $\rho(\mathbf{M})$ .

The following are a few important results from the theory of nonnegative matrices. Let  $\mathbf{M}$  be an irreducible nonnegative matrix. The matrix  $\mathbf{M}$  has a simple, real eigenvalue  $r$  on the spectral circle. Also, there exists a positive vector  $\mathbf{w}_1$  such that  $\mathbf{w}_1^T \mathbf{M} = r \mathbf{w}_1^T$ , which is the unique positive left eigenvector of  $\mathbf{M}$  up to a positive scale factor. If  $\mathbf{M}$  has only one eigenvalue on the spectral circle, then it is said to be *primitive*. If  $\mathbf{M}$  has  $h > 1$  eigenvalues with modulus equal to  $\rho(\mathbf{M})$ , then it is said to be *imprimitive* or a cyclic matrix. The value  $h$  is known as the *index of imprimitivity* or period.

Frobenius's test for the primitivity of a nonnegative matrix  $\mathbf{M}$  is that  $\mathbf{M}$  is primitive if and only if  $\mathbf{M}^p > \mathbf{0}$  for some positive integer  $p$ . There are two tests for the irreducibility of an  $m \times m$  nonnegative matrix  $\mathbf{M}$ . For each index  $(i, j)$  pair, there exists a positive integer  $p = p(i, j)$  such that  $[\mathbf{M}^p]_{ij} > 0$  if and only if  $\mathbf{M}$  is irreducible. The second test is that  $\mathbf{M}$  is irreducible if and only if  $(\mathbf{I} + \mathbf{M})^{(m-1)} > \mathbf{0}$ .

### 4.2.2 General Graph Theory

An *undirected graph*  $G = (V, E)$  consists of a finite set of vertices  $V$  and a finite collection of edges  $E$ . Each edge is an unordered pair of vertices  $\{v_i, v_j\}$  such that the edge joins vertices  $v_i$  and  $v_j$ . Given the edge  $\{v_i, v_j\}$ , we say that vertex  $v_i$  is *adjacent* to vertex  $v_j$ , and *vice versa*. A vertex and edge are *incident* with one another if the vertex is contained in the edge.

We do not give further consideration to graphs with self-loops. A *self-loop* is an edge joining a vertex to itself. Loopless graphs are commonly known as *multigraphs*, which may contain parallel edges. *Parallel edges* are multiple inclusions of an edge in the edge collection. A *simple graph* has neither self-loops nor parallel edges.

The *order* of a graph is the number of vertices and the *size* is the number of edges. The degree  $d(v_i)$  of vertex  $v_i$  is the number of edges incident with  $v_i$ . Euler's handshaking lemma states that the sum of all degrees of a graph must be twice its number of edges. It follows from the fact that each edge in the graph must contribute 2 to the sum of all degrees. Thus, in equation form for the undirected graph  $G = (V, E)$ , it may be stated that

$$\begin{aligned} \text{order}(G) &= |V| \quad \text{and} \\ \text{size}(G) &= |E| = \frac{1}{2} \sum_{v_i \in V} d(v_i). \end{aligned}$$

A *regular graph* is a graph whose vertices are all of equal degree.

In an undirected graph  $G$ , a *walk* between two vertices is an alternating sequence of incident vertices and edges. The vertices and edges in a walk need not be distinct. The number of edges in a walk is its *length*. The vertices  $v_i$  and  $v_j$  are said to be *connected* if the graph contains a walk of any length from  $v_i$  to  $v_j$ , noting that every vertex is considered connected to itself. A graph is said to be *connected* if every pair of vertices is connected, otherwise the graph is said to be *disconnected*. The vertices of a disconnected graph may be partitioned into connected components.

Unique edges are those edges which appear only once in the edge collection of the graph. A walk that *backtracks* is a walk in which a unique edge appears twice or more in-a-row in the walk.

A *closed walk* is a walk that begins and ends on the same vertex. A *cycle* is a closed walk with no repeated edges or vertices (except the initial and final vertex) of length at least two. The *girth* of a graph is the length of its shortest cycle, if it has cycles. If every vertex in a multigraph has at least degree two, then the graph contains one or more cycles.

A *leaf* is a vertex of degree one. A *tree* is a connected graph without cycles. Trees, with at least two vertices, have leaves. We call an undirected graph *leafless* if it does not contain leaves.

We will say that two graphs are *identical* if they have equal vertex sets and equal edge collections. Two graphs are said to be *isomorphic* if there exists between their vertex sets a one-to-one correspondence having the property that whenever two vertices are adjacent in one graph, the corresponding two vertices are adjacent in the other graph. This correspondence relationship is called an *isomorphism*. The isomorphism is a relabeling of the vertices that preserves the graph's structure.

The *adjacency matrix*  $\mathbf{A}(G)$  of any simple graph  $G$  of order  $n$  is the  $n \times n$  symmetric  $(0, 1)$ -matrix whose  $(i, j)$  entry is 1 if and only if  $\{v_i, v_j\}$  is an edge of  $G$ . More generally, the adjacency matrix  $\mathbf{A}(G')$  of any multigraph  $G'$  is the symmetric nonnegative matrix whose  $(i, j)$  entry indicates the number of edges joining  $v_i$  and  $v_j$  in  $G'$ . The number of walks of exactly length  $p$  between vertices  $v_i$  and  $v_j$ , in graph  $G$ , is the  $(i, j)$  entry of  $\mathbf{A}(G)^p$ . Two graphs are isomorphic if and only if there exists a symmetric permutation that relates their adjacency matrices.

The *incidence matrix*  $\mathbf{N}(G)$  of any multigraph  $G$  of order  $n$  and size  $m$  is the  $n \times m$   $(0, 1)$ -matrix whose  $(i, j)$  entry is 1 if and only if the  $i$ th vertex is incident with  $j$ th edge of  $G$ . It then follows that the multigraph's adjacency matrix may be stated as

$$\mathbf{A}(G) = \mathbf{N}(G)\mathbf{N}(G)^T - \text{diag}(d(v_1), d(v_2), \dots, d(v_n)).$$

A *cycle graph*, denoted  $C_n$ , is a graph of order  $n \geq 2$ , in which the distinct vertices of the set  $\{v_1, v_2, \dots, v_n\}$  are joined by the edges from the collection  $\{\{v_1, v_2\}, \dots, \{v_{n-1}, v_n\}, \{v_n, v_1\}\}$ . A cycle graph is a single cycle.

A *bipartite graph*  $B = (V, C, E)$  is a special case of an undirected graph in which the graph's vertices may be partitioned into two disjoint sets  $V$  and  $C$ . Each edge  $e \in E$  of a bipartite graph joins a vertex from  $V$  to a vertex from  $C$ . Hence, bipartite graphs cannot have self-loops by definition. The parity-check matrix  $\mathbf{H}$  over  $\mathbb{F}_2$  describes a bipartite graph  $B$ , called the *Tanner graph* of  $\mathbf{H}$ , in which the vertices are known as variable nodes  $V$  and check nodes  $C$ . Tanner graphs of binary codes do not have parallel edges. The parity-check matrix  $\mathbf{H}$  is related to the adjacency matrix  $\mathbf{A}(B)$  of the Tanner graph  $B$  as

$$\mathbf{A}(B) = \begin{bmatrix} \mathbf{0} & \mathbf{H}^T \\ \mathbf{H} & \mathbf{0} \end{bmatrix}. \quad (4.1)$$

A  $d_v$ -variable-regular Tanner graph is a Tanner graph whose variable nodes all have equal degree  $d_v$ , and a  $(d_v, d_c)$  regular Tanner graph is both  $d_v$ -variable-regular and has check nodes all of degree  $d_c$ .

Given a subset of the variable nodes  $\mathcal{S} \subseteq V$ , we use  $\mathcal{N}(\mathcal{S})$  to denote the set of adjacent check nodes. Given a Tanner graph  $B = (V, C, E)$  and any set of variable nodes  $\mathcal{S} \subseteq V$ , let  $B_{\mathcal{S}}$  represents the *induced subgraph* of  $\mathcal{S}$ . That is,  $B_{\mathcal{S}} = (\mathcal{S}, \mathcal{N}(\mathcal{S}), E_{\mathcal{S}})$  is a bipartite graph containing the variable nodes  $\mathcal{S}$ , the check nodes  $\mathcal{N}(\mathcal{S})$ , and the edges between them  $E_{\mathcal{S}}$ . We will frequently refer to these induced subgraphs as Tanner subgraphs with parity-check submatrix  $\mathbf{H}_{\mathcal{S}}$ . The submatrix  $\mathbf{H}_{\mathcal{S}}$  is formed by selecting the columns of  $\mathbf{H}$  as indexed by the members of the set  $\mathcal{S}$  and optionally removing any resulting all-zero rows.

The *line graph*  $\mathcal{L}(G)$  of a multigraph  $G$  is the graph whose vertices are the edges of  $G$ . Two vertices of  $\mathcal{L}(G)$  are adjacent if and only if their corresponding edges in  $G$  have a vertex (or two) in common. In fact, two parallel edges connect vertices in  $\mathcal{L}(G)$  if and only if their corresponding edges in  $G$  are parallel. We find that the line graph's adjacency matrix is related to the incidence matrix of  $G$  by  $\mathbf{A}(\mathcal{L}(G)) = \mathbf{N}(G)^T \mathbf{N}(G) - 2\mathbf{I}$ .

For the line graph  $\mathcal{L}(G)$  of multigraph  $G = (V, E)$ ,

$$\text{order}(\mathcal{L}(G)) = |E| = \frac{1}{2} \sum_{v_i \in V} d(v_i) \quad \text{and} \quad (4.2)$$

$$\begin{aligned} \text{size}(\mathcal{L}(G)) &= \frac{1}{2} \sum_{v_i \in V} d(v_i)(d(v_i) - 1) \\ &= \frac{1}{2} \sum_{v_i \in V} d(v_i)^2 - |E|. \end{aligned} \quad (4.3)$$

The *spectral radius*  $\rho(G)$  of a multigraph  $G = (V, E)$  is defined to be the spectral radius of the matrix  $\mathbf{A}(G)$ , and it is bounded by

$$\frac{1}{|V|} \sum_{v_i \in V} d(v_i) \leq \rho(G) \leq \max_{v_i \in V} d(v_i). \quad (4.4)$$

Note that  $d(v_i)$  is just the  $i$ th row (or column) sum of  $\mathbf{A}(G)$ . Since the matrix  $\mathbf{A}(G)$  is symmetric, the lower bound follows by applying the Rayleigh quotient to  $\mathbf{A}(G)$  with an all-one vector [28, pp. 176–181]. The upper bound applies since the matrix  $\mathbf{A}(G)$  is nonnegative, and is due to Frobenius [28, p. 492]. The lower bound holds with equality if and only if the all-one vector is an eigenvector of  $\mathbf{A}(G)$  corresponding to the eigenvalue  $\rho(G)$ . If  $G$  is connected, then the upper bound holds with equality if and only if  $G$  is regular. When  $G$  is connected these conditions are equivalent.

A *directed graph* or *digraph*  $D = (Z, A)$  consists of a set of vertices  $Z$  and a collection of directed edges or *arcs*  $A$ . Each arc is an ordered pair of vertices  $(z_i, z_j)$  such that the arc is directed from vertex  $z_i$  to vertex  $z_j$ . For arc  $(z_i, z_j)$ , we call the first vertex  $z_i$  its *initial vertex* and the second vertex  $z_j$  its *terminal vertex*. We will focus on *simple digraphs* which exclude self-loops and parallel arcs.

In a digraph  $D$ , a *directed walk* is an alternating sequence of vertices and arcs from  $z_i$  to  $z_j$  in  $D$  such that every arc  $a_i$  in the sequence is preceded by its initial vertex and is followed by its terminal vertex. A digraph  $D$  is said to be *strongly connected* if for any ordered pair of distinct vertices  $(z_i, z_j)$  there is a directed walk in  $D$  from  $z_i$  to  $z_j$ . For example, the digraph in Fig. 4.4d is strongly connected.

The *adjacency matrix*  $\mathbf{A}(D)$  of any simple digraph  $D$  is the  $(0, 1)$ -matrix whose  $(i, j)$  entry is 1 if and only if  $(z_i, z_j)$  is an arc of  $D$ .

**Lemma 4.1** *A  $(0, 1)$ -matrix is irreducible if and only if the associated digraph is strongly connected.*

*Proof:* See [31, p. 78]. ■

The *outdegree*  $d_i^+$  of vertex  $z_i$  is the number of arcs in digraph  $D$  with initial vertex  $z_i$ . Likewise, the *indegree*  $d_i^-$  of vertex  $z_i$  is the number of arcs with terminal vertex  $z_i$ . For the digraph  $D = (Z, A)$ , we note that

$$\begin{aligned} \text{order}(D) &= |Z| \quad \text{and} \\ \text{size}(D) &= |A| = \sum_{z_i \in Z} d_i^+ = \sum_{z_i \in Z} d_i^-. \end{aligned}$$

A *regular* digraph is a digraph whose vertices are all of equal indegree and outdegree. The *spectral radius*  $\rho(D)$  of a digraph  $D$  is defined to be the spectral radius of its matrix  $\mathbf{A}(D)$ , and it is bounded in the following lemma.

**Lemma 4.2** *Let the  $i$ th vertex  $z_i$  of digraph  $D = (Z, A)$  have outdegree  $d_i^+$ . Then the spectral radius  $\rho(D)$  of  $D$  is bounded above and below by*

$$\min_{z_i \in Z} d_i^+ \leq \rho(D) \leq \max_{z_i \in Z} d_i^+. \quad (4.5)$$

*If  $D$  is strongly connected, then the inequalities are strict unless the digraph has regular outdegree. Analogous statements hold for the indegrees.*

*Proof:* See [28, p. 492] and [30, pp. 8 and 22]. ■

The interested reader is referred to [31–34], etc., for a more complete treatment of the subject. Our use of graph theory has parallels to [20].

### 4.2.3 LDPC Codes, the AWGN Channel, and SPA Decoding

**Assumption 4.1** We are only concerned with LDPC codes over the binary field  $\mathbb{F}_2$  and with binary antipodal signaling over the AWGN channel.

The SPA decoder would be optimal with respect to minimizing the symbol error rate, if the Tanner graph had no cycles. In any “good” finite-length code, the Tanner graph will indeed have cycles [35], but we generally find that the SPA decoder does quite well. Like many, we prefer to implement our decoder simulation in the LLR-domain, as it is close to the approximations typically used in building hardware. We will assume that the reader is familiar with the bipartite (Tanner) graphs used to describe LDPC codes and SPA decoding of LDPC codes over the AWGN channel (see page 47 and the prior chapters). Please note the convention that a full iteration consists of incrementing the iteration counter, updating the check nodes’ outputs, and then updating variable nodes’ outputs.

Direct implementation of (2.8) yields numerical problems at high LLRs. As specified in IEEE 754 [36], double-precision floating-point (*i.e.*, 64-bit) computer computations maintain 53-bits of precision, while the remaining bits are for the sign and exponent. Thus, such a computer implementation results in  $\tanh(\lambda/2)$  being rounded to  $\pm 1$  for any LLR magnitude  $|\lambda| > 38.1230949 = 55 \ln 2$ . These numerical issues are discussed at length in Chapter 3. The non-saturating SPA decoder simulation we have used is based on a pairwise check node reduction which includes the small approximation by Richter *et al.* [37] described in Section 3.3.

### 4.2.4 Trapping Sets and Absorbing Sets

**Definition 4.3 (Richardson [3])** A trapping set is an error-prone set of variable nodes  $\mathcal{T}$  that depends upon the decoder’s input space and decoding algorithm. Let  $\tilde{\lambda}_l^{[i]}(\mathbf{r})$  denote the  $i$ th soft output symbol at the  $l$ th iteration given the decoder input vector  $\mathbf{r}$ . We say that symbol  $i$  is eventually correct if there exists  $L$  such that, for all  $l \geq L$ ,  $\tilde{\lambda}_l^{[i]}(\mathbf{r})$  is correct. If the set of symbols that is not eventually correct is not the empty set, we call it a trapping set,  $\mathcal{T}(\mathbf{r})$ . It is called an  $(a, b)$  trapping set if the set has  $a = |\mathcal{T}|$  variable nodes and the induced subgraph  $B_{\mathcal{T}}$  contains exactly  $b$  check nodes of odd degree.

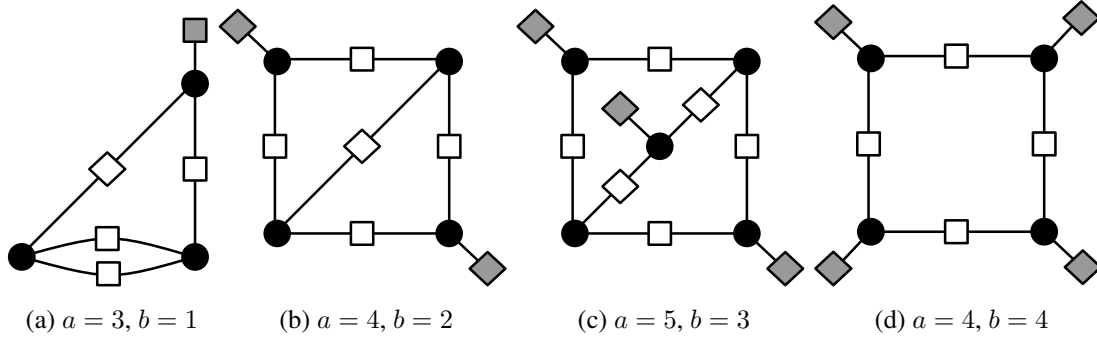


**Figure 4.2:** Four sample elementary Tanner subgraphs of a code with  $d_v = 3$ . Odd-degree check nodes are shown shaded.

From here on this chapter will use the term *trapping set*, unqualified, to mean the trapping sets defined above with respect to the AWGN channel decoded by an LLR-domain SPA decoder with saturating LLRs. Note, that while near-codewords are distinct from codewords, the trapping set definition includes codewords.

**Definition 4.4** *A Tanner subgraph or trapping set is called elementary if all the check nodes are of degree one or two [38].*

Elementary trapping sets are simple enough to be modeled with linear systems and also account for most of the error floors seen in practice. In [3, 9, 12, 38–40], the authors observe that the majority of trapping sets contributing to the error floors of belief propagation decoders are elementary trapping sets.

**Example 4.1** Four sample elementary Tanner subgraphs are shown in Fig. 4.2 for a code of variable-degree three. These are all trapping sets if they satisfy the “not eventually correct” condition of Definition 4.3.

We will define absorbing sets as they dominate the error floor of SPA decoders with saturating LLRs. Absorbing sets are a particular type of near-codeword or trapping set. The fully absorbing definition adds conditions beyond the subgraph itself to the remaining Tanner graph.

**Definition 4.5 (Dolecek *et al.* [11])** *Let  $S$  be a subset of the variable nodes  $V$  of the Tanner graph  $B = (V, C, E)$  and let  $S$  induce a subgraph having a set of odd-degree check nodes  $\mathcal{N}_o(S)$  and a set of even-degree check nodes  $\mathcal{N}_e(S)$ . If  $S$  has cardinality  $a$  and induces a subgraph with*

$|\mathcal{N}_o(\mathcal{S})| = b$ , in which each node in  $\mathcal{S}$  has strictly fewer neighbors in  $\mathcal{N}_o(\mathcal{S})$  than neighbors in  $\mathcal{N}_e(\mathcal{S})$ , then we say that  $\mathcal{S}$  is an  $(a, b)$  absorbing set. If, in addition, each node in  $V$  has strictly fewer neighbors in  $\mathcal{N}_o(\mathcal{S})$  than neighbors in the set  $C \setminus \mathcal{N}_o(\mathcal{S})$ , then we say that  $\mathcal{S}$  is an  $(a, b)$  fully absorbing set.

Fully absorbing sets are stable structures during decoding using the bit-flipping algorithm [11]. This results from every variable node of the set neighboring strictly more checks which reinforce the incorrect value than checks working to correct the variable node.

### 4.3 State-Space Model

In this section we refine and justify the existing linear system model which closely approximates the non-linear message passing of the LLR-domain SPA decoder described earlier, as applied to an elementary trapping set. Since this is a linear block code, with no loss of generality, we will frequently assume for convenience that the all-zero codeword has been sent.

The state-space model was introduced in [18, 19] to analyze elementary trapping sets. Identifying and modeling trapping sets are of interest so as to explain the observed phenomenon of the trapping sets' variable nodes being decoded incorrectly, while the variable nodes outside of the set are eventually decoded correctly by an LLR-domain SPA decoder.

**Assumption 4.2** All trapping sets of interest are elementary and contain more than one variable node. We do not consider subgraphs with a single variable node as they would not have states in the state-space model. For trapping-like behavior of single degree-1 variable nodes, see [41, p. 24].

First, we review the failure state of an elementary trapping set. Let the variable nodes of the trapping set  $\mathcal{S} \subset V$  induce the Tanner subgraph  $B_{\mathcal{S}}$ . In later iterations the condition is reached where the variable nodes  $\mathcal{S}$  within the trapping set are in error and those variable nodes outside the trapping set (*i.e.*,  $V \setminus \mathcal{S}$ ) are correct. In this condition, the subgraph's check nodes of degree one are unsatisfied and those of degree two are satisfied.

The vector of messages from the degree-one (*i.e.*, unsatisfied) check nodes  $\lambda_l^{(ex)}$  at iteration  $l$  are taken to be stochastic system inputs to the state-space model and are therefore modeled separately. We will use density evolution and simulation techniques, described in later sections, to model this input vector. The other system input vector is the intrinsic information  $\lambda$  provided by the channel to be used in variable node updates at every iteration. For the AWGN

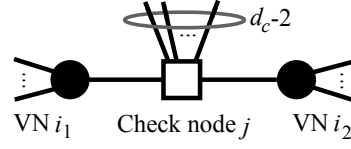


**Figure 4.3:** Illustration of a check node of degree two in the trapping set and degree  $d_c$  in the Tanner graph.

channel model, each element of  $\lambda$  is an independent and identically distributed (i.i.d.) Gaussian random variable. Both of these system inputs will be treated as column vectors of LLRs; the vector  $\lambda$  has  $a$  entries and the vector  $\lambda_l^{(ex)}$  has  $b$  entries for an  $(a, b)$  trapping set.

#### 4.3.1 Check Node Gain Model

Sun's linear model included the asymptotic approximation that every degree-two (*i.e.*, satisfied) check node output message is equal to the input message at the other edge [18]. This is based on the conditions of correct variable nodes and very high LLR values outside of the trapping set. Schlegel and Zhang introduced more accuracy to this model by applying a multiplicative gain  $g_l$  at the degree-two check nodes, where  $0 < g_l \leq 1$  [5]. This gain models the effect of the  $d_c - 2$  external variable nodes lowering the magnitude of LLR messages as they pass through the degree-two check nodes. This is illustrated in Fig. 4.3. This approach adds accuracy to modeling the early iterations, and after several iterations these gains approach 1.

**Assumption 4.3** We will assume that the external inputs to the trapping set's degree-two check nodes are i.i.d. This is generally a reasonable assumption to make as the gain computations are most significant at early iterations. By the iteration count in which significant message correlation is present on the check nodes' inputs, the gains are approximately unity.

Schlegel and Zhang compute the mean gain  $\bar{g}_l$  corresponding to each iteration count  $l \geq 1$  [5]. Thus, the gain model turns the state updates into a time-varying process. We have tried introducing a gain variance to the model, but found the effect to be very small, so we present their mean gain model<sup>1</sup>, for which the gain of the  $j$ th check node during the  $l$ th iteration is

$$g_l^{[j]} \triangleq \prod_{k \in \mathcal{N}(j) \setminus \{i_1, i_2\}} \tanh \left( \frac{\lambda_{l-1}^{[k \rightarrow j]}}{2} \right). \quad (4.6)$$

<sup>1</sup>The following includes a correction to the gain expressions of [5] related to the iteration count dependencies

Then the expected value of the check node's gain over all realizations of the channel noise vector  $\mathbf{n}$  and all check nodes  $c_j$ , such that  $d(c_j) = d_c$ , is  $\bar{g}_l(d_c) = \mathbb{E}_{\mathbf{n},j}[g_l^{[j]}]$ . Using Assumption 4.3, one forms the following simplification with respect to  $\bar{g}_l(d_c)$

$$\bar{g}_l(d_c) = \mathbb{E}_{\mathbf{n},j,k} \left[ \tanh \left( \frac{\lambda_{l-1}^{[k \rightarrow j]}}{2} \right) \right]^{d_c-2}, \quad (4.7)$$

over all  $j$  such that  $d(c_j) = d_c$  and all  $k \in \mathcal{N}(j)$ .

The application of gain at the check node may be justified using a Taylor series expansion of (2.8) that begins with letting

$$\lambda_l^{\text{out}} \triangleq f(\lambda) = 2 \tanh^{-1} \left[ \tanh \left( \frac{\lambda}{2} \right) g_l \right], \quad (4.8)$$

where  $g_l$  is the check node's gain (4.6),  $\lambda$  is the check node's input LLR from the trapping set, and  $\lambda_l^{\text{out}}$  is the check node's output LLR on the other edge within the trapping set. Noting that the first two partial derivatives of (4.8) evaluated at  $\lambda = 0$  are

$$\left. \frac{\partial f(\lambda)}{\partial \lambda} \right|_{\lambda=0} = g \quad \text{and} \quad \left. \frac{\partial^2 f(\lambda)}{\partial \lambda^2} \right|_{\lambda=0} = 0,$$

one produces the truncated series approximation  $\lambda_{\text{out}} \approx g\lambda$ . The approximation is good for  $|\lambda| \leq 2 \tanh^{-1}(g)$ . The exact expression for  $\lambda_{\text{out}}$  (4.8) saturates at an output of  $\pm 2 \tanh^{-1}(g)$ , while the approximation is linear in  $\lambda$ . Since we expect the LLRs outside of the trapping set to grow quite fast, we believe this to be an adequate approximation.

We now extend the check node gain model to include inversions through the degree-two check nodes caused by erroneous messages from outside of the trapping set. This occurs primarily during early iterations and is illustrated in Fig. 4.3. When an odd number of the  $d_c - 2$  messages from outside of the trapping set into a degree-two check node within the trapping set are erroneous, then that check node will invert the polarity of the LLR message sent from variable node  $i_1$  to  $i_2$ , as well as the message sent in the opposite direction.

We will present a simplification of the method used by Schlegel and Zhang to account for these inversions [5]. Moreover, they present a small rise in their predicted error floor instead of lowering it as we find with our technique. Schlegel and Zhang injected a stochastic cancellation signal into the state update equations, while we choose to modify the mean check node gains.

During the first iteration, messages from variable nodes may contain inversions due to channel errors. The probability that a specific input to the check node contains an inversion during iteration  $l = 1$  is just the uncoded symbol error rate

$$P_{e,1} = Q \left( \sqrt{\frac{2RE_b}{N_0}} \right). \quad (4.9)$$

Thus, utilizing Assumption 4.3, the probability of a polarity inversion in a specific check node is given at iteration  $l$  by

$$\begin{aligned} P_{\text{inv},l} &= \sum_{k \text{ odd}} \binom{d_c - 2}{k} P_{e,l}^k (1 - P_{e,l})^{d_c - 2 - k} \\ &= \frac{1 - (1 - 2P_{e,l})^{d_c - 2}}{2}, \end{aligned} \quad (4.10)$$

which is the probability of an odd number of errors in the  $d_c - 2$  input messages at iteration  $l$ . The final simplification in (4.10) is from [42, p. 38].

For additional iterations we can use density evolution [43, 44] to predict an effective  $E_b/N_0$  at the output of the variable nodes and reuse equations (4.9) and (4.10). When there is a channel inversion, the output message magnitude will likely be very low, as suggested in [5]. Hence, to model this we will equate random channel inversions to randomly setting the check node gain to zero with probability  $P_{\text{inv},l}$ . We introduce the check node's *modified mean gain* as simply

$$\bar{g}'_l = \bar{g}_l(1 - P_{\text{inv},l}). \quad (4.11)$$

For an LDPC code with regular check node degree  $d_c$ , the scalar gain  $\bar{g}'_l$  of (4.11) may be applied at each iteration  $l$  to model the external influence on the subgraph's degree-two check nodes. For an LDPC code with irregular check degree, we have a few options. If the check node degree spread is small, we may generalize (4.7) and (4.11) by also taking the expectation over the degree distribution of all the degree-two check nodes in the subgraph. Alternatively, we may maintain several versions of gain  $\bar{g}'_l$ , one for each check-degree that appears in the subgraph.

**Assumption 4.4** We assume that all the degree-two check nodes in the subgraph may be modeled by the modified mean gain  $\bar{g}'_l$ .

One may criticize this gain model with inversions as double-counting. First, inversions from outside the trapping set will reduce the check node gain through (4.7). Then, the same inversions reduce the check node gain again in (4.11). Heuristically, one may consider this as just adding weight to the significance of these inversions. We find it justified empirically in Section 4.7 and note that our earlier arguments paralleled those in [5].

### 4.3.2 Linear State-Space Model

**Assumption 4.5** We consider only codes described by  $d_v$ -variable-regular Tanner graphs with  $d_v \geq 3$ . For an examination of irregular codes, see [18].

**Assumption 4.6** In justifying the model it is assumed that the messages within the trapping set under study have little impact on the rest of the Tanner graph. This seems reasonable given that the number of unsatisfied check nodes which form the main interface between the trapping set and the rest of the Tanner graph is small.

The state-space modeling equations are shown below for input column vectors  $\lambda_l^{(\text{ex})}$  and  $\lambda$  and for output column vector  $\tilde{\lambda}_l$ , whose entries are the LLR-domain soft output decisions at iteration  $l$  [5, 18].

$$\mathbf{x}_0 = \mathbf{B}\lambda$$

$$\mathbf{x}_l = \tilde{g}_l' \mathbf{A} \mathbf{x}_{l-1} + \mathbf{B}\lambda + \mathbf{B}_{\text{ex}} \lambda_l^{(\text{ex})} \quad \text{for } l \geq 1 \quad (4.12)$$

$$\tilde{\lambda}_l = \tilde{g}_l' \mathbf{C} \mathbf{x}_{l-1} + \lambda + \mathbf{D}_{\text{ex}} \lambda_l^{(\text{ex})} \quad \text{for } l \geq 1 \quad (4.13)$$

The central part of the state-space model, of course, is the updating of the state  $\mathbf{x}_l$ . The state vector  $\mathbf{x}_l$  represents the vector of LLR messages sent along the edges from the subgraph's variable nodes toward the degree-two check nodes. The state is updated once per full iteration. One may consider that  $\tilde{g}_l' \mathbf{x}_{l-1}$  represents the LLR messages after passing through the degree-two check nodes during the first half-iteration, and  $\tilde{g}_l' \mathbf{A} \mathbf{x}_{l-1}$  represents their contribution to the variable node update during the second half-iteration of iteration  $l$ . Since  $ad_v$  equals the number of edges of the variable-regular Tanner subgraph of  $a$  variable nodes, the number of states in the model of a subgraph is  $m = ad_v - b$ . For an elementary subgraph, these  $m$  edges are incident on the subgraph's degree-two check nodes, forcing  $m$  to be an even integer.

The  $m \times a$  matrix  $\mathbf{B}$  and the  $m \times b$  matrix  $\mathbf{B}_{\text{ex}}$  are used to map  $\lambda$  and  $\lambda_l^{(\text{ex})}$ , respectively, to the appropriate entries of the state vector to match the set of variable node update equations. Since every variable node has exactly one element from  $\lambda$  participating,  $\mathbf{B}$  has a single 1 in every row, and is zero otherwise. The number of 1 entries per row of  $\mathbf{B}_{\text{ex}}$  corresponds to the number of degree-one check nodes adjacent to the corresponding variable node, which may be none. Like all the matrices appearing in the state-space equations, they are  $(0, 1)$ -matrices.

The  $m \times m$  matrix  $\mathbf{A}$  describes the dependence of the state update upon the prior state vector. Each nonzero entry  $[\mathbf{A}]_{ij} = 1$  indicates the  $j$ th edge of the prior iteration contributes to the variable node update computation of the  $i$ th edge. Thus, the matrix  $\mathbf{A}^T$  may be taken as the adjacency matrix associated with a simple digraph of order  $m$  which describes the state-variable update relationships. Given our  $d_v$ -variable-regular codes, the row weight of  $\mathbf{A}$  and the row weight of  $\mathbf{B}_{\text{ex}}$  will sum to  $d_v - 1$  for every row.

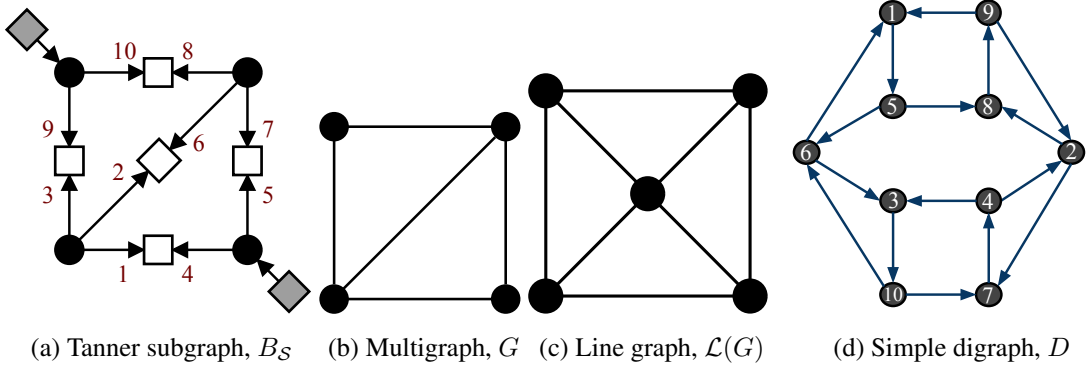


**Figure 4.4:** Several graphical descriptions of the  $(4, 2)$  trapping set with model's states numbered. Arrows have been added to the Tanner subgraph in (a) only to emphasize the direction of state variable flow. The spectral radii of  $G$ ,  $\mathcal{L}(G)$ , and  $D$  are 2.5616, 3.2316, and 1.5214, respectively.

Finally, the  $a \times m$  matrix  $\mathbf{C}$  and the  $a \times b$  matrix  $\mathbf{D}_{\text{ex}}$  are used to map  $\mathbf{x}_{l-1}$  and  $\lambda_l^{(\text{ex})}$  entries, respectively, to the corresponding entry of the soft output decision vector  $\tilde{\lambda}_l$ . The row weight of  $\mathbf{C}$  and the row weight of  $\mathbf{D}_{\text{ex}}$  will sum to the variable node degree  $d_v$  for every row.

If there is a single degree-one check node neighboring every variable node in the subgraph, such as in the  $(8, 8)$  trapping set of 802.3an, then (4.12) degenerates to the case where  $\mathbf{B} = \mathbf{B}_{\text{ex}}$  and  $\mathbf{B}$  has regular column weight as derived in [5]. Furthermore, this degenerate case produces only  $\mathbf{A}$  matrices which have uniform row and column sums with a dominant eigenvalue  $r = d_v - 2$ . Thus, our development will be significantly more general than [5].

Again, note that the check node gains form our approximate model for the behavior of degree-two check nodes within the trapping set. One drawback to any linear system approach is that saturation cannot be introduced to the state variables. Any saturation effects must be applied to the linear system inputs.

**Example 4.2** For the  $(4, 2)$  trapping set with  $d_v = 3$ , we label with integers all the edges into every degree-two check node as depicted in Fig. 4.4a. Now, we can set up the matrices used to make the linear system. We have numbered the edges in Fig. 4.4a to correspond to the order that the edges will appear in the state vector  $\mathbf{x}_l$ . The matrix  $\mathbf{A}$  describes the updating relationship of the state vector in a full iteration of the SPA decoder. Since edge 1 depends only on edges 6 and 9 during one iteration, we fill the first row of  $\mathbf{A}$  to indicate such. All the matrices associated with the trapping set of Fig. 4.4 are given below. Additionally, treating the matrix  $\mathbf{A}^T$  as an adjacency

matrix, it describes the simple digraph shown in Fig. 4.4d.

$$\begin{aligned}
 \mathbf{A} &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} & \mathbf{B} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} & \mathbf{B}_{\text{ex}} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \\
\mathbf{C} &= \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} & \mathbf{D}_{\text{ex}} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}
\end{aligned}$$

If we remove the gains  $\bar{g}_l'$  from the linear state-space equations, we have a linear time-invariant (LTI) system for which we can write a transfer function. Let the unilateral  $Z$ -transform of the state  $\mathbf{x}_l$  be denoted by  $\mathbf{X}(z)$  and the  $Z$ -transform of the input vector  $\boldsymbol{\lambda}_l^{(\text{ex})}$  be denoted by  $\boldsymbol{\Lambda}_{\text{ex}}(z)$ , where  $\mathbf{X}(z) = \sum_{k=0}^{\infty} \mathbf{x}_k z^{-k}$  [45]. Note that the other input vector  $\boldsymbol{\lambda}$  is not a function of iteration number. Finally, letting the soft output vector  $\tilde{\boldsymbol{\lambda}}_l$  of the system be denoted in the  $Z$ -domain by  $\tilde{\mathbf{A}}(z)$ , yields

$$\begin{aligned}
 \tilde{\mathbf{A}}(z) &= \left[ \mathbf{C} (z\mathbf{I} - \mathbf{A})^{-1} \mathbf{B} + \mathbf{I} \right] \frac{\boldsymbol{\lambda}}{1 - z^{-1}} \\
 &\quad + \left[ \mathbf{C} (z\mathbf{I} - \mathbf{A})^{-1} \mathbf{B}_{\text{ex}} + \mathbf{D}_{\text{ex}} \right] \boldsymbol{\Lambda}_{\text{ex}}(z).
 \end{aligned} \tag{4.14}$$

This makes evident the importance of the eigenvalues of  $\mathbf{A}$ . As the eigenvalues are the roots of the characteristic equation  $\det(\mathbf{A} - \mu\mathbf{I}) = 0$ , they correspond directly to the poles of the discrete-time LTI system in (4.14). We find that for the trapping sets addressed herein, the dominant poles are on or outside the unit circle. Hence, the LTI system is marginally stable or (more often) unstable.

We wish to take the recursive state-update equation (4.12) and develop it into an expression without feedback. The first two iterations are easy to express term-by-term, as

$$\begin{aligned} \mathbf{x}_1 &= \bar{g}'_1 \mathbf{A} \mathbf{B} \boldsymbol{\lambda} + \mathbf{B} \boldsymbol{\lambda} + \mathbf{B}_{\text{ex}} \boldsymbol{\lambda}_1^{(\text{ex})} \quad \text{and} \\ \mathbf{x}_2 &= \bar{g}'_1 \bar{g}'_2 \mathbf{A}^2 \mathbf{B} \boldsymbol{\lambda} + \bar{g}'_2 \mathbf{A} \mathbf{B} \boldsymbol{\lambda} + \mathbf{B} \boldsymbol{\lambda} \\ &\quad + \bar{g}'_2 \mathbf{A} \mathbf{B}_{\text{ex}} \boldsymbol{\lambda}_1^{(\text{ex})} + \mathbf{B}_{\text{ex}} \boldsymbol{\lambda}_2^{(\text{ex})}. \end{aligned}$$

Generalizing to an arbitrary iteration  $l > 0$ , we have

$$\mathbf{x}_l = \mathbf{A}^l \mathbf{B} \boldsymbol{\lambda} \prod_{j=1}^l \bar{g}'_j + \sum_{i=1}^l \mathbf{A}^{l-i} \left( \mathbf{B} \boldsymbol{\lambda} + \mathbf{B}_{\text{ex}} \boldsymbol{\lambda}_i^{(\text{ex})} \right) \prod_{j=i+1}^l \bar{g}'_j. \quad (4.15)$$

### 4.3.3 Graphical Assumptions and Interrelationships

We have already proposed two graphical restrictions in Assumptions 4.2 and 4.5. This subsection notes some further assumptions we wish to place on the trapping sets and describes the interrelationships among several useful graphical descriptions of the trapping set: the Tanner subgraph, its corresponding undirected multigraph, the line graph of the multigraph, and the simple digraph that corresponds directly to the state update model. Finally, we form estimates of the dominate eigenvalue for the trapping set.

**Lemma 4.6** *There exists a bijective map between the set of  $d_v$ -variable-regular elementary Tanner subgraphs (up to a relabeling of the check nodes) and the set of multigraphs  $G = (V, E)$  with vertex degrees upper bounded by  $d_v$ , i.e.,  $d(v_i) \leq d_v \forall v_i \in V$ .*

*Proof:* Given the elementary Tanner subgraph  $B_S = (\mathcal{S}, \mathcal{N}(\mathcal{S}), E_S)$ , each variable node  $v \in \mathcal{S}$  becomes a vertex  $v \in V$  of  $G$ , that is  $V = \mathcal{S}$ , and the degree-one check nodes are discarded. The check nodes of degree two become the edges of  $G$ , each joining a pair of vertices. Alternatively, let parity-check submatrix  $\mathbf{H}_S$  describe  $B_S$ ; then  $\mathbf{A}(G) = \mathbf{H}_S^T \mathbf{H}_S - d_v \mathbf{I}$ .

Given the multigraph  $G$ , each vertex  $v \in V$  of  $G$  becomes a variable node  $v \in \mathcal{S}$  of  $B_S$ . So, again  $V = \mathcal{S}$ . Each edge of  $G$  is replaced by a degree-two check node (with an arbitrary label) and two edges in  $B_S$ . Finally, degree-one check nodes are attached with single edges to variable nodes as needed until every variable node has  $d_v$  neighbors. ■

In creating the multigraph  $G$ , we have a more basic description of the potential elementary trapping set. This is illustrated in creating Fig. 4.4b from Fig. 4.4a. Either one of these

graphs is sufficient to characterize the iteration-to-iteration dependencies among the state variables in the model. The multigraph  $G = (V, E)$  corresponding to  $B_S$  has

$$\begin{aligned} \text{order}(G) = |V| &= a \quad \text{and} \\ \text{size}(G) = |E| &= \frac{ad_v - b}{2}. \end{aligned} \tag{4.16}$$

Tanner graphs with cycles of length 4, the smallest permitted in a Tanner graph, will map to parallel edges in  $G$ . Cycles of length  $k$  in  $B_S$  map to cycles of length  $k/2$  in  $G$ . The 4-cycles in the Tanner graph are often avoided by code designers.

