Channel Coding Theory

The First Class

Agenda

- Course Schedule
- Shannon Capacity Theorem
- Source Rate Separation Theory
- HW#0

- Disclaimers
  - Some pictures are scanned from other sources
  - They will be marked on the picture with the last name of the first author

E-mail

- My e-mail is

  heungno@gist.ac.kr
- I will have the whole lecture notes available at the printing shop.

Course Information

- Class hours: 10:30-12:00 am Monday, Wednesday
- Lecture room: B201
- Office hours:
  - 2:00pm ~ 4:00pm Monday,
  - 4:00apm ~ 5:00pm Tuesday.
  - Or make an appointment via e-mail.

Grade Distribution

- Two exams (Midterm#1: 20%, Final: 30%)
- Homework + Homework Grading + Class Participation (20%)
- Term Project (30%)
  - Wireless network codes
  - Compressive sensing

Homework, Class-Project Policies

- Discussion and exchange of ideas are strongly encouraged.
- On each homework and class project set, a reviewer will be assigned (will take turns).
- The job of each reviewer is to
  - grade homework/project sets,
  - type up the best homework solution(rec. WORD with Mathtype),
  - get an approval of the solution manual from me, and
  - distribute the graded homework and solution to the students within a week.

Tentative Schedule

Week #	Date	Topics	HWs	Note
1	9/1(Wed)	Introduction to Channel Codes (Shannon's 1948 paper)	HW#0	
2	9/6, 8	Galois Fields	HW#1 Out	
3	9/13, 15	Polynomials over Galois Fields	HW#2 Out	
4	9/20,	Linear Block Codes		9/21-23 Full Moon Holidays
5	9/27, 29	Linear Block Codes	HW#3 Out	
6	10/4, 6	BCH and Reed-Solomon Codes	HW#4 Out	
7	10/11, 13	BCH and Reed-Solomon Codes		
8	10/18, 20	Midterm Week		Midterm
9	10/25, 27	Convolutional Codes		
10	11/1, 3	Convolutional Codes/Trelllis Codes	HW#5	
11	11/8, 10	Turbo Codes/Turbo Decoding Makeup on 11/12(Friday)	HW#6	Asilomar Conference
12	11/15, 17	Performance Analysis of Turbo Codes		
13	11/22, 24	LDPC codes/Decoding	HW#7	
14	11/29, 12/1	Density Evolution/EXIT Charts	HW#8	
15	12/6, 8	Distance Spectrum/Tight Union Bounds		
Final Week	12/15	Final Exam on Wednesday Term paper/project program package due by Friday		

Scope of this course

- Learn and apply the channel coding theory to practical communications problems.
- Learn and simulate communications systems for the purpose of evaluating their performances.
- Be able to analyze the obtained simulation result and to predict the performance of a given system, and provide a better design.
- Once we know how to predict/evaluate the performance of a communications system, we will use these knowledge and tool sets to design a better performing communications system.
- I say this is the way how the communications theory has been evolved.

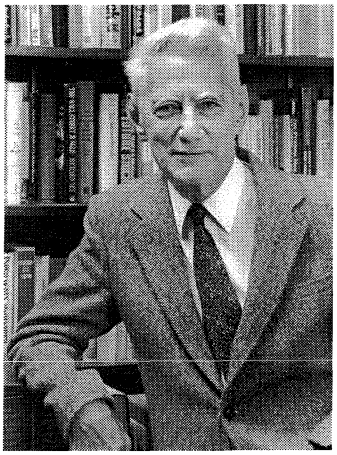
Text Books

- **Required Textbook**: Todd Moon, **Error Correction Coding: Mathematical Methods and Algorithms** A comprehensive introduction modern and classical error correction coding. Wiley, 2005. (ISBN 0-471-64800-0) (Fall 2006)
- **Reference:** Stephen B. Wicker, "Error Control Systems for Digital Communication and Storage," Prentice Hall.
- **Reference**: F.J. MacWilliams and N.J.A. Sloane, The Theory of Error-Correcting Codes, North-Holland Mathematical Library, 1977.
- **Reference**: D. Mackay, Information Theory, Inference, and Learning Algorithms, Cambridge University Press, 2003. (Downloadable at his Web-site)
- **Reference:** T. Richardson and R. Urbanke, Modern Coding Theory, Cambridge University Press, 2007.
- **Reference**: IEEE Transaction Papers to be Identified During the Course

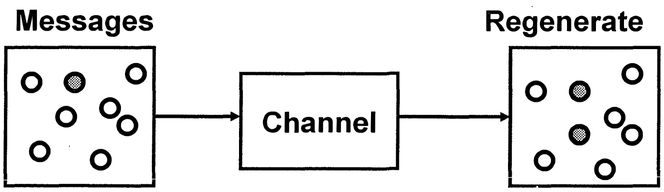
- Now, let's begin…

Claude E. Shannon (1916 - 2001)

- Math/EE Bachelor from UMich (1936)
- MSEE and Math Ph.D. from MIT (1940)
- A landmark paper "Mathematical Theory of Communications" (1948)
  - Founder of Information Theory
  - Fundamental limits on communications
  - Information quantified as a logarithmic measure
- For more info on him, make a visit to http://www.bell-labs.com/news/2001/february/26/1.html



Shannon's Perspective on Communications



- Communications: Transfer of information from a source to a receiver
- Messages (information) can have *meaning*; but they are irrelevant for the design of communications system.
- What's important then?
  - *A message is selected from a set of all possible messages and transmitted, and regenerated at the receiver*
  - The size of the message set is the amount of information
- *The capacity C of a channel is the maximum size of message set that can be transferred over the channel and can be regenerated almost error-free at the receiver.*

Digital Communication

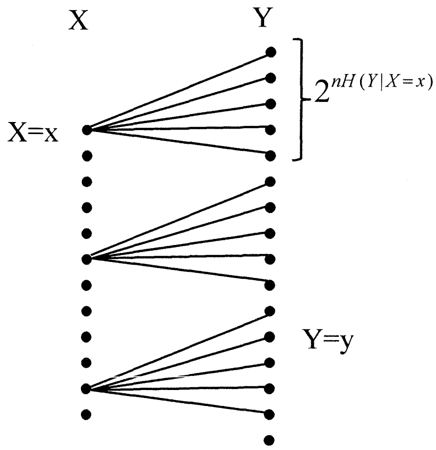
- It is to send a message index m (out of M total) over *the channel*
  - for the duration of time T, and
  - have an expectation that the same index m can be recovered almost error-free at the receiver.
- Transmission rate $R = \log_2(M)/T$ [bits/sec]
- *If* $R < C$, *then* almost error-free recovery can be achieved.
- We need to find a set of M waveforms to interface the channel.
  - An analog (physical) waveform shall be chosen to carry the messages. Why?

Main Story in Shannon's Paper

- Given a channel relation (Y = X + N), find out the size M of the input message set which results in very small P(e).
  - You are allowed to use the same channel many times, say n times.
- The strength of the noise limits the size of the input message set. (Obvious)
- Determine the range of rates $R = \log_2(M)/n$ that gives P(e) very small.
- There are $2^{nH(X)}$ typical input sequences of length n.
- We choose 2^{nR} messages randomly out of total $2^{nH(X)}$ typical words.
- We want only one message out of total 2^{nR} messages falls into the fan of $2^{nH(X|Y)}$.

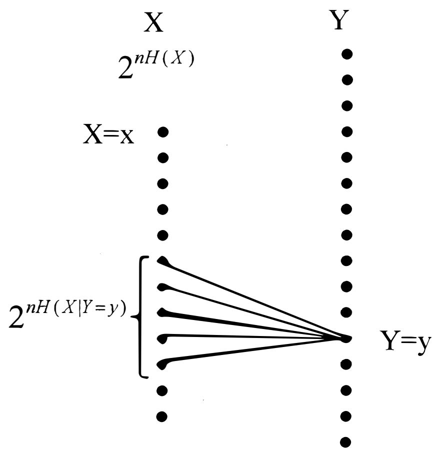
Shannon's Key Idea

- There are $2^{nH(Y)}$ *typical* outputs.
- Each input message fans out to $2^{nH(Y|X)}$
- If we select only $\frac{2^{nH(Y)}}{2^{nH(Y|X)}} = 2^{nI(X;Y)}$ number of messages, no *equivocation* would occur.
  - P(e) is close to 0 (LLN)
- So, such a codebook *exists* and can be constructed, though difficult.
  - What if it is constructed randomly?



Shannon's Key Idea: P(e) in Random Codebook Construction

- Let's select the message set(Codebook) randomly.
- And, see if we can make P(e) very small.
- Given a fan of size $2^{nH(X|Y=y)}$, decoding error occurs if there are more than one messages within the fan.
  - See the analysis in the following page



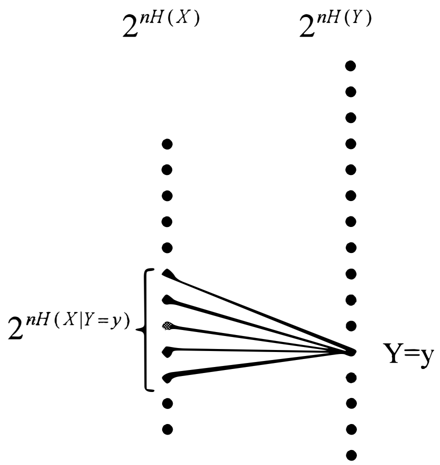
$$\Pr\{E\} = \sum_{C} P(C)\, P_e(C) = \sum_{C} P(C) \sum_{i=1}^{M} \left(\tfrac{1}{M}\right) P_e(C, x_i) = \sum_{i=1}^{M} \left(\tfrac{1}{M}\right) \sum_{C} P(C)\, P_e(C, x_i)$$

$$= \sum_{i=1}^{M} \left(\tfrac{1}{M}\right) \sum_{C} P(C)\, P_e(C, x_1) = \sum_{C} P(C)\, \lambda_1(C) = \Pr\{E \mid W=1\}$$

Shannon's Key Idea: P(e) in Random Codebook Construction (2)

$\lambda_i = \Pr\{$

- Steps:
  - Select the first message (the red dot) and send.
  - With probability close to 1, we get the typical output y.
  - Randomly select the rest of messages.
  - Consider the *fan* of y and find out the probability of decoding error.
  - Decoding error occurs when any one of $2^{nR} - 1$ other messages falls within the fan.



P(e) in Random Codebook Construction (3)

- $$\text{P(e)} = 1 - \left(1 - \frac{2^{nH(X|Y)}}{2^{nH(X)}}\right)^{2^{nR}-1}$$
  $$\leq 1 - \left(1 - 2^{-n[H(X)-H(X|Y)]}\right)^{2^{nR}}$$
  $$\approx 1 - \left(1 - 2^{nR} 2^{-n[H(X)-H(X|Y)]}\right)$$
  $$= 2^{-n[I(X;Y)-R]}$$

- Thus, as long as *R* is chosen slightly smaller than I(X; Y), P(e) decreases to zero as *n* increases.
  - Now we maximize I(X; Y) by selecting the best input distribution, and obtain the capacity, $C = \max_{p(x)} I(X; Y)$.

Channel Capacity vs. Rate Distortion

- Source-Channel Separation Theorem
  - Data compression (Rate Distortion Function) and Channel Coding can be separately done without losing optimality.

- To explain, suppose
  - A source is transmitting at an apparent data rate R [bits/sec/Hz] with source coding error probability P_b.
  - The true source rate then is $R(1 - H(P_b))$, which should be smaller than the channel capacity C for near zero transmission errors.
  - Per channel coding theorem, we must have

$$R(1 - H(p)) < C \qquad \text{[bits/sec/Hz]}$$

Capacity Lower Bounds on P_b as a function of E_b/N_o

- CLB is very useful later on for the course.
  - It provides fundamental bounds on bit error probability.

- For a fixed R, we can find the capacity lower bound on

$$R = C/(1 - H(P_b)).$$

- Now, what's left for us to find is the capacity at a certain E_b/N_o.

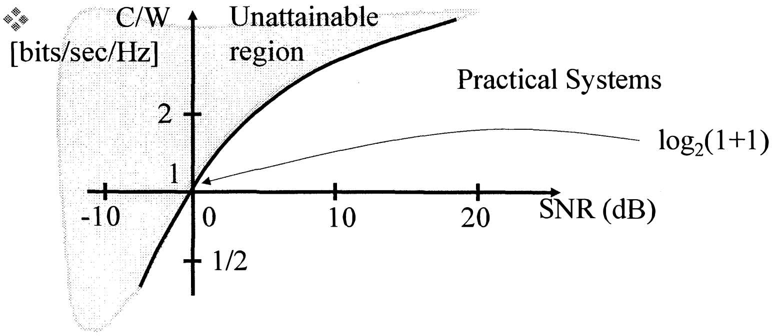
- Let's find the capacity expression for two cases
  - AWGN channel: $C(E_b/N_o)$
  - BPSK over AWGN channel: $C(E_b/N_o)$