**Example 4.3** The multigraph corresponding to the Tanner subgraph of Fig. 4.2d is the cycle graph of order 4,  $C_4$ .

**Assumption 4.7** Tanner subgraphs of interest and their associated multigraphs are connected. Those trapping sets described by disconnected Tanner subgraphs can instead be analyzed component by component. The main reason for this simplification is to reduce the number of cases in which the state update expression contains a reducible  $\mathbf{A}$  matrix.

**Assumption 4.8** We further assume that the variable nodes within the Tanner subgraphs contain from zero to  $d_v - 2$  adjacent degree-one check nodes. Equivalently, we assume that the associated multigraphs are leafless; that is, they do not contain vertices of degree one. For consideration of multigraphs with leaves, see Appendix 4.B.

A leaf in the multigraph maps to a variable node in the Tanner subgraph neighboring one degree-two check node and  $d_v - 1$  degree-one check nodes. Thus, we require here that every variable node in the Tanner subgraph must neighbor at least two degree-two check nodes. The multigraphs allowed by Assumptions 4.7 and 4.8 will be connected, leafless graphs, containing one or more cycles.

The conditions described here, Sun termed “simple trapping sets” in [18]. Our Assumption 4.8 is equivalent to elementary absorbing sets for codes of variable-degree three. However, our Assumption 4.8 is less restrictive in some ways than absorbing sets for  $d_v \geq 4$  as we allow for more degree-one check nodes per variable node.

Finally, we describe how to create the simple digraph  $D = (Z, A)$  from an undirected multigraph  $G = (V, E)$  which meets Assumptions 4.7 and 4.8. This is illustrated in creating Fig. 4.4d from Fig. 4.4b. Each edge of a connected leafless multigraph  $G$  corresponds to two vertices in  $D$ , representing the two directions of LLR message flow in the original Tanner graph.

That is, the number of vertices  $m$  in  $D$  is simply

$$m = \text{order}(D) = 2 \text{ size}(G) = ad_v - b.$$

The digraph  $D$  is a representation of the message updating process with respect to the output of the trapping set's variable nodes in one full iteration of the SPA decoder. Let edge  $e_i = \{v_j, v_k\} \in E$  map to vertices  $z_i, z_{i'} \in Z$ , with  $z_i$  representing the direction of  $e_i$  flowing from  $v_j$  to  $v_k$  and  $z_{i'}$  representing the other direction. No arcs in  $D$  join  $z_i$  to  $z_{i'}$ . Arcs initiating in  $z_i$  are directed to vertices corresponding to other edges in  $G$  flowing out of  $v_k$  and arcs terminating in  $z_i$  are directed from vertices corresponding to other edges in  $G$  flowing into  $v_j$ . Hence,

$$d_i^+ = d(v_k) - 1 = d_{i'}^-, \quad (4.17)$$

$$d_i^- = d(v_j) - 1 = d_{i'}^+, \quad \text{and} \quad (4.18)$$

$$\text{size}(D) = \sum_{z_i \in Z} d_i^- = \sum_{v_j \in V} d(v_j) [d(v_j) - 1].$$

With respect to a Tanner subgraph that meets Assumption 4.8, induced from a code of variable-degree three, the size of the associated digraph simplifies to  $6a - 4b$ . In our  $(4, 2)$  example, the digraph of Fig. 4.4d has order 10 and size 16.

We now describe a second version of the construction of the simple digraph. From the multigraph  $G$ , shown in Fig. 4.4b, we create the line graph  $\mathcal{L}(G)$  shown in Fig. 4.4c as described in Section 4.2.2. Next, we perform a special directed lifting of  $\mathcal{L}(G)$  to form the simple digraph  $D$ , shown in Fig. 4.4d. This lifting by a factor of two replaces each vertex with a pair of vertices and each edge with a pair of arcs, oriented in opposite directions. Thus, if the  $(i, j)$  entry of  $\mathbf{A}(\mathcal{L}(G))$  is  $w$  it is replaced with a  $2 \times 2$  submatrix containing  $w$  ones, where the line graph limits  $w$  to the values 0, 1, and 2. When the  $2 \times 2$  submatrix which replaces the  $(i, j)$  entry of  $\mathbf{A}(\mathcal{L}(G))$  is determined, the selection of the corresponding  $(j, i)$  submatrix follows a set of rules of correspondence, listed in Table 4.1, which ensure that the new pair of arcs are oriented in opposite directions. Thus, the subdiagonal elements can be easily determined from the superdiagonal elements. The exact construction of either one of these is more involved and requires a direction assignment consistency between, for example, Fig. 4.4a and the lifting of Fig. 4.4c.

All directed walks in  $D$  will correspond to walks in  $G$ . All walks in  $G$  that do not backtrack will correspond to directed walks in  $D$ . Backtracking is prohibited here due to the structure of SPA decoder as expressed in the message update rules, which exclude an edge's own incoming LLR from its outgoing LLR computation. For each  $k$ -cycle in  $G$ , there are two corresponding directed cycles of length  $k$  in  $D$ , one for each direction.

**Table 4.1:** Submatrix Correspondence Rules of the Directed Lifting of the Line Graph (for  $i = j$ , only the all-zero matrix is allowed)

| Entry $(i, j)$                                 | Entry $(j, i)$                                 |
|------------------------------------------------|------------------------------------------------|
| $\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$ | $\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$ |
| $\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$ | $\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$ |
| $\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$ | $\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$ |
| $\begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}$ | $\begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}$ |
| $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ | $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ |
| $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ | $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ |

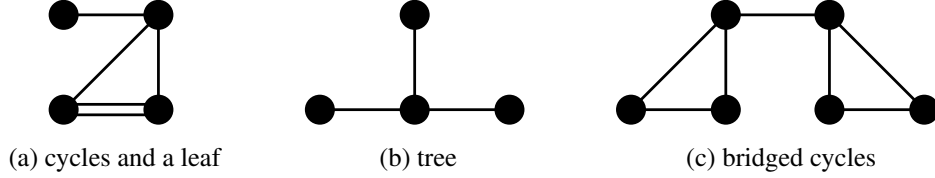


**Figure 4.5:** Three sample diagrams of undirected multigraphs.

The adjacency matrix of  $D$  is  $\mathbf{A}(D)$ , which is an  $m \times m$   $(0, 1)$ -matrix. The matrix  $\mathbf{A}$  used in state space model (4.12) is equal to  $\mathbf{A}^T(D)$  as  $D$  describes the flow of messages within the model. This connection will be convenient when discussing the properties of  $\mathbf{A}$ .

**Example 4.4** Figs. 4.2a, 4.2b, 4.2c, 4.2d, and 4.5c meet Assumptions 4.7 and 4.8. Of these, only Fig. 4.2d has a reducible  $\mathbf{A}(D)$  matrix and it may be analyzed in this chapter as explained in Appendix 4.A. The graphs of Figs. 4.5a and 4.5b fail Assumption 4.8 and have reducible  $\mathbf{A}(D)$  matrices. In the case of Fig. 4.5a, the graph may first be analyzed without the leaf, and then the leaf considered as explained in Appendix 4.B. The case of Fig. 4.5b has all zero eigenvalues as no amount of leaf removal creates a satisfactory structure. Thus, it is not analyzed in this chapter.

We are now ready to bound the spectral radius of the graphs presented in this subsection in order to closely approximate the dominate eigenvalues of  $\mathbf{A}(D)$  and hence  $\mathbf{A}$ . Using (4.4) and (4.16), we may bound the spectral radius of the starting multigraph  $G = (V, E)$  as

$$d_v - b/a \leq \rho(G) \leq \max_{v_i \in V} d(v_i), \quad (4.19)$$

where both equalities hold for connected  $G$  if and only if  $G$  is regular. The maximum value of  $d(v_i)$  is limited by the parent Tanner graph to be no greater than  $d_v$ .

To bound the spectrum of the line graph  $\mathcal{L}(G)$ , we may use the relation

$$\rho(\mathcal{L}(G)) = \rho(\mathbf{A}(G) + \mathbf{D}) - 2, \quad (4.20)$$

where  $\mathbf{D} = \text{diag}(d(v_1), d(v_2), \dots, d(v_n))$ . This follows from the incidence matrix related expressions for each type of graph and the following fact which is easy to show: Given any real matrix  $\mathbf{M}$ , the two products  $\mathbf{M}^T \mathbf{M}$  and  $\mathbf{M} \mathbf{M}^T$  have identical nonzero eigenvalues.

It is now easy to address the case when  $G$  is regular of degree  $k$ . Then,  $\rho(G) = k$  and  $k = d_v - b/a$ , where  $b/a$  must be an integer. Additionally, (4.20) implies that  $\rho(\mathcal{L}(G)) = 2k - 2$  for the line graph which is itself regular [33, p. 36]. Furthermore, we find that the corresponding digraph  $D$  is regular with indegree  $k - 1$  and outdegree  $k - 1$ , implying that its spectral radius must be  $\rho(D) = k - 1 = d_v - 1 - b/a$ . This value corresponds to Schlegel and Zhang's approximation to the spectral radius of any absorbing set [5] which is

$$\rho(D) \approx d_v - 1 - b/a. \quad (4.21)$$

Alternatively, one can view (4.21) as arising from the lower bound of (4.19), which is fairly tight, and the fact that  $\rho(D) = \rho(G) - \Delta$ , where  $\Delta$  is typically 1 or a little greater. This difference between  $\rho(D)$  and  $\rho(G)$  stems from the vertex-degree relationships in (4.17) and (4.18). We empirically examine the accuracy of (4.21) in Appendix 4.C.

We can derive an alternative approximation based on bounding  $\rho(\mathcal{L}(G))$ . From (4.2) and (4.16), we find the order of  $\mathcal{L}(G)$  to be  $(ad_v - b)/2$ . For the following few calculations, we assume that typically no more than one unsatisfied check node is adjacent to any variable node. This is always the case for absorbing sets with  $d_v = 3$  or 4, and likely mostly true in many other cases. Thus, we have that  $G$  contains  $b$  nodes of degree  $d_v - 1$  and  $a - b$  nodes of degree  $d_v$ . This yields the approximation to (4.3) that

$$\text{size}(\mathcal{L}(G)) \approx (d_v - 1)(ad_v - 2b)/2$$

and the approximate bound to the spectral radius of  $\mathcal{L}(G)$

$$\rho(\mathcal{L}(G)) \gtrsim 2 \left( d_v - 1 - \frac{b - b/d_v}{a - b/d_v} \right).$$

Finally, since the directional lifting technique of creating  $\mathbf{A}(D)$  from  $\mathbf{A}(\mathcal{L}(G))$  roughly halves the weight of each row and column, we form the new approximation

$$\rho(D) \approx d_v - 1 - \frac{b - b/d_v}{a - b/d_v}, \quad (4.22)$$

which is also examined in Appendix 4.C. Note that when  $b/a$  equals 0 or 1 the two approximations, (4.21) and (4.22), are equal.

## 4.4 Dominant Eigenvalues and Probability Model

This section will simplify the linear equation used to update the states developed in the prior section to the point that we can easily apply a probability model and predict failure rates for specific trapping sets.

### 4.4.1 Utilizing Dominant Eigenvalues

For analysis of the model, we need to simplify the powers of the matrix  $\mathbf{A}$  that appear in (4.15). This section largely generalizes the development by Schlegel and Zhang [5] with a focus on the dominant eigenvalues of the nonnegative  $m \times m$  matrix  $\mathbf{A}$  and its left eigenvectors. This approach will avoid using the assumption made by Sun, which was that  $\mathbf{A}$  is diagonalizable [18]. In our extensive search of potential trapping sets, shown in Appendix 4.C, we found a small fraction of the trapping sets have  $\mathbf{A}$  matrices which cannot be diagonalized.

Let  $\mu_k \in \mathbb{C}$  be an eigenvalue of matrix  $\mathbf{A}$ , and let  $\mathbf{w}_k$  be the left eigenvector associated with  $\mu_k$ , such that  $\mathbf{w}_k^* \mathbf{A} = \mu_k \mathbf{w}_k^*$ , where  $\mathbf{w}_k^*$  is the conjugate transpose of column vector  $\mathbf{w}_k$ . Then, by induction, for any positive integer  $i$ ,

$$\mathbf{w}_k^* \mathbf{A}^i = \mu_k^i \mathbf{w}_k^*. \quad (4.23)$$

Left-multiplying (4.15) by  $\mathbf{w}_k^*$  and using (4.23) to simplify, we derive the scalar quantity

$$\begin{aligned} \mathbf{w}_k^* \mathbf{x}_l &= \mu_k^l \mathbf{w}_k^* \mathbf{B} \boldsymbol{\lambda} \prod_{j=1}^l \bar{g}'_j \\ &+ \sum_{i=1}^l \mu_k^{l-i} \mathbf{w}_k^* \left( \mathbf{B} \boldsymbol{\lambda} + \mathbf{B}_{\text{ex}} \boldsymbol{\lambda}_i^{(\text{ex})} \right) \prod_{j=i+1}^l \bar{g}'_j. \end{aligned} \quad (4.24)$$

We now shift to using a specific left eigenvector of the nonnegative matrix  $\mathbf{A}$ , the left eigenvector  $\mathbf{w}_1$  associated with the eigenvalue of maximum modulus  $r$ . Dividing the  $m \times m$  nonnegative matrix  $\mathbf{A}$  into the following two cases, the theory of nonnegative matrices allows us to make certain statements regarding  $\mathbf{w}_1$  and  $r$  [28–31]:

1. Let the nonnegative matrix  $\mathbf{A}$  be irreducible. There is a simple eigenvalue  $r$ , such that  $r = \rho(\mathbf{A})$  and  $r > 0$ . The associated left eigenvector  $\mathbf{w}_1$  of  $r$  is positive. There are no other nonnegative left eigenvectors of  $\mathbf{A}$ , except for positive multiples of  $\mathbf{w}_1$ .

2. Let the nonnegative matrix  $\mathbf{A}$  be reducible. Letting  $\mathbf{P}$  be an  $m \times m$  permutation matrix, the matrix  $\mathbf{A}$  may be symmetrically permuted to  $\mathbf{A}' = \mathbf{PAP}^T$ , where  $\mathbf{A}'$  is in block upper triangular form. The block diagonal submatrices  $\mathbf{A}'_i$  are either irreducible square matrices or the 1-by-1 zero matrix. The spectrum of  $\mathbf{A}$  is thus the union of the individual spectra  $\sigma(\mathbf{A}'_i)$ . Appendix 4.A explains that the reducible cases allowed by Assumptions 4.7 and 4.8 have all their eigenvalues located on the unit circle, each with multiplicity two. For these cases, we can associate the positive left eigenvector  $\mathbf{w}_1 = [1, 1, \dots, 1]^T$  with the eigenvalue  $r = 1$ . Of course there exist other nonnegative eigenvectors.

Thus, given our prior assumptions, we may associate a positive left eigenvector  $\mathbf{w}_1$  with the real eigenvalue  $r$ , and this  $r$  is real with magnitude greater than or equal to that of all other eigenvalues.

Generalizing the  $\beta$  that appeared in [5], we define the error indicator  $\beta_l \triangleq \mathbf{w}_1^T \mathbf{x}_l$ . We define  $\beta_l$  this way to create a scalar indicator of trapping set error. Consider  $\beta_l$  as the scaled projection of the state vector  $\mathbf{x}_l$  onto the positive vector  $\mathbf{w}_1$ . Since the state vector  $\mathbf{x}_l$  contains the internal messages of the trapping set, the projection onto a positive vector indicates the trapping set's messages are generally either in the positive (correct) direction or negative (erroneous) direction.

Thus  $\beta_l$  is only approximately indicative of the variable nodes' decisions. Especially in the case where the system is inherently unstable,  $r > 1$ , the state variables will tend to all move toward positive or negative infinity and dominate the soft output expression (4.13). The domination of (4.13) by  $\mathbf{x}_{l-1}$  is obvious in the case that the  $\lambda_l^{(\text{ex})}$  entries are saturated to finite values. Even when this is not the case, the  $\mathbf{x}_{l-1}$  terms still dominate the soft output expression (4.13) as the state update equation (4.12) is much like (4.13), but (4.13) puts even more weight on the states.

**Example 4.5** We continue the on-going example. The nonnegative matrices  $\mathbf{A}$  describing the state update operations of Figs. 4.2a and 4.2b are both primitive, with  $r = 1.6956$  and  $1.5214$ , respectively. The nonnegative matrix  $\mathbf{A}$  describing Fig. 4.2c is imprimitive with  $h = 4$  and  $r = \sqrt{2}$ . This means the four eigenvalues on the spectral circle are  $\pm\sqrt{2}$  and  $\pm\sqrt{2}i$ . One may find that  $h = 4$  by visual inspection of Fig. 4.2c, noting that every cycle length (measured in full iterations) is divisible by four. The matrix  $\mathbf{A}$  describing the state update of Fig. 4.2d is reducible with  $r = 1$ .

Our error indicator expression simplifies if we rescale  $\beta_l$  by a positive constant to  $\beta'_l \triangleq$

$\beta_l / \left( r^l \prod_{j=1}^l \bar{g}'_j \right)$ , so

$$\beta'_l = \mathbf{w}_1^T \mathbf{B} \boldsymbol{\lambda} + \sum_{i=1}^l \frac{\mathbf{w}_1^T (\mathbf{B} \boldsymbol{\lambda} + \mathbf{B}_{\text{ex}} \boldsymbol{\lambda}_i^{(\text{ex})})}{r^i \prod_{j=1}^i \bar{g}'_j}. \quad (4.25)$$

This expression is similar to, but more general than (1) in [5], which was derived for a specific degenerate trapping set as previously described.

#### 4.4.2 Probability of Error Model

The expression for  $\beta'_l$  in (4.25) is a linear combination of stochastic vectors  $\boldsymbol{\lambda}$  and  $\boldsymbol{\lambda}_i^{(\text{ex})}$ . Since elements of  $\boldsymbol{\lambda}$  are i.i.d. Gaussian, linear operations on  $\boldsymbol{\lambda}$  will produce a Gaussian distribution. Several authors [5, 43] have used the approximation that the check node output LLRs, such as  $\boldsymbol{\lambda}_i^{(\text{ex})}$ , are Gaussian, too. The central limit theorem implies that the distribution of a linear combination of several independent check node output messages with the elements of  $\boldsymbol{\lambda}$  will be nearly Gaussian, even if the check node output LLRs are only approximately Gaussian.

**Assumption 4.9** We assume that  $\beta'_l$  has a Gaussian distribution.

Under this assumption, the probability of the failure event,  $\xi(\mathcal{S})$ , corresponding to trapping set  $\mathcal{S}$ , at iteration  $l$ , is then simply

$$\Pr \{ \xi(\mathcal{S}) \} = \Pr \{ \beta'_l < 0 \} = Q \left( \frac{\mathbb{E}[\beta'_l]}{\sqrt{\text{VAR}[\beta'_l]}} \right). \quad (4.26)$$

If  $\{\mathcal{S}_i\}$  enumerates all potential trapping sets, the union bound provides an upper bound on the error floor of the block error rate

$$P_f \lesssim \sum_i \Pr \{ \xi(\mathcal{S}_i) \}. \quad (4.27)$$

The block error rate of (4.27) is also commonly called the frame error rate (FER) or codeword error rate. Next, we wish to express the error floor as an information bit error rate (BER). Letting  $\hat{a}_i$  represent the maximum number of information bits associated with trapping set failure  $\mathcal{S}_i$  for the specific encoding technique used, we may express the BER union bound as

$$P_b \leq \sum_i \frac{\hat{a}_i}{k} \Pr \{ \xi(\mathcal{S}_i) \}, \quad (4.28)$$

where  $k$  is the number of information bits per codeword. When the codeword is encoded *systematically* the codeword bit locations are partitioned between information bits and parity bits. For

a systematic encoding, in which the bit error positions are spread uniformly with no preference to information and parity locations, we may state  $\mathbb{E}[\hat{a}_i] = a_i k/n$ , where  $a_i$  is the number of variable nodes in trapping set  $\mathcal{S}_i$  and  $n$  is the block length of the codeword. In this case<sup>2</sup>, (4.28) simplifies to

$$P_b \leq \sum_i \frac{a_i}{n} \Pr\{\xi(\mathcal{S}_i)\}. \quad (4.29)$$

#### 4.4.3 Codewords

In the case of elementary trapping sets that are also codewords,  $b = 0$  and  $\mathbf{B}_{\text{ex}}$  is not used since there are no unsatisfied check nodes. We find that the failure probability (4.26) of the codeword simplifies to

$$\Pr\{\xi(\mathcal{S}_i)\} = Q\left(\sqrt{\frac{2RE_b}{N_0}} \frac{\sum_{k=1}^a (\mathbf{w}_1^T \mathbf{B})_k}{\sqrt{\sum_{k=1}^a (\mathbf{w}_1^T \mathbf{B})_k^2}}\right). \quad (4.30)$$

This reduces further as the eigensystem of  $\mathbf{A}$  for codewords is rather simple. Every row sum and column sum of the matrix  $\mathbf{A}$  is  $d_v - 1$ , so the spectral radius is  $r = d_v - 1$ . The positive left eigenvector associated with  $r$  is proportional to the all-one vector,  $\mathbf{w}_1 \propto [1, 1, \dots, 1]^T$ . Also,  $\mathbf{B}$  has uniform row weight one and column weight  $d_v$ . Therefore,  $(\mathbf{w}_1^T \mathbf{B})_k = d_v \forall k \in \{1, \dots, a\}$ . Upon recognizing that  $a$  is just the Hamming weight  $w_H$  in the context of codewords, (4.30) simplifies to the more recognizable (see §2.4)

$$\Pr\{\xi(\mathcal{S}_i)\} = Q\left(\sqrt{\frac{2RE_b}{N_0}} w_H\right),$$

independent of iteration count  $l$ .

#### 4.4.4 Non-codewords

To further understand the general behavior of (4.26), we examine the numerator and denominator terms. Letting  $m_{\lambda(\text{ex})}^{(l)}$  be the expected value of each entry of  $\lambda_l^{(\text{ex})}$ , we can write the numerator of the Q-function argument as

$$\begin{aligned} \mathbb{E}[\beta'_l] &= m_\lambda \left(1 + \sum_{i=1}^l \frac{1}{r^i \prod_{j=1}^i \bar{g}'_j}\right) \sum_{k=1}^a (\mathbf{w}_1^T \mathbf{B})_k \\ &\quad + \sum_{i=1}^l \frac{m_{\lambda(\text{ex})}^{(i)}}{r^i \prod_{j=1}^i \bar{g}'_j} \sum_{k=1}^b (\mathbf{w}_1^T \mathbf{B}_{\text{ex}})_k. \end{aligned} \quad (4.31)$$

---

<sup>2</sup>We offer (4.29) as a correction to [5] which uses  $k$  in the denominator

**Assumption 4.10** We will assume that the entries of  $\lambda$  and  $\lambda_l^{(\text{ex})}$  are statistically independent of each other at a given iteration  $l$  as is often done in density evolution (DE) studies. Further, we assume that the entries of  $\lambda_l^{(\text{ex})}$  are independent from iteration to iteration, as was implicitly assumed in [5].

Letting  $\sigma_l^2$  be the variance of each entry of the vector  $\lambda_l^{(\text{ex})}$ , we can write the squared-denominator of the Q-function argument as

$$\begin{aligned} \text{VAR}[\beta'_l] = & 2m_\lambda \left( 1 + \sum_{i=1}^l \frac{1}{r^i \prod_{j=1}^i \bar{g}'_j} \right)^2 \sum_{k=1}^a (\mathbf{w}_1^T \mathbf{B})_k^2 \\ & + \sum_{i=1}^l \frac{\sigma_i^2}{(r^i \prod_{j=1}^i \bar{g}'_j)^2} \sum_{k=1}^b (\mathbf{w}_1^T \mathbf{B}_{\text{ex}})_k^2. \end{aligned} \quad (4.32)$$

We are interested in the behavior of (4.31) and (4.32) as the number of iterations goes toward infinity. The divergence of the first series within (4.31) is not of interest as its effect will be canceled by the first series within (4.32) when they are combined into (4.26). We will apply the Ratio Test to the second series within (4.31) to evaluate its convergence. In general, the Ratio Test is applied to the terms of a series  $a_1, \dots, a_i, a_{i+1}, \dots$  by evaluating the ratio  $\rho = \lim_{i \rightarrow \infty} |a_{i+1}/a_i|$ . Then, the series  $\sum_{i=1}^\infty a_i$  converges absolutely if  $\rho < 1$  and diverges if  $\rho > 1$ . Now, applied to the second series within (4.31),

$$\rho = \lim_{i \rightarrow \infty} \left| \frac{m_{\lambda(\text{ex})}^{(i+1)} r^i \prod_{j=1}^i \bar{g}'_j}{m_{\lambda(\text{ex})}^{(i)} r^{i+1} \prod_{j=1}^{i+1} \bar{g}'_j} \right| = \lim_{i \rightarrow \infty} \left| \frac{m_{\lambda(\text{ex})}^{(i+1)}}{m_{\lambda(\text{ex})}^{(i)} r} \right|. \quad (4.33)$$

The right-most expression follows by noting that the gains  $\bar{g}'_l$  approach 1 rapidly with increasing iteration count  $l$ . Thus, if the mean unsatisfied-check LLR  $m_{\lambda(\text{ex})}^{(i)}$  dominates  $r^i$  asymptotically (*i.e.*,  $m_{\lambda(\text{ex})}^{(i)} > Cr^i$  for every positive constant  $C$  and sufficiently large  $i$ ), then  $\rho > 1$  and (4.31) will grow without bound.

If we can be assured that (4.32) does not grow as fast as the square of (4.31), then the entire argument to the Q-function will grow without bound, driving the failure rate of the potential elementary trapping set toward zero. In that case, sufficient iterations and headroom for  $\lambda_l^{(\text{ex})}$  growth are the requirements for the model to achieve as low an error floor as desired.

Other models of error floor behavior [5, 18] have used the Gaussian approximation to LLR densities [43], which implies  $\sigma_l^2 = 2m_{\lambda(\text{ex})}^{(l)}$ . Moreover, using a numerical version of DE, Fu found that  $\sigma_l^2 < 2m_{\lambda(\text{ex})}^{(l)}$  as the LLRs get large [46]. With such an LLR variance, *i.e.*,  $\sigma_l^2 \leq cm_{\lambda(\text{ex})}^{(l)}$  for some positive  $c$ , we find that the entire argument to the Q-function grows without bound in the cases that satisfy  $m_{\lambda(\text{ex})}^{(i)} > Cr^i$ . We will find in the following sections that

this latter condition is true within the assumptions identified. However, if the variance grows as the square of the mean, then the argument to the Q-function reaches a finite limit and a nonzero error floor is produced. This is a very important issue that we will revisit later in the chapter.

#### 4.4.5 Bounds on Spectral Radius of the Matrix $\mathbf{A}$

We now need some bounds on  $r$ , the spectral radius of the  $\mathbf{A}$  matrix, in order to compare it with the growth rate of  $m_{\lambda(\text{ex})}^{(i)}$  which we will develop later. The classic bounds of Frobenius, from Lemma 4.2, are sufficient for our needs—merely note the equivalence between the out-degrees of a digraph and the row-sums of the associated adjacency matrix.

**Theorem 4.7** *Consider a variable-regular LDPC code with variable-degree  $d_v \geq 3$ . Let the trapping set  $\mathcal{S}$  induce an elementary connected subgraph  $B_{\mathcal{S}}$  with  $a \geq 2$  variable nodes and  $b > 0$  degree-one check nodes. Further, let the associated undirected multigraph  $G$  be leafless. Then, the adjacency matrix  $\mathbf{A}(D)$  of the associated simple digraph  $D$  must have spectral radius  $r$  such that  $1 \leq r < d_v - 1$ .*

*Proof:* We have ruled out codewords by requiring  $b > 0$ . The codewords of a variable-regular graph will have an  $\mathbf{A}$  matrix with uniform row sums equal to  $d_v - 1$ , as each variable node in  $B_{\mathcal{S}}$  is adjacent to  $d_v$  degree-two check nodes and the edge being updated is always excluded from computation in the SPA decoder. By assumption, there will be some  $b > 0$  degree-one check nodes. If we let  $b_{\max}$  be the greatest number of degree-one check nodes neighboring any single variable node ( $1 \leq b_{\max} \leq d_v - 2$ ), and  $b_{\min}$  be the least such number ( $0 \leq b_{\min} \leq b_{\max}$ ), then using Lemma 4.2, we can state

$$d_v - 1 - b_{\max} \leq r \leq d_v - 1 - b_{\min}. \quad (4.34)$$

If  $\mathbf{A}$  is irreducible, then either no equalities above hold or both equalities hold, the latter when  $b_{\min} = b_{\max}$ . If  $\mathbf{A}$  is irreducible and  $b_{\min} = b_{\max}$ , then  $1 \leq r \leq d_v - 2$ . If  $\mathbf{A}$  is irreducible and  $b_{\min} < b_{\max}$ , then  $1 < r < d_v - 1$ . In the case that  $\mathbf{A}$  is reducible, Appendix 4.A shows that the graphs permitted by the connected and leafless conditions only allow  $r = 1$ , which will be strictly less than  $d_v - 1$  since we have assumed  $d_v \geq 3$ . ■

**Corollary 4.8** *If in addition to the conditions given in Theorem 4.7, we also stipulate that  $b_{\min} < d_v - 2$ , then the spectral radius  $r$  is bounded by  $1 < r < d_v - 1$ .*

*Proof:* The case where  $\mathbf{A}$  is irreducible and  $b_{\min} = b_{\max}$ , now yields  $1 < r \leq d_v - 2$ . The case where  $\mathbf{A}$  is irreducible and  $b_{\min} < b_{\max}$  remains unchanged with  $1 < r < d_v - 1$ . Finally, the case where  $\mathbf{A}$  is reducible is not permitted, since Appendix 4.A shows that all the reducible matrices that are associated with connected and leafless subgraphs have  $d_v - 2$  degree-one check nodes neighboring every variable node. ■

## 4.5 Analysis Using DE without Saturation

This section analyzes LLR-domain SPA decoding without saturation to make ultimate error floor predictions. First, we must apply DE to model the contributions from nodes outside of the trapping set relevant to the behavior of our trapping set. Then we will refer back to the divergence condition of (4.33) to see if error floors are produced.

**Assumption 4.11** Density evolution (DE) assumes that the Tanner graph has no cycles and the code's block length is infinite. For our purposes, this is equivalent to the assumed independence among incoming LLRs of Assumption 4.10.

SPA decoding of LDPC codes exhibits a threshold phenomenon as the block length tends to infinity. The error rate drops very dramatically as the SNR exceeds the *decoding threshold* which can be found using DE [43]. As we are interested in the behavior of the error floor in this work, we assume that the channel SNR is always above the decoding threshold.

The application of this DE technique produces an analytical expression for the LLRs of the SPA decoder. As DE progresses above the decoding threshold, LLR values grow fast, but the question is how fast. In [18], Sun showed that mean check node outputs,  $m_{\lambda(\text{ex})}$ , grow from iteration  $l - 1$  to  $l$  as

$$m_{\lambda(\text{ex})}^{(l)} = (d_v - 1)m_{\lambda(\text{ex})}^{(l-1)} + \text{"some small value terms."} \quad (4.35)$$

In Appendix 4.D, we bound the effect of Sun's "small value terms," by developing tighter asymptotic expressions. In fact, we are able to develop bounds on the SNR region in which we can expect the growth rate of the mean extrinsic LLRs  $m_{\lambda(\text{ex})}^{(l)}$  to exceed the internal trapping set LLR growth rate. In any case, DE analysis in the high  $E_b/N_0$  regime leads to  $m_{\lambda(\text{ex})}^{(l)} > Cr^l$  for every positive constant  $C$ , any  $r$ , such that  $1 \leq r < d_v - 1$ , and sufficiently large  $l$ .

When combined with (4.33) developed in Section 4.4, the growth rates of this section show that every potential (non-codeword) elementary trapping-set error can be corrected by a

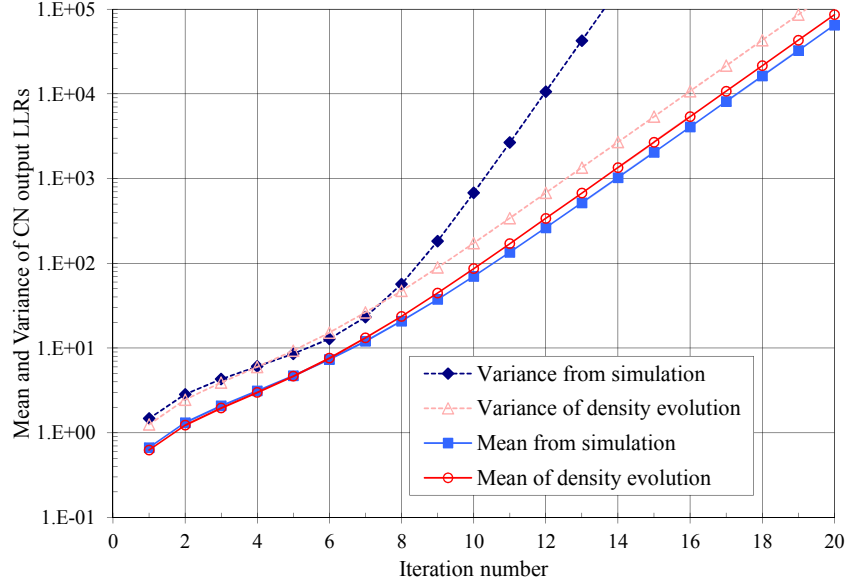


**Figure 4.6:** Check node output LLR mean and variance versus iteration number for the Margulis code at  $E_b/N_0$  of 2.8 dB.

non-saturating LLR-domain SPA decoder after a sufficient number of iterations, provided we can rely upon DE as a model for LLR growth outside of the trapping set for variable-regular ( $d_v \geq 3$ ) LDPC codes.

In Fig. 4.6 we plot the mean check node output LLRs from the SPA decoder simulation of the (2640, 1320) Margulis code, which is (3, 6)-regular, and from DE of a (3, 6)-regular code. In this simulation the all-zero codeword is transmitted over the AWGN channel and early termination of the LLR-domain decoder is disabled. We note that the mean LLR follows very closely the mean LLR predicted by DE. The variance of the check-node output LLRs, however, shows a problem. For the first seven iterations, the variance is approximately twice the mean, as predicted by the Gaussian approximation to DE. However, by the ninth iteration, the variance has clearly taken on the trend of the square of the mean. As discussed in Section 4.4, this trend can generate an error floor as the mean to standard deviation ratio of  $\beta'_l$  in (4.26) reaches a fixed value rather than growing as iterations get large, *i.e.*,

$$\frac{\mathbb{E}[\beta'_l]}{\sqrt{\text{VAR}[\beta'_l]}} = O(1). \quad (4.36)$$

The cause of the variance's departure from the behavior predicted by DE is the positively correlated nature of the LLRs after several iterations. In Fig. 4.7 we plot the average of the off-

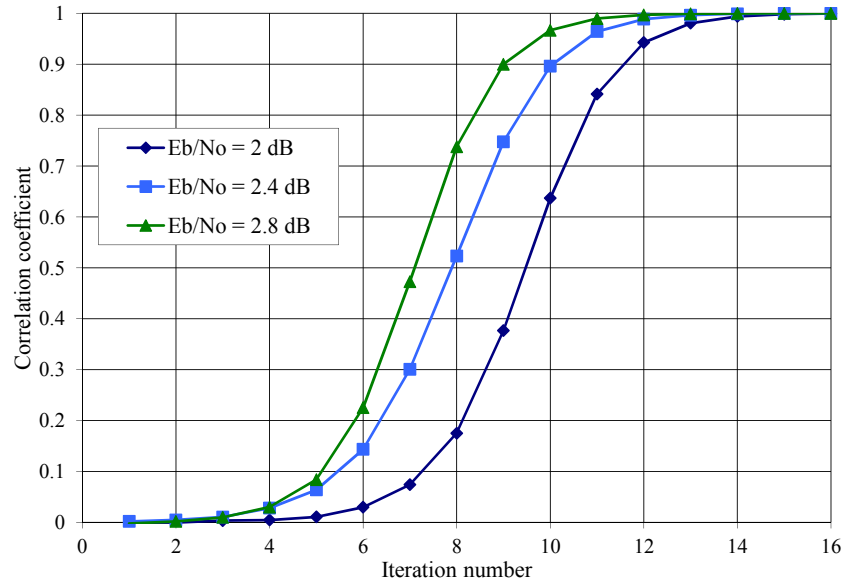


**Figure 4.7:** Correlation coefficient among the four LLR check node output metrics used as linear model inputs versus iteration number for  $(12, 4)$  trapping sets of the Margulis code as measured in LLR-domain SPA simulation.

diagonal values from the  $4 \times 4$  matrix of correlation coefficients of the four LLR check node output metrics used as linear model inputs versus the iteration number for the  $(12, 4)$  trapping sets of the Margulis code. This shows significant positive correlation appearing in the sixth iteration and progressing rapidly. Thus, the assumptions used to justify (4.32) do not hold in this case. The  $(640, 192)$  QC code, which is studied later, also shows similar variance and correlation behavior just with an earlier onset, due to its smaller girth (6 iterations vs. 8).

Also exposing flaws in our probability of error model is the skewness of  $\beta'_l$ . We have measured its skewness in the range of 0.6 to 1.8 as the correlated LLRs propagate through an SPA simulation of the Margulis code. Skewness is a measure of the asymmetry of the probability distribution of a random variable. Of course, (4.26) assumed a Gaussian distribution, which is symmetric about the mean and has a skewness of zero. We consider such large skews a significant complication for this case.

The problems presented here make it impossible to accurately use the probability model that we presented beyond the seventh iteration of decoding the Margulis code without saturation. However, in the following sections we are able to find applications of this model and use other techniques to estimate the error floor for non-saturated decoding of the Margulis code.

## 4.6 Modifying and Applying Richardson's Semi-Analytical Technique without Saturation

In this section, we introduce Richardson's semi-analytical technique to estimate the error floor by simulating in the vicinity of trapping sets [3]. Then, we describe potential improvements to the technique for it to be effective in the case of the non-saturating decoder, and present results for the error floor of the Margulis code.

Consider a specific, known TS  $\mathcal{T}$  containing  $a$  variable nodes. We will generate the noise for the variables outside the TS as we normally would for standard Monte-Carlo simulation, but we will treat the noise within the TS in a special way. Consider that the noise subvector  $\mathbf{n} = (n_1, \dots, n_a)$  containing i.i.d. zero-mean Gaussian random samples may be generated in various domains. If the elements of  $\mathbf{u}$  are i.i.d. Gaussian then so are the elements of  $\mathbf{n} = \mathbf{Q}\mathbf{u}$ , if  $\mathbf{Q}$  is orthogonal. Multiplying by an orthogonal matrix  $\mathbf{Q}$  preserves the i.i.d. nature as  $\mathbf{Q}^T \mathbf{Q} = \mathbf{I}$  and preserves the Gaussian distribution as sums of Gaussians are also Gaussian-distributed.

Richardson's technique biases the noise in the direction of the trapping set (TS), much like importance sampling. (Alternatively, one may employ importance sampling techniques to estimate the error rates of particular trapping sets, as in [26, 39, 47].) Richardson effectively introduces a special orthogonal matrix  $\mathbf{Q}$  with a column that has equal support in the  $a$  positions of the known TS. The remainder of the orthogonal matrix may be filled using Gram-Schmidt. Consider the first column of  $\mathbf{Q}$  to be  $[1/\sqrt{a}, \dots, 1/\sqrt{a}]^T$  and the first element of  $\mathbf{u}$  to be  $s\sqrt{a}$ , where  $s$  is  $N(0, \sigma^2/a)$ . This allows us to treat the Gaussian random variable  $s$  in the direction of the noise separately, while we perform Monte-Carlo simulation on the other  $a - 1$  dimensions of the noise subvectors with variance  $\sigma^2$ .

Now, we fix  $s$  at several negative values and run simulations, finding the conditional TS failure probability  $\Pr\{\xi_{\mathcal{T}}|s\}$  for each  $s$  value. Then, the overall TS failure probability  $\Pr\{\xi_{\mathcal{T}}\}$  may be computed using total probability, as

$$\Pr\{\xi_{\mathcal{T}}\} = \int \Pr\{\xi_{\mathcal{T}}|s\} p_S(s) ds. \quad (4.37)$$

Due to the high degree of overlap among TSs, when counting failures for  $\Pr\{\xi_{\mathcal{T}}|s\}$ , Richardson only counts a frame as a failure if the set of symbols which are not eventually correct exactly matches the symbol locations of  $\mathcal{T}$ , the specific TS under test.