Shannon Capacity for AWGN Channel

- The channel capacity in [bits/sec] is

$$C = W\ log_2(1 + \tfrac{P}{N}) = W\ log_2(1 + \tfrac{P}{N_o W})$$

-



Shannon Limit

- The bit rate of the transmission

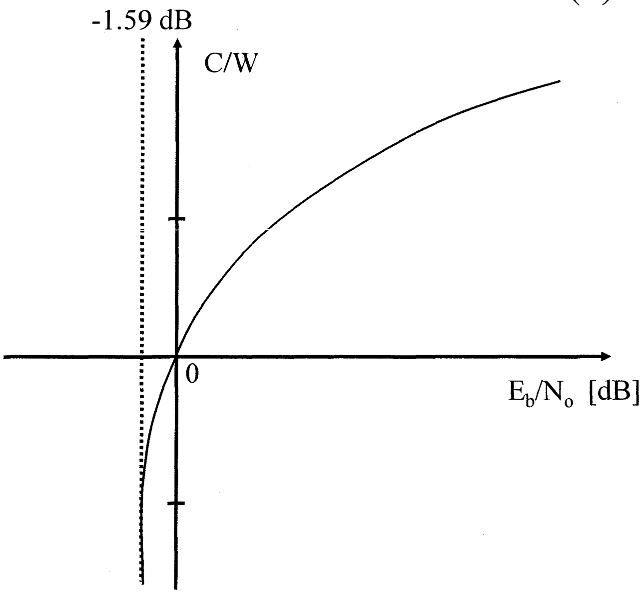
  R [bits/sec] = R_s [symbols/sec] × *k* [bits/symbol]
- Signal Power P = Energy per symbol × Baud = $E_s \times R_s$
- Energy per bit $E_b = E_s/k$
- $P = E_b \times R$
- $P/N = (E_b R)/(N_o W)$
- Thus, the spectral efficiency [C/W] is

  $C/W = \log_2[1+(E_b/N_o)(R/W)]$ [bits/sec/Hz] ---------(1)

Shannon Limit (2)

- We are interested in finding the smallest E_b/N_o such that error free transmission is possible
- In Eq. (1), substitute R for C in the right side of Eq.
- Then, we have
  $C/W = \log_2(1+(E_b/N_o)(C/W))$,
- Arranging it for E_b/N_o, we have
  $E_b/N_o = (W/C)(2^{C/W} - 1)$
- Let $x:=(C/W) \to 0$
- $E_b/N_o = \lim_{x \to 0} (1/x)(2^x - 1) = \log_e 2 = 0.693 = -1.6$ dB
- This is the ultimate limit below which no error-free transmission is possible no matter how small R/W we may choose

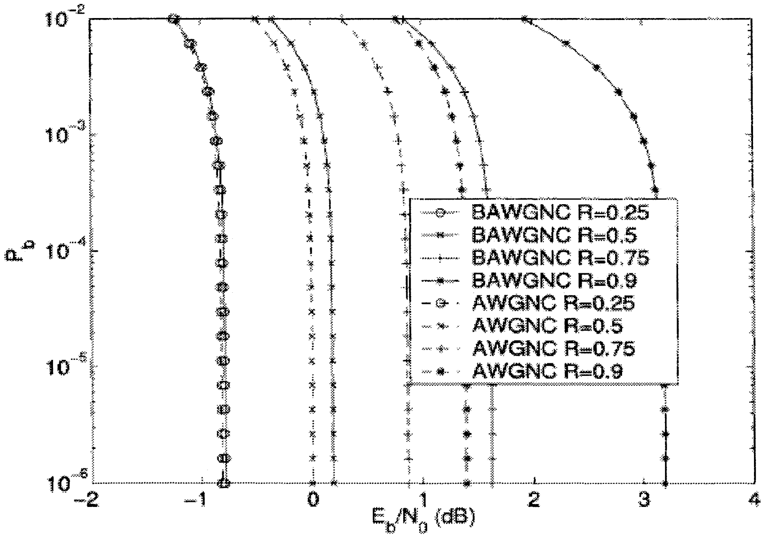
Shannon Limit (3)



BPSK over AWGN

- $Y = X + N$
  - $X = 1$ or -1
  - $N \sim$ Gaussian $(0, N_o/(2E_b))$
- $I(X; Y) = E\{\log P(Y|X)/P(Y)\}$

Capacity Lower Bounds on P_b as a function of E_b/N_o



Moon, p.52

Figure 1.24: Capacity lower bounds on P_b as a function of SNR.

Table 1.1: Historical Milestones

Year	Milestone
1948	Shannon publishes "A Mathematical Theory of Communication" [309]
1950	Hamming describes Hamming codes [137]
1954	Reed [284] and Muller [248] both present Reed-Muller codes and their decoders
1955	Elias introduces convolutional codes [76]
1957	Prange introduces cyclic codes [271]
1959	A. Hocquenghem [151] and ...
1960	Bose and Ray-Chaudhuri [36] describe BCH codes
	Reed&Solomon produce eponymous codes [286]
	Peterson provides a solution to BCH decoding [261]
1961	Peterson produces his book [260], later extended and revised by Peterson and Weldon [262]
1962	Gallager introduces LDPC codes [112]
	2400 BPS modem commercially available (4-PSK) (see [100])
1963	The Fano algorithm for decoding convolutional codes introduced [80]
	Massey unifies the study of majority logic decoding [224]
1966	Forney produces an in-depth study of concatenated codes [87] and introduces generalized minimum distance decoding [88]
1967	Berlekamp introduces a fast algorithm for BCH/Reed-Solomon decoding [22]
	Rudolph initiates the study of finite geometries for coding [299]
	4800 BPS modem commercially available (8-PSK) (see [100])
1968	Berlekamp produces *Algebraic Coding Theory* [25]
	Gallager produces *Information theory and reliable communication* [111]
1969	Jelinek describes the stack algorithm for decoding convolutional codes [165]
	Massey introduces his algorithm for BCH decoding [222]
	Reed-Muller code flies on *Mariner* deep space probes using Green machine decoder
1971	Viterbi introduces the algorithm for ML decoding of convolutional codes [359]
	9600 BPS modem commercially available (16-QAM) (see [100])
1972	The BCJR algorithm is described in the open literature [10]
1973	Forney elucidates the Viterbi algorithm [89]
1975	Sugiyama et al. propose the use of the Euclidean algorithm for decoding [324]
1977	MacWilliams and Sloane produce the encyclopedic *The Theory of Error Correcting Codes* [220]
	Voyager deep space mission uses a concatenated RS/convolutional code (see [231])
1978	Wolf introduces a trellis description of block codes [377]
1980	14,400 BPS modem commercially available (64-QAM) (see [100])
	Sony and Phillips standardize the compact disc, including a shortened Reed-Solomon code
1981	Goppa introduces algebraic-geometry codes [123, 124]
1982	Ungerboeck describes trellis-coded modulation [345]
1983	Lin & Costello produce their engineering textbook [203]
	Blahut publishes his textbook [33]
1984	14,400 BPS TCM modem commercially available (128-TCM) (see [100])
1985	19,200 BPS TCM modem commercially available (160-TCM) (see [100])
1993	Berrou, Glavieux, and Thitimajshima announce turbo codes [28]
1994	The $\mathbb{Z}_4$ linearity of families of nonlinear codes is announced [138]
1995	MacKay resuscitates LDPC codes [218]
	Wicker publishes his textbook [373]
1996	33,600 BPS modem (V.34) modem is commercially available (see [98])
1998	Alamouti describes a space-time code [3]
1999	Guruswami and Sudan present a list decoder for RS and AG codes [128]
2000	Aji and McEliece [2] (and others [195]) synthesize several decoding algorithms using message passing ideas
2002	Hanzo, Liew, and Yeap characterize turbo algorithms in [141]
2003	Koetter and Vardy extend the GS algorithm for soft-decision decoding of RS codes [191]
2004	Lin&Costello second edition [204]
2005	Moon produces what is hoped to be a valuable book!

Moon, pg 41

HW#0

MATLAB BPSK Sim.

Coded BPSK Sim

- Read Chapter 1
- Review the following items
  - Entropy, Conditional Entropy, Mutual Information
  - Source Encoder/Decoder
  - Channel Encoder/Decoder
  - BPSK and its probability of error
  - Gaussian Channel
  - ML vs. MAP~ which one is better?
  - Union Bounds
  - Binary Symmetric Channel
  - Hamming Distance
  - What is a code?
  - Minimum Distance of a Code
  - Coding Gain?
  - Channel Coding Theorem and its Proof
  - Capacity of AWGN
  - Difference between E_b and E_s
- Reproduce Figure 1.24 using MATLAB.

P1.15
P1.16
P1.24
P1.26
P1.27
P1.28
P1.29
P1.30
P1.31

References

- C.E. Shannon, A Mathematical Theory of Communications, *The Bell System Technical Journal,* Vol. 27, pp. 379–423, 623–656, July, October, 1948.
- D. Costello and D. Forney, Channel Coding: The Road to Channel Capacity, Proceedings of IEEE, vol.96, no.6, June 2007.
- Todd Moon, **Error Correction Coding: Mathematical Methods and Algorithms** A comprehensive introduction modern and classical error correction coding. Wiley, 2005.

Galois Fields

Évariste Galois [evaʁist galwa] (1811 ~ 1832):
Died from a wound obtained in a duel over a lover.

References: Ch.2, Ch.3Wicker 1995, Ch. 2 of Moon, Wikipedia

Why study **Galois Fields**?

- In the first class, we reviewed the channel capacity.
- The capacity of a channel can be approached via a channel code.
- The lesson we learned from the channel coding theorem is that we should use vectors, rather than single symbols to communicate over a noisy channel.
- What is a channel code then?
  - A channel code is a mapping from a vector space of messages to a bigger dimensional vector space of codewords.
    - Bigger dimension to accommodate redundancy information

Why study **Galois Fields**?

- Algebraic codes developed early in 50s and 60s are mapping from a vector space of dimension m, over a finite field GF(q^m), into a vector space of dimension $n>m$.
- A code is a subset of vectors of length n.
  - Larger distances are obtained between codewords.
- Reed-Solomon codes
  - $n = q^m - 1$
  - Over GF(q^m)
  - For a t-error correcting code, $t*m$ bits can be corrected.

Algebra

- The word "algebra" comes from Arabic language.
  - It means "restoration."
- In mathematics, it means a branch of mathematics that deals with addition, subtraction, multiplication and division.

Different Representations of GF Elements

- An element in GF can be represented as *vectors* or as *polynomials*.

유한체에는
Vector는 곱하기
없는데?

- Ex) Polynomials of degree less than 2 over GF(2)

  $x^2 + 1$, $x+1$,
  - Sum of the two polynomials = $x^2 + x$
- Factorization of polynomials
- Codewords = polynomials = vectors = curves over GF

Original Question asked in GF Study

- "Why is there no formula for the roots of a fifth or higher degree polynomial equation in terms of the coefficients of the polynomial, using only the usual algebraic operations (addition, subtraction, multiplication, division) and application of radicals (square roots, cube roots, etc)?" [Wikipedia]

- Galois theory answers this question, as well as others, i.e.,
  - Why we can do that for degree less than four?

- Ex) Roots of $x^2 - 4x + 1 = 0$.
  - $a = 2 + \text{sqrt}(3)$, $b = 2 - \text{sqrt}(3)$.
  - $a+b = 4$ and $ab=1$.

- Ex2) Roots of $x^2 + x + 1 = 0$.

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2}$$

- Now, let's begin

Loose Definitions, for now

- (*Abelian*) *Group* is a set of objects that can be "added."
- *Field* is a set of objects that can be "added," and "multiplied."
- *Vector space* is a set of n-tuples, defined over a field, in which the vector addition and the scalar multiplication are well defined.

Set

- Collection of objects, or elements
- Cardinality, the number of objects
- Consider a *binary* operation on two set elements which yields a third element.

Group G

- A set of objects on which a binary operation + satisfies the following 4 axioms:
  - Closure: $a, b \in G \Rightarrow a + b \in G$
  - Associativity: $(a + b) + c = a + (b + c)$
  - Identity: for all $a \in G$, $\exists$ an *identity* element such that $a + e = e + a = a$.
  - Inverse: for all $a \in G$, $\exists$ an (*unique*) element $a^{-1} \in G$ such that $a + a^{-1} = a^{-1} + a = e$.
- The cardinality of a group is called the *order* of the group.
- Finite group, we call, when the order $< \infty$.

Associativity

- Addition and multiplication are associative.
  - Ex)
- Subtraction and division are not associative.
  - Ex)

Abellian Group

- A group is said to be commutative (or *Abellian*) if it also satisfies
  - Commutativity: for all $a, b \in G$, $a + b = b + a$.
  - Most groups considered in this class will be commutative.
  - cf) The group of invertible matrix is non Abellian, i.e.

$$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = \begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix}$$

Examples of Groups

- The set of integers, under integer addition (but not under multiplicative inverse).
- The positive rational numbers, under ordinary multiplication.
- A set of integers {0, 1, 2, …, *m*-1}, under modulo-*m* addition.
- A set of integers {1, 2, …, *p*-1}, under modulo-*p* multiplication *iff p* is prime.
  - cf. A set {0, 1, 2, …, *p*-1} is not a group under modulo-*p* mult, if *p* not a prime.

Identity = 1.
$a \bullet a^{-1} = 1$
not integer

Identity = 1

$\forall i \in \{0,1,2,3\}$
Do we have i^{-1}?

mod 4

*	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	0	2
3	0	3	2	1

Ex) {0, 1, 2, 3}

$2*3 \bmod 4 = 6 \bmod 4 = 2$.

$2*1 \bmod 4 = 2$

1 is identity!

2 does not have an inverse.

Ass.?

$*\ (2*3)*3 = 2$

$*\ (2*(3*3)) = 2*1 = 2$.

$0 + 4 = 1$

$*\ (2*2)*3 = 0$

$*\ 2*(2*3) = 2*2 = 0$.

Equivalent Classes

- Two integers a and b are in the same *equivalent-class* modulo-m if $a = x\,m + b$ for some integer x.
- Example
  - *Addition modulo* $m = 3$ gives three distinctive equivalent classes (*labeled* with the smallest non-negative integers)
    - $0 \leftrightarrow \{\ldots, -3, 0, 3, 6, \ldots\}$
    - $1 \leftrightarrow \{\ldots, -2, 1, 4, 7, \ldots\}$
    - $2 \leftrightarrow \{\ldots, -1, 2, 5, 8, \ldots\}$
- "Equivalent" in the sense that
  - $0 + 2 = 2 \bmod 3$
  - $3 + 2 = 2 \bmod 3$
  - 0 can be substituted with 3 in operations w/out changing the outcome of the operation.