The dominant TSs of the (2640, 1320) Margulis LDPC code are the (12, 4) and (14, 4) elementary TSs, shown in Figs. 4.8a and 4.8b [2, 3, 12]. In the case of the (14, 4) TS of the Margulis code, the not-eventually-correct symbols exactly match the symbols in  $\mathcal{T}$ , when testing

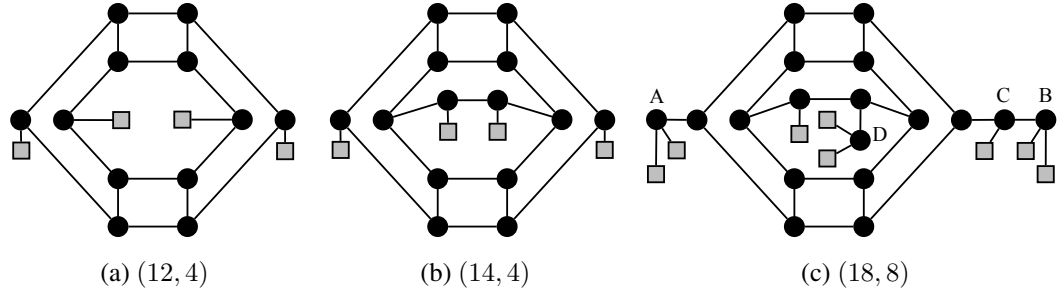


**Figure 4.8:** Three elementary TSs of a  $d_v = 3$  code shown with degree-one check nodes shaded and with degree-two check nodes omitted. The two on the left are absorbing sets and dominate the error floor of the saturated SPA decoder for the Margulis code, while the one on the right fails is not an absorbing set.

the saturated decoder at a rate of about 3 in 4 failures or greater (at  $s = -1.35$ ). Note that this rate drops substantially as  $s$  approaches  $-1$ . However, for a non-saturating decoder, this ratio became less than 1 in  $10^5$  (at  $s = -1.35$ ) in our tests, indicating that the not-eventually-correct condition is not effective when saturation is eliminated. This ratio was so low when running Richardson’s technique for  $\mathcal{T}$  set to a  $(12, 4)$  TS without saturation, that we could not measure it. When saturation is present, failures are generally forced to occur on absorbing sets, and this is often the set under test when  $s \leq -1.25$ . Without saturation present, the failures occur on a wide variety of Tanner subgraphs. Most of the failing subgraphs during non-saturating simulation are not absorbing sets and most are larger than the original  $a$  variables under test. In fact, the two smallest failing structures that we captured at high SNR were the  $(14, 4)$  absorbing set of Fig. 4.8b and the  $(18, 8)$  subgraph shown in Fig. 4.8c. Since the set of general Tanner subgraphs within the Margulis code is vast [8], rerunning the error floor estimate for each type of subgraph with such a low capture rate would be impractical. We propose two solutions to this.

The first solution would be to ignore the not-eventually-correct criteria and simply count all failures at the risk of overestimating the error floor. Instead, we propose to estimate the error floor by introducing saturation in the final iterations of failed frames so that they may fail on absorbing sets. For example, iterations with LLR saturation would force corrections to the labeled variable nodes of Fig. 4.8c, as the unsatisfied check nodes outnumber the satisfied checks, driving the failed structure to the  $(14, 4)$  absorbing set contained within. Since this approach reduces the number of incorrect bits per failure, it perhaps has useful applications. It appears effective, as we see that most failures are reduced to small absorbing sets in practice.

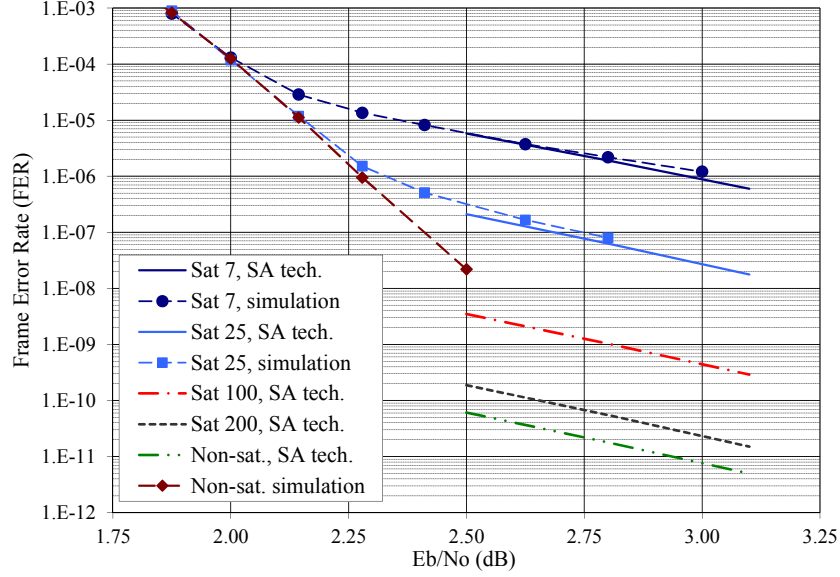


**Figure 4.9:** FER vs.  $E_b/N_0$  in dB for Margulis LDPC Code in AWGN. The semi-analytical (SA) technique only estimates errors due to the (12,4) and (14,4) trapping sets at  $E_b/N_0 = 2.8$  dB.

We prefer this strategy because we are interested in establishing a lower bound on the floor for the non-saturating case.

As there may be many variable nodes to correct to get to the absorbing set contained within the failed structure we allow for 20 iterations of saturated SPA decoding before beginning the “eventually correct” logic which runs for 12 additional iterations. We chose to run the saturation phase at an LLR limit of 25. Our results are shown in Fig. 4.9 where all simulations were run for a maximum of 200 iterations, except that the non-saturating semi-analytical simulations were run for 32 additional iterations as just described. The semi-analytical technique was run at  $E_b/N_0 = 2.8$  dB in all cases, and was extrapolated locally as suggested in [3]. Richardson’s extrapolation assumes that the conditional failure probability  $\Pr\{\xi_{\mathcal{T}}|s\}$  is insensitive to SNR changes. The extrapolations appear to produce parallel lines in the log-log plot (*i.e.*, the logarithm of the error rate versus the  $E_b/N_0$  measured in decibels, which is also logarithmic) due to the character of the integration in (4.37). The dominating part of the integrand is about 5 standard deviations of  $s$  below the zero-mean point in all the conditions studied. Thus, small changes to the standard deviation of  $s$  in (4.37) produce a multiplicative effect on the estimated error rate regardless of the specific decoder configuration.

Fig. 4.9 shows an estimated error floor at  $E_b/N_0 = 2.8$  dB of  $2 \cdot 10^{-11}$  FER due to

these two TSs, which is incrementally better than the  $6 \cdot 10^{-11}$  achieved by limiting the LLR magnitudes to 200 at the check node output. In making this error floor prediction we assume that no other TSs become dominant in these conditions.

Several additional observations are worth noting. Increasing the maximum number of iterations of the non-saturating simulation to 1000 reduces the error floor by about a factor of 3. Additionally, we have applied the semi-analytical techniques of this section to estimate the combined FER contributions of the (12, 4) and (14, 4) TSs to be  $4 \cdot 10^{-17}$  FER at  $E_b/N_0 = 4$  dB. This shows that the error floor falls significantly faster than Richardson's extrapolation predicts, since the conditional failure probability  $\Pr\{\xi_{\mathcal{T}}|s\}$  appears to decrease significantly with increasing SNR. As shown in Appendix 4.D, at higher SNR, the beneficial LLR growth starts earlier and grows faster giving the code beyond the TS a higher probability to correct the channel error along a TS for a non-saturating decoder. However, an SPA decoder with limited LLR dynamic range will not leverage these effects that depend on SNR.

Table 4.2 shows the ratio of the error floor contributions of the (12, 4) TS to the contributions of the (14, 4) TS at  $E_b/N_0 = 2.8$  dB. We note that when saturation limits are low the (12, 4) TS dominates the error floor, but when the limits are raised (14, 4) dominates. There are two potential reasons for this. The first is that the larger maximum eigenvalue value of (14, 4) is more sensitive to saturation even though it is less likely to initially trigger due to the greater number of variable nodes involved. Secondly, an effect we have noticed is that a growing fraction of (12, 4) TS failures are subsumed by the (14, 4) absorbing sets which contain them as LLR limits are raised. Additionally, we notice that the (14, 4) TSs dominate even more as the SNR is increased for a non-saturating decoder.

Finally, we would like to know if we are seeing a “floor” or not for the non-saturating SPA simulation results. Extrapolating the FER of the waterfall region of Fig. 4.9 at a constant log-log slope out to  $E_b/N_0 = 4.0$  dB yields an FER of about  $5 \cdot 10^{-20}$ . Thus, the FER contribution of the (12, 4) and (14, 4) TSs at  $4 \cdot 10^{-17}$  FER overpowers the FER of the extrapolated waterfall region at an  $E_b/N_0$  of 4.0 dB. Therefore, early evidence predicts that we will indeed see a floor, but this floor is much lower and has about twice the log-log slope versus SNR than what was previously referred to as a “floor” for the Margulis code. Thus, our floor estimates are about 5 orders of magnitude lower at  $E_b/N_0 = 2.8$  dB than prior SPA simulation results, and about 8 orders of magnitude lower at 4.0 dB.

**Table 4.2:** Ratio of  $(12, 4)$  trapping set error floor contribution to  $(14, 4)$  for the Margulis code at  $E_b/N_0 = 2.8$  dB using Richardson’s semi-analytical technique.

| Decoder                            | Ratio    |
|------------------------------------|----------|
| Saturated SPA, at $\pm 7$ CN out   | 4.6 : 1  |
| Richardson’s 5-bit [3]             | 3.3 : 1  |
| Saturated SPA, at $\pm 15$ CN out  | 3.3 : 1  |
| Saturated SPA, at $\pm 25$ CN out  | 1.7 : 1  |
| Saturated SPA, at $\pm 50$ CN out  | 0.7 : 1  |
| Saturated SPA, at $\pm 100$ CN out | 0.6 : 1  |
| Saturated SPA, at $\pm 200$ CN out | 0.46 : 1 |
| Non-saturating SPA decoder         | 0.4 : 1  |

## 4.7 Numerical Results with Saturation

This section readdresses the prediction model, this time for the case of saturated decoding. We present the techniques used to produce the linear system’s inputs, and then compare predicted error floors to simulation results for four LDPC codes. Two numerical models are described for use as system inputs for the mean and variance of the unsatisfied, degree-one check nodes within the trapping set (TS). These models also generate the mean gain  $\bar{g}_l$  of the degree-two check nodes required by the model presented in Section 4.3. This completes the necessary tools to produce error floor predictions when the LLR metrics are saturated within the LLR-domain SPA decoder.

The first technique is the numerical version of density evolution (DE) known as discretized density evolution (DDE) described in [44]. This technique utilizes a probability mass function (pmf) that closely approximates the continuous probability density function (pdf) of node input messages and output messages. This technique is selected for the saturated case as it allows us to precisely model the behavior of LLRs as they saturate since it works directly on the pmfs. At a particular  $E_b/N_0$  value, we capture the mean and variance of the values we require at each iteration for insertion into the model. See Table 4.3 for an example, which also lists the mean check node gain, which have been computed following (4.7).

We present our technique to contrast it with that presented in [5]. Schlegel and Zhang

suggest recording only the mean LLRs from a (continuous) DE analysis and computing the variance based on the consistent Gaussian distribution assumption [43]. The consistent Gaussian approximation<sup>3</sup> is that  $\sigma_l^2 = 2 m_{\lambda(\text{ex})}^{(l)}$ . In this section, we have purposely avoided the consistency assumption since the effect of LLR saturation will void it as seen in Table 4.3.

The second technique is to simply run the LLR-domain SPA decoder with early termination turned off and collect the required statistics. Configuring the AWGN channel for a particular  $E_b/N_0$  value, we capture the mean and variance of the check nodes' output LLRs and the mean check node gain computed as in (4.7) at each iteration. For an example of the results using this technique see Table 4.4. We find 100 frames to be sufficient, and of course very quick to produce.

The results in the two tables are similar, as expected. Starting at iteration six or seven, the LLR variance from the SPA simulation is significantly larger than from the DDE technique, due to cycles present in the simulation of the finite length code. This is similar to Fig. 4.6, but under different saturation conditions. We would expect that a code with an even a larger girth would have an even later onset of variance divergence. Interestingly, it was found the two techniques produced nearly identical error floor predictions in spite of the differences. We chose to present the results obtained using the SPA simulation technique in the figures that follow.

The first code we examine is the (640, 192) quasi-cyclic (QC) LDPC code from [12], with  $d_v = 5$  and irregular check degree. The dominant TS for a saturating LLR-domain SPA decoder is the (5, 5) elementary TS shown in [12]. The  $\mathbf{A}$  matrix for this TS is primitive and has maximum eigenvalue  $r = 3$ . The multiplicity of this TS in the code is just 64 and all such sets have disjoint sets of variable nodes. Figs. 4.10, 4.11 and 4.12 show FER results for simulation and error floor prediction for this code at two levels of LLR saturation. Our error floor prediction model shows very fast convergence, attributed to the large eigenvalue (*i.e.*,  $r \gg 1$ ), so we stop the model at 14 iterations.

The FER simulation results shown in these figures are produced by our LLR-domain SPA decoder, running on a computer, implemented in double-precision floating-point. This SPA decoder is non-saturating, in that computations are organized such that saturation will not occur (see §4.2.3). We have not witnessed our non-saturating decoder fail on the (5, 5) TSs, which were reported to dominate in [12]. Next, saturation was intentionally introduced into our decoder at levels shown in the figures' legends. The intentional saturation was introduced at a point corresponding to the output of the complete check node update. We simulate the

---

<sup>3</sup>The mean and variance relationship of the LLRs was printed erroneously in [5] as  $m = 2\sigma^2$

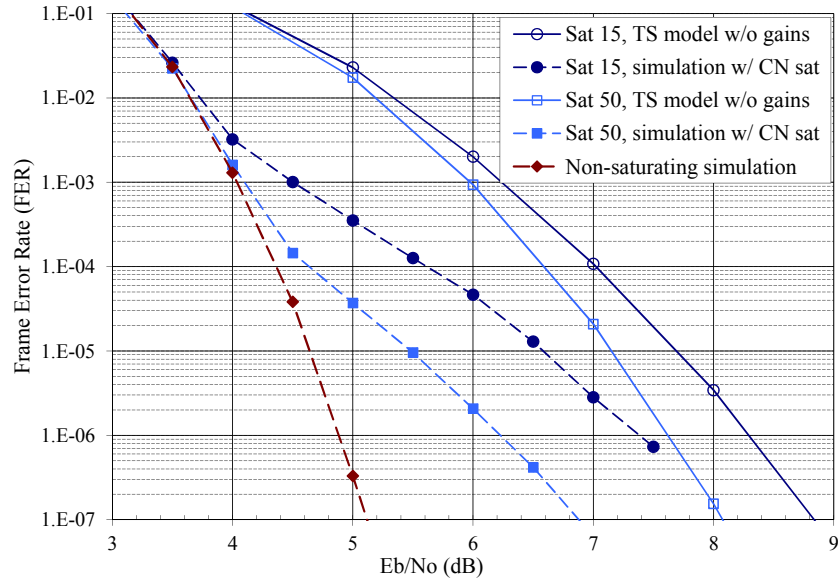


**Figure 4.10:** FER vs.  $E_b/N_0$  in dB for the (640, 192) QC code with simulation and the TS model for predicting error floors using unit gains in the model and no check node inversions.

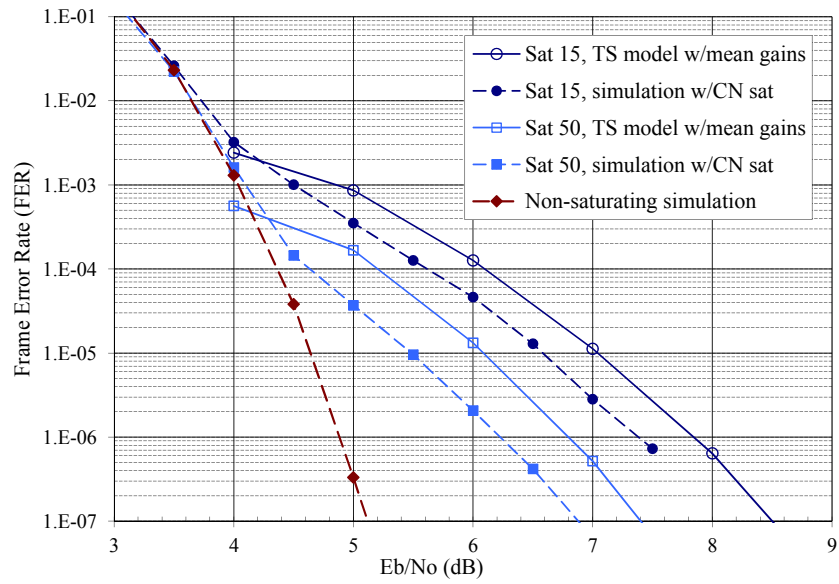


**Figure 4.11:** FER vs.  $E_b/N_0$  in dB for the (640, 192) QC code with simulation and the TS model for predicting error floors using mean gains in the model and no check node inversions.

**Table 4.3:** DDE numerical results by iteration for (3, 6)-regular code at an  $E_b/N_0$  of 2.8 dB with LLR saturation set to  $\pm 25$  at check node output. LLR pdfs discretized to pmfs with a resolution of 50/2047. In this case,  $m_\lambda = 3.8109$ .

| $l$ | $m_{\lambda(\text{ex})}^{(l)}$ | $\sigma_l^2$ | $\bar{g}_l$ |
|-----|--------------------------------|--------------|-------------|
| 1   | 0.669                          | 1.47         | 0.3242      |
| 2   | 1.315                          | 2.81         | 0.4898      |
| 3   | 2.08                           | 4.24         | 0.6243      |
| 4   | 3.11                           | 5.94         | 0.7472      |
| 5   | 4.66                           | 8.05         | 0.8586      |
| 6   | 7.21                           | 10.74        | 0.9446      |
| 7   | 11.66                          | 14.22        | 0.9891      |
| 8   | 19.67                          | 16.59        | 0.9995      |
| 9   | 24.91                          | 0.47         | 1.0000      |
| 10  | 25.00                          | 0.00         | 1.0000      |

transmission of the all-zero codeword for simplicity and take any other decoding result to be decoding failure or frame error. The simulations were run for a maximum of 200 iterations.

Fig. 4.10 shows the error floors predicted by (4.27), but with all check node gains forced to unity. Fig. 4.11 shows the same conditions, but now with the mean gains of (4.7) applied to the model. Fig. 4.12 again shows the same conditions, but this time with the modified mean gains of (4.11) applied to account for polarity inversions within the degree-two check nodes. Comparing the first two figures, one notices that the mean gains provide a significant improvement in the model's accuracy, to within 0.5 dB of simulation. Comparing Figs. 4.11 and 4.12, one notices that the channel polarity model represents an improvement in the model's accuracy to within  $\pm 0.1$  dB of simulation.

Heuristically, the reason that error floor predictions drop when check node gains are introduced has to do with incorrect state buildup in early iterations. With unity gains, incorrect channel inputs may propagate rapidly in the highly interconnected states of the TS. With small gains in early iterations this is held off for several iterations, giving the unsatisfied check nodes a better chance to correct the system. As we previously noted, during early iterations polarity

**Table 4.4:** SPA simulation results by iteration for the (2640, 1320) Margulis code which is (3, 6)-regular at an  $E_b/N_0$  of 2.8 dB with LLR saturation set to  $\pm 25$  at check node output. In this case,  $m_\lambda = 3.8109$ .

| $l$ | $m_{\lambda(\text{ex})}^{(l)}$ | $\sigma_l^2$ | $\bar{g}_l$ |
|-----|--------------------------------|--------------|-------------|
| 1   | 0.675                          | 1.49         | 0.3245      |
| 2   | 1.324                          | 2.88         | 0.4897      |
| 3   | 2.09                           | 4.36         | 0.6219      |
| 4   | 3.14                           | 6.19         | 0.7425      |
| 5   | 4.73                           | 8.83         | 0.8505      |
| 6   | 7.37                           | 13.65        | 0.9338      |
| 7   | 12.08                          | 25.82        | 0.9793      |
| 8   | 19.13                          | 33.94        | 0.9956      |
| 9   | 23.66                          | 11.86        | 0.9995      |
| 10  | 24.83                          | 1.31         | 1.0000      |

inversions through the degree-two check nodes act to effectively lower the check node gain. Therefore, as we would expect, introducing the inversions lowers the predicted error floor.

The SPA decoder results presented in [12] would appear slightly above the simulation curves with saturation set to 15 shown in Figs. 4.10 to 4.12. The FER ratio between our non-saturated SPA simulation and the SPA simulation of [12] is three orders of magnitude at an  $E_b/N_0$  of 5 dB and growing very rapidly.

The second code analyzed is the (2640, 1320) Margulis LDPC code. Recall that it is a (3, 6)-regular code with dominant TSs reported to be the (12, 4) and (14, 4) elementary TSs [2, 3, 12]. The  $\mathbf{A}$  matrices for these two TSs are both imprimitive, with  $h = 2$ , and have maximum eigenvalues  $r$  of 1.6956 and 1.7606, respectively. We ran the model for 16 iterations. The multiplicity of each of these TSs is 1320 with lots of overlap between variable nodes of differing sets. In fact, each (14, 4) TS contains all 12 variable nodes and 2 of the unsatisfied check nodes of a (12, 4) TS, as can be seen in Fig. 4.8. Thus, when we compute the failure probability of one (12, 4) TS, it largely includes the failure probability of the (14, 4) TS that contains it. Therefore, Fig. 4.13 shows the error floors predicted for just the (12, 4) TSs, utilizing the model

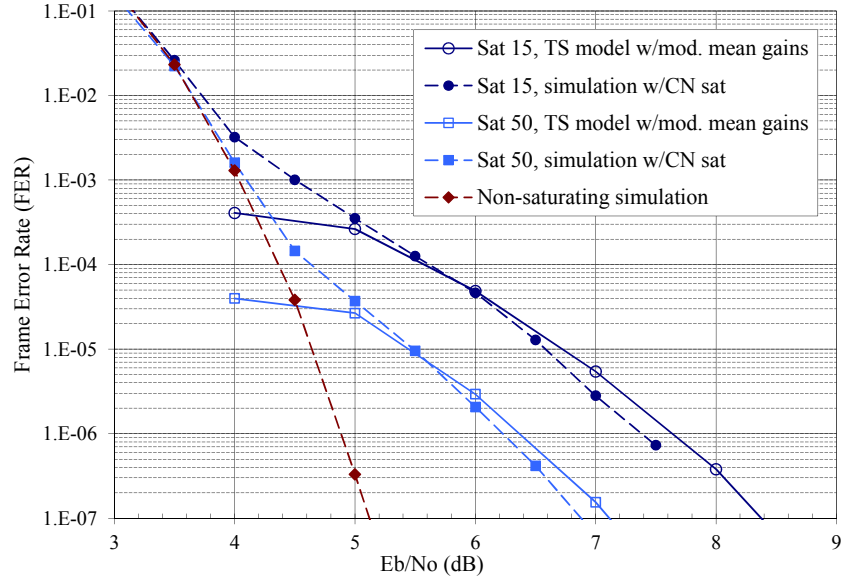


**Figure 4.12:** FER vs.  $E_b/N_0$  in dB for the (640, 192) QC code by simulation and the TS model for predicting error floors using the modified mean gains of (4.11) in the model which takes into account check node inversions.

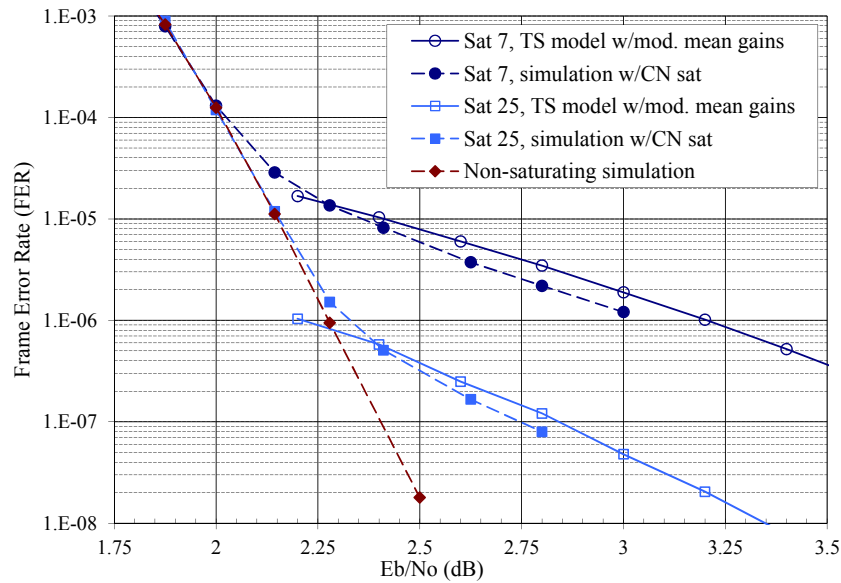


**Figure 4.13:** FER vs.  $E_b/N_0$  in dB for (2640, 1320) Margulis code with simulation and TS model predicted floor using modified mean gains.

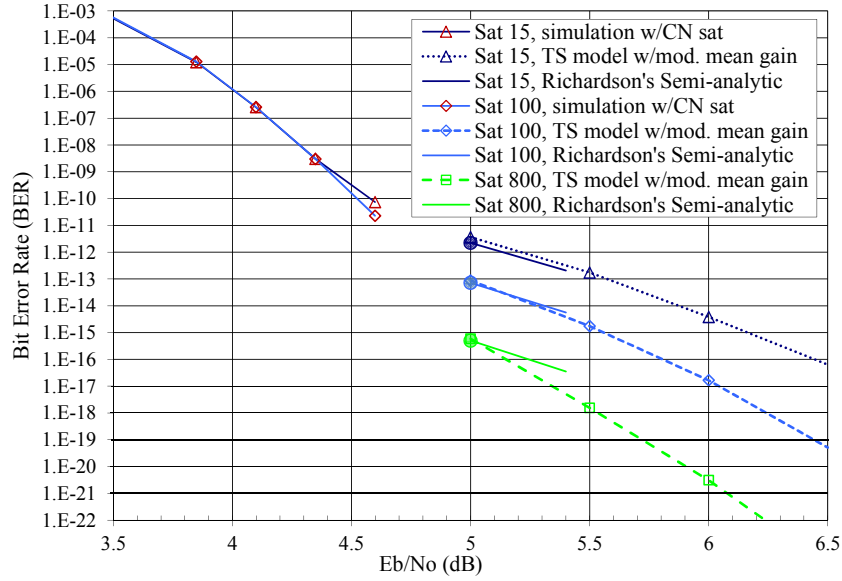


**Figure 4.14:** BER vs.  $E_b/N_0$  in dB for (2048, 1723) LDPC code from 802.3an with simulation, Richardson's semi-analytical (8, 8) floor estimation, and TS model predicted (8, 8) floor using modified mean gains.

with modified mean gains. We applied the inversion model at every iteration. We find that the state-space model overestimates the simulation's error floor by 0.15 dB or less.

The third code analyzed is the (2048, 1723) Reed-Solomon based LDPC code [48] from the 802.3an standard. It is a (6, 32)-regular code with dominant TSs reported to be the (8, 8) elementary TS [5, 25]. The  $\mathbf{A}$  matrix for this TS is primitive, with a maximum eigenvalue  $r = 4$ . The linear model for this TS converges very fast, in just 3 iterations when saturation is set to 15 and a few more iterations at higher saturation limits. We conservatively ran the model for 16 iterations, including the inversions at every iteration. The multiplicity of the (8, 8) TS is 14 272 [5]. Fig. 4.14 shows the predicted BER error floors of this code utilizing the model with modified mean gains if we assume there are no other TSs present in this code.

Since a standard Monte Carlo simulation of this low error floor takes a prohibitively large amount of CPU time, we have used Richardson's semi-analytical technique to estimate the error floor [3]. We have run this semi-analytical technique, which was introduced in Section 4.6, at an  $E_b/N_0$  of 5.0 dB (the solid circles on Fig. 4.14) and used Richardson's method of extrapolating the results to 5.4 dB (the solid lines). We have only configured the semi-analytical technique to measure the (8, 8) TSs. For the saturation levels of 15, 100, and 800, Richardson's technique required 0.9, 9.5, and 1300 hours of CPU time, respectively. We used less than 2 minutes of

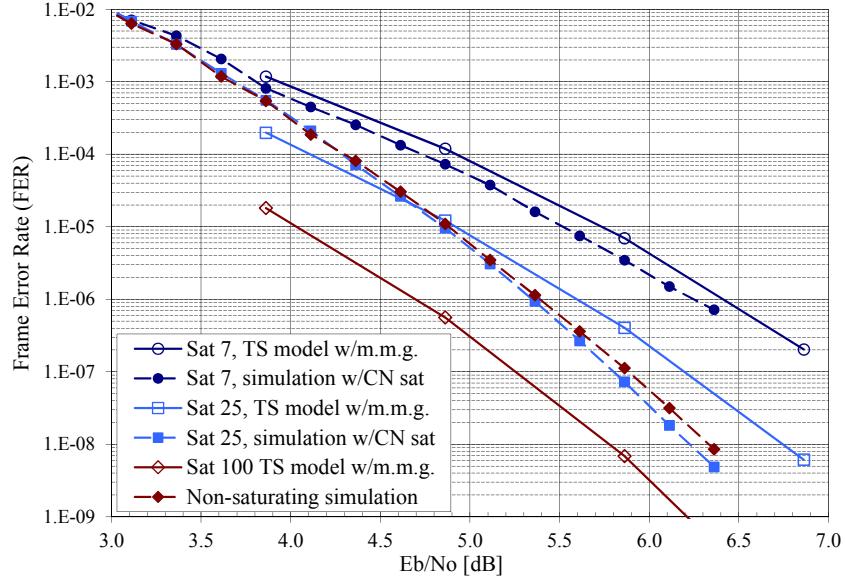


**Figure 4.15:** FER vs.  $E_b/N_0$  in dB for (155, 64) QC-LDPC code with simulation and TS model predicted floor using modified mean gains.

CPU time to collect the 1000-frames worth of LLR statistics that we used to produce each error floor prediction point. Our model's predictions appear to have an accuracy of better than 0.1 dB at 5.0 dB.

The fourth code analyzed is the (155, 64) QC-LDPC code by Tanner et al. [49]. It is a (3, 5)-regular code with the dominant TS reported to be an (8, 2) elementary TS, with multiplicity 465 [26, 27], despite the presence of several other small TSs. Additionally, Declercq et al. worked to reduce the floor of this code for the BSC channel in [50]. We are interested in understanding if the model breaks down for a code of such short block length. The  $\mathbf{A}$  matrix for this TS is primitive, with a maximum eigenvalue  $r = 1.7870$ . Fig. 4.15 shows that the predicted FER error floors of this code utilizing the (8, 2) model with modified mean gains agrees with simulation to within 0.2 dB at an LLR saturation level of 7. At this level of saturation an error floor is created; for  $E_b/N_0$  values of at least 4.86 dB, we measure 69% percent of the failures to be the (8, 2) TSs and most of the remaining failures to be other small TSs. However, at an LLR saturation level of 25, the error floor appears to be overcome since we measure only 6.8% of the failed frames are on (8, 2) TSs. Given this small fraction of (8, 2) failures captured in simulation, we must say that the linear model overestimated TS failures by about 0.9 dB at  $10^{-7}$ .

Our findings lead us to remark that the error rate results for this (155, 64) code in [26, 27] may be misleading. In that work an error floor is plotted at a saturation level of 100 based only

on the  $(8, 2)$  TSs in an SNR region where we believe they will only cause a small fraction of the overall errors observed in practice (5 to 7% perhaps). In fact, we find that some 55% of the failed frames at a saturation level of 100 contain 11 or more symbols in error. Another odd feature of Fig. 4.15 is that for the simulation without a saturation limit, the FER actually degrades a little compared to a saturation level of 25. Although it is not shown, the simulation of an LLR saturation level of 100 overlaps the non-saturating simulation. We attribute the oddities of this example to the shallow nature of waterfall region of the performance curve.

We chose these four LDPC codes as they are quite different and their error floors have appeared in prior studies. The TSs in these codes have a broad range of multiplicities and degree of overlap among them. Interestingly, the Margulis code has two dominant TSs, both with imprimitive  $\mathbf{A}$  matrices, while the other codes are dominated by single TSs with primitive  $\mathbf{A}$  matrices. Regardless, we find that the state-space model with modified mean gains estimates the simulation's error floor reasonably well and very quickly.

Surprisingly, we find that there are situations in which LLR saturation at the fairly high values of 25, 50 and 100 produce noticeable error floors in simulation and model prediction. Finally, we note that the terms in (4.32) containing  $\sigma_i^2$  have a small impact on the overall results.

## 4.8 Conclusion

We have drawn into question the error floor levels published in [2, 3, 12] and their root cause. We have shown that the error floor levels are caused by the interaction of non-codeword trapping sets and the numerical effects of handling highly-certain messages in a belief propagation decoder, such as the LLR-domain SPA decoder. We have shown that when care is taken in processing those messages the error floor level may be lowered by many orders of magnitude.

We have added some clarity to the situation of two distinctly different applications of the linear system model. On one hand, Sun introduced the state-space model and proved that many error floors do not exist for infinite-length codes without cycles [18]. On the other hand, Schlegel and Zhang improved upon the model and predicted nonzero error floors due to the dominant  $(8, 8)$  trapping sets of the 802.3an code [5].

With respect to Sun's work, we are in agreement with Sun's conclusion as applied to variable-regular LDPC codes without cycles. We have tightened Sun's asymptotic analysis to derive the SNR thresholds and SNR-LLR regions in which the beneficial metrics from the degree-one check nodes grow fast enough to overcome the potentially faulty channel information that triggers a trapping set failure. Further, we have shown empirically that the assumptions

in the probability of error model do not hold for codes with cycles. For these codes, we showed that the effects of positively correlated LLRs when allowed to grow (without saturation) drive the ratio of the mean to standard deviation to a finite value possibly implying an error floor. This error floor level is difficult to estimate due to the positively correlated LLRs and the substantially non-Gaussian distribution of our error indicator.

We are also able to put Schlegel and Zhang’s work into perspective. While not emphasized, their error rate results showing a nonzero error floor for a trapping set were for a decoder that saturated (*i.e.*, “clipped”) the LLR values. They made no claims with regard to the effects of removing saturation in [5]. Their addition of mean gains to the model makes a substantial improvement to error floor prediction as we have shown. Our realization of the model is more general than theirs, in that we do not require that  $\mathbf{B} = \mathbf{B}_{\text{ex}}$ , that  $\mathbf{B}$  has regular column weight, and that  $\mathbf{A}$  has regular row and column weight<sup>4</sup>. Additionally, we have simplified considerably the incorporation of polarity inversion by the degree-two check nodes into the model. We have presented two new methods to collect the statistics needed to drive the model. We have shown that the reason that the model predicts saturated performance relatively well is that by the iteration count at which LLRs become substantially correlated, their variance is driven to zero by the effect of saturation. This means that the model derived for the case with no cycles actually works rather well for the case with cycles when there is a significant saturation effect present. In our experiments the model with modified mean gains estimated the FER floor of floating-point simulation to within 0.2 dB for several different LDPC codes.

Altogether, we have reached a better understanding of the trapping set failure mechanisms. Trapping set failures can be frequently corrected when the beneficial unsatisfied check LLRs eventually overpower the trapping set’s own detrimental LLR growth. Achieving this reliably can take significant LLR metric dynamic range. The greater the LLRs are allowed to grow the more often channel inversions aligned to trapping sets can be overcome. Chen *et al.* had the insightful observation in [23], “...that the error floor is caused by the combined effects of short cycles in the graph representation of the code and of clipping.”

This phenomenon also helps explain prior observations. Others have noted that “stable” trapping set failures converge very rapidly (*cf.* §15.3.1 of [51]), often in less than ten iterations. We now realize that this is due to the saturation point of the beneficial unsatisfied check nodes being reached and bringing the correction process to a halt. When it is reached and the states are still in error the structure is very stable when it is an absorbing set. Absorbing sets guarantee

---

<sup>4</sup>If  $\mathbf{A}$  has regular row or column weight, then its spectral radius  $\rho(\mathbf{A})$  is integral.

that unsatisfied check messages are always outnumbered by internal messages at any variable node within the set. Metric saturation is why the bit-flipping decoder analogy was made in [11] to explain absorbing sets. The process of passing real-valued messages in an LLR-domain SPA decoder degenerates to passing positively or negatively saturated messages where the outcomes can be explained by bit-flipping decoder logic. We have shown that when saturation degenerates the SPA decoder to a bit flipping decoder, it actually helps to correct variable nodes in the trapping set until an absorbing set is reached. We have used this observation to modify Richardson's semi-analytical technique so that it may estimate error floors of a non-saturating SPA decoder.

The results make clear that when presenting error floors, future researchers should document the levels of any LLR saturation.

## 4.A Appendix: Reducible $\mathbf{A}$ Matrices

In this Appendix we address the special case of reducible  $\mathbf{A}$  matrices.

**Lemma 4.9** *Let  $G$  be a connected multigraph of order  $n \geq 2$ . Then  $G$  is isomorphic to the cycle graph  $C_n$  if and only if every vertex is degree two.*

*Proof:* By definition a cycle graph has vertices of degree two. Since isomorphisms preserve structure, graphs isomorphic to a cycle graph also have vertices of degree two. We have proved that graphs isomorphic to a cycle graph are a sufficient condition for every vertex to be degree two. To prove necessity we will take a walk on  $G$ , having every vertex of degree two. The walk will not allow backtracks, so at every intermediate vertex there is only one edge to choose to continue walking since every vertex is incident to exactly two edges. The walk will start on vertex  $v_1$ , and end when we return to  $v_1$ . Since the graph is connected this non-backtracking walk must visit every vertex. There are exactly two walks which satisfy these conditions corresponding to the choice of the starting edge selected leaving the first vertex  $v_1$ . If we relabel the vertices we visit in order as we walk  $v_2, v_3$ , etc., our relabeled multigraph is clearly a cycle graph, proving that  $G$  is isomorphic to the cycle graph  $C_n$ . ■

**Theorem 4.10** *Consider a connected multigraph  $G$  which meets Assumption 4.8.  $G$  is either a cycle graph or has an associated  $\mathbf{A}(D)$  matrix that is irreducible.*

*Proof:* Multigraphs with any vertices of degree one are not allowed by Assumption 4.8. Connected multigraphs with all vertices of degree two are cyclic by Lemma 4.9. So now we will

show any possible remaining connected multigraphs must have an irreducible adjacency matrix  $\mathbf{A}(D)$ .

The remaining multigraphs must have at least one vertex of degree greater than two. Also, all vertex degrees must sum to an even number, as every edge joins two vertices. This yields a connected multigraph with at least two cycles. Since a walk without backtracking through a connected multigraph with two cycles is able to reverse direction, every edge may be visited in either direction. Thus, when the edges of  $G$  are expanded to be vertices of the digraph  $D$ ,  $D$  will be strongly connected. Since  $D$  is strongly connected, its associated adjacency matrix  $\mathbf{A}(D)$  will be irreducible by Lemma 4.1. ■

Of course, since  $\mathbf{A}(D)$  is irreducible so must be  $\mathbf{A}(D)^T$ , which we use as  $\mathbf{A}$  in our state-space model.

Therefore, the remainder of this appendix need only address the properties of cycle graphs. The expansion of the edges of cycle graph  $G = C_a$  to a digraph  $D$  will produce two disconnected directed cycles of length  $a$ , one associated with the clockwise non-backtracking walk of  $G$  and one with the counter-clockwise non-backtracking walk of  $G$ . Thus,  $\mathbf{A}(D)$  may be symmetrically permuted to  $\mathbf{P}\mathbf{A}(D)\mathbf{P}^T = \begin{bmatrix} \mathbf{A}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_2 \end{bmatrix}$ , in which both  $\mathbf{A}_1$  and  $\mathbf{A}_2$  are irreducible. Since the component digraphs are each directed cycles of length  $a$ , for some  $\mathbf{P}$ , both adjacency submatrices  $\mathbf{A}_1$  and  $\mathbf{A}_2$  are  $a \times a$  permutation matrices that implement a circular shift by one position. Thus, both  $\mathbf{A}_1$  and  $\mathbf{A}_2$  are imprimitive with  $h = a$ . Since every row and column of  $\mathbf{A}_1$  and  $\mathbf{A}_2$  have weight one, all their eigenvalues lie on the unit circle,  $r = \rho(\mathbf{A}_1) = \rho(\mathbf{A}_2) = 1$ . The eigenvectors of  $\mathbf{A}_1$  and  $\mathbf{A}_2$  must be identical. The eigenvectors of  $\mathbf{A}_1$  may be enlarged to be the eigenvectors of  $\mathbf{A}$  by appending  $a$  zeros. As  $\mathbf{A}_1$  is a permutation matrix, the eigenvalue  $r = 1$  is associated with the all-one eigenvector (or a scalar multiple thereof):  $\mathbf{A}_1 \mathbf{v}_{1,1} = r \mathbf{v}_{1,1}$ , where  $\mathbf{v}_{1,1} = [1, 1, \dots, 1]^T$ . By a similar argument the associated left eigenvector of  $\mathbf{A}_1$  is  $\mathbf{w}_{1,1} = [1, 1, \dots, 1]^T$ . Similar statements hold for  $\mathbf{A}_2$ .

Define  $\tilde{\mathbf{w}}_{1,1}$  to be  $\mathbf{w}_{1,1}$  appended with  $a$  zeros, and  $\tilde{\mathbf{w}}_{2,1}$  to be  $\mathbf{w}_{2,1}$  prefixed with  $a$  zeros. Now, let us complete the derivations in Section 4.4 using  $\beta_l \triangleq \mathbf{w}_1^T \mathbf{x}_l$ , with  $\mathbf{w}_1 \triangleq \tilde{\mathbf{w}}_{1,1} + \tilde{\mathbf{w}}_{2,1}$ . Since  $\tilde{\mathbf{w}}_{1,1}$  and  $\tilde{\mathbf{w}}_{2,1}$  are associated with the same eigenvalue, their sum  $\mathbf{w}_1 = [1, 1, \dots, 1]^T$  is also a left eigenvector of  $\mathbf{A}$ . Note that  $\mathbf{w}_1$  is not the only positive left eigenvector of  $\mathbf{A}$ , as any linear combination  $c_1 \tilde{\mathbf{w}}_{1,1} + c_2 \tilde{\mathbf{w}}_{2,1}$  with  $c_i > 0$  would also be a positive left eigenvector. For

the cycle graphs under consideration, (4.25) becomes

$$\begin{aligned} \beta'_l &= (\tilde{\mathbf{w}}_{1,1}^T + \tilde{\mathbf{w}}_{2,1}^T) \mathbf{B} \boldsymbol{\lambda} \prod_{j=1}^l g_j \\ &+ \sum_{i=1}^l (\tilde{\mathbf{w}}_{1,1}^T + \tilde{\mathbf{w}}_{2,1}^T) \left( \mathbf{B} \boldsymbol{\lambda} + \mathbf{B}_{\text{ex}} \boldsymbol{\lambda}_i^{(\text{ex})} \right) \prod_{j=i+1}^l g_j. \end{aligned} \quad (4.38)$$

This is equivalent to the other expressions derived in Section 4.4 since  $\mathbf{w}_1 = \tilde{\mathbf{w}}_{1,1} + \tilde{\mathbf{w}}_{2,1}$ , noting that  $r = 1$  here. With these considerations we may use the results of Section 4.4 and later for the allowed reducible  $\mathbf{A}$  matrices. Relative to the variable-regular codes addressed in this chapter, the cycle graph  $C_a$  corresponds to the  $(a, b)$  trapping set with  $a \geq 2$  and  $b = (d_v - 2)a$ .