The Order of a Group Element

- Let $g \in G$ with a group operation $*$.
- *ord*(g) is defined to be the smallest integer t such that

$$g^t := \underbrace{g * g * \cdots * g}_{t} = e$$

- *Examples*
  - Group *of order* 2 under modulo 3 multiplication $\{1, 2\}$.
  - The *order* of element 1 is, *ord(*1) = 1.
  - The *order* of element 2 is, *ord*(2) = 2.

Subgroup S of a group G

- When a subset S of a group G forms itself a group, it is called a subgroup of G.
  - Closure: If any $a, b \in S$, then $a * b \in S$.
  - Inverse: If any $a \in S$, then there exists $a^{-1} \in S$.
  - *S is a subgroup of G, if the closure and inverse conditions are met.*
  - We say *proper* if $S \subset G$, but not $S = G$.
- Example:
  - The group of integers under *modulo 9 addition* contains the proper subgroups $\{0\}$, $\{0,3,6\}$.
- A simple way to construct a subgroup S of a group G
  - Take any element $g \in G$.
  - A subgroup S generated by g is $\{g, g^2, g^3, \ldots, g^{ord(g)}\}$.

Coset Decomposition of G

- Given a group G and a subgroup $S = \{s_1, s_2, \ldots, s_n\}$

Coset Leaders

Choose any element in G which has not appeared in above rows

$s_1=e$	s_2	s_3	$\ldots$	s_n
g_2*s_1	g_2*s_2	g_2*s_3	$\ldots$	g_2*s_n
g_3*s_1	g_3*s_2	g_3*s_3	$\ldots$	g_3*s_n
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
g_q*s_1	g_q*s_2	g_q*s_3	$\ldots$	g_q*s_n

$q \times n =$ the order of G

Every element of G appears once and only once in a coset decomposition.

Lagrange's Theorem

- If S is a subgroup of G, the ord(S) devides the ord(G) without remainder.
  - notation: $\mathrm{ord}(S) \mid \mathrm{ord}(G) = \mathrm{ord}(G) \bmod \mathrm{ord}(S) = 0$

Coset Decomposition Example

- G = {0, 1, 2, …, 8} is an *Abellian* group under modulo 9 addition.
- S = {0, 3, 6} is a subgroup.
- Cosets are {0, 3, 6}, {1, 4, 7}, and {2, 5, 8}.
- The coset leaders are 0, 1, 2.

Ex)

0 3 6
4 7 1
2 5 8

나열해주기는 coset은 똑같이. coset leaders 는 바뀔수 있어!

Ring R

- A ring R is a collection of elements with additive + and multiplicative * operations with the four properties
  - R is a *commutative group* under +.
  - Closure under *: For any $a, b \in$ R, the product $a * b \in$ R.
  - Associative * operation: $(a * b) * c = a * (b * c)$.
  - * distributes over +: $a * (b + c) = a*b + a*c$.
- A ring is *commutative* if * commutes.
  - $a * b = b * c$.
- A ring with identity → Field? No.!
  - * has an identity element (labeled as "1").

Every non-zero element 들이 모두 inverse가 있어야! Field.

Examples of Ring

- The set of integers does not form a field since most integers do not have multiplicative inverse ($3 \times 1/3 = 1$, but 1/3 is not an integer).
- The integers under mod-m multi and addition form a commutative ring with identity.
- The set of all polynomials with binary coefficients form a commutative ring with identity under standard mod-2 polynomial addition and multiplication.

Field F

- A set of elements F on which + and * are defined is a *field iff*
  - F is a *communicative* group under the additive operator + (The additive identity element is labeled as "0").
  - $F - \{0\}$ is a *commutative* group under * (The multiplicative identity element is labeled as "1").
  - The distributive law holds: $a*(b+c) = a*b + a*c$.
- It is a *commutative ring* with identity in which every non-zero element has a multiplicative *inverse.*

Examples of Fields

- Infinite Fields
  - The real numbers
  - The complex numbers
  - The set of rational numbers.

Galois Fields GF(q)

- A field of q-elements, when exists, is unique (There is only one.), and denoted as *the* GF(q), q is finite.
  - Discovered by Evariste Galois.
- The integers {0, 1, 2, …, q-1}, where q is prime, is the GF(q) under modulo-q addition and multiplication.
- This field can be sufficiently represented by the addition and multiplicative tables.

Examples of GF(q)

- GF(2)

+	0	1
0	0	1
1	1	0

*	0	1
0	0	0
1	0	1

- GF(3)

+	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

*	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

GF(4) = {0, 1, 2, 3}

- Note that 4 is not a prime, but is a certain power of a prime, $2^2=4$.
- Note that here + and * are not modulo operations:

+	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

*	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	3	1
3	0	3	1	2

- GF(q^m), where q is a prime and $m > 1$, can be constructed with more complex operations than simple modulo arithmetic.

Vector Spaces

- V is a set of vectors.
- F is a field of elements called scalars.
- Binary vector addition +.
  - For $v_1, v_2 \in V$, $v_1 + v_2 = v \in V$.
- Binary scalar multiplication *
  - For $v_1 \in V$ and $a \in F$, $a*v_1 = v \in V$.
- V is called a vector space over F if the followings hold:
  1. V is a group under the vector addition +.
  2. For any $a \in$ F and v $\in$ V, $a * v = u \in$ V.
  3. $a*(v_1 + v_2) = a*v_1 + a*v_2$ and $(a+b)*v = a*v + b*v$.
  4. Associativity: $(a*b)*V = a*(b*V)$.
  5. $1*v = v$, 1 is also the scalar multiplication identity.

n-tuple vector space V_n

- $\mathrm{v} = (v_0, v_1, \ldots, v_{n-1})$
- Example: V_3 over GF(2)
  - (0, 0, 0), (0, 0, 1), …
- The linear combinations of a spanning set G include all vectors in G.
- A spanning set G which has minimal cardinality *is* a *basis* for *a V.*
- The Inner product *operator* $\cdot$:
- *Dual* spaces of a vector space
  - $S^{\perp}$ is the dual space of S *iff* for all $\mathrm{v}_1 \in S^{\perp}$ and for all $\mathrm{v}_2 \in S$, $\mathrm{v}_1 \cdot \mathrm{v}_2 = 0$.

Theorem 1: ***Order of an element* divides q-1.**

- For GF(q), "order" is defined for multiplication.
- Let $\beta \in \mathrm{GF}^*(q) = \{1, 2, \ldots, q\text{-}1\}$.
  - Ord(β) is the smallest integer $t > 0$ such that $\beta^t = 1$.
- t divides q-1 without remainder (Theorem 1).
  - Note that $\{\beta, \beta^2, \beta^3, \ldots, \beta^t = 1\}$ forms a subgroup of GF$^*(q)$ under multiplication.
  - As a subgroup, its size must divide the size of the original group (Lagrange's Theorem).
  - This determines the range of possible orders.
    - Example: {1, 3, 5, 15} is the range of orders for ~~any non-zero~~ element in GF(16).

15를 나누는 약수

Greatest Common Divisor: GCD(a, b)

- The set of integers forms a ring under usual mult and add.
- Division Algorithm: For any pair of integers $a \geq b \neq 0$, there is a unique pair of integers Q (the quotient) and r (the remainder) such that
  - $a = Q\,b + r$, where $0 \leq r < |b|$
  - Cf. $a = r \bmod b$ (a is congruent to r modulo b.)
  - $b|a$ when $r = 0$ (b divides a w/out remainder; b is a factor of a)
- For $a,\ b \in Z$, GCD(a, b) := the largest divisor m of a and b, i.e., $m|a$ and $m|b$.

Euclidean Algorithm
Theorem 2: r_n is the $GCD(a, b)$.

- Iterative algorithm to find the GCD(a, b), for $a \geq b > 0$
- Lemma-1: Any common factor c of a and b is a factor of r.
  - We can write $a=cx$ and $b=cy$ for $x, y > 0$.
  - $r = c(x - yQ)$ from $cx = cyQ + r$
  - Note that $(x - yQ)$ is non negative integer.
- Lemma-2: Any common factor d of b and r is a factor of a.
  - We can write $b=dx$ and $r=d$ then $a = dxQ + dz = d\,(xQ+z)$.

dz

dz

- Algorithm:
  - $a = b\,Q_1 + r_1$ $\quad (0 \leq r_1 < b)$
  - $b = r_1\,Q_2 + r_2$ $\quad (0 \leq r_2 < r_1)$
  - $r_1 = r_2\,Q_2 + r_3$ $\quad (0 \leq r_3 < r_2)$
  - …
  - $r_{n-2} = r_{n-1}\,Q_n + r_n$ $\quad (0 \leq r_n < r_{n-1})$
  - $r_{n-1} = r_n\,Q_{n+1}$ (stop when zero remainder)

$6 = \frac{12}{2}$

Corollary

- For any $a, b \in Z$, $\exists\, x, y \in Z$ such that

$$GCD(a, b) = ax + by$$

$r_3 = r_4 Q_5 \Rightarrow 12 = 6(2)$

$r_2 = r_3 Q_4 + r_4 \Rightarrow 18 = 12(1) + 6 \Rightarrow 6 = 18 - 12(1)$

$18 = 6\cdot(2) + 6$

$r_1 = r_2 Q_3 + r_3 \Rightarrow 48 = 18(2) + 12$

$48 = (6\cdot2+6)(2)+12$

$b = r_1 Q_2 + r_2 \Rightarrow 66 = 48(1) + 18$

$= [(6\cdot2+6)(2)+12] + 18$

$a = b Q_1 + r_1 \Rightarrow 180 = 66(2) + 48$

$= 18 - [48 - 18(2)]$

$= 18(3) - 48$

$= 3(66-48) - (48)$

$= 3(66) - 4(48)$

$= 3(66) - 4(180-66(2))$

$= 11(66) - 4(180)$

$= 726 - 720$

$= \boxed{6}$

GCD Example

GCD(66, 180) =

$b = 66,\ a = 180$

$66 \overline{)180}$ $Q_1 = 2$, 132, $r_1 = 48$

$48 \overline{)66}$ $Q_2 = 1$, 48, $r_2 = 18$

$18 \overline{)48}$ $Q_3 = 2$, 36, $r_3 = 12$

$12 \overline{)18}$ $Q_4 = 1$, 12, $r_4 = 6$

$6 \overline{)12}$ $Q_5 = 2$, 12, 0

$\boxed{6} = r_4 = GCD(66, 180)$

$= (+3)(66) + (-1)(180)$

Theorem 3
If ord(α)= t, then $\underbrace{\text{ord}(\alpha^i)}_{x}$ = t /GCD(t, i).

Let α and α^i be elements of GF(q). If ord(α)= t, then ord(α^i) = t /GCD(t, i).

- Proof: Show x divides and is divisible by t /GCD(t, i)
  - Consider $(\alpha^i)^{t/\text{GCD}(t,i)} = (\alpha^t)^{i/\text{GCD}(t,i)} = (1)^{i/\text{GCD}(t,i)} = 1$.
  - Thus, $x \mid t/\text{GCD}(t, i)$.
  - Now note $(\alpha^i)^x = \alpha^{ix} = 1$ since x is the order of α^i by definition.
  - This implies $t \mid ix$, which implies $t/\text{GCD}(t, i) \mid x$.
    - Note $qi+rt=\text{GCD}(i, t)$ for integers q and r and $tu=ix$ for integer u.
    - Multiply both side by x, then $xqi+xrt=x\text{GCD}(i, t)$.
    - Replace ix with tu on the left and have $(uq+xr)t = x\text{GCD}(i, t)$.
    - Divide both side by GCD(i,t), which shows $t/\text{GCD}(t, i) \mid x$.

The Euler Totient Function ϕ(t)

- $$\phi(t) := \left|\left\{1 \le i < t \,\middle|\, GCD(i,t) = 1\right\}\right|$$
  $$= t\prod_{p}\left(1 - \tfrac{1}{p}\right)$$
  where $p \in \{0 < p < t : p \text{ is prime and } p \mid t\}$

- Totative of t is a positive integer less than t that is relatively prime to t.

- Totient = # of totatives

Examples

- $\phi(1) := 1$
- $\phi(2) = 1$ {1}
- $\phi(3) = 2$ {1, 2}
- $\phi(4) = 2$ {1, 2, 3}
- $\phi(5) = 4$ {1, 2, 3, 4}
- $\phi(6) = 2$ {1, 2, 3, 4, 5}
- $\phi(7) = 6$ {1, 2, 3, 4, 5, 6}
- $\phi(8) = 4$ {1, 2, 3, 4, 5, 6, 7}
- $\phi(9) = 6$,
- $\phi(10) = 4$, …, etc.

Euler Totient Function (Cont'd)

- Properties

(1) $\phi(p) = p - 1$, when p is prime.

(2) $\phi(p_1 \cdot p_2) = \phi(p_1) \cdot \phi(p_2) = (p_1 - 1)\,(p_2 - 1)$, when p_1 and p_2 are distinct primes.

(1) $\phi(p^m) = p^{m-1}\,(p-1)$, for prime p.

(2) $\phi(p_1^{\,m}\, p_2^{\,m}) = p_1^{\,m-1}\, p_2^{\,m-1}\,(p_1 - 1)\,(p_2 - 1)$, for distinct primes p_1 and p_2.

• Let GF(q)=16, how many elements have order 2? → 0
3? → $\phi(3)=2$

Multiplicative Structure of GFs (Theorem 4)

In GF(q), $t \mid q$-1 means there are $\phi(t)$ elements of order t.