## 4.B Appendix: Adding Leaves and Branches

We now expand the set of multigraphs addressed in this work to include those with leaves and branches, which were previously eliminated by Assumption 4.8. Thus, the variable nodes in the associated Tanner subgraphs will be permitted to have  $d_v - 1$  adjacent degree-one check nodes. In [18], Sun referred to the graphs addressed in this appendix as “trapping sets with external data nodes.”

We will describe the elementary Tanner graphs of this appendix from the perspective of their associated multigraphs using the mapping of Lemma 4.6. Let a *base* graph be a multigraph that meets Assumptions 4.2, 4.7, and 4.8. Such a graph contains one or more cycles. A *leaf* is a vertex of degree one. A *branch* is a vertex of degree two or more outside of the base graph, and is not contained in any cycles. Note that branches become leaves upon removing their attached leaves and/or branches.

**Example 4.6** Fig. 4.5a contains one leaf on a base graph and Fig. 4.5b contains just three leaves and no base graph. The  $(18, 8)$  failing structure of Fig. 4.8c has leaves A, B, and D and branch C on a base graph that is the  $(14, 4)$  trapping set of Fig. 4.8b.

**Assumption 4.12 (Replacement for Assumption 4.8)** Multigraphs of interest must contain a base graph and zero or more leaves and branches.

We require a base graph because without a base graph the eigenvalues would all be zero, as will become apparent shortly. Our new assumption allows the variable nodes within the associated Tanner subgraphs to neighbor up to  $d_v - 1$  degree-one check nodes.

**Theorem 4.11** *Let  $G = (V, E)$  be a multigraph meeting Assumptions 4.2, 4.7, and 4.12 containing base graph  $G_B$ ,  $n$  leaves, with  $n \geq 1$ , and zero or more branches. Let  $D = (Z, A)$  and  $D_B$  be the associated digraphs, created as described in Section 4.3.3. Then, the adjacency matrix  $\mathbf{A}(D)$  is reducible and its spectrum contains the eigenvalues of  $\mathbf{A}(D_B)$  and zeros.*

*Proof:* Each additional leaf in  $G$  adds one edge to  $G$ . Let the edge  $e_i \in E$  join leaf  $v_k$  to vertex  $v_j$  in  $G$ , where  $v_j$  is not a leaf. The edge  $e_i$  maps to vertices  $z_i, z_i' \in Z$  in  $D$ . These vertices are not strongly connected to the digraph as  $d_i^+ = 0$  and  $d_{i'}^- = 0$ , and hence  $\mathbf{A}(D)$  must be reducible by Lemma 4.1.

Every leaf in  $G$  creates an all-zero column in  $\mathbf{A}(D)$  due to  $d_{i'}^- = 0$  and an all-zero row due to  $d_i^+ = 0$ . The all-zero columns may be symmetrically permuted to the left (*i.e.*, their corresponding vertices are relabeled with the lowest possible values) and the all-zero rows to the bottom to form

$$\mathbf{P} \mathbf{A}(D) \mathbf{P}^T = \begin{bmatrix} \mathbf{0} & \mathbf{Y}_1 & \mathbf{Y}_2 \\ \mathbf{0} & \mathbf{B} & \mathbf{Y}_3 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}, \quad (4.39)$$

where the block diagonal contains square submatrices. The  $n \times n$  zero matrices at both ends of the block diagonal correspond to the  $n$  leaves in  $D$ .

The eigenvalues of  $\mathbf{A}(D)$  are the roots of the characteristic equation  $\det(\mathbf{A}(D) - \mu \mathbf{I}) = 0$ , which simplifies to  $\det(\mathbf{B} - \mu \mathbf{I}) \mu^{2n} = 0$  by use of the expansion by minors along every zero row and column. Thus, the eigenvalues of  $\mathbf{A}(D)$  are the eigenvalues of  $\mathbf{B}$  and  $2n$  zeros. In case  $G$  contains both leaves and branches, the removal of one layer of leaves exposes a new layer of leaves and the operations in this paragraph must be repeated until we work down to  $\mathbf{B} = \mathbf{A}(D_B)$ , which will be irreducible except for the case addressed in Appendix 4.A in which  $G_B$  is a cycle graph. ■

By showing that the nonzero eigenvalues are preserved by the addition of leaves and branches, it is simple to argue that Theorem 4.7 and Corollary 4.8, which bound the spectral radius of  $\mathbf{A}(D)$ , still hold if Assumption 4.8 is replaced by 4.12.

The only weakness with respect to our prior development is that we can no longer assume that the left eigenvector  $\mathbf{w}_1$  is positive as  $\mathbf{A}(D)$  may now be reducible. In practice, we have found that the new entries added to  $\mathbf{w}_1$  by the addition of leaves and branches are half zero and half positive. This may be proved by applying the Subinvariance Theorem in [30, p. 23] to (4.39). The presence of zero entries in  $\mathbf{w}_1$  weakens our prior claims on the error indicator  $\beta_l \triangleq \mathbf{w}_1^T \mathbf{x}_l$ . Now, the error indicator is most effective on the variable nodes corresponding to

the base graph and less effective on the variable nodes corresponding to the branches and leaves.

Finally, we do not see much practical motivation to predict the error floors of graphs with branches and leaves. We would prefer to run predictions on the base graphs contained within. Since the graphs with branches and leaves have the same spectral radius as their base graph they are no more likely to fail in theory. In fact, they should be less likely to fail as more channel values are involved and more unsatisfied check nodes are working to correct the graph.

## 4.C Appendix: Tables of Absorbing Set Structure

As opposed to studying the trapping sets that dominate several specific codes, in this appendix we present an overview of all subgraphs that may become troublesome trapping sets which meet specific conditions for particular large classes of codes. This can serve to put some perspective on the structural parameters that have been discussed in this chapter such as:  $a$ ,  $b$ , the index of imprimitivity  $h$ , and the spectral radius  $r$ .

The variable-regular codes we examine are assumed to be described by Tanner graphs that are 4-cycle-free. We divide the information into tables by their variable-degree  $d_v$ . The further conditions we put on the subgraphs of interest are described in Assumptions 4.2 and 4.7 and Definition 4.5; the subgraphs must be elementary, connected and absorbing, respectively.

**Table 4.5:** Connected, elementary absorbing sets from the set of 4-cycle-free, degree-3 variable-regular codes, with  $r_{\max} > 1.3$

| $(a, b)$ | Num.   | $h_{\max}$ | $r_{\min}$ | $r_{\max}$ |
|----------|--------|------------|------------|------------|
| 4,0      | 1      | 1          | 2          | 2          |
| 4,2      | 1      | 1          | 1.521      | 1.521      |
| 5,1      | 1      | 1          | 1.829      | 1.829      |
| 5,3      | 2      | 4          | 1.414      | 1.424      |
| 6,0      | 2      | 2          | 2          | 2          |
| 6,2      | 4      | 2          | 1.696      | 1.729      |
| 6,4      | 4      | 2          | 1.348      | 1.361      |
| 7,1      | 4      | 1          | 1.883      | 1.888      |
| 7,3      | 10     | 2          | 1.599      | 1.665      |
| 7,5      | 6      | 2          | 1.298      | 1.316      |
| 8,0      | 5      | 2          | 2          | 2          |
| 8,2      | 19     | 2          | 1.780      | 1.870      |
| 8,4      | 25     | 2          | 1.521      | 1.622      |
| 9,1      | 19     | 1          | 1.911      | 1.929      |
| 9,3      | 63     | 2          | 1.696      | 1.851      |
| 9,5      | 52     | 2          | 1.463      | 1.592      |
| 10,0     | 19     | 2          | 2          | 2          |
| 10,2     | 113    | 2          | 1.829      | 1.911      |
| 10,4     | 198    | 2          | 1.629      | 1.841      |
| 10,6     | 109    | 4          | 1.414      | 1.570      |
| 11,1     | 114    | 1          | 1.929      | 1.947      |
| 11,3     | 482    | 2          | 1.758      | 1.899      |
| 11,5     | 536    | 2          | 1.571      | 1.836      |
| 11,7     | 197    | 2          | 1.379      | 1.555      |
| 12,0     | 85     | 2          | 2          | 2          |
| 12,2     | 835    | 2          | 1.861      | 1.940      |
| 12,4     | 1,892  | 2          | 1.696      | 1.894      |
| 12,6     | 1,373  | 2          | 1.521      | 1.833      |
| 12,8     | 351    | 2          | 1.348      | 1.545      |
| 13,1     | 839    | 1          | 1.941      | 1.961      |
| 13,3     | 4,541  | 2          | 1.799      | 1.934      |
| 13,5     | 6,374  | 2          | 1.645      | 1.891      |
| 13,7     | 3,159  | 2          | 1.481      | 1.831      |
| 13,9     | 581    | 2          | 1.321      | 1.537      |
| 14,0     | 509    | 2          | 2          | 2          |
| 14,2     | 7,589  | 2          | 1.883      | 1.954      |
| 14,4     | 21,434 | 2          | 1.745      | 1.931      |
| 14,6     | 19,587 | 2          | 1.599      | 1.889      |
| 14,8     | 6,879  | 2          | 1.446      | 1.830      |
| 14,10    | 931    | 4          | 1.298      | 1.532      |

**Table 4.6:** Connected, elementary absorbing sets from the set of 4-cycle-free, degree-4 variable-regular codes

| $(a, b)$ | Num.   | $h_{\max}$ | $r_{\min}$ | $r_{\max}$ |
|----------|--------|------------|------------|------------|
| 4,4      | 1      | 1          | 2          | 2          |
| 5,0      | 1      | 1          | 3          | 3          |
| 5,2      | 1      | 1          | 2.629      | 2.629      |
| 5,4      | 1      | 1          | 2.219      | 2.219      |
| 6,0      | 1      | 1          | 3          | 3          |
| 6,2      | 2      | 1          | 2.697      | 2.710      |
| 6,4      | 3      | 1          | 2.355      | 2.367      |
| 6,6      | 2      | 2          | 2          | 2          |
| 7,0      | 2      | 1          | 3          | 3          |
| 7,2      | 7      | 1          | 2.744      | 2.762      |
| 7,4      | 11     | 2          | 2.449      | 2.480      |
| 7,6      | 4      | 1          | 2.159      | 2.160      |
| 8,0      | 6      | 2          | 3          | 3          |
| 8,2      | 28     | 2          | 2.778      | 2.805      |
| 8,4      | 50     | 2          | 2.525      | 2.585      |
| 8,6      | 28     | 2          | 2.272      | 2.296      |
| 8,8      | 5      | 2          | 2          | 2          |
| 9,0      | 16     | 1          | 3          | 3          |
| 9,2      | 126    | 1          | 2.805      | 2.850      |
| 9,4      | 285    | 2          | 2.584      | 2.728      |
| 9,6      | 177    | 2          | 2.355      | 2.429      |
| 9,8      | 27     | 1          | 2.126      | 2.141      |
| 10,0     | 59     | 2          | 3          | 3          |
| 10,2     | 719    | 2          | 2.826      | 2.886      |
| 10,4     | 1,915  | 2          | 2.629      | 2.821      |
| 10,6     | 1,404  | 2          | 2.422      | 2.648      |
| 10,8     | 298    | 2          | 2.219      | 2.313      |
| 10,10    | 19     | 2          | 2          | 2          |
| 11,0     | 265    | 1          | 3          | 3          |
| 11,2     | 4,721  | 1          | 2.843      | 2.903      |
| 11,4     | 14,569 | 2          | 2.667      | 2.852      |
| 11,6     | 12,458 | 2          | 2.478      | 2.788      |
| 11,8     | 3,231  | 2          | 2.295      | 2.458      |
| 11,10    | 208    | 1          | 2.104      | 2.127      |

**Table 4.7:** Connected, elementary absorbing sets from the set of 4-cycle-free, degree-5 variable-regular codes, with  $r_{\max} > 2.6$

| $(a, b)$ | Num.   | $h_{\max}$ | $r_{\min}$ | $r_{\max}$ |
|----------|--------|------------|------------|------------|
| 5,5      | 1      | 1          | 3          | 3          |
| 5,7      | 1      | 1          | 2.629      | 2.629      |
| 6,0      | 1      | 1          | 4          | 4          |
| 6,2      | 1      | 1          | 3.693      | 3.693      |
| 6,4      | 2      | 1          | 3.360      | 3.403      |
| 6,6      | 4      | 1          | 3          | 3.112      |
| 6,8      | 5      | 1          | 2.697      | 2.767      |
| 7,1      | 1      | 1          | 3.875      | 3.875      |
| 7,3      | 5      | 1          | 3.596      | 3.669      |
| 7,5      | 14     | 1          | 3.307      | 3.427      |
| 7,7      | 23     | 1          | 3          | 3.130      |
| 7,9      | 25     | 1          | 2.744      | 2.827      |
| 8,0      | 3      | 1          | 4          | 4          |
| 8,2      | 16     | 1          | 3.775      | 3.827      |
| 8,4      | 68     | 1          | 3.522      | 3.673      |
| 8,6      | 165    | 1          | 3.271      | 3.465      |
| 8,8      | 252    | 2          | 3          | 3.208      |
| 8,10     | 232    | 2          | 2.778      | 2.918      |
| 8,12     | 124    | 2          | 2.525      | 2.619      |
| 9,1      | 28     | 1          | 3.904      | 3.905      |
| 9,3      | 276    | 1          | 3.693      | 3.769      |
| 9,5      | 1,151  | 2          | 3.464      | 3.665      |
| 9,7      | 2,541  | 2          | 3.243      | 3.521      |
| 9,9      | 3,284  | 2          | 3          | 3.289      |
| 9,11     | 2,541  | 2          | 2.805      | 3.071      |
| 9,13     | 1,150  | 2          | 2.584      | 2.751      |
| 10,0     | 60     | 2          | 4          | 4          |
| 10,2     | 1,188  | 2          | 3.822      | 3.870      |
| 10,4     | 8,435  | 2          | 3.626      | 3.789      |
| 10,6     | 27,706 | 2          | 3.421      | 3.810      |
| 10,8     | 49,991 | 2          | 3.220      | 3.717      |
| 10,10    | 53,884 | 2          | 3          | 3.440      |
| 10,12    | 35,721 | 2          | 2.826      | 3.191      |
| 10,14    | 14,308 | 2          | 2.629      | 3.009      |
| 10,16    | 3,224  | 2          | 2.422      | 2.652      |

Rather than find every single graph that meets our parameters we can simplify the search considerably. As we are just interested in a graph's structure as opposed to its labeling we need only identify one of the graphs among several that are isomorphic to the others. This is known as partitioning the graphs into *equivalence classes*.

Graph theory tools such as “geng” [52] can generate non-isomorphic simple graph descriptions very quickly satisfying sets of parameters such as ours. We run this tool once for each row of the tables. Each time we configure it to find undirected graphs of order  $a$ , size  $(ad_v - b)/2$ , and with vertex degrees in the interval  $[\lceil d_v/2 \rceil, d_v]$ . The size we have specified is a direct result of Euler's handshaking lemma. By limiting the tool output to simple graphs, we eliminate multigraphs of girth 2 and their equivalent Tanner subgraphs of girth 4. The range of vertex degrees specified ensures that we get absorbing sets. We then process each graph in a custom tool that converts it to the adjacency matrix of the associated directed graph to find its spectral radius  $r$  and index of imprimitivity  $h$ .

Table 4.5 shows the graphs found based on the set of 4-cycle-free codes with  $d_v = 3$ . The graphical equivalence classes found are divided into rows by their  $(a, b)$  parameters listed in the first column. The second column of each row presents the number of equivalence classes found that meet the  $(a, b)$  parameters and all our other assumptions. The third column shows the maximum index of imprimitivity  $h$  over the  $(a, b)$  equivalence classes. The fourth and fifth columns show the minimum and maximum spectral radius  $r$  over the  $(a, b)$  equivalence classes. To save space we have left out the weakest  $(a, b)$  pairs with  $r_{\max} \leq 1.3$ . Tables 4.6 and 4.7 present similar results based on codes with  $d_v = 4$  and 5, respectively, omitting the  $(a, b)$  pairs with  $r_{\max} \leq 2.6$  for  $d_v = 5$ .

These tables present enough information to comment on the approximations to the spectral radius  $r$  developed in Section 4.3.3. The extreme relative estimation errors are summarized in Table 4.8. We find empirically that the approximation of (4.21) does not overestimate the spectral radius, but generally underestimates it. For 90% of the  $d_v = 3$  equivalence classes, this approximation deviates by no more than 6.0% below the true value of the spectral radius. The approximation of (4.22) generally produces larger estimates of the spectral radius, sometimes overestimating the true value. For the  $d_v = 3$  equivalence classes shown in Table 4.5, the mean approximation error is  $-3.5\%$  for (4.21) and  $+1.3\%$  for (4.22). For both estimators, we measure a standard deviation of the relative error of about 2.0%. Since we find that the correlation coefficient between the relative errors of these two estimates to be 0.732, we may reduce the standard deviation and bias by combining them. In summary, it appears that (4.22) and the combined

**Table 4.8:** Extreme Relative Estimation Errors for the Spectral Radius

| Originating<br>Table | With respect<br>to (4.21) and |            | With respect<br>to (4.22) and |                     |
|----------------------|-------------------------------|------------|-------------------------------|---------------------|
|                      | $r_{\min}$                    | $r_{\max}$ | $r_{\min}$                    | $r_{\max}$          |
| 4.5 ( $d_v = 3$ )    | 0.0%                          | -21.9%     | 6.1%                          | -16.4%              |
| 4.6 ( $d_v = 4$ )    | 0.0%                          | -12.0%     | 2.1%                          | -9.4%               |
| 4.7 ( $d_v = 5$ )    | 0.0%                          | -13.9%     | 1.0%                          | -20.2% <sup>a</sup> |
| 4.7 with $b \leq a$  | 0.0%                          | -13.9%     | 1.0%                          | -12.9%              |

<sup>a</sup>Discovered over the relatively weak (10, 16) trapping sets.

estimator are marginally better estimators than (4.21).

As discussed in Section 4.3.3, the approximation (4.22) assumes that the trapping set contains mostly zero or one unsatisfied check node per variable node, which is not always true in Table 4.7 (*i.e.*,  $d_v = 5$ ). When the assumption does not hold, the approximation of (4.22) can underestimate the spectral radius significantly, as demonstrated by the extreme value within Table 4.8. In this case, the -20.2% error occurred over the relatively weak (10, 16) trapping sets. If we limit the processing of Table 4.7 to its rows such that  $b \leq a$ , *i.e.*, the strong trapping sets, then the extreme relative error for this estimator would be reduced to -12.9% as shown in the final row of Table 4.8.

## 4.D Appendix: Density Evolution in the Error Floor Region

This appendix explores density evolution for LDPC codes in the AWGN channel at signal-to-noise ratios (SNRs) that are significantly above the decoding threshold. Sun's analytical development, which used density evolution (DE), was asymptotic in log-likelihood ratio (LLR) values and the number of iterations [18]. Here, we develop the governing expressions which closely approximate the LLR growth behavior at moderate amplitudes. We then produce bounds on the SNR region in which we can expect the necessary growth rate of the mean LLR which to eliminate error floors. We find our predictions produced for the mean using these techniques to be accurate in the error floor region. Unfortunately, the model of error floor prediction also depends upon the variance of the LLRs, and it is in this respect that our DE analysis is

inconclusive.

DE was introduced in [43, 53] as a technique to estimate the distribution of messages in an SPA decoder. The technique was then used to find the decoding threshold of LDPC codes with specific degree distributions in the limit as the block length goes to infinity. For large block lengths, the error rate typically drops very dramatically as the SNR exceeds the decoding threshold. As we are interested in the behavior of the error floor in this work, we assume that the channel SNR is always above the decoding threshold.

In the limit, as assumed in DE, the block length goes to infinity and the Tanner graph is free of cycles. However, in any “good” finite-length code, the Tanner graph will indeed have cycles [35]. On the other hand, we generally find the SPA decoder does quite well even though it implicitly uses the cycle-free assumption.

As DE progresses above the decoding threshold, LLR values grow fast, but the question is how fast. In [18], Sun showed that mean check node output LLRs,  $m_{\lambda(\text{ex})}$ , grow from iteration  $l - 1$  to  $l$  as

$$m_{\lambda(\text{ex})}^{(l)} = (d_v - 1)m_{\lambda(\text{ex})}^{(l-1)} + \text{“some small value terms.”}$$

We seek to bound the effect of Sun’s “small value terms,” by developing the governing expressions rather than the asymptotic result. We follow the development in [18], but we also develop bounds on the SNR region in which we can expect the growth in  $m_{\lambda(\text{ex})}^{(l)}$  necessary to eliminate error floors.

We begin by using the consistent Gaussian approximation to AWGN DE [43]. The update rule at iteration  $l$  for the unsatisfied check nodes’ output mean  $m_{\lambda(\text{ex})}$  is

$$m_{\lambda(\text{ex})}^{(l)} = \phi^{-1} \left( 1 - \left[ 1 - \phi \left( m_{\lambda} + (d_v - 1)m_{\lambda(\text{ex})}^{(l-1)} \right) \right]^{d_c - 1} \right), \quad (4.40)$$

where  $\phi(x)$  is defined by the integral

$$\phi(x) \triangleq \begin{cases} 1 - \frac{1}{\sqrt{4\pi x}} \int_{\mathbb{R}} \tanh \frac{u}{2} e^{-\frac{(u-x)^2}{4x}} du, & \text{if } x > 0 \\ 1, & \text{if } x = 0. \end{cases}$$

Note that  $\phi(x)$  is continuous and monotonically decreasing from 1 to 0, as  $x$  increases from 0 toward  $\infty$ .

For any  $x > 0$ , the function  $\phi(x)$  is bounded [43] by

$$\sqrt{\frac{\pi}{x}} e^{-x/4} \left( 1 - \frac{3}{x} \right) < \phi(x) < \sqrt{\frac{\pi}{x}} e^{-x/4} \left( 1 - \frac{1}{7x} \right). \quad (4.41)$$

These bounds tighten significantly as  $x$  increases. We will take the lower bound of (4.41) and use it to upper bound  $1 - \phi(x)$  as

$$1 - \delta \sqrt{\frac{\pi}{x}} e^{-x/4} > 1 - \phi(x), \quad (4.42)$$

where  $\delta \triangleq 1 - 3/x$ , noting that  $\delta$  approaches 1 for large  $x$ . We take a small digression before proceeding with the main theorem to develop inequalities that we will require.

**Lemma 4.12** *For any  $K \in \mathbb{R}$  and  $x, \beta > 0$  there exists a unique  $k_1 \in \mathbb{R}$  such that  $K = k_1 + \beta \ln(1 + k_1/x)$ . Moreover,  $k_1 > -x$ .*

*Proof:* The function  $f(k_1) = k_1 + \beta \ln(1 + k_1/x)$  is well-defined only on the domain  $k_1 > -x$ . It is continuous and monotonically increasing on that domain, and  $f(k_1) \rightarrow -\infty$  as  $k_1 \rightarrow -x$  from the right. ■

**Lemma 4.13** *Let  $m + \beta \ln m > x + \beta \ln x + K$  where  $m, x, \beta, K \in \mathbb{R}$  with  $m, x, \beta > 0$ . Then  $m > x + \frac{K}{1+\beta/x}$ .*

*Proof:* First we will prove by contrapositive that  $m > x + k_1$ , where  $K = k_1 + \beta \ln(1 + k_1/x)$ . From Lemma 4.12 we know for any  $K$  we can find a unique real  $k_1 > -x$ .

Assume that  $m \leq x + k_1$ . Then we can write  $\beta \ln m \leq \beta \ln(x + k_1) = \beta \ln x + \beta \ln(1 + k_1/x)$ . Adding these two inequalities together yields

$$m + \beta \ln m \leq x + \beta \ln x + k_1 + \beta \ln(1 + k_1/x).$$

Recognizing that the right two terms combine to equal  $K$ , we have an expression that exactly contradicts our recent assumption. Thus,  $m > x + k_1$ .

Since we know for any  $z > -1$  that  $z \geq \ln(1 + z)$ , we can write

$$K = k_1 + \beta \ln(1 + k_1/x) \leq k_1 + \beta k_1/x.$$

This implies  $k_1 \geq \frac{K}{1+\beta/x}$ . Finally, we combine these two inequalities to produce the desired result, i.e.,

$$m > x + k_1 \geq x + \frac{K}{1 + \beta/x}.$$

■

**Theorem 4.14** *Given a  $(d_v, d_c)$ -regular LDPC code, with  $d_v \geq 3$ , at sufficiently high  $E_b/N_0$ , after a finite number of iterations, DE predicts that the SPA decoder's check node output mean  $m_{\lambda(\text{ex})}^{(l)}$  dominates  $(d_v - 1)^l$  asymptotically.*

*Proof:* We start by restating (4.40) in the form

$$1 - \phi\left(m_{\lambda(\text{ex})}^{(l)}\right) = \left[1 - \phi\left(m_{\lambda} + (d_v - 1)m_{\lambda(\text{ex})}^{(l-1)}\right)\right]^{d_c - 1}. \quad (4.43)$$

First, we would like to apply the upper bound of (4.42) to the LHS of (4.43), where  $x = m_{\lambda(\text{ex})}^{(l)}$ . To do this we must assume that  $m_{\lambda(\text{ex})}^{(l)} > 3$  so that  $\delta > 0$ . It will take some number of iterations for  $\delta$  to get close to 1, depending upon the AWGN channel's  $E_b/N_0$ , but when the SNR exceeds the decoding threshold, we are assured that the LLRs will continue growing by DE. Now,

$$1 - \delta \sqrt{\frac{\pi}{m_{\lambda(\text{ex})}^{(l)}}} e^{-m_{\lambda(\text{ex})}^{(l)}/4} > \left[1 - \phi\left(m_{\lambda} + (d_v - 1)m_{\lambda(\text{ex})}^{(l-1)}\right)\right]^{d_c - 1},$$

where  $\delta = 1 - 3/m_{\lambda(\text{ex})}^{(l)}$ .

Next, we can truncate the binomial expansion to a simple bound, in the form shown below for small  $\phi(x)$ . Specifically, for any positive integer  $n$ ,

$$\begin{aligned} (1 - a)^n &= 1 - na + \binom{n}{2}a^2 - \binom{n}{3}a^3 \dots + (-a)^n \\ &> 1 - na, \quad \text{for small enough } a. \end{aligned} \quad (4.44)$$

For example, by the time the LLRs have reached  $m_{\lambda} + (d_v - 1)m_{\lambda(\text{ex})}^{(i-1)} > 8$ , we are assured that  $\phi(\cdot) < 0.09$  by (4.41), which is small enough for the truncated binomial (4.44) to hold as a positive lower bound for  $d_c \leq 12$ . Codes with higher check degree will require larger LLRs for (4.44) to hold. Utilizing this binomial bound we now have

$$\sqrt{\frac{\pi}{m_{\lambda(\text{ex})}^{(l)}}} e^{-m_{\lambda(\text{ex})}^{(l)}/4} < \frac{d_c - 1}{\delta} \phi\left(m_{\lambda} + (d_v - 1)m_{\lambda(\text{ex})}^{(l-1)}\right). \quad (4.45)$$

Next, we apply the upper bound of (4.41) with  $x = m_{\lambda} + (d_v - 1)m_{\lambda(\text{ex})}^{(l-1)}$  to the right-hand side of (4.45), take the logarithm of both sides, and scale by  $-4$  to find

$$\begin{aligned} 2 \ln m_{\lambda(\text{ex})}^{(l)} + m_{\lambda(\text{ex})}^{(l)} &> \\ &- 4 \ln \left( \frac{(d_c - 1) \left(1 - \frac{1}{7x}\right)}{\delta} \right) + 2 \ln x + x. \end{aligned} \quad (4.46)$$

Applying Lemma 4.13 with  $\beta = 2$  to (4.46) yields

$$\begin{aligned} m_{\lambda(\text{ex})}^{(l)} &> x - 4 \frac{\ln \left\{ \frac{d_c - 1}{\delta} \left(1 - \frac{1}{7x}\right) \right\}}{1 + 2/x} \\ &= m_{\lambda} + (d_v - 1)m_{\lambda(\text{ex})}^{(l-1)} - 4 \frac{\ln \left\{ \frac{d_c - 1}{\delta} \left(1 - \frac{1}{7x}\right) \right\}}{1 + 2/x}. \end{aligned} \quad (4.47)$$

Examining (4.47), we see that if the  $m_\lambda$  term exceeds the magnitude of the final term which is negative, we can guarantee  $m_{\lambda(\text{ex})}^{(l)}$  growth by at least a factor of  $d_v - 1$  at each iteration. Noting that the mean channel LLR is  $m_\lambda = 4RE_b/N_0$  in the AWGN channel, the condition needed to satisfy  $m_{\lambda(\text{ex})}^{(l)} > C(d_v - 1)^l$  is

$$R \frac{E_b}{N_0} > \frac{\ln \left\{ \frac{d_c - 1}{\delta} \left( 1 - \frac{1}{7x} \right) \right\}}{1 + 2/x}, \quad (4.48)$$

where  $x = m_\lambda + (d_v - 1)m_{\lambda(\text{ex})}^{(l-1)} \gg 0$ , and  $x$  is increasing with each iteration. Since the two factors containing  $x$  reduce the right-hand side of (4.48) for all  $x > 1/7$ , we can make the condition (4.48) simpler, but stronger as

$$R \frac{E_b}{N_0} > \ln \frac{d_c - 1}{\delta}, \quad (4.49)$$

where  $\delta = 1 - 3/m_{\lambda(\text{ex})}^{(l)} > 0$ . The value of  $\delta$  approaches 1 as the number of iterations increases. If our earlier assumptions on the magnitude of the LLRs and (4.49) are satisfied, we have reached the LLR growth regime of  $m_{\lambda(\text{ex})}^{(l)} > (d_v - 1)m_{\lambda(\text{ex})}^{(l-1)}$ . ■

With  $\delta = 1$ , (4.49) has appeared in [54, p. 38], [55, 56] for several LDPC decoder algorithms as a breakout channel condition in which the error rate tends toward zero as the iteration count and block length increase toward infinity.

**Example 4.7** For any  $(3, 6)$ -regular code,  $R = 0.5$ , we compute the  $E_b/N_0$  threshold of (4.49) to be 5.077 dB, assuming that  $\delta \approx 1$ . At any  $E_b/N_0$  greater than 5.077 dB, we can guarantee that  $m_{\lambda(\text{ex})}^{(i)} > C(d_v - 1)^i$  is reached after enough iterations to satisfy the assumptions made in this section. This is a sufficient extrinsic LLR growth rate to overcome any potential elementary trapping set, because the trapping set's LLR growth rate is limited by  $r < d_v - 1$ . However, we have only guaranteed it in a mean sense.

**Corollary 4.15** *Given a  $(d_v, d_c)$ -regular LDPC code, with  $d_v \geq 3$ , at any  $E_b/N_0$  above the decoding threshold, after a finite number of iterations  $l$ , DE predicts that the SPA decoder's check node output mean  $m_{\lambda(\text{ex})}^{(l)}$  dominates  $r^l$  asymptotically, for any  $r$ , such that  $1 \leq r < d_v - 1$ .*

*Proof:* The condition we want to prove is

$$\frac{m_{\lambda(\text{ex})}^{(l)}}{m_{\lambda(\text{ex})}^{(l-1)}} > r, \quad (4.50)$$

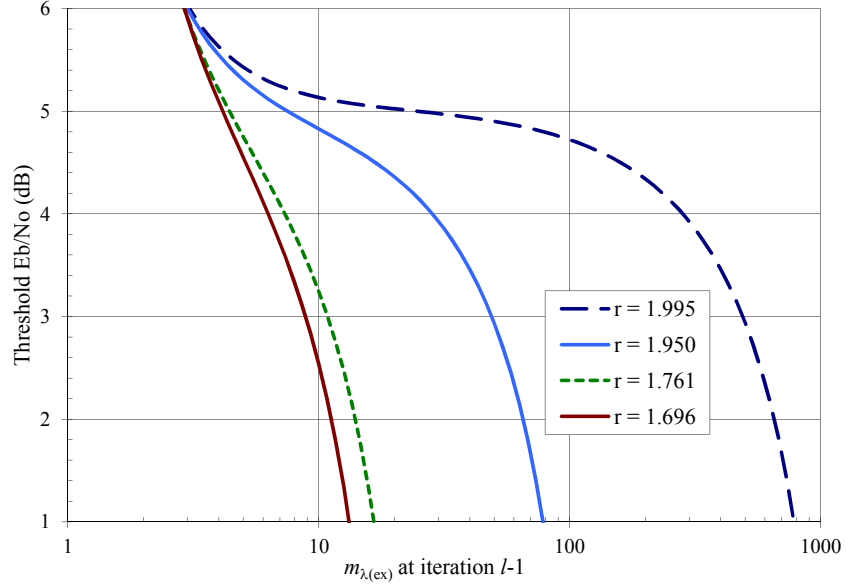


**Figure 4.16:** Threshold  $E_b/N_0$  in dB vs.  $m_{\lambda(\text{ex})}^{(l-1)}$  to achieve the LLR growth needed to overcome a trapping set characterized by  $r$  as specified in the legend for  $(3, 6)$ -regular codes such as Margulis.

which, by (4.47) of the previous proof, will be true if the following stronger condition holds

$$d_v - 1 + \frac{m_\lambda}{m_{\lambda(\text{ex})}^{(l-1)}} - \frac{4}{m_{\lambda(\text{ex})}^{(l-1)}} \frac{\ln \left\{ \frac{d_c - 1}{\delta} \left( 1 - \frac{1}{7x} \right) \right\}}{1 + 2/x} > r, \quad (4.51)$$

where  $x = m_\lambda + (d_v - 1)m_{\lambda(\text{ex})}^{(l-1)}$ . Define  $\epsilon$  such that  $\epsilon \triangleq d_v - 1 - r > 0$ . Rearranging terms, we have the equivalent condition

$$R \frac{E_b}{N_0} > \frac{\ln \left\{ \frac{d_c - 1}{\delta} \left( 1 - \frac{1}{7x} \right) \right\}}{1 + 2/x} - \frac{\epsilon m_{\lambda(\text{ex})}^{(l-1)}}{4}, \quad (4.52)$$

which will be true for some  $l$  given any positive growth in  $m_{\lambda(\text{ex})}^{(l)}$ . ■

Once triggered, the most difficult elementary trapping sets to overcome, that are not codewords, have a spectral radius very close to  $d_v - 1$ , which is the upper bound on spectral radii (see §4.4.5). To approach such an extreme spectral radius, the trapping set would require a very large  $a$  and small  $b$  parameter. The  $E_b/N_0$  required to reach the LLR growth regime of  $m_{\lambda(\text{ex})}^{(i)} > Cr^i$ , which is necessary to overcome the trapping behavior of a specific trapping set, may be significantly less than the threshold of (4.49) as the next example illustrates.

**Example 4.8** Continuing the prior example, we note that the  $(2640, 1320)$  Margulis code is  $(3, 6)$ -regular, with the most troublesome trapping sets reportedly being  $(12, 4)$  and  $(14, 4)$

[2, 3, 12]. We have computed their maximum eigenvalues  $r$  to be 1.6956 and 1.7606, respectively. Fig. 4.16 shows the behavior of the threshold pair  $(E_b/N_0, m_{\lambda(\text{ex})}^{(l-1)})$ , that is required by (4.52) to achieve a growth rate in the mean equivalent to the specific eigenvalues mentioned and two hypothetical extreme eigenvalues of 1.995 and 1.95. Note that this figure suffers some approximation errors at the smallest LLRs shown. From Fig. 4.16 we see that for the (12, 4) trapping set at an  $E_b/N_0$  of 2.8 dB we will require  $m_{\lambda(\text{ex})}^{(l-1)} > 9.3$  to enter the LLR growth regime of  $m_{\lambda(\text{ex})}^{(i)} > C(1.6956)^i$  necessary to eventually overcome that trapping set. This growth rate condition is satisfied at the start of the eighth iteration.

**Theorem 4.16** *Given a variable-regular, but check-irregular  $(d_v, \rho(x))$  LDPC code, with  $d_v \geq 3$ , at sufficiently high  $E_b/N_0$ , after a finite number of iterations, DE predicts that the SPA decoder's check node output mean  $m_{\lambda(\text{ex})}^{(l)}$  dominates  $(d_v - 1)^l$  asymptotically.*

*Proof:* Let  $d_r$  denote the maximum check degree for the graph and let  $\rho(x)$  represent a polynomial with coefficients  $\rho_j, j \in \{2, \dots, d_r\}$ , denoting the fraction of edges belonging to degree- $j$  check nodes, as in (2.4). If we separate  $m_{\lambda(\text{ex})}^{(l)}$  into its components we can reuse our results from Theorem 4.14. Given

$$m_{\lambda(\text{ex})}^{(l)} = \sum_{j=2}^{d_r} \rho_j m_{j, \lambda(\text{ex})}^{(l)}, \quad (4.53)$$

where

$$m_{j, \lambda(\text{ex})}^{(l)} = \phi^{-1} \left( 1 - [1 - \phi(x)]^{j-1} \right), \quad (4.54)$$

$$x = m_\lambda + (d_v - 1)m_{\lambda(\text{ex})}^{(l-1)}, \quad (4.55)$$

and letting  $\delta_j \triangleq 1 - 3/m_{j, \lambda(\text{ex})}^{(l)}$ , (4.47) implies

$$m_{\lambda(\text{ex})}^{(l)} > m_\lambda + (d_v - 1)m_{\lambda(\text{ex})}^{(l-1)} - \sum_{j=2}^{d_r} \rho_j \frac{4 \ln \left\{ \frac{j-1}{\delta_j} \left( 1 - \frac{1}{7x} \right) \right\}}{1 + 2/x}. \quad (4.56)$$

As before, the growth condition is always met with an even stronger condition on the SNR, stated as

$$R \frac{E_b}{N_0} > \sum_{j=2}^{d_r} \rho_j \ln \frac{j-1}{\delta_j}. \quad (4.57)$$

When (4.57) is satisfied, then  $m_{\lambda(\text{ex})}^{(l)} > (d_v - 1)m_{\lambda(\text{ex})}^{(l-1)}$ . ■

**Corollary 4.17** *Given a variable-regular, but check irregular  $(d_v, \rho(x))$  LDPC code, with  $d_v \geq 3$ , at any  $E_b/N_0$  above the decoding threshold, after a finite number of iterations  $l$ , DE predicts that the SPA decoder's check node output mean  $m_{\lambda(\text{ex})}^{(l)}$  dominates  $r^l$  asymptotically, for any  $r$ , such that  $1 \leq r < d_v - 1$ .*

*Proof:* Similar to Corollary 4.15. ■

When combined with the model developed in Section 4.4, the results of this section suggest that every potential (non-codeword) elementary trapping set error can be corrected by a non-saturating LLR-domain SPA decoder after a sufficient number of iterations, provided certain assumptions hold. These assumptions include the extrinsic LLR distribution is approximately Gaussian and the LLR variance  $\sigma_{l(\text{ex})}^2$  grows at a rate no faster than  $m_{\lambda(\text{ex})}^{(l)}$  each iteration  $l$ . These assumptions come from the consistent Gaussian approximation to AWGN DE [43], which assumes that the Tanner graph does not contain cycles as it takes the statistical distributions to be independent among incoming edges. Since useful finite-length LDPC codes *do* contain cycles, it is worth questioning the use of DE to predict error floors.

## Acknowledgement

This chapter is in part a reprint of the papers: Brian K. Butler and Paul H. Siegel, “Error Floor Approximation for LDPC Codes in the AWGN Channel,” submitted to *IEEE Transactions on Information Theory*, 2012 and Brian K. Butler and Paul H. Siegel, “Error Floor Approximation for LDPC Codes in the AWGN Channel,” in *Proceedings of the 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton 2011)*, Monticello, IL, pp. 203–211, Sept. 28–30, 2011. The dissertation author was the primary author of these papers.

## Bibliography

- [1] C. Di, D. Proietti, I. E. Telatar, T. J. Richardson, and R. L. Urbanke, “Finite length analysis of low-density parity-check codes,” *IEEE Trans. Inf. Theory*, vol. 48, no. 6, pp. 1570–1579, Jun. 2002.
- [2] D. J. C. MacKay and M. S. Postol, “Weaknesses of Margulis and Ramanujan–Margulis low-density parity-check codes,” in *Proc. of the 2nd Irish Conf. on the Math. Foundations of Comput. Sci. and Inf. Tech. (MFCSIT)*, ser. Electronic Notes in Theoretical Comput. Sci., vol. 74, Galway, Ireland, 2003.
- [3] T. J. Richardson, “Error-floors of LDPC codes,” in *Proc. of the 41st Annu. Allerton Conf.*, Monticello, IL, Oct. 2003, pp. 1426–1435.

- [4] C.-C. Wang, S. R. Kulkarni, and H. V. Poor, "Finding all small error-prone substructures in LDPC codes," *IEEE Trans. Inf. Theory*, vol. 55, no. 5, pp. 1976–1999, May 2009.
- [5] C. Schlegel and S. Zhang, "On the dynamics of the error floor behavior in (regular) LDPC codes," *IEEE Trans. Inf. Theory*, vol. 56, no. 7, pp. 3248–3264, Jul. 2010.
- [6] S. Abu-Surra, D. DeClercq, D. Divsalar, and W. E. Ryan, "Trapping set enumerators for specific LDPC codes," in *Inf. Theory and Appl. Workshop (ITA)*, San Diego, CA, Feb. 2010.
- [7] X. Zhang and P. H. Siegel, "Efficient algorithms to find all small error-prone substructures in LDPC codes," in *Proc. IEEE Global Telecommun. Conf.*, Houston, TX, Dec. 2011, pp. 1–6.
- [8] M. Karimi and A. H. Banihashemi, "Efficient algorithm for finding dominant trapping sets of LDPC codes," *IEEE Trans. Inf. Theory*, vol. 58, no. 11, pp. 6942–6958, Nov. 2012.
- [9] O. Milenkovic, E. Soljanin, and P. Whiting, "Asymptotic spectra of trapping sets in regular and irregular LDPC code ensembles," *IEEE Trans. Inf. Theory*, vol. 53, no. 1, pp. 39–55, Jan. 2007.
- [10] S. Abu-Surra, D. Divsalar, and W. E. Ryan, "Enumerators for protograph-based ensembles of LDPC and generalized LDPC codes," *IEEE Trans. Inf. Theory*, vol. 57, no. 2, pp. 858–886, Feb. 2011.
- [11] L. Dolecek, Z. Zhang, V. Anantharam, M. J. Wainwright, and B. Nikolić, "Analysis of absorbing sets and fully absorbing sets of array-based LDPC codes," *IEEE Trans. Inf. Theory*, vol. 56, no. 1, pp. 181–201, Jan. 2010.
- [12] Y. Han and W. E. Ryan, "Low-floor decoders for LDPC codes," *IEEE Trans. Commun.*, vol. 57, no. 6, pp. 1663–1673, Jun. 2009.
- [13] S. Laendner and O. Milenkovic, "Algorithmic and combinatorial analysis of trapping sets in structured LDPC codes," in *Proc. Int. Conf. on Wireless Netw., Commun. and Mobile Computing*, Maui, HI, Jun. 2005, pp. 630–635 vol.1.
- [14] Z. Zhang, V. Anantharam, M. J. Wainwright, and B. Nikolić, "An efficient 10GBASE-T Ethernet LDPC decoder design with low error floors," *IEEE J. Solid-State Circuits*, vol. 45, no. 4, pp. 843–855, Apr. 2010.
- [15] A. I. Vila Casado, M. Griot, and R. D. Wesel, "LDPC decoders with informed dynamic scheduling," *IEEE Trans. Commun.*, vol. 58, no. 12, pp. 3470–3479, Dec. 2010.
- [16] E. Sharon, O. Fainzilber, and S. Litsyn, "Decreasing error floor in LDPC codes by parity-check matrix extensions," in *Proc. IEEE Int. Symp. on Inform. Theory*, Seoul, South Korea, Jun.-Jul. 2009, pp. 374–378.
- [17] J. S. Yedidia, Y. Wang, and S. C. Draper, "Divide and concur and difference-map BP decoders for LDPC codes," *IEEE Trans. Inf. Theory*, vol. 57, no. 2, pp. 786–802, Feb. 2011.
- [18] J. Sun, "Studies on graph-based coding systems," Ph.D. dissertation, Dept. Elect. Eng., Ohio State Univ., Columbus, 2004.

- [19] J. Sun, O. Y. Takeshita, and M. P. Fitz, "Analysis of trapping sets for LDPC codes using a linear system model," in *Proc. of the 42nd Annu. Allerton Conf.*, Monticello, IL, Sep.-Oct. 2004, pp. 1701–1702.
- [20] J. Brevik and M. E. O'Sullivan, "The sum-product algorithm for degree-2 check nodes and trapping sets," Oct. 2010, unpublished.
- [21] H. Xiao, A. H. Banihashemi, and M. Karimi, "Error rate estimation of low-density parity-check codes decoded by quantized soft-decision iterative algorithms," *IEEE Trans. Commun.*, vol. 61, no. 2, pp. 474–484, Feb. 2013.
- [22] J. Thorpe, "Low-complexity approximations to belief propagation for LDPC codes," Oct. 2002, unpublished. [Online]. Available: <http://www.systems.caltech.edu/~jeremy/research/papers/>
- [23] J. Chen and M. P. C. Fossorier, "Density evolution for BP-based decoding algorithms of LDPC codes and their quantized versions," in *Proc. IEEE Global Telecommun. Conf.*, vol. 2, Taipei, Taiwan, Nov. 2002, pp. 1378–1382.
- [24] J. Zhao, F. Zarkeshvari, and A. H. Banihashemi, "On implementation of min-sum algorithm and its modifications for decoding low-density parity-check (LDPC) codes," *IEEE Commun. Lett.*, vol. 53, no. 4, pp. 549–554, Apr. 2005.
- [25] Z. Zhang, L. Dolecek, B. Nikolić, V. Anantharam, and M. J. Wainwright, "Design of LDPC decoders for improved low error rate performance: Quantization and algorithm choices," *IEEE Trans. Wireless Commun.*, vol. 8, no. 11, pp. 3258–3258, Nov. 2009.
- [26] S. Zhang and C. Schlegel, "Causes and dynamics of LDPC error floors on AWGN channels," in *Proc. of the 49th Annu. Allerton Conf.*, Monticello, IL, Sep. 2011, pp. 1025–1032.
- [27] —, "Controlling the error floor in LDPC decoding," Sep. 2012, submitted to *IEEE Trans. on Commun.* [Online]. Available: <http://arxiv.org/abs/1209.0532>
- [28] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge, UK: Cambridge Univ. Press, 1985.
- [29] C. D. Meyer, *Matrix Analysis and Applied Linear Algebra*. Philadelphia, PA: SIAM, 2000.
- [30] E. Seneta, *Non-negative Matrices and Markov Chains*. New York: Springer, 1981.
- [31] H. Minc, *Nonnegative Matrices*. New York: John Wiley & Sons, 1988.
- [32] R. J. Trudeau, *Introduction to Graph Theory*. Kent, OH: Kent State Univ. Press, 1976.
- [33] R. A. Brualdi and H. J. Ryser, *Combinatorial Matrix Theory*. Cambridge, UK: Cambridge Univ. Press, 1991.
- [34] D. Cvetković, P. Rowlinson, and S. Simić, *An Introduction to the Theory of Graph Spectra*. Cambridge, UK: Cambridge Univ. Press, 2010.

- [35] T. Etzion, A. Trachtenberg, and A. Vardy, "Which codes have cycle-free Tanner graphs?" *IEEE Trans. Inf. Theory*, vol. 45, no. 6, pp. 2173–2181, Sep. 1999.
- [36] *IEEE Standard for Floating-Point Arithmetic*, IEEE Std. 754-2008, Aug 29, 2008.
- [37] G. Richter, G. Schmidt, M. Bossert, and E. Costa, "Optimization of a reduced-complexity decoding algorithm for LDPC codes by density evolution," in *Proc. IEEE Int. Conf. on Commun.*, vol. 1, Seoul, South Korea, May 2005, pp. 642–646.
- [38] S. Ländner, T. Hehn, O. Milenkovic, and J. B. Huber, "The trapping redundancy of linear block codes," *IEEE Trans. Inf. Theory*, vol. 55, no. 1, pp. 53–63, Jan. 2009.
- [39] C. A. Cole, S. G. Wilson, E. K. Hall, and T. R. Giallorenzi, "A general method for finding low error rates of LDPC codes," May 2006. [Online]. Available: <http://arxiv.org/abs/cs/0605051>
- [40] Y. Zhang and W. E. Ryan, "Toward low LDPC-code floors: A case study," *IEEE Trans. Commun.*, vol. 57, no. 6, pp. 1566–1573, Jun. 2009.
- [41] J. Hamkins, "Performance of low-density parity-check coded modulation," Jet Propulsion Laboratory, Pasadena, CA, Tech. Rep. INP Progress Report 42-184, Feb. 15, 2011.
- [42] R. G. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA: M.I.T. Press, 1963.
- [43] S.-Y. Chung, T. J. Richardson, and R. L. Urbanke, "Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 657–670, Feb. 2001.
- [44] S.-Y. Chung, G. D. Forney, Jr., T. J. Richardson, and R. L. Urbanke, "On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit," *IEEE Commun. Lett.*, vol. 5, no. 2, pp. 58–60, Feb. 2001.
- [45] T. Kailath, *Linear Systems*. Upper Saddle River, NJ: Prentice Hall, 1980.
- [46] M. Fu, "On Gaussian approximation for density evolution of low-density parity-check codes," in *Proc. IEEE Int. Conf. on Commun.*, vol. 3, Istanbul, Turkey, Jun. 2006, pp. 1107–1112.
- [47] L. Dolecek, P. Lee, Z. Zhang, V. Anantharam, B. Nikolić, and M. J. Wainwright, "Predicting error floors of structured LDPC codes: deterministic bounds and estimates," *IEEE J. Sel. Areas Commun.*, vol. 27, no. 6, pp. 908–917, Aug. 2009.
- [48] I. Djurdjevic, J. Xu, K. Abdel-Ghaffar, and S. Lin, "Class of low-density parity-check codes constructed based on Reed-Solomon codes with two information symbols," *IEEE Commun. Lett.*, vol. 7, no. 7, pp. 317–319, Jul. 2003.
- [49] R. M. Tanner, D. Sridhara, A. Sridharan, T. E. Fuja, and D. J. Costello, Jr., "LDPC block and convolutional codes based on circulant matrices," *IEEE Trans. Inf. Theory*, vol. 50, no. 12, pp. 2966–2984, Dec. 2004.

- [50] D. Declercq, L. Danjean, E. Li, S. K. Planjery, and B. Vasić, "Finite alphabet iterative decoding (FAID) of the (155,64,20) Tanner code," in *Proc. 6th Int. Symp. Turbo Codes*, Brest, France, Sep. 2010, pp. 11–15.
- [51] W. E. Ryan and S. Lin, *Channel Codes: Classical and Modern*. Cambridge, UK: Cambridge Univ. Press, 2009.
- [52] B. D. McKay. (2009, Nov. 4) Nauty user's guide (version 2.4). [Online]. Available: <http://cs.anu.edu.au/~bdm/nauty/>
- [53] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 599–618, Feb. 2001.
- [54] N. Wiberg, "Codes and decoding on general graphs," Ph.D. dissertation, Dept. Elect. Eng., Linköping Univ., Linköping, Sweden, Apr. 1996.
- [55] M. Lentmaier, D. V. Truhachev, K. S. Zigangirov, and D. J. Costello, Jr., "An analysis of the block error probability performance of iterative decoding," *IEEE Trans. Inf. Theory*, vol. 51, no. 11, pp. 3834–3855, Nov. 2005.
- [56] R. Koetter and P. O. Vontobel, "On the block error probability of LP decoding of LDPC codes," in *Inf. Theory and Appl. Workshop (ITA)*, San Diego, CA, Feb. 2006.

## Chapter 5

# Bounds on the Minimum Distance of Punctured Quasi-Cyclic LDPC Codes

Recent work by Divsalar *et al.* has shown that properly designed protograph-based low-density parity-check (LDPC) codes typically have minimum (Hamming) distance linearly increasing with block length. This fact rests on ensemble arguments over all possible expansions of the base protograph. However, when implementation complexity is considered, the expansions are frequently selected from a smaller class of structured expansions. For example, protograph expansion by cyclically shifting connections generates a quasi-cyclic (QC) code. Other recent work by Smarandache and Vontobel has provided upper bounds on the minimum distance of QC codes. In this chapter, we generalize these bounds to punctured QC codes and then show how to tighten these for certain classes of codes. We then evaluate these upper bounds for the family of protograph codes known as AR4JA codes that have been recommended for use in deep space communications in a standard established by the Consultative Committee for Space Data Systems. At block lengths larger than 4400 bits, these upper bounds fall well below the ensemble lower bounds.

### 5.1 Introduction

This chapter also concerns the error floor of low-density parity-check (LDPC) codes. The error floor can be attributed, in part, to the existence of certain properties of the Tanner graph that is associated with a chosen parity-check matrix and upon which the decoder operates. Techniques that reduce the occurrence of short cycles in the Tanner graph, for example [1, 2],

have been shown to mitigate the error floor phenomenon. Specifically, the ACE algorithm [3] for placing edges in a graph-based code brings down the error floor substantially by preventing short cycles from clustering around low-degree variable nodes.

Another property of a code that limits its error performance at high SNR is the minimum (Hamming) distance between codewords. The minimum distance is also important in understanding the likelihood of undetected errors, a critical concern in many applications. Yet, relatively little attention has been paid to analyzing the minimum distance of LDPC codes and to developing LDPC code design methodologies that ensure large minimum distance. MacKay and Davey introduced upper bounds on the minimum distance for a class of codes that included quasi-cyclic (QC) LDPC codes in [4]. Notable later work appears in [5, 6]. Of particular relevance to this work are the upper bounds of Smarandache and Vontobel [7] which allow for more variation in the underlying protograph used to represent the code.

Another line of research has shown that most codes in the ensemble of protograph-based codes characterized by a limited number of degree-two variable nodes have minimum distance that increases linearly with block length [8–10]. The family of LDPC codes known as AR4JA codes, recommended for deep-space communications by the Consultative Committee for Space Data Systems (CCSDS) [11], are obtained by puncturing QC-LDPC codes designed from protographs in this ensemble. The selected protographs are expanded to the actual codes (in two stages) using the ACE algorithm to place the edges with QC constraints.

In this chapter, we extend the bounds of [7] to the general class of punctured QC-LDPC codes, and show that these bounds can be tightened in cases where the protomatrices associated with the underlying protograph contain many zero entries. Much of our methodology parallels [7], with our extensions motivated by an interest in bounding the actual minimum distance of the AR4JA codes specified in the CCSDS standard. Somewhat surprisingly, the application of our methodology to the protomatrices underlying the AR4JA constructions for code rates  $1/2$ ,  $2/3$ , and  $4/5$  yields upper bounds of 66, 58, and 56, respectively, independent of the code block length. For large block lengths, these bounds fall well short of the linearly-growing ensemble lower bound mentioned above. Finally, using slight modifications of previously proposed search techniques, we identify specific codewords for each of these code rates at two of the standardized code lengths. The weights of these codewords validate the upper bounds and suggest that the upper bounds may be fairly tight.

The remainder of the chapter is organized as follows. Section 5.2 provides a review of protograph-based LDPC code design, as well as the specific family of AR4JA codes. Sec-

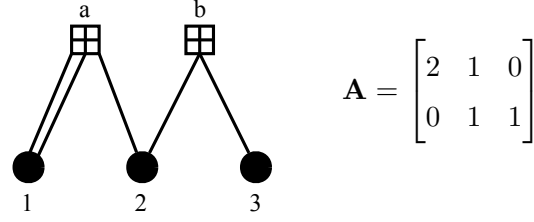


**Figure 5.1:** The protograph  $G$  and the corresponding protomatrix  $\mathbf{A}$ .

tion 5.3 provides the necessary mathematical background on the polynomial representation and properties of QC-LDPC codes obtained by the expansion of protographs in which the expansion is based on circulant matrices. In Section 5.4, we review the upper bounds on the minimum-distance of QC-LDPC codes in [7], and then develop the necessary algebraic results to generalize these bounds to punctured QC-LDPC codes. Section 5.5 describes techniques that can produce tighter upper bounds for protographs with specific properties, including some of the AR4JA protographs. In Section 5.6, we apply our methods to calculate upper bounds on the minimum distance of the codes in the CCSDS standard, which are obtained by a two-step QC expansion of the AR4JA protographs. In Section 5.7, we use computer search to find low-weight codewords for several AR4JA codes, compare them to our length-independent bounds, and reconcile them with the ensemble minimum-distance lower bounds [8–10]. We also examine the girth of AR4JA codes. Section 5.8 concludes the chapter.

## 5.2 Protographs and the AR4JA Family of Codes

Protographs were introduced as a way to impart structure to the interconnectivity of graph-based codes [12]. Protographs themselves are a subset of multi-edge type graphs [13, ch. 7].

A *protograph* is essentially a Tanner graph with a relatively small number of nodes. More specifically, a protograph,  $G = (V, C, E)$ , consists of a set of variable nodes  $V$ , a set of check nodes  $C$ , and a collection of edges  $E$ . Each edge,  $e \in E$ , connects a variable node,  $v_e \in V$ , to a check node,  $c_e \in C$ . Protographs have the additional property that parallel edges are permitted. Moreover, variable nodes in  $V$  can be designated as punctured; *i.e.*, the corresponding bits are not included in the transmitted codeword.

A simple protograph  $G$  with three variable nodes, two check nodes, and five edges is shown in Fig. 5.1. The accompanying *protomatrix*  $\mathbf{A}$  fully describes the protograph structure. The entry in the  $j$ th row and  $i$ th column of the protomatrix  $\mathbf{A}$  indicates the number of edges

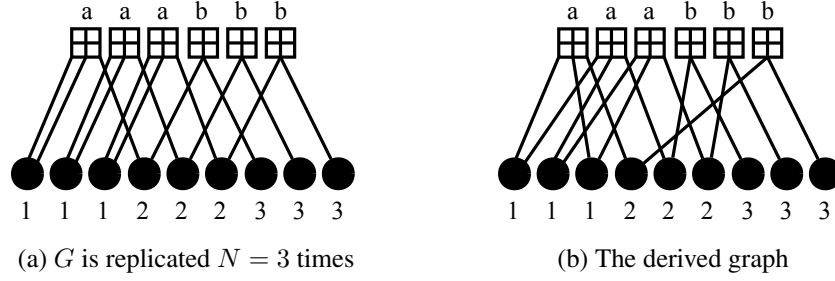


**Figure 5.2:** To obtain the derived graph, the protograph  $G$  is replicated  $N = 3$  times and the edge connections within each edge set are permuted.

connecting the  $j$ th check node to the  $i$ th variable node within the corresponding protograph.

A *derived graph* is constructed by replicating the protograph a specified number of times and interconnecting the copies of the variable and check nodes in a manner consistent with the topology of  $G$ . In this example, all copies of check node  $a$  are called “type  $a$ ” check nodes. Similarly, all copies of variable node 1 are called “type 1” variable nodes. The replicas of an edge connecting check node  $a$  and variable node 1 form a so-called edge set, and their connected check nodes may be permuted within the set of “type  $a$ ” check nodes. (The term “type,” when used in Section 5.6 to classify matrices as in [7], is unrelated.) The corresponding linear code is referred to as a *protograph code*.

The application of this interconnection procedure to all replicas ensures that node degrees and connectivity by node types of the original protograph are maintained in the resulting derived graph. The corresponding iterative decoder implementation may, in fact, be less complex than that of “random” LDPC codes because of the structure imposed on the node interconnections.

Figs. 5.2a and 5.2b illustrate the process of making  $N = 3$  copies of the protograph of Fig. 5.1 and interconnecting them to generate the derived graph. The parity-check matrix corresponding to the derived graph of Fig. 5.2b is shown below, divided into submatrices so that the relationship to the protomatrix  $\mathbf{A}$  of Fig. 5.1 is evident:

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

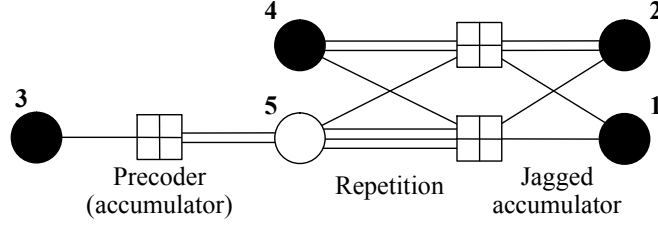


**Figure 5.3:** AR4JA protograph, rate-1/2.

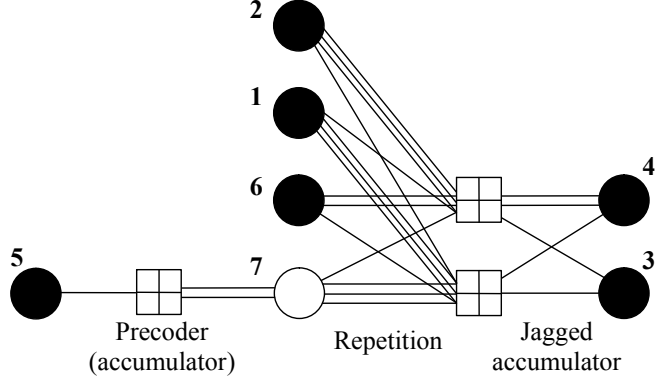


**Figure 5.4:** AR4JA protograph, rate-2/3.

Protograph-based code design allows for the introduction of degree-one variable nodes and punctured variable nodes in a structured way. With regard to degree-one nodes, recall that the optimization of irregular LDPC codes by density evolution typically avoids degree-one variable nodes. This is because they induce an error rate floor in randomly constructed codes, even as the block length grows toward infinity [13, p. 161]. However, density evolution often produces a significant fraction of degree-two variable nodes. This suggests that the incorporation of degree-one variable nodes may offer a potential benefit in code performance, as noted in [13, p. 382].

The protograph for the rate  $r = 1/2$  AR4JA code [8] is shown in Fig. 5.3. We follow the convention of representing the transmitted variable nodes as solid circles and the punctured variable nodes as unfilled circles. The protograph of the rate-1/2 code is extended to rate-2/3 by adding two degree-four variable nodes as shown in Fig. 5.4. The corresponding protomatrices are

$$\mathbf{A}_{r=1/2} = \begin{bmatrix} 0 & 0 & 1 & 0 & 2 \\ 1 & 1 & 0 & 1 & 3 \\ 1 & 2 & 0 & 2 & 1 \end{bmatrix} \quad (5.1)$$

and

$$\mathbf{A}_{r=2/3} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 2 \\ 3 & 1 & 1 & 1 & 0 & 1 & 3 \\ 1 & 3 & 1 & 2 & 0 & 2 & 1 \end{bmatrix}, \quad (5.2)$$

respectively. The numerical labels of the variable nodes in the figures correspond to the columns in the protomatrices, enumerated from left to right, and the check nodes correspond to the rows of the protomatrix.

The AR4JA family of protographs further extends the code rate options by adding an additional four degree-4 variable nodes. The rate-4/5 protograph has 11 variable nodes altogether, as can be seen from its protomatrix,

$$\mathbf{A}_{r=4/5} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 2 \\ 3 & 1 & 3 & 1 & 3 & 1 & 1 & 1 & 0 & 1 & 3 \\ 1 & 3 & 1 & 3 & 1 & 3 & 1 & 2 & 0 & 2 & 1 \end{bmatrix}. \quad (5.3)$$

For all of these rate options, the variable nodes in the derived graph that correspond to copies of the degree-6 variable node in the protograph (equivalently, the right-most column of the protomatrix) are punctured.

The name *AR4JA* is derived from the operations reflected in the protograph structure and indicated in Figs. 5.3 and 5.4. As can be seen, the protograph embodies several features similar to those of an Accumulate-Repeat-Accumulate (ARA) code. For the AR4JA code construction, a partial precoding by accumulation (A) is followed by “repetition 4 times” (R4), culminating in a “jagged” accumulation (JA). The jagged accumulation differs from a standard accumulation, which contains degree-two variable nodes only, by the additional edge at the upper right of the protograph. In fact, the switch from a standard accumulation stage to the jagged accumulation stage, which reduces the number of degree-2 variable nodes, allows the AR4JA protographs to meet the criterion of the ensemble of protographs with linearly increasing minimum distance. More specifically, the techniques of Divsalar and other researchers [8–10, 14] can be used to calculate the asymptotic ensemble weight enumerators for protograph-based codes, from which the *typical relative minimum distance*  $\delta_{\min}$  can be found. They prove that the minimum distance  $d_{\min}$  of most of the codes in the ensemble derived from the protograph exceeds  $\delta_{\min}n$ , where  $n$  is the block length of the code. For the rate-1/2 AR4JA protomatrix in (5.1),  $\delta_{\min} = 0.015$  [8].

### 5.3 QC Expansion and Polynomial Representation

The codewords of a block code may be divided into non-overlapping *subblocks* of  $N$  consecutive symbols. A *quasi-cyclic* (QC) code is a linear block code having the property that applying identical circular shifts to every subblock of a codeword yields a codeword. QC codes are a generalization of conventional cyclic block codes and are simple to encode [15, §8.14].

A binary QC-LDPC code of length  $n = LN$  can be described by an  $m \times n$  sparse parity-check matrix  $\mathbf{H} \in \mathbb{F}_2^{m \times n}$ , with  $m = JN$ , which is composed of  $N \times N$  circulant submatrices. A right *circulant matrix* is a square matrix with each successive row right-shifted circularly one position relative to the row above. Therefore, circulant matrices can be completely described by a single row or column. As in [7], we use the description corresponding to the left-most column.

A binary QC-LDPC code can also be described in polynomial form, since there exists an isomorphism between the commutative ring of  $N \times N$  circulant binary matrices and the commutative ring of binary polynomials modulo  $x^N - 1$ , i.e.,  $\mathbb{F}_2[x]/\langle x^N - 1 \rangle$ . Addition and multiplication in the latter ring correspond to, respectively, addition and multiplication of polynomials in  $\mathbb{F}_2[x]$ , modulo  $x^N - 1$ .

The isomorphism between  $N \times N$  binary circulant matrices and polynomial residues in the quotient ring maps a matrix to the polynomial in which the coefficients in order of increasing degree correspond to the entries in the left-most matrix column taken from top to bottom. Under this isomorphism, the  $N \times N$  identity matrix maps to the multiplicative identity in the polynomial quotient ring, namely 1. A few examples of the mapping (indicated by  $\mapsto$ ) for  $N = 3$  are shown below:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \mapsto 1 \quad \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \mapsto x \quad \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} \mapsto 1 + x^2.$$

This isomorphism requires that care be taken when representing multiplication of a circulant matrix  $\mathbf{M}$  by a binary vector  $\mathbf{v} = (v_0, v_1, \dots, v_{N-1})$ . If we associate the polynomial  $M(x)$  with the matrix  $\mathbf{M}$  under the isomorphism just described, and let  $v(x) = v_0 + v_1x + \dots + v_{N-1}x^{N-1}$  represent the vector  $\mathbf{v}$ , then the product  $(\mathbf{M}\mathbf{v}^T)^T = \mathbf{v}\mathbf{M}^T$  maps to the polynomial  $M(x)v(x)$  modulo  $x^N - 1$ , and the product  $\mathbf{v}\mathbf{M}$  maps to the polynomial  $x^N M(x^{-1})v(x)$  modulo  $x^N - 1$ . Note that from this paragraph onward, all indexing begins at zero and all vectors are row vectors.

Given a polynomial residue  $a(x) \in \mathbb{F}_2[x]/\langle x^N - 1 \rangle$ , we define its weight  $\text{wt}(a(x)) \in \mathbb{Z}$

to be the number of nonzero coefficients. Thus, the weight  $\text{wt}(a(x))$  of the polynomial  $a(x)$  is equal to the Hamming weight  $w_H(\mathbf{a})$  of the corresponding binary vector of coefficients  $\mathbf{a}$ . For a length- $L$  vector of elements in the ring,  $\mathbf{a}(x) = (a_0(x), a_1(x), \dots, a_{L-1}(x))$ , we define its Hamming weight to be the sum of the weights of its components, *i.e.*,  $w_H(\mathbf{a}(x)) = \sum_{i=0}^{L-1} \text{wt}(a_i(x))$ . Throughout this work, computations implicitly shift to integer arithmetic upon taking the weight.

In using a ring, there are a few important points to bear in mind. The elements in a ring  $R$  need not have a multiplicative inverse; the ones that do are called *units*. The ring may include zero divisors, where a *zero divisor* (or *factor of zero*) is a nonzero element  $a \in R$ , such that  $ab = 0$  for some  $b \in R$ ,  $b \neq 0$ . For example, the elements 2 and 3 in  $R = \mathbb{Z}/6\mathbb{Z}$ , the ring of integers modulo 6, are zero divisors since  $2 \cdot 3 = 0$  modulo 6. Note that units cannot be zero divisors. This follows by noting that if  $a^{-1}$  exists and  $ab = 0$  for  $b \neq 0$ , then  $a^{-1}(ab) = a^{-1} \cdot 0$  simplifies to  $b = 0$ , a contradiction. Moreover, all *finite* rings, such as  $\mathbb{Z}/6\mathbb{Z}$  and  $\mathbb{F}_2[x]/\langle x^N - 1 \rangle$ , have the following important property.

**Lemma 5.1** *Every nonzero element in a finite ring with unity is either a unit or a zero divisor [16, p. 205].*

*Proof:* Let  $R$  be a ring with elements  $\{0, 1, r_1, r_2, \dots, r_n\}$ . Unity, denoted by 1, is always a unit. Take any  $r_i$  and note that the set of all products between  $r_i$  and the elements of  $R$ . If the products are all distinct, then  $r_i$  is a unit since one of the products equals 1. If they are not all distinct, then  $r_i r_j = r_i r_k$  implies that  $r_i(r_j - r_k) = 0$ , where  $r_j \neq r_k$ . In this case,  $r_i$  and  $(r_j - r_k)$  are zero divisors. ■

The elements of weight one in the polynomial quotient ring  $\mathbb{F}_2[x]/\langle x^N - 1 \rangle$  are the monomials, all of which are units in the ring. Specifically, the inverse of the monomial  $a(x) = x^i$ ,  $0 \leq i < N$ , is the monomial  $(a(x))^{-1} = x^{(N-i) \bmod N}$ . Under the isomorphism defined above, the monomials in the ring correspond to *cyclic permutation matrices*, which are binary circulant matrices with a single one in each row and each column.

For any  $N > 0$ , the nonzero elements in the ring  $\mathbb{F}_2[x]/\langle x^N - 1 \rangle$  that have even weight are zero divisors. For instance, the product of the polynomial  $x^{N-1} + x^{N-2} + \dots + x + 1$  and any even weight polynomial is zero. Odd weight polynomials, on the other hand, may sometimes be zero divisors, such as  $x^3 + x + 1$  in the ring  $\mathbb{F}_2[x]/\langle x^7 - 1 \rangle$ .

As we are interested in the connection between protographs and QC-LDPC codes, we

focus on parity-check matrices  $\mathbf{H}$  that are in  $J \times L$  block matrix form, that is

$$\mathbf{H} \triangleq \begin{bmatrix} \mathbf{H}_{0,0} & \cdots & \mathbf{H}_{0,L-1} \\ \vdots & \ddots & \vdots \\ \mathbf{H}_{J-1,0} & \cdots & \mathbf{H}_{J-1,L-1} \end{bmatrix},$$

where each submatrix  $\mathbf{H}_{j,i}$  is an  $N \times N$  binary circulant matrix. Let  $h_{j,i,s} \in \mathbb{F}_2$  be the left-most entry in the  $s$ th row of the submatrix  $\mathbf{H}_{j,i}$ . We can then write  $\mathbf{H}_{j,i} = \sum_{s=0}^{N-1} h_{j,i,s} \mathbf{I}_s$ , where  $\mathbf{I}_s$  is the  $N \times N$  identity matrix circularly left-shifted by  $s$  positions. Now, using the same convention as above for identifying matrices with polynomial residues, we can associate with  $\mathbf{H}$  the *polynomial parity-check matrix*  $\mathbf{H}(x)$ , where  $\mathbf{H}(x) \in (\mathbb{F}_2[x]/\langle x^N - 1 \rangle)^{J \times L}$ ,

$$\mathbf{H}(x) \triangleq \begin{bmatrix} h_{0,0}(x) & \cdots & h_{0,L-1}(x) \\ \vdots & \ddots & \vdots \\ h_{J-1,0}(x) & \cdots & h_{J-1,L-1}(x) \end{bmatrix},$$

and  $h_{j,i}(x) \triangleq \sum_{s=0}^{N-1} h_{j,i,s} x^s$ .

We will be interested in the weight of each polynomial entry of  $\mathbf{H}(x)$ , or, equivalently, the row or column sum of each submatrix of  $\mathbf{H}$ . The *weight matrix* of  $\mathbf{H}(x)$ , which is a  $J \times L$  matrix of nonnegative integers, is defined as

$$\text{wt}(\mathbf{H}(x)) \triangleq \begin{bmatrix} \text{wt}(h_{0,0}(x)) & \cdots & \text{wt}(h_{0,L-1}(x)) \\ \vdots & \ddots & \vdots \\ \text{wt}(h_{J-1,0}(x)) & \cdots & \text{wt}(h_{J-1,L-1}(x)) \end{bmatrix}.$$

Note that for a protograph-based QC-LDPC code, the weight matrix of the associated polynomial parity-check matrix  $\text{wt}(\mathbf{H}(x))$  is precisely the corresponding protomatrix,  $\mathbf{A} = \text{wt}(\mathbf{H}(x))$ . It is also convenient to represent codewords of QC codes in polynomial form. In particular, the set of codewords, which is the set of vectors  $\mathbf{c}$  such that  $\mathbf{H}\mathbf{c}^T = \mathbf{0}^T$  over  $\mathbb{F}_2$ , maps to the set of polynomial vectors  $\mathbf{c}(x) \in (\mathbb{F}_2[x]/\langle x^N - 1 \rangle)^L$  satisfying  $\mathbf{H}(x)\mathbf{c}(x)^T = \mathbf{0}^T$ . Under this identification, the entries  $c_i(x)$  of the polynomial vector  $\mathbf{c}(x) = (c_0(x), c_1(x), \dots, c_{L-1}(x))$  correspond to the length- $N$  subblocks of the codeword  $\mathbf{c}$  that were defined at the start of this section.

## 5.4 Minimum Distance Bounds for QC Codes

In this section we review the upper bounds on the minimum (Hamming) distance of QC-LDPC codes that were established in [7] and then extend them to punctured QC-LDPC codes.

### 5.4.1 Upper Bounds for QC Codes

We will use the shorthand notation  $[L]$  to indicate the set of  $L$  consecutive integers,  $\{0, 1, 2, \dots, L-1\}$ . We also let  $\mathcal{S} \setminus i$  denote all the elements of  $\mathcal{S}$ , excluding the element  $i$ . We denote by  $\mathbf{A}_{\mathcal{S}}$  the submatrix of  $\mathbf{A}$  containing the columns indicated by the index set  $\mathcal{S}$ . Similarly,  $\mathbf{a}_{\mathcal{S}}$  denotes the subvector containing the elements of the vector  $\mathbf{a}$  indicated by the index set  $\mathcal{S}$ .

The permanent of a  $J \times J$  matrix  $\mathbf{B} = [b_{j,i}]$  over a ring is defined to be

$$\text{perm}(\mathbf{B}) \triangleq \sum_{\sigma} \prod_{j \in [J]} b_{j, \sigma(j)},$$

where the summation is over all  $J!$  permutations  $\sigma$  of the set  $[J]$ , and  $\sigma(j)$  is the  $j$ th entry of the permuted set  $\sigma([J])$ . The definition of the permanent resembles that of the determinant of a square matrix;

$$\det(\mathbf{B}) \triangleq \sum_{\sigma} \text{sign}(\sigma) \prod_{j \in [J]} b_{j, \sigma(j)},$$

where  $\text{sign}(\sigma)$  equals  $+1$  if  $\sigma$  is an even permutation and  $-1$  if  $\sigma$  is an odd permutation. (Recall that an even permutation is obtained by applying an even number of transpositions of pairs of elements to the sequence  $\{0, 1, 2, \dots, J-1\}$ .) When the elements of  $\mathbf{B}$  belong to a ring of characteristic two, where addition and subtraction are interchangeable,  $\text{perm}(\mathbf{B}) = \det(\mathbf{B})$ . Like the determinant, the permanent may be computed recursively by taking the cofactor expansion along any row or column. That is, for any  $j \in [J]$ , the cofactor expansion of the permanent of matrix  $\mathbf{B}$  along the  $j$ th row is

$$\text{perm}(\mathbf{B}) = \sum_{i \in [J]} b_{j,i} \cdot \text{perm}(\mathbf{B}'_{[J] \setminus i}), \quad (5.4)$$

with  $\mathbf{B}'$  denoting the submatrix of  $\mathbf{B}$  with the  $j$ th row removed. (Note that the subscript  $[J] \setminus i$  on  $\mathbf{B}'$  in (5.4) removes the  $i$ th column.)

In the derivation of upper bounds on the minimum distance, we will make use of the notion of dependence of vectors over a commutative ring  $R$ . This will require an appeal to some special properties of the ring of matrices over a finite commutative ring with unity. We remark that, in [7], an alternative approach that exploits the connection between QC block codes and convolutional codes was used in the derivation of the upper bounds on the minimum distance.

Let  $\mathcal{U}$  be a set of vectors over  $R$ ; that is,  $\mathcal{U} = \{\mathbf{u}_0, \dots, \mathbf{u}_{n-1}\}$ , where  $\mathbf{u}_i \in R^N$ . The set  $\mathcal{U}$  is said to be *dependent*<sup>1</sup> if there exist elements  $r_0, \dots, r_{n-1} \in R$ , not all zero, such that the

---

<sup>1</sup>Some authors use the term “linearly dependent” in this context.

linear combination

$$r_0 \mathbf{u}_0 + \cdots + r_{n-1} \mathbf{u}_{n-1} = (0, \dots, 0). \quad (5.5)$$

If no such set of elements exists, the set  $\mathcal{U}$  is said to be *independent*. (See, for example, [17, p. 454].) We note that this independence test may be applied to the set of row vectors of a matrix with elements in  $R$ . A *dependent row* is any row in (5.5) which is multiplied by a nonzero scalar.

**Lemma 5.2** *Let  $\mathbf{B}$  be a square matrix over a finite commutative ring with unity. Then  $\det(\mathbf{B})$  equals zero or is a zero divisor if and only if the set of row vectors of  $\mathbf{B}$  is dependent.*

*Proof:* The proof of Lemma 5.2 can be found in the appendix, along with some illustrative examples. ■

We now review a technique from [7] for explicitly constructing codewords of a QC code specified by a polynomial parity-check matrix.

**Lemma 5.3 (Lemma 6 [7])** *Let  $\mathcal{C}$  be a QC code with polynomial parity-check matrix  $\mathbf{H}(x) \in (\mathbb{F}_2[x]/\langle x^N - 1 \rangle)^{J \times L}$ . Let  $\mathcal{S}$  be an arbitrary size- $(J+1)$  subset of  $[L]$  and let  $\mathbf{c}(x)$  be a length- $L$  vector whose elements are in the ring  $\mathbb{F}_2[x]/\langle x^N - 1 \rangle$  and are given by*

$$c_i(x) \triangleq \begin{cases} \text{perm}(\mathbf{H}_{\mathcal{S} \setminus i}(x)) & \text{if } i \in \mathcal{S} \\ 0 & \text{otherwise.} \end{cases}$$

*Then  $\mathbf{c}(x)$  is a codeword in  $\mathcal{C}$ .*

*Proof:* For any  $j \in [J]$ , let the  $j$ th row of  $\mathbf{H}(x)$  be  $\mathbf{h}_j(x)$ . Then,

$$\begin{aligned} \mathbf{h}_j(x) \mathbf{c}(x)^T &= \sum_{i \in [L]} h_{j,i}(x) \cdot c_i(x) \\ &= \sum_{i \in \mathcal{S}} h_{j,i}(x) \text{perm}(\mathbf{H}_{\mathcal{S} \setminus i}(x)) \end{aligned} \quad (5.6)$$

$$= \text{perm} \begin{bmatrix} \mathbf{h}_{j,\mathcal{S}}(x) \\ \mathbf{H}_{\mathcal{S}}(x) \end{bmatrix} \quad (5.7)$$

$$= \det \begin{bmatrix} \mathbf{h}_{j,\mathcal{S}}(x) \\ \mathbf{H}_{\mathcal{S}}(x) \end{bmatrix} = 0, \quad (5.8)$$

where computations are in the ring  $\mathbb{F}_2[x]/\langle x^N - 1 \rangle$ . The cofactor expansion of (5.7) is (5.6). As the elements belong to a ring of characteristic two, the permanent in (5.7) equals the determinant in (5.8). The determinant shown must be zero as it contains a repeated row. Since every row of

$\mathbf{H}(x)$  has zero inner product with  $\mathbf{c}(x)$ , we conclude that  $\mathbf{H}(x)\mathbf{c}(x)^T = \mathbf{0}^T$ . Therefore,  $\mathbf{c}(x)$  is a codeword in  $\mathcal{C}$ . ■

While the  $\min$  function applied to a collection of nonnegative real numbers returns the minimum, we will require a variant of this function, denoted  $\min^*$ , defined as follows. For a finite collection of nonnegative real numbers  $\mathcal{R}$ , let  $\mathcal{R}^+ \subseteq \mathcal{R}$  be the subset of positive elements of  $\mathcal{R}$ . We define

$$\min^* \mathcal{R} \triangleq \begin{cases} \min \mathcal{R}^+ & \text{if } \mathcal{R}^+ \neq \emptyset \\ +\infty & \text{if } \mathcal{R}^+ = \emptyset. \end{cases}$$

The minimum distance of the QC code  $\mathcal{C}$  can then be written as

$$d_{\min}(\mathcal{C}) = \min_{\mathbf{c}(x) \in \mathcal{C}}^* w_H(\mathbf{c}(x)), \quad (5.9)$$

where we have used  $\min^*$  to exclude the all-zero codeword.

We now develop two upper bounds on the minimum distance. The first, based upon [7], uses Lemma 5.3 to produce low-weight codewords from the polynomial parity-check matrix of the code. We will generate as many codewords as possible and apply (5.9) to achieve an upper bound on the minimum distance.