Proof:

- Suppose $t = \text{ord}(\alpha)$, $\alpha \in GF(q)$, then $\{\alpha, \alpha^2, \ldots, \alpha^t\}$ contains all the solutions to $x^t - 1 = 0$. each is distinct. each is a root of $x^t-1=0$.
- Then, all the elements of order t are in $\{\alpha, \alpha^2, \ldots, \alpha^t\}$.
- We know $\text{ord}(\beta) = t/\text{GCD}(i, t)$ for ~~all~~ any $\beta = \alpha^i$. (Thm 3) o.k.
  - $\text{Ord}(\alpha^i) = t$, ~~iff~~ GCD$(i, t) = 1$. o.k. if
- By definition, then, there are ϕ(t) such elements in $\{\alpha, \alpha^2, \ldots, \alpha^t\}$. ∃(t+1)

Why? ∃ are t roots total. But ~~all of~~ each of $\{\alpha, \alpha^2, \ldots, \alpha^t\}$ is a root. Each is distinct!

$2 \nmid \{1, 2, 3, \ldots, t\}$

In every GF(q), there exists a *primitive element.*

- An element with *order* q-1 in GF(q) is called *primitive*.
- In every GF(q), there are $\phi(q\text{-}1)$ primitive elements.
  - Corollary to Theorem 4.
  - Every GF(q) has at least one primitive element ($\phi(t) \geq 1$)
  - If $\alpha \in GF(q)$ is a primitive element, then

$$\underbrace{1, \alpha, \alpha^2, \alpha^3, \ldots,}_{\substack{\text{q-1 distinct} \\ \text{non-zero} \\ \text{elements in GF}(q)}} \underbrace{\alpha^{q-1}=1, \alpha^q, \alpha^{q+1}, \ldots}_{\text{Repeating}}$$

  - $\{1, \alpha, \alpha^2, \alpha^3, \ldots, \alpha^{q-1}\} = GF(q) - \{0\} =: GF^*(q)$

Quiz
• Every Galois field has at least one primitive element.

Quiz (Thm 4) • Take an element $\alpha \in GF(q)$ with order t.
All elements in the set $\{\alpha, \alpha^2, \dots, \alpha^t\}$ have order t.

T/F.

• $\exists$ one elements in the set $\{\alpha, \alpha^2, \dots, \alpha^t\}$ whose order is t.
more than

Example

- GF(7) = {0, 1, 2, 3, 4, 5, 6}
  - $GF^*(7) = \{1, 2, \dots, 6\}$
  - Possible orders $t \mid (q-1) = \{1, 2, 3, 6\}$.
  - # of elements of order t
  - Order t: 1, 2, 3, 6
  - $\phi(t) = 1, 1, 2, 2$
  - Primitive elements are 3 and 5
    - Ex) 1, 3, $3^2=2$ mod 7, $3^3 = 6$, $3^5 = 5$, $3^6 = 1$.

$3^4 =$

i	3^i	Ord(3^i) = 6/GCD(i,6)
0	1	1
1	3	6
2	2	3
3	6	2
4	4	3
5	5	6

~~This set is considered~~

$\{0, 1, 2(1), 3(1), \dots, (j-1)(1)\}$ ⇒ Then first one to repeat is zero.
⇒ no element is repetitive.
⇒ Elements are distinct.

Additive Structure of GFs

Take
- "1" ~~is~~ the multiplicative identity.
- Now consider

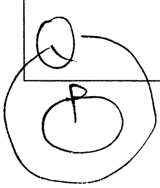
  0, 1, 1+1, 1+1+1, 1+1+1+1, …

  2(1), 3(1), 4(1), …

  just a way of representing the sum.
- The sequence must repeat (finite field).
- The first one to repeat is zero.
  - Suppose j(1) is the fist one to repeat.
  - Then, j(1) = k(1) for some $0 \le k < j$.
  - Then, k must be zero; o.w., (j-k)(1)=0 is an earlier repetition than j.

Proof by Contradiction.

To prove $P \Rightarrow Q$, show $P \wedge \sim Q$ something wrong.



~~Contrapositive~~

$P \Rightarrow Q \equiv \sim Q \Rightarrow \sim P$

yes!

Theorem 5

- The *characteristic* p of a GF(q) is the smallest integer, i.e., $p(1)=0$.

Suppose ~~If~~ $p \neq Q$
Let p be characteristic
but $p = p_1 p_2$.
for some p_1, p_2 prime.
Characteristic $p(1) = p_1(1) p_2(1) = 0$
$\therefore$ either $p_1(1) = 0$ or $p_2(1) = 0$.
which contradict the supposition p_1, p_2 prime.

- Theorem 5: The characteristic p of a GF(q) must be a prime integer.

If p is characteristic of GF(q), p prime.

- $Z_p = \{0, 1, 2, \ldots, p\text{-}1\}$, p prime, is the GF(p) under mod-p mult./add.

$Z_p = \{0, 1, 2(1), 3(1), \ldots, (p-1)1\}$

Theorem 6

- The order (size) q of a GF(q) must be a power of a prime.

Polynomials over GF(q): GF(q)[x]

- $a_0 + a_1 x^1 + a_2 x^2 + \dots + a_n x^n$
- $a_i \in \mathrm{GF}(q)$.
- Addition, Multiplication in GF(q).

$$\left(\sum_{i=0}^{n} a_i x^i\right) + \left(\sum_{i=0}^{n} b_i x^i\right) = \sum_{i=0}^{n} (a_i + b_i) x^i$$

$$\left(\sum a_i x^i\right)\left(\sum b_i x^i\right) = \sum_i \sum_{j=0}^{i} (a_j b_{i-j})_{GF(q)}\, x^i$$

ex) $a(x) = x^2+1$, $b(x) = x+1 \in GF(2)[x]$

$a(x) + b(x) = x^2 + x + 0$

$a(x)b(x) = (x^2+1)(x+1)$

$= x^3+x^2+x+1$

Monic if $a_n = 1$.

Irreducible Polynomials in GF(q)

- Definition: A polynomial $f(x)$ is *irreducible* in GF(q) if $f(x)$ cannot be factored into a product of lower-degree polynomials in GF(q)[x].
- An irreducible polynomial $f(x)$ has no roots in GF(q).
- Examples:

x^2+1 over $R[x]$

$(x^2+1) = (x+1)(x+1)$ over $GF(2)[x]$

(x^2+1) ~ irreducible over $GF(3)[x]$

$x^2+x+x+1$

$= x^2+2x+1$

$(x^2+x+1)^2$ has no roots in $GF(2)[x]$ but is irreducible.

Every polynomial is irreducible in some ring of polynomials.

Primitive Polynomials

- Definition: An irreducible polynomial $f(x) \in \mathrm{GF}(q)[x]$ of degree m, is called *primitive* if the smallest positive integer n for which $f(x) \mid (x^n - 1)$ is $n = q^m - 1$.
  - $f(x)$ has no roots in $\mathrm{GF}(q)$
  - $f(x)$ has m roots in $\mathrm{GF}(q^m)$ which are primitive elements in $\mathrm{GF}(q^m)$.

primitive elements

- Examples

Ex) x^3+x+1 is primitive in $GF(2)[x]$

$x^7-1 = (x^3+x+1)(x^4+x^2+x+1)$

is the smallest poly of the form x^n-1

and $7 = 2^3-1$

Ex) x^4+x+1 is primitive in $GF(2)[x]$ since the smallest poly of the form x^n-1 is $x^{2^4-1}-1$.