**Theorem 5.4 (Theorem 7 [7])** *Let  $\mathcal{C}$  be a QC code with the polynomial parity-check matrix  $\mathbf{H}(x) \in (\mathbb{F}_2[x]/\langle x^N - 1 \rangle)^{J \times L}$ . Then the minimum distance of  $\mathcal{C}$  satisfies the upper bound*

$$d_{\min}(\mathcal{C}) \leq \min_{\substack{\mathcal{S} \subseteq [L] \\ |\mathcal{S}|=J+1}}^* \sum_{i \in \mathcal{S}} \text{wt}(\text{perm}(\mathbf{H}_{\mathcal{S} \setminus i}(x))). \quad (5.10)$$

*Proof:* Let  $\mathcal{S}$  be a subset of  $[L]$  of size- $(J + 1)$  and apply Lemma 5.3 to construct a codeword  $\mathbf{c}(x)$  in  $\mathcal{C}$ . The weight of  $\mathbf{c}(x)$  is

$$\begin{aligned} w_H(\mathbf{c}(x)) &= \sum_{i \in [L]} \text{wt}(c_i(x)) \\ &= \sum_{i \in \mathcal{S}} \text{wt}(\text{perm}(\mathbf{H}_{\mathcal{S} \setminus i}(x))). \end{aligned} \quad (5.11)$$

The upper bound (5.10) follows immediately by combining (5.9) and (5.11) and noting that, in general, only a strict subset of the codewords can be generated by Lemma 5.3. We must use the  $\min^*$  function because, for some choices of the set  $\mathcal{S}$ , the construction in Lemma 5.3 will yield the all-zero codeword, and we have to exclude those sets from the calculation of the upper bound. ■

The second upper bound on the minimum distance makes use of an upper bound on the weight of the permanent of a matrix over the ring  $\mathbb{F}_2[x]/\langle x^N - 1 \rangle$ , as described in the following lemma.

**Lemma 5.5** *Let  $\mathbf{B}(x)$  be a  $J \times J$  matrix with elements  $b_{j,i}(x)$  in the ring  $\mathbb{F}_2[x]/\langle x^N - 1 \rangle$ . Then the weight of the permanent of  $\mathbf{B}$  satisfies the upper bound*

$$\text{wt}(\text{perm}(\mathbf{B}(x))) \leq \text{perm}(\text{wt}(\mathbf{B}(x))).$$

*Proof:* Let the polynomials  $a(x)$  and  $b(x)$  be in the ring  $\mathbb{F}_2[x]/\langle x^N - 1 \rangle$ . We know that  $\text{wt}[a(x) + b(x)] \leq \text{wt}(a(x)) + \text{wt}(b(x))$ , as the maximum number of nonzero coefficients in the sum  $a(x) + b(x)$  is  $\text{wt}(a(x)) + \text{wt}(b(x))$ .

Similarly, we know that  $\text{wt}[a(x) \cdot b(x)] \leq \text{wt}(a(x)) \cdot \text{wt}(b(x))$ , as the maximum number of nonzero coefficients in the product  $a(x) \cdot b(x)$  is  $\text{wt}(a(x)) \cdot \text{wt}(b(x))$ . Therefore,

$$\begin{aligned} \text{wt}(\text{perm}(\mathbf{B}(x))) &= \text{wt} \left[ \sum_{\sigma} \prod_{j \in [J]} b_{j, \sigma(j)}(x) \right] \\ &\leq \sum_{\sigma} \text{wt} \left[ \prod_{j \in [J]} b_{j, \sigma(j)}(x) \right] \\ &\leq \sum_{\sigma} \prod_{j \in [J]} \text{wt}(b_{j, \sigma(j)}(x)) \\ &= \text{perm}(\text{wt}(\mathbf{B}(x))). \end{aligned}$$

■

The second upper bound on the minimum distance, described in the next theorem, is expressed in terms of the weight matrix of the code.

**Theorem 5.6 (Theorem 8 in [7])** *Let  $\mathcal{C}$  be a QC code with polynomial parity-check matrix  $\mathbf{H}(x) \in (\mathbb{F}_2[x]/\langle x^N - 1 \rangle)^{J \times L}$  and let  $\mathbf{A} \triangleq \text{wt}(\mathbf{H}(x))$ . Then the minimum distance of  $\mathcal{C}$  satisfies the upper bound*

$$d_{\min}(\mathcal{C}) \leq \min_{\substack{\mathcal{S} \subseteq [L] \\ |\mathcal{S}|=J+1}}^* \sum_{i \in \mathcal{S}} \text{perm}(\mathbf{A}_{\mathcal{S} \setminus i}) \quad (\text{in } \mathbb{Z}). \quad (5.12)$$

*Proof:* The proof follows from Theorem 5.4 and Lemma 5.5. The only subtlety arises from the fact that the sets  $\mathcal{S}$  excluded from (5.10) by the  $\min^*$  function may not be excluded from (5.12) by the application of  $\min^*$ , i.e., there might be sets  $\mathcal{S}$  such that  $\sum_{i \in \mathcal{S}} \text{wt}(\text{perm}(\mathbf{H}_{\mathcal{S} \setminus i}(x))) = 0$  but  $\sum_{i \in \mathcal{S}} \text{perm}(\mathbf{A}_{\mathcal{S} \setminus i}) > 0$ . The resolution of this potential complication can be achieved by reference to Theorem 8 in [7]. ■

**Remark 5.1** Alternatively, we may prove the correctness of this upper bound by following the arguments presented in the proof of Theorem 5.10, below. Since Theorem 5.10 includes puncturing, consider the set  $\mathcal{P}$  to be empty for this case.

### 5.4.2 Upper Bounds for Punctured QC Codes

We now extend the preceding upper bounds on the minimum distance to the class of punctured QC codes. The puncturing strategy of the AR4JA codes is prompted by the addition of the precoder to the underlying protographs, a modification that generally improves the decoding threshold [8, 9]. Note that puncturing whole subblocks of the polynomial codeword  $\mathbf{c}(x)$  preserves quasi-cyclicity. The set  $\mathcal{P} \subset [L]$  indexes the subblocks of the polynomial codeword  $\mathbf{c}(x)$  which are not transmitted. Indices of  $\mathcal{P}$  may also be associated with columns of the  $J \times L$  polynomial parity-check matrix  $\mathbf{H}(x)$ .

We begin with a QC code  $\mathcal{C}$  based upon  $\mathbf{H}(x)$ . Next, we define a new QC code  $\mathcal{C}'$  by puncturing the components of  $\mathbf{c}(x)$  that are indexed by  $\mathcal{P}$ . We mark the subblocks to be punctured with the symbol “ $\varphi$ ” as re-indexing would introduce unnecessary notational complexity, and we define  $\text{wt}(\varphi) = 0$ , since the punctured symbols are not transmitted.

**Lemma 5.7** *Let  $\mathcal{C}'$  be a punctured QC code constructed by puncturing subblocks of the QC code  $\mathcal{C}$ , defined by the polynomial parity-check matrix  $\mathbf{H}(x) \in (\mathbb{F}_2[x]/\langle x^N - 1 \rangle)^{J \times L}$ . Let the subblocks of  $\mathcal{C}$  indexed by the set  $\mathcal{P}$ ,  $\mathcal{P} \subset [L]$ , be punctured. Let  $\mathcal{S}$  be an arbitrary size- $(J + 1)$  subset of  $[L]$ . Let the length- $L$  vector  $\mathbf{c}'(x) = (c'_0(x), c'_1(x), \dots, c'_{L-1}(x))$ , with  $c'_i(x) \in \mathbb{F}_2[x]/\langle x^N - 1 \rangle \cup \{\varphi\}$ , be defined by*

$$c'_i(x) \triangleq \begin{cases} \text{perm}(\mathbf{H}_{\mathcal{S} \setminus i}(x)) & \text{if } i \in \mathcal{S} \setminus \mathcal{P} \\ \varphi & \text{if } i \in \mathcal{P} \\ 0 & \text{otherwise.} \end{cases}$$

*Then  $\mathbf{c}'(x)$  is a codeword of the punctured code  $\mathcal{C}'$ .*

*Proof:* This follows by noting that  $\mathbf{c}'(x)$  is obtained by puncturing the subblocks indexed by  $\mathcal{P}$  from the codeword  $\mathbf{c}(x)$  of Lemma 5.3. ■

**Theorem 5.8** *Let  $\mathcal{C}'$  be a punctured QC code constructed by puncturing subblocks of the QC code  $\mathcal{C}$  with polynomial parity-check matrix  $\mathbf{H}(x) \in (\mathbb{F}_2[x]/\langle x^N - 1 \rangle)^{J \times L}$ . Let the subblocks of*

$\mathcal{C}$  indexed by the set  $\mathcal{P}$ ,  $\mathcal{P} \subset [L]$ , be punctured. Then

$$d_{\min}(\mathcal{C}') \leq \min_{\substack{\mathcal{S} \subseteq [L] \\ |\mathcal{S}|=J+1}}^* \sum_{i \in \mathcal{S} \setminus \mathcal{P}} \text{wt}(\text{perm}(\mathbf{H}_{\mathcal{S} \setminus i}(x))). \quad (5.13)$$

*Proof:* Let  $\mathcal{S}$  be a subset of  $[L]$  of size- $(J+1)$ , and apply Lemma 5.7 to construct a codeword  $\mathbf{c}'(x)$  in  $\mathcal{C}'$ . The weight of this codeword is

$$\begin{aligned} w_H(\mathbf{c}'(x)) &= \sum_{i \in [L]} \text{wt}(c'_i(x)) \\ &= \sum_{i \in \mathcal{S} \setminus \mathcal{P}} \text{wt}(\text{perm}(\mathbf{H}_{\mathcal{S} \setminus i}(x))), \end{aligned} \quad (5.14)$$

where we use the fact that  $\text{wt}(\varphi) = 0$ . Combining (5.14) with (5.9), we obtain the upper bound (5.13), again noting that we obtain an upper bound because in general only a strict subset of the codewords can be generated by Lemma 5.7. ■

Care must be taken to ensure that the puncturing operation does not reduce the dimensionality of the code, that is, the base-2 logarithm of the number of distinct codewords. This is a requirement in the results that follow. Clearly, if puncturing a nonzero codeword of  $\mathcal{C}$  produces the all-zero codeword of  $\mathcal{C}'$ , dimensionality will be lost with respect to the original code.

**Lemma 5.9** *Let  $\mathcal{C}'$  be a punctured QC code constructed by puncturing subblocks of the QC code  $\mathcal{C}$ , while maintaining the dimensionality of  $\mathcal{C}$ . Let the length- $L$  vector  $\mathbf{c}(x)$  be a codeword of  $\mathcal{C}$  and  $\mathbf{c}'(x)$  be a codeword of  $\mathcal{C}'$  obtained by puncturing  $\mathbf{c}(x)$ . Then,  $c'_i(x) \in \{0, \varphi\} \forall i \in [L]$  if and only if  $\mathbf{c}(x) = \mathbf{0}$ .*

*Proof:* The necessity of the condition  $\mathbf{c}(x) = \mathbf{0}$  follows from the requirement that the dimensionality of the original code be maintained. The sufficiency of the condition follows directly from the fact that puncturing the all-zero codeword of  $\mathcal{C}$  produces the all-zero codeword of  $\mathcal{C}'$ . ■

Since the contribution of each subblock to the weight of the codeword is nonnegative, the weight of any particular punctured codeword must be less than or equal to its weight before puncturing, as can be seen by comparison of (5.14) to (5.11). Moreover, Lemma 5.9 implies that the upper bound of Theorem 5.8 will always be less than or equal to the upper bound of Theorem 5.4, where no puncturing is used.

**Theorem 5.10** *Let  $\mathcal{C}'$  be a punctured QC code constructed by puncturing subblocks of the QC code  $\mathcal{C}$ , defined by the polynomial parity-check matrix  $\mathbf{H}(x) \in (\mathbb{F}_2[x]/\langle x^N - 1 \rangle)^{J \times L}$  and let*

$\mathbf{A} \triangleq \text{wt}(\mathbf{H}(x))$ . Let the subblocks of  $\mathcal{C}$  indexed by the set  $\mathcal{P}$ ,  $\mathcal{P} \subset [L]$ , be punctured, while maintaining the dimensionality of the code. Then

$$d_{\min}(\mathcal{C}') \leq \min_{\substack{\mathcal{S} \subseteq [L] \\ |\mathcal{S}|=J+1}}^* \sum_{i \in \mathcal{S} \setminus \mathcal{P}} \text{perm}(\mathbf{A}_{\mathcal{S} \setminus i}) \quad (\text{in } \mathbb{Z}). \quad (5.15)$$

*Proof:* The proof techniques we use are similar to those used in the proof of Theorem 8 in [7], while avoiding the use of intermediate bounds based on convolutional codes. Let  $\mathcal{S}$  be a subset of  $[L]$  of size- $(J+1)$ , and apply Lemma 5.7 to construct a codeword,  $\mathbf{c}'(x)$ , in code  $\mathcal{C}'$ . From (5.14) we obtain

$$\begin{aligned} w_{\mathbf{H}}(\mathbf{c}'(x)) &= \sum_{i \in \mathcal{S} \setminus \mathcal{P}} \text{wt}(\text{perm}(\mathbf{H}_{\mathcal{S} \setminus i}(x))) \\ &\leq \sum_{i \in \mathcal{S} \setminus \mathcal{P}} \text{perm}(\text{wt}(\mathbf{H}_{\mathcal{S} \setminus i}(x))) \\ &= \sum_{i \in \mathcal{S} \setminus \mathcal{P}} \text{perm}(\mathbf{A}_{\mathcal{S} \setminus i}), \end{aligned}$$

where we invoked Lemma 5.5 to obtain the inequality in the second step.

We now show the validity of using the  $\min^*$  function in the upper bound. The potential complication arises from the fact that for specific choices of the set  $\mathcal{S}$ , namely those which yield the all-zero codeword in Lemma 5.7, the  $\min^*$  function may not exclude their contribution to (5.15), even though it does so in the bound (5.13). So, assume that a specific choice of  $\mathcal{S}$  yields  $w_{\mathbf{H}}(\mathbf{c}'(x)) = 0$  according to (5.14), but produces a nonzero value for  $\sum_{i \in \mathcal{S} \setminus \mathcal{P}} \text{perm}(\mathbf{A}_{\mathcal{S} \setminus i})$ . We must show there exists a nonzero codeword  $\mathbf{c}^{*'}(x)$  for which  $w_{\mathbf{H}}(\mathbf{c}^{*'}(x)) \leq \sum_{i \in \mathcal{S} \setminus \mathcal{P}} \text{perm}(\mathbf{A}_{\mathcal{S} \setminus i})$ . Thus, for such specific choices of  $\mathcal{S} \subseteq [L]$ , we assume in the remainder of the proof that  $c'_i(x) \in \{0, \varphi\} \forall i \in [L]$ . By Lemma 5.9, we know that  $\mathbf{c}(x) = \mathbf{0}$  for this  $\mathcal{S}$ . By Lemma 5.3, we know that every  $J \times J$  submatrix of  $\mathbf{H}_{\mathcal{S}}(x)$  must have a zero permanent and determinant. We now consider two cases.

**Case 1** If  $\sum_{i \in \mathcal{S} \setminus \mathcal{P}} \text{perm}(\mathbf{A}_{\mathcal{S} \setminus i}) = 0$ , then this specific  $\mathcal{S}$  has no effect on the bound (5.15), as the zero result will be discarded by the  $\min^*$  function.

**Case 2** Alternatively, if  $\sum_{i \in \mathcal{S} \setminus \mathcal{P}} \text{perm}(\mathbf{A}_{\mathcal{S} \setminus i}) > 0$ , there are no all-zero rows in  $\mathbf{A}_{\mathcal{S}}$  and, therefore, none in  $\mathbf{H}_{\mathcal{S}}(x)$ . However, we know that every  $J \times J$  submatrix of  $\mathbf{H}_{\mathcal{S}}(x)$  has a zero determinant for the specific  $\mathcal{S}$  that generates  $\mathbf{c}(x) = \mathbf{0}$ . By Lemma 5.2, this implies that the set of rows of  $\mathbf{H}_{\mathcal{S}}(x)$  is dependent. We analyze this case further by setting aside the  $t$ th row of

$\mathbf{H}_{\mathcal{S}}(x)$ ,  $\mathbf{h}_{t,\mathcal{S}}(x)$ , preferring a dependent row to be the  $t$ th row<sup>2</sup>. We form a new matrix,  $\mathbf{H}'(x)$ , with the remaining  $J - 1$  rows of  $\mathbf{H}(x)$  and a matrix,  $\mathbf{A}'$ , with the corresponding  $J - 1$  rows of  $\mathbf{A}$ . Because of the assumption that  $\sum_{i \in \mathcal{S} \setminus \mathcal{P}} \text{perm}(\mathbf{A}_{\mathcal{S} \setminus i}) > 0$ , there must be at least one index  $i \in \mathcal{S} \setminus \mathcal{P}$ , such that  $\text{perm}(\mathbf{A}_{\mathcal{S} \setminus i}) > 0$ . The cofactor expansion of this term along row  $t$  contains a term

$$a_{t,i^*} \cdot \text{perm}(\mathbf{A}'_{(\mathcal{S} \setminus i) \setminus i^*}) > 0, \quad (5.16)$$

for some  $i^* \in \mathcal{S} \setminus i$ , where the positive integer  $a_{t,i^*}$  is the entry in the  $t$ th row and  $i^*$ th column of  $\mathbf{A}$ . Let  $\mathcal{S}^* \triangleq \mathcal{S} \setminus i^*$ .

Proceeding, we now assume that  $\mathbf{H}'_{\mathcal{S}^*}(x)$  contains at least one  $(J - 1) \times (J - 1)$  submatrix with nonzero permanent. (If this is not true, we repeat the row removal process above, which may need to be repeated several times. In the extreme case, these reductions could continue until we get a  $1 \times 2$  matrix  $\mathbf{H}'_{\mathcal{S}^*}(x)$ , having at least one nonzero entry.) Then applying Lemma 5.3, with  $\mathbf{H}(x) = \mathbf{H}'(x)$  and  $\mathcal{S} = \mathcal{S}^*$ , we generate a nonzero vector,  $\mathbf{c}^*(x)$ , with components

$$c_i^*(x) = \begin{cases} \text{perm}(\mathbf{H}'_{\mathcal{S}^* \setminus i}(x)) & \text{if } i \in \mathcal{S}^* \\ 0 & \text{otherwise.} \end{cases}$$

The proof of Lemma 5.3 implies that  $\mathbf{H}'(x)\mathbf{c}^*(x)^T = \mathbf{0}^T$ . Multiplying the removed row of the parity-check matrix by the vector  $\mathbf{c}^*(x)$  yields

$$\begin{aligned} \mathbf{h}_t(x)\mathbf{c}^*(x)^T &= \sum_{i \in [L]} h_{t,i}(x) \cdot c_i^*(x) \\ &= \sum_{i \in \mathcal{S}^*} h_{t,i}(x) \text{perm}(\mathbf{H}'_{\mathcal{S}^* \setminus i}(x)) \\ &= \text{perm}(\mathbf{H}_{\mathcal{S}^*}(x)) = 0, \end{aligned}$$

since all  $J \times J$  submatrices of  $\mathbf{H}_{\mathcal{S}}(x)$  were assumed to have a zero permanent. Therefore, the nonzero vector  $\mathbf{c}^*(x)$  is a codeword in  $\mathcal{C}$ .

By puncturing  $\mathbf{c}^*(x)$  we generate another nonzero vector,  $\mathbf{c}^{*'}(x)$ , which is a codeword

---

<sup>2</sup>The proof holds no matter which row is chosen for removal; however, the row removal process terminates more rapidly if a dependent row is chosen.

in  $\mathcal{C}'$ . The Hamming weight of this codeword satisfies the upper bound

$$\begin{aligned} w_H(\mathbf{c}^{*'}(x)) &= \sum_{i \in \mathcal{S}^* \setminus \mathcal{P}} \text{wt} \left( \text{perm} \left( \mathbf{H}'_{\mathcal{S}^* \setminus i}(x) \right) \right) \\ &\leq \sum_{i \in \mathcal{S}^* \setminus \mathcal{P}} \text{perm} \left( \mathbf{A}'_{\mathcal{S}^* \setminus i} \right) \end{aligned} \quad (5.17)$$

$$\leq a_{t,i^*} \cdot \sum_{i \in \mathcal{S}^* \setminus \mathcal{P}} \text{perm} \left( \mathbf{A}'_{\mathcal{S}^* \setminus i} \right) \quad (5.18)$$

$$\leq \sum_{i \in \mathcal{S}^* \setminus \mathcal{P}} \sum_{j \in \mathcal{S} \setminus i} a_{t,j} \cdot \text{perm} \left( \mathbf{A}'_{(\mathcal{S} \setminus i) \setminus j} \right) \quad (5.19)$$

$$\begin{aligned} &= \sum_{i \in \mathcal{S}^* \setminus \mathcal{P}} \text{perm} \left( \mathbf{A}_{\mathcal{S} \setminus i} \right) \\ &\leq \sum_{i \in \mathcal{S} \setminus \mathcal{P}} \text{perm} \left( \mathbf{A}_{\mathcal{S} \setminus i} \right). \end{aligned} \quad (5.20)$$

Applying Lemma 5.5 produces (5.17). Using the fact that  $a_{t,i^*} \geq 1$ , a consequence of (5.16), we upper bound (5.17) by (5.18). Next, we further upper bound (5.18) by (5.19) by adding additional nonnegative terms to (5.19). We recognize that (5.19) contains the sum of the cofactor expansions of each addend of (5.20).

We can now conclude that, even if a set  $\mathcal{S}$  generates the all-zero codeword in Lemma 5.7, it still yields a valid upper bound on the minimum distance of the punctured code, *i.e.*,

$$d_{\min}(\mathcal{C}') \leq \sum_{i \in \mathcal{S} \setminus \mathcal{P}} \text{perm} \left( \mathbf{A}_{\mathcal{S} \setminus i} \right),$$

provided that  $\text{perm}(\mathbf{A}_{\mathcal{S} \setminus i})$  is positive for at least one  $i \in \mathcal{S} \setminus \mathcal{P}$ . The validity of the upper bound in (5.15) follows. ■

## 5.5 Tighter Bounds on the Minimum Distance

Examining the AR4JA protomatrices for rate-2/3 in (5.2) and rate-4/5 in (5.3), we see cases where the selection of  $J+1 = 4$  columns of the weight matrix  $\mathbf{A}$  will produce a submatrix  $\mathbf{A}_{\mathcal{S}}$  containing an all-zero top row. This particular selection of  $\mathcal{S}$  produces the all-zero codeword by the codeword construction of Lemmas 5.3 and 5.7, and, thus, will have no effect on the upper bounds of Theorems 5.4, 5.6, 5.8, and 5.10. We can improve those bounds by finding nonzero codewords after row elimination, as in the proof of Theorem 5.10.

In the interest of brevity, we will state the following theorems in a way that applies to both unpunctured and punctured codes. In the unpunctured case, it is understood that the set  $\mathcal{P}$

is empty.

**Lemma 5.11** *Let  $\mathcal{C}'$  be a QC code constructed by optionally puncturing subblocks of the QC code  $\mathcal{C}$ , defined by the polynomial parity-check matrix  $\mathbf{H}(x) \in (\mathbb{F}_2[x]/\langle x^N - 1 \rangle)^{J \times L}$ . Let the subblocks of  $\mathcal{C}$  indexed by the set  $\mathcal{P}$ ,  $\mathcal{P} \subset [L]$ , be punctured. Let  $\mathbf{H}'(x)$  be a submatrix of  $\mathbf{H}(x)$  with rows  $\mathbf{h}_t(x)$ ,  $t \in \mathcal{T} \subset [J]$ , removed. Let  $\mathcal{S}$  be a subset of  $[L]$  of size  $J + 1 - |\mathcal{T}|$ , such that*

$$\text{perm} \begin{bmatrix} \mathbf{h}_{t,\mathcal{S}}(x) \\ \mathbf{H}'_{\mathcal{S}}(x) \end{bmatrix} = 0 \quad \forall t \in \mathcal{T}. \quad (5.21)$$

*Let the components of the length- $L$  vector  $\mathbf{c}'(x) = (c'_0(x), c'_1(x), \dots, c'_{L-1}(x))$ , with  $c'_i(x) \in \mathbb{F}_2[x]/\langle x^N - 1 \rangle \cup \{\varphi\}$ , be defined as*

$$c'_i(x) = \begin{cases} \text{perm}(\mathbf{H}'_{\mathcal{S} \setminus i}(x)) & \text{if } i \in \mathcal{S} \setminus \mathcal{P} \\ \varphi & \text{if } i \in \mathcal{P} \\ 0 & \text{otherwise.} \end{cases}$$

*Then  $\mathbf{c}'(x)$  is a codeword in  $\mathcal{C}'$ .*

*Proof:* We consider two cases.

**Case 1** If  $\mathcal{P} = \emptyset$  (the code is unpunctured), then  $\mathcal{C} = \mathcal{C}'$ . We first examine every retained row of  $\mathbf{H}(x)$ , denoted by  $\mathbf{h}_j(x)$ , where  $j \in [J]$  and  $j \notin \mathcal{T}$ . The inner product of  $\mathbf{h}_j(x)$  with the vector  $\mathbf{c}'(x)$  is

$$\begin{aligned} \mathbf{h}_j(x) \mathbf{c}'(x)^T &= \sum_{i \in \mathcal{S}} h_{j,i}(x) \text{perm}(\mathbf{H}'_{\mathcal{S} \setminus i}(x)) \\ &= \text{perm} \begin{bmatrix} \mathbf{h}_{j,\mathcal{S}}(x) \\ \mathbf{H}'_{\mathcal{S}}(x) \end{bmatrix} \\ &= \det \begin{bmatrix} \mathbf{h}_{j,\mathcal{S}}(x) \\ \mathbf{H}'_{\mathcal{S}}(x) \end{bmatrix} = 0, \end{aligned}$$

since the determinant expression contains a repeated row. Next, for every row  $\mathbf{h}_t(x)$  removed from  $\mathbf{H}(x)$ , i.e., every row  $\mathbf{h}_t(x)$  in which  $t \in \mathcal{T}$ , we have

$$\mathbf{h}_t(x) \mathbf{c}'(x)^T = \text{perm} \begin{bmatrix} \mathbf{h}_{t,\mathcal{S}}(x) \\ \mathbf{H}'_{\mathcal{S}}(x) \end{bmatrix} = 0,$$

because the permanent was assumed to be zero in (5.21). Since all rows of the original polynomial parity-check matrix  $\mathbf{H}(x)$  have been accounted for,  $\mathbf{H}(x) \mathbf{c}'(x)^T = \mathbf{0}^T$  and  $\mathbf{c}'(x)$  is a codeword in  $\mathcal{C} = \mathcal{C}'$ .

**Case 2** If the code  $\mathcal{C}'$  is punctured, let the components of the length- $L$  vector  $\mathbf{c}(x) = (c_0(x), c_1(x), \dots, c_{L-1}(x))$ , with  $c_i(x) \in \mathbb{F}_2[x]/\langle x^N - 1 \rangle$ , be defined as

$$c_i(x) = \begin{cases} \text{perm}(\mathbf{H}'_{\mathcal{S} \setminus i}(x)) & \text{if } i \in \mathcal{S} \\ 0 & \text{otherwise.} \end{cases}$$

The proof follows by noting that  $\mathbf{c}'(x)$  is obtained by puncturing subblocks indexed by  $\mathcal{P}$  from the unpunctured codeword  $\mathbf{c}(x)$ , above. Since Case 1 establishes that  $\mathbf{c}(x) \in \mathcal{C}$ , we conclude that  $\mathbf{c}'(x) \in \mathcal{C}'$ . ■

Not only does Lemma 5.11 remove all-zero rows from  $\mathbf{H}_{\mathcal{S}}(x)$ , it also helps produce lower weight codewords in more general conditions, as the following example shows.

**Example 5.1** Consider the polynomial parity-check matrix

$$\mathbf{H}(x) = \begin{bmatrix} 0 & 0 & 0 & f_1(x) \\ x^a & x^b & x^c & f_2(x) \\ x^a & x^b & x^d & f_3(x) \end{bmatrix},$$

where  $f_i(x)$ ,  $i = 1, 2, 3$  are arbitrarily chosen polynomials. Since  $\text{perm}(\mathbf{H}_{\mathcal{S}}(x)) = 0$  with the column set  $\mathcal{S} = \{0, 1, 2\}$ , as required by (5.21), we proceed with single row removal on  $\mathbf{H}(x)$ . Application of Lemma 5.11 for all possible choices of  $\mathcal{T}$  with  $|\mathcal{T}| = 1$  yields the codewords  $\mathbf{c}(x) = \mathbf{0}$  and

$$\mathbf{c}(x) = (x^{b+d} + x^{b+c}, x^{a+d} + x^{a+c}, 0, 0) \bmod x^N - 1.$$

However, with careful consideration, we see that Lemma 5.11 will let us delete two sub-rows when the column set is  $\mathcal{S} = \{0, 1\}$ . This produces the obvious codeword  $\mathbf{c}(x) = (x^b, x^a, 0, 0)$ , when  $\mathcal{T} = \{0, 1\}$  or  $\mathcal{T} = \{0, 2\}$ .

**Theorem 5.12** Let  $\mathcal{C}'$  be a QC code constructed by optionally puncturing subblocks of the QC code  $\mathcal{C}$ , defined by the polynomial parity-check matrix  $\mathbf{H}(x) \in (\mathbb{F}_2[x]/\langle x^N - 1 \rangle)^{J \times L}$ . Let the subblocks of  $\mathcal{C}$  indexed by the set  $\mathcal{P}$ ,  $\mathcal{P} \subset [L]$ , be punctured. Let  $\mathbf{H}'(x)$  be a submatrix of  $\mathbf{H}(x)$  with rows  $\mathbf{h}_t(x)$ ,  $t \in \mathcal{T} \subset [J]$ , removed. Let  $\mathcal{S}$  be a subset of  $[L]$  of size  $J + 1 - |\mathcal{T}|$ , such that (5.21) holds. Then

$$d_{\min}(\mathcal{C}') \leq \min_{\mathcal{S}, \mathcal{T}}^* \sum_{i \in \mathcal{S} \setminus \mathcal{P}} \text{wt}(\text{perm}(\mathbf{H}'_{\mathcal{S} \setminus i}(x))). \quad (5.22)$$

*Proof:* The proof mirrors the proof of Theorem 5.8, with the weight of the resulting codeword now given by

$$w_H(\mathbf{c}'(x)) = \sum_{i \in S \setminus \mathcal{P}} \text{wt} \left( \text{perm} \left( \mathbf{H}'_{S \setminus i}(x) \right) \right).$$

■

Note that the minimization in (5.22) requires the removal of every possible set of rows  $\mathcal{T}$  and every set of retained columns  $\mathcal{S}$  for which  $|\mathcal{S}| + |\mathcal{T}| = J + 1$  and (5.21) holds. In the case of single row removal ( $|\mathcal{T}| = 1$ ), any row may be removed, as our requirement (5.21) degenerates to the condition  $\text{perm}(\mathbf{H}_S(x)) = 0$ , which is independent of the row selected for removal. For multiple row removal, the conditions are more complex to evaluate as each row in the set to be removed must be tested individually to verify that (5.21) holds.

For a specified expansion factor  $N$ , Lemma 5.11 and Theorem 5.12 impose certain conditions on the set  $\mathcal{T}$  that allow for the removal of rows from the polynomial parity-check matrix. However, these conditions cannot be directly translated into a form applicable to the nonnegative weight matrix  $\mathbf{A}$ , which is independent of  $N$ . Therefore, the following theorem uses the stricter condition that the sub-row  $\mathbf{a}_{t,S}$  is all-zero before removal.

**Theorem 5.13** *Let  $\mathcal{C}'$  be a QC code constructed by optionally puncturing subblocks of the QC code  $\mathcal{C}$ , defined by the polynomial parity-check matrix  $\mathbf{H}(x) \in (\mathbb{F}_2[x]/\langle x^N - 1 \rangle)^{J \times L}$  and let  $\mathbf{A} \triangleq \text{wt}(\mathbf{H}(x))$ . Let the subblocks of  $\mathcal{C}$  indexed by the set  $\mathcal{P}$ ,  $\mathcal{P} \subset [L]$ , be punctured, while maintaining the dimensionality of the code. Let  $\mathbf{A}'$  be a submatrix of  $\mathbf{A}$  with rows  $\mathbf{a}_t$ ,  $t \in \mathcal{T} \subset [J]$ , removed. Let  $\mathcal{S}$  be a subset of  $[L]$  of size  $J + 1 - |\mathcal{T}|$ , such that the sub-rows  $\mathbf{a}_{t,S} = \mathbf{0} \forall t \in \mathcal{T}$ . Then*

$$d_{\min}(\mathcal{C}') \leq \min_{\mathcal{S}, \mathcal{T}}^* \sum_{i \in S \setminus \mathcal{P}} \text{perm}(\mathbf{A}'_{S \setminus i}) \quad (\text{in } \mathbb{Z}).$$

*Proof:* Let  $\mathbf{H}'(x)$  be the submatrix of  $\mathbf{H}(x)$  with rows  $\mathbf{h}_t(x)$ ,  $t \in \mathcal{T}$ , removed. The  $|\mathcal{T}|$  sub-rows of the weight matrix  $\mathbf{A}$  to be removed are all-zero (i.e.,  $\mathbf{a}_{t,S} = \mathbf{0} \forall t \in \mathcal{T}$ ) if and only if the corresponding sub-rows of the polynomial parity-check matrix  $\mathbf{H}(x)$  are all-zero (i.e.,  $\mathbf{h}_{t,S}(x) = \mathbf{0} \forall t \in \mathcal{T}$ ). The latter condition implies that (5.21) holds and we may apply Lemma 5.11 with this  $\mathcal{S}$  and  $\mathcal{T}$  and construct a codeword  $\mathbf{c}'(x)$  in the code  $\mathcal{C}'$ . By Theorem 5.12, the weight of  $\mathbf{c}'(x)$  is

$$\begin{aligned} w_H(\mathbf{c}'(x)) &= \sum_{i \in S \setminus \mathcal{P}} \text{wt} \left( \text{perm} \left( \mathbf{H}'_{S \setminus i}(x) \right) \right) \\ &\leq \sum_{i \in S \setminus \mathcal{P}} \text{perm}(\mathbf{A}'_{S \setminus i}), \end{aligned}$$

where Lemma 5.5 is applied to obtain the inequality. Once again, the use of the  $\min^*$  function in the bound must be validated by consideration of the all-zero codewords discarded in Theorem 5.12. The reasoning largely parallels that used in the proof of Theorem 5.10, but with  $|\mathcal{T}|$  sub-rows of  $\mathbf{H}_S(x)$  guaranteed to be all-zero. We omit the details. ■

**Example 5.2** The benefits of Theorem 5.13 may be seen by considering the weight matrix given by

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 2 \\ 1 & 2 & 2 & 1 \end{bmatrix}.$$

Treating the code as unpunctured, Theorem 5.10 produces a minimum distance upper bound of  $+\infty$ , since all  $3 \times 3$  submatrices of  $\mathbf{A}$  have a zero permanent. Theorem 5.13 produces a much tighter bound of 3, when  $\mathbf{A}'_S = \begin{bmatrix} 1 & 2 \end{bmatrix}$ .

**Example 5.3** We now consider the weight matrix

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 3 & 0 & 3 \\ 1 & 1 & 0 & 1 & 3 \\ 1 & 2 & 0 & 2 & 1 \end{bmatrix}$$

which resembles the rate-1/2 AR4JA protomatrix in (5.1). Treating the code as unpunctured, Theorem 5.10 produces a minimum distance upper bound of 30, while Theorem 5.13 produces a substantially tighter upper bound of 10. The reason for the difference is that the permanents of Theorem 5.10 produce large values with the top row of  $\mathbf{A}$  present. Theorem 5.13 will remove the top row of  $\mathbf{A}$  when the chosen column set is  $S = \{0, 1, 3\}$ , yielding the tighter bound.

## 5.6 QC Expansion of the AR4JA Family of Codes

A direct QC expansion of the AR4JA protographs shown in Figs. 5.3 and 5.4 will generate a QC-LDPC code. Applying Theorems 5.10 and 5.13 to the AR4JA protomatrices (5.1) – (5.3) yields an upper bound of 10 on the minimum distance for all code rates, independent of block length. A minimum distance of 10 or less is rather small for the large block lengths desired, motivating the consideration of a more involved expansion procedure.

In fact, the construction of the AR4JA codes defined in [11] makes use of a two-step expansion process. After a first QC expansion (“lifting”) by a factor of 4, a larger weight matrix

is obtained, as illustrated for rate-1/2 by the matrix

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ \hline 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ \hline 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}. \quad (5.23)$$

This is a so-called *type-1* weight matrix—that is, it contains only ones and zeros—implying that the associated protograph does not have parallel edges [7].

According to the CCSDS standard, the weight matrices so obtained, such as (5.23), are considered to be protomatrices themselves, that are then expanded in a second QC expansion to create QC-LDPC codes with three block lengths, corresponding to  $k = 1024$ , 4096, and 16 384 information bits. For example, quasi-cyclically expanding (5.23) by a factor of  $N = 128$  and puncturing the last 4 columns of (5.23) yields the  $(n, k) = (2048, 1024)$  AR4JA code. In this final expansion, the binary parity-check matrix  $\mathbf{H}$  is constructed by replacing each 1 entry in (5.23) by a cyclic permutation matrix selected using a variation on the ACE algorithm. These codes are QC with a subblock size equal to the second step expansion factor (*e.g.*,  $N = 128$ ). In other words, the two-step process is not equivalent to any single-step QC expansion.

To compute length-independent minimum distance bounds for the AR4JA codes specified in the CCSDS standard using the techniques we have developed in this chapter, the protomatrices such as (5.23) should be used. The resulting upper bounds, shown in Table 5.1, range from 56 to 66. Note that, prior to [7], the tightest known upper bound on the minimum distance of QC-LDPC codes was  $(J + 1)!$  in the case when the protomatrices are all-ones [4, 6]. (We find in [7] that  $d_{\min} \leq (J + 1)!$  also holds for the more general case of type-1 protomatrices.) For the protomatrix of (5.23), where  $J = 12$ , this would yield the extremely loose upper bound  $d_{\min} \leq 6.2 \cdot 10^9$ .

This example points to the potential advantage of a two-step QC expansion, as suggested by the increase from 10 to the range 56 – 66 in the minimum distance upper bound. It also illustrates the strength of the general class of hierarchical QC-LDPC codes that have recently

**Table 5.1:** Minimum distance of AR4JA protomatrices after first QC expansion (independent of block length).

| <b>Code<br/>Rate <math>r</math></b> | <b>Upper Bounds by<br/>Theorems 5.10 and 5.13</b> | <b>Num. of sets <math>\mathcal{S}</math> of<br/>size <math>J + 1</math> in <math>[L]</math></b> |
|-------------------------------------|---------------------------------------------------|-------------------------------------------------------------------------------------------------|
| 1/2                                 | 66                                                | $7.8 \cdot 10^4$                                                                                |
| 2/3                                 | 58                                                | $3.7 \cdot 10^7$                                                                                |
| 4/5                                 | 56 <sup>a</sup>                                   | $5.2 \cdot 10^{10}$                                                                             |

<sup>a</sup>Computations are not exhaustive in sets  $\mathcal{S}$  due to complexity.

been examined in [18].

In order to reduce the computation time required to produce the results in Table 5.1, a number of techniques were used. For larger weight matrices, if we assume that calculations are dominated by the computation time  $t_J$  for the  $J \times J$  permanent, then the total time to evaluate Theorem 5.10 is  $t_J(J+1)\binom{L}{J+1}$ . The final term  $\binom{L}{J+1}$  is the number of sets  $\mathcal{S}$  in the weight matrix  $\mathbf{A}$  and is shown in the right column of Table 5.1 for AR4JA. For the computations of interest, with  $J = 12$ , we built a simple recursive routine with  $t_{12} = 44\mu s$  for computing sparse permanents (as measured on a 2.6 GHz CPU). Thus, the estimated time for computing the rate-1/2 results in Table 5.1 is 44s, while we measured an actual run-time of 53s, including bookkeeping and set manipulations. On the other hand, for the rate-4/5 construction, the estimated time to completely evaluate the upper bound of Theorem 5.10 is 344 days. Therefore, we selectively directed the computations, yielding the results shown in Table 5.1. These selective computations were performed by choosing sets  $\mathcal{S}$  which, based upon the findings from the rate-2/3 code, we thought would yield the smallest value permanents in evaluating (5.15). Additional efforts at each code rate to recompute the minimum distance bounds using the row elimination logic of Theorem 5.13 did not yield tighter results with the AR4JA weight matrices.

## 5.7 Numerical Results Obtained by Search

In this section, we present numerical results on the minimum distance of AR4JA codes obtained by means of computer search for low-weight codewords. We also examine bounds on the girth of AR4JA codes.

**Table 5.2:** Distance of CCSDS AR4JA parity-check matrices.

| Code<br>Rate<br>$r$ | Minimum Distance<br>U.B. by Searching |            | Stopping Distance<br>U.B. by Searching |                 |
|---------------------|---------------------------------------|------------|----------------------------------------|-----------------|
|                     | $k = 1024$                            | $k = 4096$ | $k = 1024$                             | $k = 4096$      |
|                     |                                       |            |                                        |                 |
| 1/2                 | 52                                    | 63         | 50                                     | 63 <sup>a</sup> |
| 2/3                 | 26                                    | –          | 23                                     | 62 <sup>b</sup> |
| 4/5                 | 13                                    | 27         | 11                                     | 25              |

<sup>a</sup>The smallest stopping set found was a codeword.

<sup>b</sup>Beyond the upper bound shown in Table 5.1.

**Table 5.3:** Estimated weight spectrum for CCSDS AR4JA rate-4/5,  $k = 1024$  using search parameters:  $I = 150$  and  $T = 10$ .

|                     | Hamming Weight |     |     |     | Search  |
|---------------------|----------------|-----|-----|-----|---------|
|                     | 13             | 14  | 15  | 16  | Time    |
| Number of Codewords | 32             | 256 | 128 | 400 | 1.5 hrs |

**Table 5.4:** Estimated weight spectrum for CCSDS AR4JA rate-4/5,  $k = 4096$  using search parameters:  $I = 300$  and  $T = 19$ .

|                     | Hamming Weight |    |    | Search |
|---------------------|----------------|----|----|--------|
|                     | 27             | 28 | 29 | Time   |
| Number of Codewords | 128            | 0  | 0  | 82 hrs |

### 5.7.1 Distance Bounds from Codeword Search for AR4JA Codes

Several papers, including [19–21], have described search techniques to find the minimum distance and/or stopping distance of general LDPC codes. To validate our bounds on the minimum Hamming distance, we utilized the error impulse and decoding algorithm of [19] to conduct a non-exhaustive search for small stopping sets, and then examined the results to

**Table 5.5:** Estimated weight spectrum for CCSDS AR4JA rate-2/3,  $k = 1024$  using search parameters:  $I = 300$  and  $T = 19$ .

|                            | Hamming Weight |     |    |    | Search |
|----------------------------|----------------|-----|----|----|--------|
|                            | 26             | 29  | 31 | 32 | Time   |
| <b>Number of Codewords</b> | 64             | 128 | 64 | 64 | 5 hrs  |

identify the codewords. We modified the algorithm to take advantage of QC symmetry by skipping impulse combinations which are identical after cyclically shifting every subblock. We also broadened the search space by increasing the value of the parameters in [19] corresponding to the number of iterations  $I$  and the maximum threshold  $T$ . In addition, before the erasure decoding step, we erased the punctured symbols in addition to the symbols already erased by the algorithm. The resulting upper bounds on the minimum distance and the stopping distance obtained by this search methodology are summarized in Table 5.2.

It should be noted that, in [19], the search algorithm was applied to rate-1/2 codes up to a minimum distance of 19, and the estimated weight spectrum results were listed only up to a maximum codeword weight of 25. It may be the case, then, that the application of this method to some of the AR4JA codes under consideration here may be pushing the algorithm beyond its effective range. Thus, further searching may turn up lower weight codewords and stopping sets than we have found.

We note that, as a consequence of the quasi-cyclicity of the code, when a codeword of a QC-LDPC code is located by our search technique, it is an indication of a group of codewords with the same weight. For instance, the low-weight codewords of the rate-4/5,  $k = 1024$  CCSDS AR4JA code generally occur as a set of  $N = 32$  cyclically-shifted versions of a base codeword. On occasion, when all subblocks are periodic with a common period, the cyclically-shifted codeword returns to the base codeword after only a fraction of  $N$  shifts. Accounting for this, we tabulated all distinct low-weight codewords found using our search algorithm for three of the AR4JA parity-check matrices given in the CCSDS standard. The estimated weight spectra, based upon the codewords we identified during our search, are shown in Tables 5.3 – 5.5.

Fig. 5.5 summarizes our minimum distance results for rate-1/2 AR4JA codes as a function of the block length  $n$ . First, the upper bound of 66 obtained from the weight matrix using Theorems 5.10 and 5.13 is plotted as a horizontal line. The points indicated by the symbol  $\times$

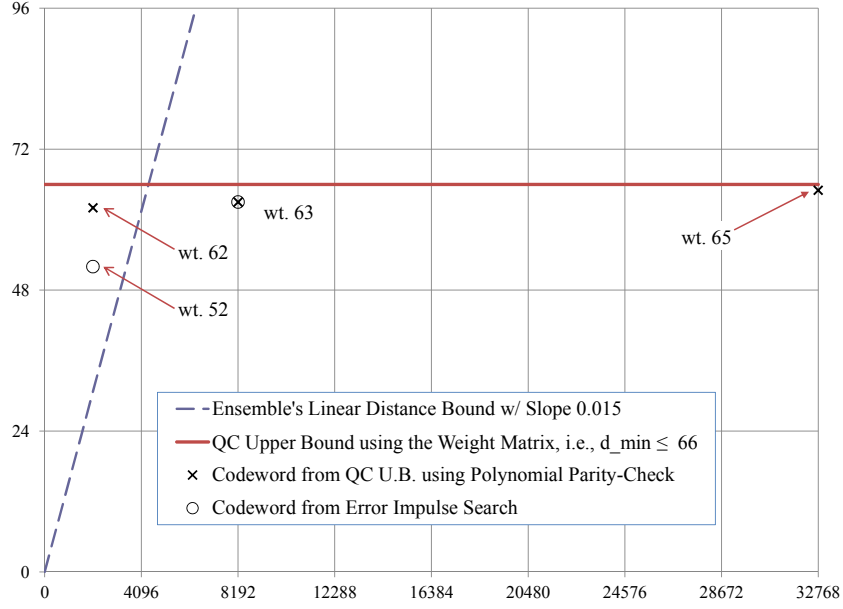


**Figure 5.5:** Minimum distance vs. block length (in bits) for rate-1/2 AR4JA.

represent the minimum distance upper bounds based on the QC polynomial parity-check matrices from Theorem 5.8. These bounds are 62, 63, and 65 for the block lengths 2048, 8192, and 32 768, respectively. Finally, the codeword weights found by our search technique, as shown in Table 5.2, are designated by the symbol  $\circ$ . Note that the smallest codeword weights found in our search are fairly close to the length-independent bound of 66 for all block lengths. We note that while our codeword searches generally used error impulse pairs, the codeword of weight 52 at the smallest block length was found using impulse triplets.

Divsalar *et al.* showed that most codes in the ensemble of certain protograph-based codes, including the AR4JA codes, have minimum distance linearly increasing with block length [8–10]. Specifically, by upper bounding the ensemble average weight enumerator, they were able to prove that  $\Pr \{d_{\min} < \delta_{\min} n\} \rightarrow 0$  exponentially fast as  $n \rightarrow \infty$ , for some constant  $\delta_{\min} > 0$ . For the rate-1/2 AR4JA-based ensemble of codes, they computed the value  $\delta_{\min} = 0.015$ . The corresponding linear growth of the ensemble minimum-distance is plotted in Fig. 5.5 as a dashed line.

It can be seen that for the QC-AR4JA codes that appear in the standard, our bounds are tighter than the linearly increasing ensemble bound for  $n \geq 4400$  bits. By examining the probability that a random expansion is quasi-cyclic, we can shed some light on this possibly surprising situation. Consider the expansion of each 1 entry in a protomatrix such as (5.23) by a factor  $N$ . There are  $N$  cyclic permutation matrices to choose from and  $N!$  general permutation

matrices. Thus, a randomly chosen permutation matrix has only a probability of  $1/(N-1)!$  of being cyclic. This probability goes to zero super-exponentially fast. Since the QC class of expansions is such a small fraction of the ensemble of all possible expansions, one cannot claim with certainty that the probabilistic bound of Divsalar *et al.* applies to the resulting class of codes.

### 5.7.2 Bounds on the Girth of AR4JA Codes

In this section, we show that the two-step expansion approach was essential in order to achieve girths beyond 6 for larger block lengths of the rate-1/2 AR4JA codes. Recall that the *girth* of a code denotes the length of the shortest cycle in its Tanner graph. Table 5.6 summarizes the results of our calculations of the girth of the AR4JA codes. The girths of the standardized codes for each block length and code rate are shown in the table. Also shown, in parentheses, are upper bounds obtained by the tree method of [14, 22], which we briefly describe. In the protograph, the neighborhood of any node can be diagrammed as a tree. We can measure how tall this tree is at a given number of nodes corresponding to the specified block length. We do this for each node type in the protograph and select the smallest as an upper bound on girth that would apply to any possible expansion method. These are the upper bounds recorded in the table.

We also determined upper bounds on the girth that derive from properties of QC expansions. Since the AR4JA protomatrices of (5.1) – (5.3) all contain the element 3, the girth of the derived graphs obtained by QC expansion cannot exceed 6 (see, for example [7]). However, since the codes in the CCSDS standard use a two-step expansion, we must examine the intermediate protomatrices such as (5.23) to evaluate the girth. We find that they contain the submatrix  $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}^T$  at every code rate. This limits the girth of the QC expansion to a maximum of 12, independent of block length [6, 23, 24], as shown in the final row of Table 5.6.

## 5.8 Conclusion

This work has extended the upper bounds on the minimum distance of QC-LDPC codes developed in [7] to the class of punctured QC-LDPC codes. We have also tightened those distance bounds in situations where the codes are derived from protographs whose protomatrices contain many zeros.

We evaluated the minimum distance upper bounds for the AR4JA codes specified in the

**Table 5.6:** Girth of the CCSDS AR4JA codes.

| <b>Information</b><br><b>Bits <math>k</math></b> | <b>Measured Girth (Upper Bound)</b> |           |           |
|--------------------------------------------------|-------------------------------------|-----------|-----------|
|                                                  | $r = 1/2$                           | $r = 2/3$ | $r = 4/5$ |
| 1024                                             | 6 (12)                              | 4 (10)    | 4 (8)     |
| 4096                                             | 8 (14)                              | 6 (10)    | 4 (10)    |
| 16384                                            | 10 (16)                             | 6 (12)    | 4 (10)    |
| <b>QC Limit of</b><br><b>Protomatrix</b>         | 12                                  | 12        | 12        |

(#) Denotes upper bound computed by tree method [14, 22]

CCSDS standard for deep space communication. Our results show that the use of a two-step expansion in the definition of these codes was critical to achieve reasonably high minimum distance. On the other hand, we have also shown that the minimum distance of the standardized QC AR4JA codes does not grow with block length, even though the asymptotic ensemble minimum distance of AR4JA codes grows linearly in the block length [8–10]. Nevertheless, the minimum distance of the CCSDS codes is likely high enough for practical purposes.

The bounds developed here and in [7] can be useful tools in evaluating future QC-LDPC code designs, both punctured and unpunctured.

## 5.A Appendix: Proof of Lemma 5.2

In this appendix we state several properties of matrices over a commutative ring with unity. Several terms defined in Sections 5.3 and 5.4 will be used here, such as determinant, unit, zero divisor, and independence.

### 5.A.1 Matrices Over Commutative Rings

Let  $R$  be a commutative ring and let  $\mathcal{M}_N(R)$  be the ring of  $N \times N$  matrices over  $R$ . The determinant of a matrix  $\mathbf{A} \in \mathcal{M}_N(R)$  is denoted by  $\det(\mathbf{A})$ . Given matrices  $\mathbf{A}, \mathbf{B} \in \mathcal{M}_N(R)$ , we have the identity [17, §12.2]

$$\det(\mathbf{AB}) = \det(\mathbf{A}) \det(\mathbf{B}). \quad (5.24)$$

If  $R$  is a commutative ring with unit element 1, then a matrix  $\mathbf{A}$  is a unit in  $\mathcal{M}_N(R)$  if and only if  $\det(\mathbf{A})$  is a unit in  $R$ . To see this, suppose that  $\mathbf{A}$  has inverse  $\mathbf{A}^{-1}$ . From (5.24), we conclude that  $\det(\mathbf{A}^{-1})\det(\mathbf{A}) = \det(\mathbf{I}) = 1$ , where  $\mathbf{I}$  is the  $N \times N$  identity matrix. This implies that  $\det(\mathbf{A})$  is a unit in  $R$ . Conversely, from Cramer's rule, we have  $\det(\mathbf{A})\mathbf{I} = \mathbf{A} \operatorname{adj}(\mathbf{A})$ , where  $\operatorname{adj}(\mathbf{A})$  is the adjugate (or classical adjoint) of  $\mathbf{A}$  [17]. If  $\det(\mathbf{A})$  is a unit in  $R$ , then  $\mathbf{A}$  is invertible, with inverse  $\mathbf{A}^{-1} = (\det(\mathbf{A}))^{-1} \operatorname{adj}(\mathbf{A})$ .

Consider the free module of rank  $N$  over  $R$ , that is,  $R^N = \{\mathbf{r} = (r_0, \dots, r_{N-1}) \mid r_i \in R, \forall i = 0, \dots, N-1\}$ . The set  $\mathcal{V} = \{\mathbf{v}_0, \dots, \mathbf{v}_{N-1}\}$ , where  $\mathbf{v}_i \in R^N$ ,  $i = 0, \dots, N-1$ , generates  $R^N$  if every  $\mathbf{r} \in R^N$  can be expressed as a linear combination of members of the set  $\mathcal{V}$ , i.e.,  $\mathbf{r} = \sum_{i=0}^{N-1} a_i \mathbf{v}_i$ , with  $a_i \in R$ , for all  $i = 0, \dots, N-1$ . Furthermore, a generating set  $\mathcal{V}$  is a *basis* for  $R^N$  if it is also independent. The *standard basis*  $\{\mathbf{e}_0, \dots, \mathbf{e}_{N-1}\}$  of  $R^N$  is simply the set of rows of the  $N \times N$  identity matrix  $\mathbf{I}$ .

The following theorem is a reformulation of material from [25, ch. 5].

**Theorem 5.14** *Let  $R$  be a commutative ring with unity, and let  $\mathbf{A} \in \mathcal{M}_N(R)$  have rows  $\mathbf{v}_0, \dots, \mathbf{v}_{N-1} \in R^N$  and columns  $\mathbf{w}_0, \dots, \mathbf{w}_{N-1} \in R^N$ . The following statements are equivalent.*

- 1)  $\mathbf{A}$  is invertible, i.e., a unit in  $\mathcal{M}_N(R)$ .
- 2)  $\det \mathbf{A}$  is a unit in  $R$ .
- 3) The set  $\{\mathbf{v}_0, \dots, \mathbf{v}_{N-1}\}$  generates  $R^N$ .
- 4) The set  $\{\mathbf{v}_0, \dots, \mathbf{v}_{N-1}\}$  is a basis for  $R^N$ .
- 3') The set  $\{\mathbf{w}_0, \dots, \mathbf{w}_{N-1}\}$  generates  $R^N$ .
- 4') The set  $\{\mathbf{w}_0, \dots, \mathbf{w}_{N-1}\}$  is a basis for  $R^N$ .

*Proof:* We have already shown in the discussion above the equivalence between 1) and 2). We now prove that 1) implies 3). Namely, if  $\mathbf{A}$  is invertible, then for any  $\mathbf{r} \in R^N$  we can write

$$\mathbf{r} = \mathbf{r}\mathbf{I} = (\mathbf{r}\mathbf{A}^{-1})\mathbf{A}.$$

Conversely, if the rows of  $\mathbf{A}$  generate  $R^N$ , then they must generate the elements of the standard basis. Given vectors  $\{\mathbf{u}_0, \dots, \mathbf{u}_{N-1}\} \in R^N$  satisfying  $\mathbf{u}_i \mathbf{A} = \mathbf{e}_i$ , for all  $i = 0, \dots, N-1$ , the inverse matrix  $\mathbf{A}^{-1}$  is the matrix whose rows are the vectors  $\mathbf{u}_0, \dots, \mathbf{u}_{N-1}$ .

Next, we prove that 1) implies 4). Specifically, if the matrix  $\mathbf{A}$  is invertible, then it is not a zero divisor in  $\mathcal{M}_N(R)$ . Therefore, if a matrix  $\mathbf{B} \in \mathcal{M}_N(R)$  satisfies  $\mathbf{BA} = \mathbf{0}$ , then  $\mathbf{B} = \mathbf{0}$ , proving that the rows of  $\mathbf{A}$  are independent. Conversely, 4) implies 3) by definition, which was shown above to imply 1).

Finally, the equivalence between 3) and 3') and between 4) and 4') follows from the fact that  $\det(\mathbf{A}) = \det(\mathbf{A}^T)$ . This implies that  $\det(\mathbf{A}^T)$  is a unit in  $R$  if and only if  $\det(\mathbf{A})$  is a unit. ■

### 5.A.2 Finite Commutative Rings

We now consider *finite* commutative rings with unity, which include those rings central to this work,  $\mathbb{F}_2[x]/\langle x^N - 1 \rangle$ . As shown in Lemma 5.1, every nonzero element of a finite ring with unity must be either a unit or a zero divisor.

**Lemma 5.15** *Let  $R$  be a finite commutative ring with unity and let  $\mathbf{A} \in \mathcal{M}_N(R)$ . The rows  $\mathbf{v}_0, \dots, \mathbf{v}_{N-1}$  of  $\mathbf{A}$  generate  $R^N$  if and only if they are independent.*

*Proof:* If the rows of  $\mathbf{A}$  generate  $R^N$ , then, by Theorem 5.14, the rows of  $\mathbf{A}$  are also independent. Conversely, suppose the rows of  $\mathbf{A}$  are independent. If they do not generate  $R^N$ , then, by the pigeonhole principle, there exist distinct vectors  $\mathbf{r}, \mathbf{s} \in R^N$  such that  $\mathbf{rA} = \mathbf{sA}$ . This implies that  $\mathbf{rA} - \mathbf{sA} = (\mathbf{r} - \mathbf{s})\mathbf{A} = (0, \dots, 0)$ . This contradicts the assumed independence of the rows of  $\mathbf{A}$ , so the rows of  $\mathbf{A}$  must generate  $R^N$ . ■

Let  $R$  be a finite commutative ring with unity and  $\mathbf{A} \in \mathcal{M}_N(R)$ . Theorem 5.14 and Lemma 5.15 imply that the determinant  $\det(\mathbf{A})$  is a unit in  $R$  if and only if the rows of  $\mathbf{A}$  are independent. Moreover, by inverting the previous logic statement and applying Lemma 5.1, we find that the determinant  $\det(\mathbf{A})$  is zero or a zero divisor in  $R$  if and only if the rows of  $\mathbf{A}$  are dependent. This completes the proof of Lemma 5.2.

We now present two examples over several commutative rings with unity.

**Example 5.4 (Integer and integer quotient rings)** Define the matrix

$$\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix},$$

over the commutative ring  $R$  with unity, noting that  $\det(\mathbf{A}) = 2$ .

(a) Consider  $R = \mathbb{Z}$ , the ring of integers. This ring is infinite and does not have zero divisors. The element  $\det(\mathbf{A}) = 2$  is not a unit in  $\mathbb{Z}$ . By Theorem 5.14, the matrix  $\mathbf{A}$  is not

invertible and, therefore, its rows cannot generate all of  $\mathbb{Z}^2$ . In particular, no linear combination of the rows of  $\mathbf{A}$  can generate odd integers in the second coordinate. On the other hand, the rows are independent, as can be easily verified.

(b) Next, consider  $R = \mathbb{Z}/9\mathbb{Z}$ , the finite ring of integers modulo 9. Here, the ring element 2 is a unit, since  $2 \cdot 5 = 1$ . By Theorem 5.14,  $\mathbf{A}$  is invertible and, therefore, its rows generate  $R^2$ . In accordance with Lemma 5.15, the rows are independent.

(c) Finally, consider  $R = \mathbb{Z}/6\mathbb{Z}$ , the finite ring of integers modulo 6. In this case, 2 is a zero divisor, so, by Theorem 5.14, the rows of  $\mathbf{A}$  do not generate  $R^2$ . By Lemma 5.15, there is at least one dependent row. Specifically, multiplying the second row by the scalar  $3 \in R$  yields the zero element  $(0, 0) \in R^2$ .

**Example 5.5 (Polynomial and polynomial quotient rings)** Define the matrix

$$\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 0 & 1+x \end{bmatrix},$$

over the commutative ring  $R$  with unity, noting that  $\det(\mathbf{A}) = 1+x$ .

(a) Consider  $R = \mathbb{F}_2[x]$ , the ring of polynomials with binary coefficients. This ring is infinite and does not have zero divisors. In this ring,  $\det(\mathbf{A}) = 1+x$  is not a unit. Therefore, by Theorem 5.14, the matrix  $\mathbf{A}$  is not invertible and its rows do not generate  $R^2$ . Yet, the rows are independent.

(b) Lastly, consider the finite quotient ring  $R = \mathbb{F}_2[x]/\langle x^N - 1 \rangle$ . In this ring,  $1+x$  is a zero divisor. Therefore,  $\mathbf{A}$  is not invertible and the rows do not generate  $R^2$ . Since  $R$  is finite, Lemma 5.15 implies that the set of rows is dependent. Specifically, multiplying the second row by the scalar  $x^{N-1} + x^{N-2} + \cdots + x + 1 \in R$  yields the zero element  $(0, 0) \in R^2$ .

## Acknowledgement

This chapter is in part a reprint of the papers: Brian K. Butler and Paul H. Siegel, “Bounds on the Minimum Distance of Punctured Quasi-Cyclic LDPC Codes,” *IEEE Transactions on Information Theory*, to appear, vol. 59, 2013 and Brian K. Butler and Paul H. Siegel, “On Distance Properties of Quasi-Cyclic Protograph-Based LDPC Codes,” in *Proceedings of the IEEE International Symposium on Information Theory (ISIT 2010)*, Austin, TX, pp. 809–813, June 13–18, 2010. The dissertation author was the primary author of these papers.

## Bibliography

- [1] T. J. Richardson, "Error-floors of LDPC codes," in *Proc. of the 41st Annu. Allerton Conf.*, Monticello, IL, Oct. 2003, pp. 1426–1435.
- [2] X.-Y. Hu, E. Eleftheriou, and D. M. Arnold, "Regular and irregular progressive edge-growth Tanner graphs," *IEEE Trans. Inf. Theory*, vol. 51, no. 1, pp. 386–398, Jan. 2005.
- [3] T. Tian, C. R. Jones, J. D. Villasenor, and R. D. Wesel, "Selective avoidance of cycles in irregular LDPC code construction," *IEEE Trans. Commun.*, vol. 52, no. 8, pp. 1242–1247, Aug. 2004.
- [4] D. J. C. MacKay and M. C. Davey, "Evaluation of Gallager codes for short block length and high rate applications," in *Codes, Systems and Graphical Models (Minneapolis, MN, 1999)*, B. Marcus and J. Rosenthal, Eds. New York: Springer-Verlag, 2000, pp. 113–130.
- [5] R. M. Tanner, D. Sridhara, A. Sridharan, T. E. Fuja, and D. J. Costello, Jr., "LDPC block and convolutional codes based on circulant matrices," *IEEE Trans. Inf. Theory*, vol. 50, no. 12, pp. 2966–2984, Dec. 2004.
- [6] M. P. C. Fossorier, "Quasi-cyclic low-density parity-check codes from circulant permutation matrices," *IEEE Trans. Inf. Theory*, vol. 50, no. 8, pp. 1788–1793, Aug. 2004.
- [7] R. Smarandache and P. O. Vontobel, "Quasi-cyclic LDPC codes: Influence of proto- and Tanner-graph structure on minimum Hamming distance upper bounds," *IEEE Trans. Inf. Theory*, vol. 58, no. 2, pp. 585–607, Feb. 2012.
- [8] D. Divsalar, S. Dolinar, and C. R. Jones, "Construction of protograph LDPC codes with linear minimum distance," in *Proc. IEEE Int. Symp. on Inform. Theory*, Seattle, WA, Jul. 2006, pp. 664–668.
- [9] D. Divsalar, S. Dolinar, C. R. Jones, and K. Andrews, "Capacity-approaching protograph codes," *IEEE J. Sel. Areas Commun.*, vol. 27, no. 6, pp. 876–888, Aug. 2009.
- [10] S. Abu-Surra, D. Divsalar, and W. E. Ryan, "Enumerators for protograph-based ensembles of LDPC and generalized LDPC codes," *IEEE Trans. Inf. Theory*, vol. 57, no. 2, pp. 858–886, Feb. 2011.
- [11] CCSDS, *TM Synchronization and Channel Coding*, Recommended Standard 131.0-B-2, Consultative Committee for Space Data Systems, Reston, VA, Aug. 2011. [Online]. Available: <http://public.ccsds.org/publications/archive/131x0b2ec1.pdf>
- [12] J. Thorpe, "Low density parity check (LDPC) codes constructed from protographs," Jet Propulsion Laboratory, Pasadena, CA, Tech. Rep. INP Progress Report 42-154, Aug. 15, 2003.
- [13] T. J. Richardson and R. L. Urbanke, *Modern Coding Theory*. Cambridge, UK: Cambridge Univ. Press, 2008.

- [14] S. L. Sweatlock, "Asymptotic weight analysis of LDPC code ensembles," Ph.D. dissertation, Dept. Appl. and Comput. Math., Calif. Inst. Tech., Pasadena, Apr. 2008. [Online]. Available: <http://thesis.library.caltech.edu/1898/>
- [15] W. W. Peterson and E. J. Weldon, Jr., *Error-Correcting Codes*, 2nd ed. Cambridge, MA: The MIT Press, 1972.
- [16] C. C. Pinter, *A Book of Abstract Algebra*, 2nd ed. New York: McGraw-Hill, 1990.
- [17] M. Artin, *Algebra*. Englewood Cliffs, NJ: Prentice Hall, 1991.
- [18] Y. Wang, S. C. Draper, and J. S. Yedidia, "Hierarchical and high-girth QC LDPC codes," to appear in *IEEE Trans. Inf. Theory*, 2013. [Online]. Available: <http://dx.doi.org/10.1109/TIT.2013.2253512>
- [19] G. Richter, "Finding small stopping sets in the Tanner graphs of LDPC codes," in *Proc. 4th Int. Symp. on Turbo Codes*, Munich, Germany, Apr. 2006, pp. 1–5.
- [20] X.-Y. Hu, M. P. C. Fossorier, and E. Eleftheriou, "On the computation of the minimum distance of low-density parity-check codes," in *Proc. IEEE Int. Conf. on Commun.*, vol. 2, Paris, Jun. 2004, pp. 767–771.
- [21] D. Declercq and M. P. C. Fossorier, "Improved impulse method to evaluate the low weight profile of sparse binary linear codes," in *Proc. IEEE Int. Symp. on Inform. Theory*, Toronto, Canada, Jul. 2008, pp. 1963–1967.
- [22] R. G. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA: M.I.T. Press, 1963.
- [23] H. Park, S. Hong, J.-S. No, and D.-J. Shin, "Protograph design with multiple edges for regular QC LDPC codes having large girth," in *Proc. IEEE Int. Symp. on Inform. Theory*, St. Petersburg, Russia, Aug. 2011, pp. 918–922.
- [24] R. M. Tanner, D. Sridhara, and T. E. Fuja, "A class of group-structured LDPC codes," in *Proc. Int. Symp. Commun. Theory and Appl.*, Ambleside, U.K., Jul. 2001, pp. 365–370.
- [25] E. H. Connell, *Elements of Abstract and Linear Algebra*, Mar. 2004, unpublished. [Online]. Available: <http://www.math.miami.edu/~ec/book/>

## Chapter 6

# Sharp Bounds on the Spectral Radius of Nonnegative Matrices and Digraphs

The spectral radius  $\rho(\mathbf{A})$  of a matrix  $\mathbf{A}$  is the maximum modulus of its eigenvalues. We present bounds on  $\rho(\mathbf{A})$  that are often tighter and are applicable to a larger class of nonnegative matrices than previously reported. The bounds are particularly suited to matrices which are sparse.

We complete the chapter by applying these bounds to digraphs and deriving their associated equality conditions which relate to the outdegree regularity of the digraph. Finally, we show that the equality conditions may be achieved only for very specific values of the digraph's spectral radius.

We were motivated to explore these bounds due to interest in understanding the dynamic behavior of trapping sets. As described in Chapter 4, the detrimental LLR growth rate is limited by the spectral radius of the trapping set's digraph description.

### 6.1 Introduction

Let  $\mathbf{A} = (a_{ij})$ ,  $1 \leq i, j \leq n$  be an  $n \times n$  nonnegative matrix, *i.e.*, a matrix in which the elements are real and greater than or equal to zero. The eigenvalues of  $\mathbf{A}$  are the complex roots of the characteristic equation  $\det(\mathbf{A} - \mu\mathbf{I}) = 0$ . The set of distinct eigenvalues is called the *spectrum* of  $\mathbf{A}$ , denoted  $\sigma(\mathbf{A}) = \{\mu_1, \dots, \mu_m\}$ , and the *spectral radius* (or *Perron root* in this context) of  $\mathbf{A}$  is the real number  $\rho(\mathbf{A}) = \max \{|\mu| : \mu \in \sigma(\mathbf{A})\}$ .

We say that the complex column vector  $\mathbf{x} = (x_1, \dots, x_n)^T$  is *nonnegative*, or  $\mathbf{x} \geq \mathbf{0}$ ,

if every element is real and greater than or equal to zero. If every element  $x_i$  of the vector  $\mathbf{x}$  is real and strictly greater than zero, we say that  $\mathbf{x}$  is *positive* or  $\mathbf{x} > \mathbf{0}$ . Furthermore, there is a nonnegative vector  $\mathbf{x} \neq \mathbf{0}$ , such that  $\mathbf{Ax} = \rho(\mathbf{A})\mathbf{x}$ . It is well known [1, p. 503] that for any nonnegative matrix  $\mathbf{A}$ , its spectral radius  $\rho(\mathbf{A})$  is an eigenvalue. Moreover, if  $\mathbf{A}$  has a positive eigenvector, the corresponding eigenvalue is  $\rho(\mathbf{A})$  [1, p. 493].

In this chapter, we derive several new bounds on the spectral radius of nonnegative matrices which we then use to bound the spectral radius of a large class of digraphs. Our results generalize those found in Zhang & Li [2], Kolotilina [3], Xu & Xu [4], and Güngör & Das [5]. With respect to the bounds of Liu [6], we find new equality conditions when they are applied to digraphs. (For a survey of prior work on the spectral radius of digraphs, see Brualdi [7].)

The nonnegative  $n \times n$  matrix  $\mathbf{A}$ , with  $n \geq 2$ , is said to be *reducible* if there exists a permutation matrix  $\mathbf{P}$  such that  $\mathbf{PAP}^T = \begin{bmatrix} \mathbf{X} & \mathbf{Y} \\ \mathbf{0} & \mathbf{Z} \end{bmatrix}$  where  $\mathbf{X}$  and  $\mathbf{Z}$  are square submatrices. Otherwise,  $\mathbf{A}$  is said to be *irreducible*.

Let  $r_i(\mathbf{A})$  denote the sum of the elements along the  $i$ th row of  $\mathbf{A}$ ; that is

$$r_i(\mathbf{A}) = \sum_{j=1}^n a_{ij}$$

for  $i \in \{1, \dots, n\}$ . The following two classical results bound the spectral radius.

**Theorem 6.1 (Frobenius)** *Let  $\mathbf{A} = (a_{ij})$  be an  $n \times n$  nonnegative matrix with spectral radius  $\rho(\mathbf{A})$  and row sums  $r_i(\mathbf{A})$ ,  $i \in \{1, \dots, n\}$ . Then*

$$\min_i r_i(\mathbf{A}) \leq \rho(\mathbf{A}) \leq \max_i r_i(\mathbf{A}). \quad (6.1)$$

*Moreover, if  $\mathbf{A}$  is an irreducible matrix, then equality holds on either side (and hence both sides) of (6.1) if and only if all row sums of  $\mathbf{A}$  are equal.*

*Proof:* See Minc [8, pp. 24–26]. ■

**Theorem 6.2 ([1, 2])** *Let  $\mathbf{A}$  be an  $n \times n$  nonnegative matrix with spectral radius  $\rho(\mathbf{A})$  and  $\mathbf{x} = (x_1, \dots, x_n)^T$  be a positive column vector. Then*

$$\min_{1 \leq i \leq n} \left[ \frac{(\mathbf{Ax})_i}{x_i} \right] \leq \rho(\mathbf{A}) \leq \max_{1 \leq i \leq n} \left[ \frac{(\mathbf{Ax})_i}{x_i} \right]. \quad (6.2)$$

*Moreover, if  $\mathbf{A}$  is an irreducible matrix, then equality holds on either side (and hence both sides) of (6.2) if and only if the vector  $\mathbf{x}$  is an eigenvector of  $\mathbf{A}$  corresponding to  $\rho(\mathbf{A})$ .*

*Proof:* By the assumption that  $x_i > 0$ , for all  $i \in \{1, \dots, n\}$ , the diagonal matrix  $\mathbf{D} = \text{diag}(x_1, \dots, x_n)$  is invertible. Since  $\mathbf{A}$  and  $\mathbf{D}^{-1}\mathbf{A}\mathbf{D}$  are similar matrices, they have identical eigenvalues, and therefore  $\rho(\mathbf{A}) = \rho(\mathbf{D}^{-1}\mathbf{A}\mathbf{D})$ . The row sums of  $\mathbf{D}^{-1}\mathbf{A}\mathbf{D}$  are given by

$$r_i(\mathbf{D}^{-1}\mathbf{A}\mathbf{D}) = \sum_{j=1}^n \frac{a_{ij}x_j}{x_i} = \frac{(\mathbf{A}\mathbf{x})_i}{x_i}, \quad (6.3)$$

for all  $i \in \{1, \dots, n\}$ . Thus, Theorem 6.1, with  $\mathbf{D}^{-1}\mathbf{A}\mathbf{D}$  substituted for  $\mathbf{A}$ , implies (6.2).

Since  $\mathbf{A}$  and  $\mathbf{D}^{-1}\mathbf{A}\mathbf{D}$  have identical *sparsity patterns* (i.e., the locations of its nonzero entries),  $\mathbf{A}$  is irreducible if and only if  $\mathbf{D}^{-1}\mathbf{A}\mathbf{D}$  is irreducible. Therefore, if either equality in (6.2) holds, then by the equality condition of Theorem 6.1, (6.3) equals  $\rho(\mathbf{A})$  for all  $i$ , which yields  $\mathbf{A}\mathbf{x} = \rho(\mathbf{A})\mathbf{x}$ , as desired. Conversely, if  $\mathbf{x} > \mathbf{0}$  and  $\mathbf{A}\mathbf{x} = \rho(\mathbf{A})\mathbf{x}$ , then the row sums in (6.3) are equal to  $\rho(\mathbf{A})$  for all  $i$ , forcing equality on both sides of (6.2). ■

In the following section, we present new bounds, along with equality conditions, on the spectral radius of general nonnegative matrices. We further refine the equality conditions in Section 6.3 and then apply these bounds and equality conditions to general digraphs in Section 6.4, illustrating them with a detailed example. Lastly, we conclude the chapter.

## 6.2 Bounds on the Spectral Radius of Nonnegative Matrices

In this section, we characterize the spectral radius of nonnegative matrices with nonzero row sums. It is well known [4, 6] that deleting the zero rows and their corresponding columns (i.e., the columns having the same indices as the zero rows) leaves unaffected the nonzero entries in the spectrum of a matrix. Since the column removal may reveal new all-zero rows, this process may have to be applied multiple times to finally produce a matrix with nonzero row sums. Once this is achieved, the bounds of this section may be applied to the reduced matrix.

Let  $\mathbf{A} = (a_{ij})$  be an  $n \times n$  matrix. We denote the  $(i, j)$ th entry of matrix  $\mathbf{A}^k$  by  $a_{ij}^{(k)}$ , noting that

$$a_{ij}^{(k)} = \overbrace{\sum_{s=1}^n \sum_{t=1}^n \cdots \sum_{y=1}^n}^{k-1 \text{ sums}} \overbrace{a_{is}a_{st} \cdots a_{yj}}^{k \text{ terms}} \quad \text{and} \quad a_{ij}^{(0)} = \delta_{ij},$$

where  $\delta_{ij}$  is the Kronecker delta (i.e., 1 if  $i = j$  and 0 if  $i \neq j$ ). Let  $r_i(\mathbf{A}^k)$  denote the sum of the  $i$ th row of  $\mathbf{A}^k$ , that is,  $r_i(\mathbf{A}^k) = \sum_{j=1}^n a_{ij}^{(k)}$ . Using the fact that, for any  $n \times n$  matrix  $\mathbf{B}$ , the row sums of the product  $\mathbf{A}\mathbf{B}$  are given by

$$r_i(\mathbf{A}\mathbf{B}) = \sum_{j=1}^n \sum_{k=1}^n a_{ij}b_{jk} = \sum_{j=1}^n a_{ij}r_j(\mathbf{B}), \quad (6.4)$$

one can derive additional useful row-sum expressions such as

$$r_i(\mathbf{A}^k) = \sum_{j=1}^n a_{ij}^{(k-s)} r_j(\mathbf{A}^s),$$

for all  $0 \leq s \leq k$ . We will make frequent use of the column vector  $\mathbf{x} = (r_1(\mathbf{A}^k), \dots, r_n(\mathbf{A}^k))^T$  and the diagonal matrix  $\mathbf{D} = \text{diag}(r_1(\mathbf{A}^k), \dots, r_n(\mathbf{A}^k))$ , for some integer  $k \geq 0$ . Then,

$$(\mathbf{A}^s \mathbf{x})_i = \sum_{j=1}^n a_{ij}^{(s)} r_j(\mathbf{A}^k) = r_i(\mathbf{A}^{s+k}) \quad (6.5)$$

and, assuming that the row sums of  $\mathbf{A}^k$  are nonzero,

$$r_i(\mathbf{D}^{-1} \mathbf{A}^s \mathbf{D}) = \frac{\sum_{j=1}^n a_{ij}^{(s)} r_j(\mathbf{A}^k)}{r_i(\mathbf{A}^k)} = \frac{r_i(\mathbf{A}^{s+k})}{r_i(\mathbf{A}^k)}, \quad (6.6)$$

for any  $s \geq 0$  and all  $i \in \{1, \dots, n\}$ . Also, as Liu [6] remarked, if the row sums of a nonnegative matrix  $\mathbf{A}$  are nonzero, then so are the row sums of  $\mathbf{A}^k$ , for  $k \geq 0$ . We now prove a theorem that provides a generalization of the bounds in Xu & Xu [4], but first we will need the following two lemmas. Recall from Perron-Frobenius theory that irreducible matrices have a unique (up to a scale factor) positive eigenvector corresponding to the spectral radius.

**Lemma 6.3** *Let  $\mathbf{A}$  be an  $n \times n$  nonnegative matrix with spectral radius  $\rho(\mathbf{A})$ . If  $\mathbf{A}^L$  is irreducible, for some  $L > 0$ , then  $\mathbf{A}$  is also irreducible and the positive eigenvectors of  $\mathbf{A}$  and  $\mathbf{A}^L$  agree up to a scale factor.*

*Proof:* First, take  $\mathbf{A}$  to be reducible. In this case, there exists a permutation matrix  $\mathbf{P}$  such that  $\mathbf{PAP}^T = \begin{pmatrix} \mathbf{X} & \mathbf{Y} \\ \mathbf{0} & \mathbf{Z} \end{pmatrix}$ . Clearly  $\mathbf{PA}^L \mathbf{P}^T$  is also upper triangular, and hence  $\mathbf{A}^L$  is reducible too. As this is a contradiction,  $\mathbf{A}$  is irreducible.

Since  $\mathbf{A}$  is irreducible, it has a unique positive eigenvector, which we denote  $\mathbf{x}$ , corresponding to  $\rho(\mathbf{A})$ . Repeatedly left-multiplying  $\mathbf{Ax} = \rho(\mathbf{A})\mathbf{x}$  by  $\mathbf{A}$  implies that  $\mathbf{A}^s \mathbf{x} = \rho(\mathbf{A})^s \mathbf{x} = \rho(\mathbf{A}^s) \mathbf{x}$ , for any integer  $s \geq 0$ . In particular, setting  $s = L$ , we conclude that  $\mathbf{x}$  is the unique (up to a scale factor) positive eigenvector of  $\mathbf{A}^L$  ■

**Lemma 6.4** *Let  $\mathbf{A}$  be an  $n \times n$  matrix with spectral radius  $\rho(\mathbf{A}) \neq 0$ . If, for some  $k \geq 0$ ,  $\mathbf{x} = (r_1(\mathbf{A}^k), \dots, r_n(\mathbf{A}^k))^T$  is an eigenvector of  $\mathbf{A}$  corresponding to  $\rho(\mathbf{A})$ , then so is  $\mathbf{y} = (r_1(\mathbf{A}^{k+1}), \dots, r_n(\mathbf{A}^{k+1}))^T$ , and  $\mathbf{y} = \rho(\mathbf{A})\mathbf{x}$ .*

*Proof:* Since  $(\mathbf{Ax})_i = \sum_{j=1}^n a_{ij} r_j(\mathbf{A}^k) = r_i(\mathbf{A}^{k+1}) = y_i$  for all  $i \in \{1, \dots, n\}$ , then  $\mathbf{Ax} = \mathbf{y} = \rho(\mathbf{A})\mathbf{x}$ . ■

**Theorem 6.5** Let  $\mathbf{A}$  be an  $n \times n$  nonnegative matrix with spectral radius  $\rho(\mathbf{A})$  and nonzero row sums. Then, for any integers  $M > 0$ ,  $N \geq 0$ , and  $k \geq 0$ ,

$$\begin{aligned} \min_{\substack{1 \leq i \leq n \\ 1 \leq j \leq n}} \left[ \left( \frac{r_i(\mathbf{A}^{k+M}) r_j(\mathbf{A}^{k+N})}{r_i(\mathbf{A}^k) r_j(\mathbf{A}^k)} \right)^{\frac{1}{M+N}} : a_{ij}^{(M)} > 0 \right] &\leq \rho(\mathbf{A}) \\ &\leq \max_{\substack{1 \leq i \leq n \\ 1 \leq j \leq n}} \left[ \left( \frac{r_i(\mathbf{A}^{k+M}) r_j(\mathbf{A}^{k+N})}{r_i(\mathbf{A}^k) r_j(\mathbf{A}^k)} \right)^{\frac{1}{M+N}} : a_{ij}^{(M)} > 0 \right]. \end{aligned} \quad (6.7)$$

Moreover, if  $\mathbf{A}^{M+N}$  is an irreducible matrix, then equality holds in either side (and hence both sides) of (6.7) if and only if  $\mathbf{x} = (r_1(\mathbf{A}^k), \dots, r_n(\mathbf{A}^k))^T$  is an eigenvector of  $\mathbf{A}$ .

*Proof:* Define the invertible diagonal matrix  $\mathbf{D} = \text{diag}(r_1(\mathbf{A}^k), \dots, r_n(\mathbf{A}^k))$ . Since  $\mathbf{A}^{M+N}$  and  $\mathbf{D}^{-1} \mathbf{A}^{M+N} \mathbf{D}$  are similar matrices, we have

$$\rho(\mathbf{A}^{M+N}) = \rho(\mathbf{D}^{-1} \mathbf{A}^{M+N} \mathbf{D}) \quad (6.8)$$

$$\leq \max_{1 \leq i \leq n} r_i(\mathbf{D}^{-1} \mathbf{A}^{M+N} \mathbf{D}) \quad (6.9)$$

where the inequality in (6.9) follows from Theorem 6.1. The row sums in (6.9) can be formulated as

$$\begin{aligned} r_i(\mathbf{D}^{-1} \mathbf{A}^M \mathbf{A}^N \mathbf{D}) &= r_i((\mathbf{D}^{-1} \mathbf{A}^M \mathbf{D})(\mathbf{D}^{-1} \mathbf{A}^N \mathbf{D})) \\ &= \sum_{j=1}^n (\mathbf{D}^{-1} \mathbf{A}^M \mathbf{D})_{ij} r_j(\mathbf{D}^{-1} \mathbf{A}^N \mathbf{D}) \end{aligned} \quad (6.10)$$

$$\leq r_i(\mathbf{D}^{-1} \mathbf{A}^M \mathbf{D}) \max_j \left\{ r_j(\mathbf{D}^{-1} \mathbf{A}^N \mathbf{D}) : a_{ij}^{(M)} > 0 \right\} \quad (6.11)$$

$$= \max_j \left\{ r_i(\mathbf{D}^{-1} \mathbf{A}^M \mathbf{D}) r_j(\mathbf{D}^{-1} \mathbf{A}^N \mathbf{D}) : a_{ij}^{(M)} > 0 \right\}, \quad (6.12)$$

where (6.10) follows from (6.4). The restricted maximizations in (6.11) and (6.12) make use of the fact that the sparsity patterns of  $\mathbf{A}^M$  and  $\mathbf{D}^{-1} \mathbf{A}^M \mathbf{D}$  are the same.

Applying (6.6) to the factors in the product in (6.12), with  $s = M$  and  $s = N$ , respectively, we conclude that

$$\rho(\mathbf{A}^{M+N}) \leq \max_{\substack{1 \leq i \leq n \\ 1 \leq j \leq n}} \left[ \frac{r_i(\mathbf{A}^{k+M}) r_j(\mathbf{A}^{k+N})}{r_i(\mathbf{A}^k) r_j(\mathbf{A}^k)} : a_{ij}^{(M)} > 0 \right].$$

Since  $\rho(\mathbf{A}^{M+N}) = \rho(\mathbf{A})^{M+N}$ , taking the  $(M+N)$ th root of the inequality above yields the upper bound in (6.7). The proof of the lower bound in (6.7) is completely analogous.

We now show the equality condition, assuming that  $\mathbf{A}^{M+N}$  is irreducible. If equality holds in the upper bound of (6.7), then (6.9) must also hold with equality. Then, by the equality condition of Theorem 6.1 applied to  $\mathbf{D}^{-1}\mathbf{A}^{M+N}\mathbf{D}$ , we conclude that, in fact,

$$\rho(\mathbf{D}^{-1}\mathbf{A}^{M+N}\mathbf{D}) = r_i(\mathbf{D}^{-1}\mathbf{A}^{M+N}\mathbf{D}),$$

for all  $i \in \{1, \dots, n\}$ . Referring to (6.6), we know that  $r_i(\mathbf{D}^{-1}\mathbf{A}^{M+N}\mathbf{D}) = (\mathbf{A}^{M+N}\mathbf{x})_i/x_i$ , for all  $i$ , where  $\mathbf{x} = (r_1(\mathbf{A}^k), \dots, r_n(\mathbf{A}^k))^T$ . This shows that  $\mathbf{x}$  is a positive eigenvector of  $\mathbf{A}^{M+N}$ . Now we apply Lemma 6.3 to show that  $\mathbf{x}$  is an eigenvector of  $\mathbf{A}$ , as desired.

Conversely, suppose that  $\mathbf{x} = (r_1(\mathbf{A}^k), \dots, r_n(\mathbf{A}^k))^T$  is an eigenvector of  $\mathbf{A}$ . Then Lemma 6.4 implies  $\rho(\mathbf{A})^s = r_i(\mathbf{A}^{k+s})/r_i(\mathbf{A}^k)$ , for all  $i \in \{1, \dots, n\}$  and  $s \geq 0$ . This shows that both the upper and lower bounds in (6.7) hold with equality. ■

**Remark 6.1** The nonzero row-sum assumption is not required for  $k = 0$ .

The bounds of Xu & Xu [4] are a special case of Theorem 6.5, where  $k = M = N = 1$ . As another special case, in which  $N = 0$ , we recover the following bounds due to Liu [6].

**Corollary 6.6 (Liu [3, 6, 9])** *For any integer  $L > 0$ ,*

$$\min_{1 \leq i \leq n} \left[ \frac{r_i(\mathbf{A}^{k+L})}{r_i(\mathbf{A}^k)} \right]^{1/L} \leq \rho(\mathbf{A}) \leq \max_{1 \leq i \leq n} \left[ \frac{r_i(\mathbf{A}^{k+L})}{r_i(\mathbf{A}^k)} \right]^{1/L}. \quad (6.13)$$

*Moreover, if  $\mathbf{A}^L$  is an irreducible matrix, then equality holds on either side (and hence both sides) of (6.13) if and only if  $\mathbf{x} = (r_1(\mathbf{A}^k), \dots, r_n(\mathbf{A}^k))^T$  is an eigenvector of  $\mathbf{A}$ .*

Since the new bounds of Theorem 6.5 depend upon the sparsity pattern of  $\mathbf{A}^M$ , they may produce sharper bounds than the bounds of Liu when critical entries of  $\mathbf{A}$  are zero. However, from the proof of Theorem 6.5 it is clear that Corollary 6.6 with  $L = M + N$  is at least as tight as Theorem 6.5. Nevertheless, in some applications it may be prohibitively complex to compute  $r_i(\mathbf{A}^{M+N+k})$  for Corollary 6.6 as opposed to examining the sparsity pattern of  $\mathbf{A}^M$  and computing  $r_i(\mathbf{A}^{M+k})$  and  $r_i(\mathbf{A}^{N+k})$  as required by Theorem 6.5.

Next, we briefly generalize bounds of a similar form developed by Kolotilina [3, §5].

**Theorem 6.7** *Let  $\mathbf{A} = (a_{ij})$  be an  $n \times n$  nonnegative matrix with spectral radius  $\rho(\mathbf{A})$  and*

row sums  $r_1(\mathbf{A}), \dots, r_n(\mathbf{A})$ , all nonzero. Then

$$\begin{aligned} \min_{\substack{1 \leq i \leq n \\ 1 \leq j \leq n}} \left[ \frac{r_i^\alpha(\mathbf{A}^{k+L}) r_j^{1-\alpha}(\mathbf{A}^{k+L})}{r_i^\alpha(\mathbf{A}^k) r_j^{1-\alpha}(\mathbf{A}^k)} : a_{ij}^{(L)} > 0 \right] &\leq \rho(\mathbf{A}^L) \\ &\leq \max_{\substack{1 \leq i \leq n \\ 1 \leq j \leq n}} \left[ \frac{r_i^\alpha(\mathbf{A}^{k+L}) r_j^{1-\alpha}(\mathbf{A}^{k+L})}{r_i^\alpha(\mathbf{A}^k) r_j^{1-\alpha}(\mathbf{A}^k)} : a_{ij}^{(L)} > 0 \right] \end{aligned} \quad (6.14)$$

for any integer  $L \geq 1$  and  $\alpha$  such that  $0 \leq \alpha \leq 1$ .

*Proof:* We apply the bounds

$$\min_{\substack{1 \leq i \leq n \\ 1 \leq j \leq n}} \left[ r_i^\alpha(\mathbf{A}) r_j^{1-\alpha}(\mathbf{A}) : a_{ij} > 0 \right] \leq \rho(\mathbf{A}) \leq \max_{\substack{1 \leq i \leq n \\ 1 \leq j \leq n}} \left[ r_i^\alpha(\mathbf{A}) r_j^{1-\alpha}(\mathbf{A}) : a_{ij} > 0 \right],$$

which were proved in [3, §5], to  $\mathbf{D}^{-1} \mathbf{A}^L \mathbf{D}$ , where the matrix  $\mathbf{D} = \text{diag}(r_1(\mathbf{A}^k), \dots, r_n(\mathbf{A}^k))$ .

■

**Remark 6.2** Note that (6.14) with  $\alpha = 0.5$  is equivalent to (6.7) with  $L = M = N$ .

The theorems and corollaries presented in this section and the previous have been structured around row sums. Since  $\rho(\mathbf{A}^T) = \rho(\mathbf{A})$ , similar bounds may be obtained starting with the column sums and the left eigenvectors of  $\mathbf{A}$ .

### 6.3 Further Equality Conditions on the Spectral Radius Bounds

In this section we develop alternative equality conditions for (6.7) by generalizing the proofs in Zhang & Li [2, §2]. Like Zhang & Li, we divide the equality conditions for these bounds into two cases corresponding to whether  $\mathbf{A}^{M+N}$  is irreducible or reducible. We address the former first, which readily follows from (6.5).

**Corollary 6.8 (to Theorem 6.5)** *If  $\mathbf{A}^{M+N}$  is an irreducible matrix, then equality holds on either side (and hence both sides) of (6.7) if and only if  $\rho(\mathbf{A}) = r_i(\mathbf{A}^{k+1})/r_i(\mathbf{A}^k)$  for all  $i \in \{1, \dots, n\}$ .*

The case in which  $\mathbf{A}^{M+N}$  is reducible requires some background concerning imprimitive matrices, which we review next. A nonnegative irreducible matrix  $\mathbf{A}$  having only one eigenvalue with a modulus equal to  $\rho(\mathbf{A})$  is said to be *primitive*. If a nonnegative irreducible matrix  $\mathbf{A}$  has  $h > 1$  eigenvalues with modulus  $\rho(\mathbf{A})$ , it is said to be *imprimitive* or a cyclic matrix, and  $h$  is known as the *index of imprimitivity*.

**Lemma 6.9** (See [10, §3.4].) *Let  $\mathbf{A}$  be an  $n \times n$  irreducible nonnegative matrix with index of imprimitivity equal to  $h$ . Let  $L > 0$  be an integer and  $r$  be the greatest common divisor (gcd) of  $h$  and  $L$ . Then  $\mathbf{A}^L$  is reducible if and only if  $r > 1$ . In general, there is a permutation matrix  $\mathbf{P}$  that symmetrically permutes  $\mathbf{A}^L$  to the block diagonal matrix*

$$\mathbf{P}\mathbf{A}^L\mathbf{P}^T = \begin{bmatrix} \mathbf{C}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{C}_r \end{bmatrix}, \quad (6.15)$$

where each  $\mathbf{C}_\ell$  matrix is an  $n_\ell \times n_\ell$  irreducible nonnegative matrix. Furthermore, for  $r > 1$ ,  $\mathbf{P}$  also symmetrically permutes  $\mathbf{A}$  to form

$$\mathbf{P}\mathbf{A}\mathbf{P}^T = \begin{bmatrix} \mathbf{0} & \mathbf{A}_{12} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_{23} & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{A}_{r-1,r} \\ \mathbf{A}_{r,1} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \end{bmatrix}, \quad (6.16)$$

where the all-zero submatrices along the diagonal are square and of order  $n_1, \dots, n_r$ , respectively. When (6.16) holds with  $r > 1$ , we say that  $\mathbf{A}$  is  $r$ -cyclic. The block (i.e., submatrix)  $\mathbf{A}_{\ell,m}$  is  $n_\ell \times n_m$ , for all  $\ell \in \{1, \dots, r\}$  and  $m = (\ell \bmod r) + 1$ . Moreover,

$$\begin{aligned} \mathbf{C}_1 &= [\mathbf{A}_{12}\mathbf{A}_{23} \cdots \mathbf{A}_{r-1,r}\mathbf{A}_{r,1}]^{(L/r)} \\ \mathbf{C}_2 &= [\mathbf{A}_{23}\mathbf{A}_{34} \cdots \mathbf{A}_{r,1}\mathbf{A}_{12}]^{(L/r)} \\ &\dots \\ \mathbf{C}_r &= [\mathbf{A}_{r,1}\mathbf{A}_{12} \cdots \mathbf{A}_{r-2,r-1}\mathbf{A}_{r-1,r}]^{(L/r)}, \end{aligned}$$

and  $\rho(\mathbf{A}^L) = \rho(\mathbf{C}_1) = \cdots = \rho(\mathbf{C}_r)$ .

**Remark 6.3** Note that for a given matrix, its  $r$  value may vary depending on the specified value of  $L$ , since  $r = \gcd(h, L)$ .

Recall from Perron-Frobenius theory that each square nonnegative matrix  $\mathbf{A}$  has at least one nonnegative eigenvector  $\mathbf{x} \neq \mathbf{0}$ , such that  $\mathbf{A}\mathbf{x} = \rho(\mathbf{A})\mathbf{x}$  [1, p. 503]. Additionally, if  $\mathbf{A}$  is reducible, it may have a positive eigenvector, or even multiple linearly independent positive eigenvectors associated with  $\rho(\mathbf{A})$ . In the case of the reducible matrix  $\mathbf{A}^L$  of Lemma 6.9, having eigenvalue  $\rho(\mathbf{A}^L)$  with an algebraic multiplicity equal to  $r$ , there are  $r$  linearly independent

positive eigenvectors of  $\mathbf{A}^L$  corresponding to  $\rho(\mathbf{A}^L)$ , a fact which we utilize to prove the next theorem.

**Theorem 6.10** *Let  $\mathbf{A}$  be an  $n \times n$  irreducible nonnegative matrix with spectral radius  $\rho(\mathbf{A})$ , index of imprimitivity equal to  $h$ , and nonzero row sums. Let  $M > 0$ ,  $N \geq 0$ , and  $k \geq 0$  be integers and  $r = \gcd(h, M + N)$ . If  $\mathbf{A}^{M+N}$  is reducible ( $r > 1$ ), then equality holds on either side (and hence both sides) of (6.7) if and only if*

$$\frac{r_i(\mathbf{A}^{k+1})}{r_i(\mathbf{A}^k)} = c_{m(i)}, \quad (6.17)$$

for all  $i \in \{1, \dots, n\}$ , where the  $i$ th row of  $\mathbf{A}$  has been assigned to the  $\ell$ th block,  $\ell \in \{1, \dots, r\}$ , using the mapping  $\ell = m(i)$  according to (6.15) and (6.16), and  $c_\ell$  is a constant for the  $\ell$ th block. Moreover,  $\rho(\mathbf{A})^r = \prod_{\ell=1}^r c_\ell$ .

*Proof:* Let  $\mathbf{x} = (r_1(\mathbf{A}^k), \dots, r_n(\mathbf{A}^k))^T$ . Without loss of generality we will assume that  $\mathbf{A}$  is in the form of (6.16) and  $\mathbf{A}^{M+N}$  is in block diagonal form (6.15), where each  $\mathbf{C}_\ell$  is  $n_\ell \times n_\ell$  and irreducible. Let  $\mathbf{x}$  be divided into  $r$  subvectors, such that  $\mathbf{x} = (\mathbf{w}_1^T, \dots, \mathbf{w}_r^T)^T$  and the  $\ell$ th subvector  $\mathbf{w}_\ell$  has  $n_\ell$  elements.

If either equality holds in (6.7), then by applying Lemma 6.9 and Theorem 6.2 block-by-block, it can be shown that  $\mathbf{x}$  is a positive eigenvector of  $\mathbf{A}^{M+N}$ . Now, we show that  $\mathcal{X} = \{(g_1 \mathbf{w}_1^T, \dots, g_r \mathbf{w}_r^T)^T : g_1, \dots, g_r \in \mathbb{R}^+\}$ , is the complete set of positive eigenvectors of  $\mathbf{A}^{M+N}$ , where  $\mathbb{R}^+$  is the set of real, positive numbers. If we evaluate  $\mathbf{y}$  in  $\mathbf{A}^{M+N} \mathbf{y} = \rho(\mathbf{A}^{M+N}) \mathbf{y}$ , where  $\mathbf{y} \in \mathcal{X}$ , it readily reduces to  $g_\ell \mathbf{C}_\ell \mathbf{w}_\ell = \rho(\mathbf{A}^{M+N}) g_\ell \mathbf{w}_\ell$  or simply  $\mathbf{C}_\ell \mathbf{w}_\ell = \rho(\mathbf{A}^{M+N}) \mathbf{w}_\ell$ , for all  $\ell \in \{1, \dots, r\}$  due to the block diagonal form of  $\mathbf{A}^{M+N}$ . Since  $\mathbf{C}_\ell$  is irreducible,  $\mathbf{w}_\ell$  is the unique positive eigenvector of  $\mathbf{C}_\ell$ , up to a positive scale factor, for all  $\ell \in \{1, \dots, r\}$ . Hence, there can be no other positive eigenvectors of  $\mathbf{A}^{M+N}$  beyond set  $\mathcal{X}$ .

Since  $\mathbf{A}$  is irreducible, it must have a unique (up to a scale factor) positive eigenvector that we shall call  $\mathbf{y}'$ . Additionally,  $\mathbf{y}'$  is in the set  $\mathcal{X}$ , because every eigenvector of  $\mathbf{A}$  must also be an eigenvector of  $\mathbf{A}^{M+N}$ . For the first block, with  $1 \leq i \leq n_1$ , we find

$$\frac{r_i(\mathbf{A}^{k+1})}{r_i(\mathbf{A}^k)} = \frac{(\mathbf{A}\mathbf{x})_i}{x_i} = \frac{(\mathbf{A}_{12}\mathbf{w}_2)_i}{(\mathbf{w}_1)_i} = \frac{\rho(\mathbf{A})g_1(\mathbf{w}_1)_i}{g_2(\mathbf{w}_1)_i} = \rho(\mathbf{A})\frac{g_1}{g_2}, \quad (6.18)$$

where we take  $g_1, \dots, g_r$  to be real, positive constants that depend upon  $\mathbf{y}'$ . Note that (6.18) is constant within the first block. The other blocks ( $1 < \ell \leq r$ ) follow similarly, confirming (6.17).

Conversely, suppose that (6.17) is true for all  $i \in \{1, \dots, n\}$ . Then, for the first block ( $\ell = 1$ ), using (6.4), (6.5), and (6.17), we see that

$$\begin{aligned} \frac{r_i(\mathbf{A}^{k+L})}{r_i(\mathbf{A}^k)} &= \frac{\sum_{j=1}^n a_{ij}^{(L-1)} r_j(\mathbf{A}^{k+1})}{r_i(\mathbf{A}^k)} = \frac{c_1 \sum_{j=1}^n a_{ij}^{(L-1)} r_j(\mathbf{A}^k)}{r_i(\mathbf{A}^k)} = \dots \\ &= \frac{c_1 c_2 \dots c_L r_i(\mathbf{A}^k)}{r_i(\mathbf{A}^k)} = c_1 c_2 \dots c_L, \end{aligned}$$

for all  $1 \leq i \leq n_1$  and  $1 \leq L \leq r$ . For values of  $L$  greater than  $r$  the indexing of  $c_\ell$  must wrap around to 1. Thus, for the  $i$ th row, which is in block  $m(i)$ , we have

$$\frac{r_i(\mathbf{A}^{k+M})}{r_i(\mathbf{A}^k)} = \prod_{\ell=m(i)}^{m(i)+M-1} c_{[(\ell-1) \bmod r]+1} \quad \text{and} \quad \frac{r_i(\mathbf{A}^{k+N})}{r_i(\mathbf{A}^k)} = \prod_{\ell=m(i)}^{m(i)+N-1} c_{[(\ell-1) \bmod r]+1}, \quad (6.19)$$

for all  $i \in \{1, \dots, n\}$ . Recognize that the values of (6.19) are still dependent on the row index  $i$ . Next, we limit our consideration of (6.19) to rows  $i$  and  $j$ , respectively, such that  $a_{ij}^{(M)} > 0$  as in (6.7). Thus, row  $j$  is in block  $m(j)$ , where  $m(j) = [(m(i) + M - 1) \bmod r] + 1$ . Therefore, forming the product of the row-sum ratios in (6.19), with such a restriction, yields

$$\begin{aligned} \frac{r_i(\mathbf{A}^{k+M})}{r_i(\mathbf{A}^k)} \frac{r_j(\mathbf{A}^{k+N})}{r_j(\mathbf{A}^k)} &= \prod_{\ell=m(i)}^{m(i)+M-1} c_{[(\ell-1) \bmod r]+1} \prod_{t=m(i)+M}^{m(i)+M+N-1} c_{[(t-1) \bmod r]+1} \\ &= \left( \prod_{\ell=1}^r c_\ell \right)^{(M+N)/r}, \end{aligned} \quad (6.20)$$

for all  $i \in \{1, \dots, n\}$  and all  $j$  such that  $a_{ij}^{(M)} > 0$ . That is, the product of constants in (6.20) includes exactly  $(M+N)/r$  occurrences of each constant  $c_\ell$ . This results in a value for (6.20) that is independent of  $i$  and  $m(i)$  and, hence, equality is true on both sides of (6.7) with  $\rho(\mathbf{A})^r = \prod_{\ell=1}^r c_\ell$ . ■

## 6.4 Bounds Applied to Digraphs

In this section, we cast the results of previous sections in graph-theoretic terms and derive further equality conditions related to the integrality of the adjacency matrix. We first review some basic concepts and terminology related to digraphs. (For a more complete treatment, we refer the reader to [8, §4.3] and [10, chap. 3].) Let  $G = (V, E)$  be a directed graph, or *digraph*, with a nonempty set of  $n$  vertices,  $V = \{v_1, \dots, v_n\}$ , and a collection  $E$  of directed edges or

*arcs*. The digraph is called *simple* if it contains no self-loops or multiarcs. A self-loop is an arc joining a vertex to itself and multiarcs are two or more arcs with the same initial and terminal vertices. In contrast to the results of Zhang & Li [2], Xu & Xu [4], and Güngör & Das [5] which we generalize below, our results are not limited to simple digraphs.

The *adjacency matrix*  $\mathbf{A}(G)$  of any digraph  $G$  is the nonnegative matrix whose  $(i, j)$ th entry  $a_{ij}$  is the number of arcs directed from vertex  $v_i$  to vertex  $v_j$  in  $G$ . The *spectral radius*  $\rho(G)$  of digraph  $G$  is defined to be the spectral radius of  $\mathbf{A}(G)$ . In a digraph  $G$ , a *directed walk* is an alternating sequence of vertices and arcs from  $v_i$  to  $v_j$  in  $G$  such that every arc in the sequence is preceded by its initial vertex and is followed by its terminal vertex. A digraph  $G$  is said to be *strongly connected* if for any ordered pair of distinct vertices  $(v_i, v_j)$  there is a directed walk in  $G$  from  $v_i$  to  $v_j$ . The *length* of a directed walk is the number of arcs in the sequence. The number of distinct directed walks from  $v_i$  to  $v_j$  of length  $k$  in  $G$  is equal to the  $(i, j)$ th entry of  $\mathbf{A}(G)^k$  and is denoted by  $W_k(i, j)$ . The digraph  $G$  is strongly connected if and only if  $\mathbf{A}(G)$  is irreducible. A strongly connected digraph  $G$  is also characterized by an *index of imprimitivity*  $h(G)$  which is equal to the index of imprimitivity of  $\mathbf{A}(G)$ . Furthermore, a digraph  $G$  is classified as *cyclically  $r$ -partite* when  $r > 1$  and  $r$  divides  $h(G)$ ; see, for example, [10, §3.4].

The *outdegree*  $d_i^+$  of vertex  $v_i \in V$  in the digraph  $G = (V, E)$  is defined to be the number of arcs in  $E$  with initial vertex  $v_i$ . Thus, the outdegree of vertex  $v_i$  is equal to the  $i$ th row sum of the adjacency matrix  $\mathbf{A}(G)$ . This concept can be generalized to the  *$k$ -outdegree*  $d_i^{k+}$ , which is the number of directed walks of length  $k$  with starting vertex  $v_i$ . That is,  $d_i^{k+} \triangleq \sum_{j=1}^n W_k(i, j)$  for any  $k > 0$  and  $d_i^{0+} \triangleq 1$ .

A vertex  $v_i$  with no outgoing arcs (*i.e.*,  $d_i^+ = 0$ ) is known as a *sink*. Sinks correspond to zero rows in  $\mathbf{A}(G)$ . Thus, the results of Sections 6.2 and 6.3 directly apply to general digraphs without sinks. The first theorem in this section bounds the spectral radii of these digraphs, while the final corollary will show that equality in the bounds may only be achieved for very limited values of  $\rho(G)$ .

We need to introduce terminology to capture the equality conditions of Section 6.3 in a digraph context. With respect to Corollary 6.8, we will call digraph  $G = (V, E)$  *average  $\kappa$ -outdegree regular* if

$$\frac{d_i^{\kappa+}}{d_i^{(\kappa-1)+}} = c, \text{ for all } v_i \in V,$$

where  $\kappa \geq 1$ . Thus for  $\kappa = 2$  our definition matches that of Zhang & Li. If  $G$  is cyclically  $r$ -partite, the set of vertices  $V$  may be partitioned into  $r$  disjoint subsets  $V = V_1 \cup V_2 \cup \dots \cup V_r$  according to (6.16). With respect to Theorem 6.10, we will call digraph  $G$  *average  $\kappa$ -outdegree*

$r$ -quasiregular if  $G$  is cyclically  $r$ -partite and

$$\frac{d_i^{\kappa+}}{d_i^{(\kappa-1)+}} = c_j, \text{ for all } v_i \in V_j$$

and all  $j \in \{1, \dots, r\}$ , where  $\kappa \geq 1$ . To be cyclically  $r$ -partite, all arcs joining vertices in  $V_j$  either initiate in  $V_{[(j-2) \bmod r]+1}$  or terminate in  $V_{(j \bmod r)+1}$ . Thus for  $r = 2$ , this condition degenerates to the bipartite-semiregular condition of Zhang & Li. For  $\kappa = 1$ , we suggest dropping the word “average” from these two new terms to be consistent with prior terminology.

Now we are ready to formulate the equality conditions of Theorem 6.5 in digraph terms.

**Theorem 6.11** *Let  $G = (V, E)$  be a digraph with spectral radius  $\rho(G)$ ,  $n$  vertices, and no sinks. Then, for any integers  $M > 0$ ,  $N \geq 0$ , and  $k \geq 0$ ,*

$$\begin{aligned} \min_{\substack{1 \leq i \leq n \\ 1 \leq j \leq n}} \left[ \left( \frac{d_i^{(k+M)+} d_j^{(k+N)+}}{d_i^{k+} d_j^{k+}} \right)^{\frac{1}{M+N}} : W_M(i, j) > 0 \right] &\leq \rho(G) \\ &\leq \max_{\substack{1 \leq i \leq n \\ 1 \leq j \leq n}} \left[ \left( \frac{d_i^{(k+M)+} d_j^{(k+N)+}}{d_i^{k+} d_j^{k+}} \right)^{\frac{1}{M+N}} : W_M(i, j) > 0 \right]. \end{aligned} \quad (6.21)$$

Moreover, if  $G$  is strongly connected, then equality in (6.21) holds if and only if  $G$  is average  $(k+1)$ -outdegree regular or average  $(k+1)$ -outdegree  $r$ -quasiregular or both, where  $r = \gcd(M+N, h(G))$  and  $h(G)$  is the index of imprimitivity of  $G$ .

*Proof:* We apply Theorem 6.5 to the adjacency matrix  $\mathbf{A}(G)$  to yield (6.21). Corollary 6.8 and Theorem 6.10 justify the equality conditions. ■

**Remark 6.4** In practice, one will likely apply (6.21) with  $M = 1$ , for simplicity in its evaluation. When  $M = 1$ , the condition that  $W_M(i, j) > 0$  is equivalent to  $\{v_i, v_j\} \in E$ .

As a corollary, we find the following new equality conditions on the bounds of Liu.

**Corollary 6.12** *For any integer  $L > 0$  and  $r = \gcd(L, h(G))$ , the following bounds are satisfied with the same equality conditions as Theorem 6.11,*

$$\min_{1 \leq i \leq n} \left[ \frac{d_i^{(k+L)+}}{d_i^{k+}} \right]^{1/L} \leq \rho(G) \leq \max_{1 \leq i \leq n} \left[ \frac{d_i^{(k+L)+}}{d_i^{k+}} \right]^{1/L}. \quad (6.22)$$

The following lemma presents another useful result from Liu [6] that we will use shortly. It shows that, when indexed by  $k$ , the upper and lower bounds of (6.13) and (6.22) form monotonically non-increasing and non-decreasing sequences, respectively.

**Lemma 6.13 (Liu [6, Theorem 3.3])** *Let  $\mathbf{A}$  be an  $n \times n$  nonnegative matrix with nonzero row sums. Then, for any integer  $L > 0$ ,*

$$\min_{1 \leq j \leq n} \left[ \frac{r_j(\mathbf{A}^{k+L})}{r_j(\mathbf{A}^k)} \right] \leq \frac{r_i(\mathbf{A}^{k+L+1})}{r_i(\mathbf{A}^{k+1})} \leq \max_{1 \leq j \leq n} \left[ \frac{r_j(\mathbf{A}^{k+L})}{r_j(\mathbf{A}^k)} \right],$$

for all  $k \geq 0$  and  $i \in \{1, \dots, n\}$ .

**Lemma 6.14** *Let  $c$  be a rational number and  $d$  be a nonzero, finite real number. If the sequence  $\{d \cdot c^j\}_{j=0}^\infty$  contains only integers, then  $c$  is also an integer.*

*Proof:* Let  $c = p/q$ , where  $p$  and  $q > 0$  are coprime integers. Since  $d \cdot p^j/q^j$  is an integer,  $d$  is a multiple of  $q^j$ , which cannot hold for all  $j \geq 0$  unless  $q = 1$ . ■

**Theorem 6.15** *Let  $G$  be a strongly connected digraph. If equality holds in (6.22) for  $G$  and some  $k = t$ , then equality holds for all  $k \geq t$  and  $\rho(G)$  is the  $r$ th root of an integer, where  $r = \gcd(L, h(G))$ .*

*Proof:* Lemma 6.13 proves the first part by showing that the bounds are monotonic in  $k$ . In the case that  $\mathbf{A}^L$  is irreducible ( $r = 1$ ) and equality holds in (6.22), then by Corollary 6.8,  $c = \rho(G)$  is a rational number and  $d_i^{(j+t)+} = d_i^{t+} \rho(G)^j$  holds for all  $j \geq 0$  and all  $i \in \{1, \dots, n\}$ . Since the  $k$ -outdegree  $d_i^{k+}$  of any vertex is integral, then by Lemma 6.14,  $\rho(G)$  is an integer. In the case that  $\mathbf{A}^L$  is reducible ( $r > 1$ ) and equality holds in (6.22), then by Theorem 6.10,  $\rho(G)^r$  is a rational number and  $d_i^{(rj+t)+} = d_i^{t+} \rho(G)^{rj}$  holds for all  $j \geq 0$  and all  $i \in \{1, \dots, n\}$ . Thus,  $\rho(G)^r$  is an integer. ■

**Corollary 6.16** *Let  $G$  be a strongly connected digraph. If equality holds in (6.21) for  $G$  and some  $M > 0$ ,  $N \geq 0$ , and  $k \geq 0$ , then  $\rho(G)$  is the  $r$ th root of an integer, where  $r = \gcd(M + N, h(G))$ .*

*Proof:* As discussed in Section 6.2, the bounds of Corollary 6.6 with  $L = M + N$  are at least as tight as the bounds of Theorem 6.5. Therefore, with respect to the upper bounds,

$$\rho(G)^{M+N} \leq \max_{1 \leq i \leq n} \left[ \frac{d_i^{(k+M+N)+}}{d_i^{k+}} \right] \leq \max_{\substack{1 \leq i \leq n \\ 1 \leq j \leq n}} \left[ \frac{d_i^{(k+M)+} d_j^{(k+N)+}}{d_i^{k+} d_j^{k+}} : W_M(i, j) > 0 \right]. \quad (6.23)$$

Assuming equality holds on the right side of (6.21), then equality holds throughout (6.23), and we may apply Theorem 6.15. ■

Just as deleting the zero rows and their corresponding columns preserved the spectral radius in Section 6.2, the removal of any sinks from the digraph  $G$  leaves  $\rho(G)$  undisturbed. Thus, this simple modification allows us to extend the bounds of this section to general digraphs. Additionally, removing sources from the digraph may tighten the bounds.

**Example 6.1** The order-5 example presented in [4] and [5] provides a useful illustration. The digraph  $G_1 = (V, E)$  and its associated adjacency matrix  $\mathbf{A}(G_1)$  are shown in Fig. 6.1. The

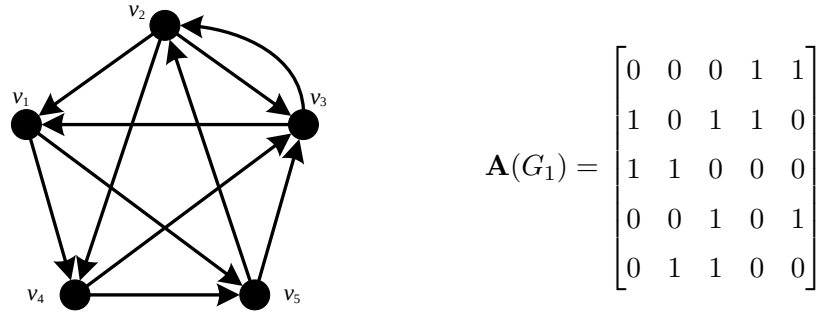


**Figure 6.1:** The digraph  $G_1$  and its  $5 \times 5$  adjacency matrix  $\mathbf{A}(G_1)$ .

spectral radius of  $G_1$  is  $\rho(G_1) \approx 2.193399638$ . First, we examine Corollary 6.12. Table 6.1 shows the quantities corresponding to each vertex  $v_i \in V$  needed to evaluate (6.22), for all values of  $(k, L)$  such that  $k + L \leq 4$ . The minimum and maximum of each, shown on the right side of the table, form the bounds on  $\rho(G_1)$ . The bounds corresponding to  $(k, L) = (1, 2)$  are the tightest of the bounds here for  $k + L \leq 3$ . When extended to  $k + L = 4$ , the bounds using  $(k, L) = (0, 4)$  and  $(1, 3)$  yield the tightest lower and upper bounds, respectively, as indicated with a “†”.

Theorem 6.11, with  $M = 1$ , yields the bounds shown in Table 6.2. In three of four cases the bounds of Theorem 6.11 with  $N = 1$  produced tighter bounds than Corollary 6.12 with  $L = 1$ . Also, the bounds indicated with a “†” are tighter than the bounds of the first table for the same maximum order of outdegree computed.

Finally, we show the Kolotilina-based bounds of Theorem 6.7 in Table 6.3. Since we limited Theorem 6.11 to the case where  $M = 1$  in order to keep the sparsity pattern determination simple, in evaluating bounds of Theorem 6.7 we limit consideration of  $L$  to 1. In Table 6.3 we have used an “\*” to indicate which bounds are independent of the  $\alpha$  parameter. The best lower bounds of Theorem 6.11 were equal to those produced by Corollary 6.12 but tighter than

**Table 6.1:** Intermediate computations and bounds on  $\rho(G_1)$  from Corollary 6.12 for all  $(k, L)$ , such that  $k + L \leq 4$ .

| $(k, L)$ <b>Parameters</b>                 | $i = 1$ | $i = 2$ | $i = 3$ | $i = 4$ | $i = 5$ | min     | max     |
|--------------------------------------------|---------|---------|---------|---------|---------|---------|---------|
| $d_i^+ \quad (0, 1)$                       | 2       | 3       | 2       | 2       | 2       | 2       | 3       |
| $\sqrt{d_i^{2+}} \quad (0, 2)$             | 2       | 2.4495  | 2.2361  | 2       | 2.2361  | 2       | 2.4495  |
| $d_i^{2+}/d_i^{1+} \quad (1, 1)$           | 2       | 2       | 2.5     | 2       | 2.5     | 2       | 2.5     |
| $\sqrt[3]{d_i^{3+}} \quad (0, 3)$          | 2.0801  | 2.3513  | 2.1544  | 2.1544  | 2.2240  | 2.0801  | 2.3513  |
| $\sqrt{d_i^{3+}/d_i^{1+}} \quad (1, 2)$    | 2.1213  | 2.0817  | 2.2361  | 2.2361  | 2.3452  | 2.0817  | 2.3452  |
| $d_i^{3+}/d_i^{2+} \quad (2, 1)$           | 2.25    | 2.1667  | 2       | 2.5     | 2.2     | 2       | 2.5     |
| $\sqrt[4]{d_i^{4+}} \quad (0, 4)$          | 2.1407  | 2.3206  | 2.1657  | 2.1407  | 2.1899  | 2.1407† | 2.3206  |
| $\sqrt[3]{d_i^{4+}/d_i^{1+}} \quad (1, 3)$ | 2.1898  | 2.1302  | 2.2240  | 2.1898  | 2.2572  | 2.1302  | 2.2572† |
| $\sqrt{d_i^{4+}/d_i^{2+}} \quad (2, 2)$    | 2.2913  | 2.1985  | 2.0976  | 2.2913  | 2.1448  | 2.0976  | 2.2913  |
| $d_i^{4+}/d_i^{3+} \quad (3, 1)$           | 2.3333  | 2.2308  | 2.2     | 2.1     | 2.0909  | 2.0909  | 2.3333  |

those produced by Theorem 6.7. Also, Theorem 6.11 produced tighter upper bounds compared with either Theorem 6.7 or Corollary 6.12 when the maximum order of outdegree was limited to three. However, Theorem 6.11 produced the loosest upper bounds when the maximum order of outdegree was relaxed to four.

In this example, the worst set of  $(k, N)$  parameters for Theorem 6.11 is from Xu & Xu [4] (*i.e.*,  $(1, 1)$ ). However, we have generally found that the best set of parameters depends on the digraph selected. For digraphs that are sparser than  $G_1$ , the advantages of Theorems 6.11 and 6.7 will be even more evident.

We note that the bipartite condition (*i.e.*, cyclically  $r$ -partite with  $r = 2$ ) was sometimes unmentioned in [4] and [5] when defining the “semiregular” digraph property [11]. Its necessity is apparent in this example. The digraph  $G_1$  might be outdegree semiregular and average 2-outdegree semiregular by some definitions, but it is not bipartite and hence does not meet the bounds with equality.

**Table 6.2:** Lower and upper bounds on  $\rho(G_1)$  from Theorem 6.11 with  $M = 1$ .

| $(k, N)$<br><b>Parameters</b> | <b>Lower Bound</b><br>on $\rho(G_1)$ | <b>Upper Bound</b><br>on $\rho(G_1)$ |
|-------------------------------|--------------------------------------|--------------------------------------|
| (0, 1)                        | 2                                    | 2.4495                               |
| (0, 2)                        | 2                                    | 2.4662                               |
| (1, 1)                        | 2                                    | 2.5                                  |
| (0, 3)                        | 2.0598                               | 2.3403‡                              |
| (1, 2)                        | 2.0801                               | 2.3208‡                              |
| (2, 1)                        | 2.0817                               | 2.3717                               |
| (0, 4)                        | 2.1118                               | 2.3116                               |
| (1, 3)                        | 2.1407                               | 2.2900                               |
| (2, 2)                        | 2.1204                               | 2.2774                               |
| (3, 1)                        | 2.0954                               | 2.2815                               |

**Table 6.3:** Lower and upper bounds on  $\rho(G_1)$  from Theorem 6.7 with  $L = 1$ .

| $k$<br><b>Parameter</b> | <b>Lower Bound</b><br>on $\rho(G_1)$ | <b>Upper Bound</b><br>on $\rho(G_1)$ |
|-------------------------|--------------------------------------|--------------------------------------|
| 0                       | 2*                                   | 2.4495@ $\alpha = 0.50$              |
| 1                       | 2*                                   | 2.5000*                              |
| 2                       | 2.0801@ $\alpha = 0.50$              | 2.3602@ $\alpha = 0.55$              |
| 3                       | 2.0993@ $\alpha = 0.92$              | 2.2611@ $\alpha = 0.70$              |

## 6.5 Conclusions

We have generalized the bounds and equality conditions of several prior works regarding the spectral radius of nonnegative matrices and digraphs. Much of the earlier work applied to irreducible matrices and strongly-connected simple digraphs. We have generalized these to a larger set of bounds and a more general set of digraphs. Finally, we have shown that the equality conditions of the bounds, when applied to strongly connected digraphs, may only be met when the spectral radius is the  $r$ th root of an integer.

## Acknowledgement

This chapter is in part a reprint of the paper: B. K. Butler and P. H. Siegel, “Sharp Bounds on the Spectral Radius of Nonnegative Matrices and Digraphs,” to appear in *Linear Algebra Appl.* 439 (2013) 1468–1478. The dissertation author was the primary author of this paper.

## Bibliography

- [1] R.A. Horn, C.R. Johnson, *Matrix Analysis*, Cambridge University Press, Cambridge, 1985.
- [2] X.-D. Zhang, J.-S. Li, Spectral radius of non-negative matrices and digraphs, *Acta Math. Sin.* 18 (2002) 293–300.
- [3] L.Yu. Kolotilina, Bounds and inequalities for the Perron root of a nonnegative matrix, *J. Math. Sci.* 121 (2004) 2481–2507.
- [4] G.-H. Xu, C.-Q. Xu, Sharp bounds for the spectral radius of digraphs, *Linear Algebra Appl.* 430 (2009) 1607–1612.
- [5] A.D. Güngör, K.Ch. Das, Improved upper and lower bounds for the spectral radius of digraphs, *Appl. Math. Comput.* 216 (2010) 791–799.
- [6] S.-L. Liu, Bounds for the greatest characteristic root of a nonnegative matrix, *Linear Algebra Appl.* 239 (1996) 151–160.
- [7] R.A. Brualdi, Spectra of digraphs, *Linear Algebra Appl.* 432 (2010) 2181–2213.
- [8] H. Minc, *Nonnegative Matrices*, John Wiley & Sons, New York, 1988.
- [9] G.-H. Xu, K.-F. Fang, J. Shen, Bounds on the spectral radii of digraphs in terms of walks, *Appl. Math. Comput.* 219 (2012) 3721–3728.
- [10] R.A. Brualdi, H.J. Ryser, *Combinatorial Matrix Theory*, Cambridge University Press, Cambridge, 1991.

- [11] R.A. Brualdi, From the Editor-in-Chief: Errata to Sharp bounds for the spectral radius of digraphs, *Linear Algebra Appl.* 438 (2013) 3–4.