Theorem 7

- The roots $\{\alpha_j\}$ of an m-th degree primitive polynomial $p(x) \in \mathrm{GF}(p)[x]$ have order $p^m - 1$.

$$
\begin{array}{r|l}
 & x^4+x^2+x+1 \\
\hline
x^3+x+1 & x^7-1 \\
 & x^7+x^5+x^4 \\
\hline
 & x^5+x^4+1 \\
 & x^5+x^3+x^2 \\
\hline
 & x^4+x^3+x^2+1 \\
 & x^4+x^2+x \\
\hline
 & x^3+x+1 \\
 & x^3+x+1 \\
\hline
 & 0
\end{array}
$$

Construction of GF(8)

- $p(x) = x^3 + x + 1$ is primitive in GF(2)[x].
- Let α be a root of p(x), i.e., $\alpha^3+\alpha+1 = 0$ or equivalently $\alpha^3 = \alpha + 1$.

- Addition

  $\alpha^4 + \alpha^5$

  $= (\alpha^2+\alpha)+(\alpha^2+\alpha+1)$

  $= 1$
- Multiplication

  $\alpha^4 \times \alpha^5 = \alpha^{4+5 \bmod 7} = \alpha^2$

  or

  $= (\alpha^2+\alpha)(\alpha^2+\alpha+1)$

  $= \alpha^4+\alpha \bmod \alpha^3+\alpha+1 = \alpha^2$

Exponential Representation	Polynomial Representation	Vector Space
α^0	1	(1, 0 ,0)
α^1	α	(0, 1, 0)
α^2	α^2	(0, 0, 1)
α^3	$\alpha + 1$	(1, 1, 0)
α^4	$\alpha^2 + \alpha$	(0, 1, 1)
α^5	$\alpha^2 + \alpha + 1$	(1, 1, 1)
α^6	$\alpha^2 + 1$	(1, 0 ,1)
0	0	(0, 0, 0)

Construction of GF(4)

- $p(x) = x^2 + x + 1$ is primitive in GF(2)[x].
- Let α be a root of p(x), i.e., $\alpha^2+\alpha+1 = 0$ or $\alpha^2 = \alpha + 1$.

Exponential Representation	Polynomial Representation	Vector Space	Label
α^0	1	(1, 0)	1
α^1	α	(0, 1)	2
α^2	$\alpha + 1$	(1, 1)	3
0	0	(0, 0)	0

+	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	2	0

x	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	3	1
3	0	3	1	2

Summary

- Every Galois field has at least one primitive element.
- The size of GF(q) is a power of a prime.
- The smallest subfield of GF(p^m) is GF(p), where p is a prime and called the characteristic of GF(p^m).
- Primitive polynomial p(x) of degree $m \in$ GF(p)[x] has no roots in GF(p) but has m-roots $\in$ GF(p^m).
- The roots of primitive polynomials p(x) are the primitive elements in GF(p^m).

HW#1

(Due Wednesday, 9/15)

- P2.1, P2.4, P2.5, P2.18, P2.19, P2.20, P2.22, P2.25

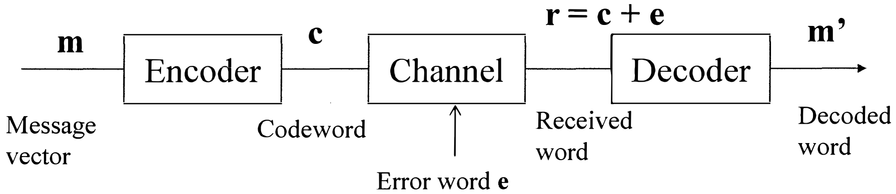
Linear Cyclic Codes

References: Moon Ch.3, Ch. 4,
Wicker Ch. 4, Ch.5,

Agenda

- (n, k) block codes
- (n, k) linear codes
- (n, k) cyclic linear codes

Coded System



- **m**, **c**, **r** and **m'** are vectors: From Shannon's work, we know block processing of information over noisy channel helps, rather than processing them in a bit-by-bit manner.

- As the size of the block increases, the larger is the potential to achieving the capacity; but the greater is the difficulty in decoding.

Kinds of Errors

- Soft errors
  - AWGN channel
- Hard errors
  - Quantizations
- Erasures
  - Loss of symbols, packets

(n, k) Linear Block Codes

- The block length n.
- The message length k.

- The code is q-ary when each coordiante takes a value from GF(q).

- Rate of the code $R = \log_q(M)/n$, and k=nR.
- Redundancy $r = n - \underbrace{\log_q(M)}_{k}$

(n, k) Linear Block Code

- A code $\mathcal{C}$ spans a vector space with dimension k. *It is a collection of M codewords.*

- *Linearity*: For any $a, b \in$ GF(q) and any $\mathbf{v}, \mathbf{u} \in \mathcal{C}$, $a\mathbf{v} \in \mathcal{C}$ and $a\mathbf{v}+b\mathbf{u} = \mathbf{c} \in \mathcal{C}$.
  - If $\mathbf{c}$ is a codeword, $0\mathbf{c} = \mathbf{0}$ is a codeword.
  - Let $d(\mathbf{v}, \mathbf{u})$ denote Hamming distance between any two different codewords $\mathbf{v}, \mathbf{u} \in \mathcal{C}$ and $w(\mathbf{v})$ Hamming weight of codeword $\mathbf{v}$ respectively.
  - Then,
    $$\begin{aligned} d_{min} &= \min d(\mathbf{v}, \mathbf{u}) \\ &= \min w(\mathbf{v} + \mathbf{u}) \\ &= \min w(\mathbf{c}=\mathbf{v} + \mathbf{u}) \\ &= \min w(\mathbf{c}), \text{ over all non-zero } \mathbf{c} \in \mathcal{C} \end{aligned}$$
  - It is the minimum weight of non-zero codeword.

A code = a generator matrix = a parity check matrix

- A linear block code can be defined by either a generator matrix or a parity-check matrix.
- Generator matrix is obtained by k linearly independent codewords.
  - The rows of **G**, generator matrix [nR x n] of a code, span the code space
- The rows of **H**, parity check matrix [$n(1-R)$ x n], span the linear space perpendicular to the row space of **G**.
  - There are $n(1-R)$ number of simultaneous linear homogeneous parity check equations.
  - There are $n(1-R)$ rows of **H** which span the null space of the code.

$$\mathbf{G}\mathbf{H}^T = 0$$

Distance Spectrum

- Hamming weight of a codeword is the number of non-zero coordinates.
- A_h is the number of codewords with weight h, h=0, 1, 2, …, n, in a code $\mathcal{C}$.
- Distance spectrum $\{A_h, h=0, 1, 2, \ldots, n\}$ is a collection of A_h.
- Polynomial representation is useful.
  - $$A(z) = \sum_{h=0}^{n} A_h z^h$$
  - $$A(z)\big|_{z=1} = \sum_{h=0}^{n} A_h = 2^k$$

MacWilliams Identity

Moon, pg. 95

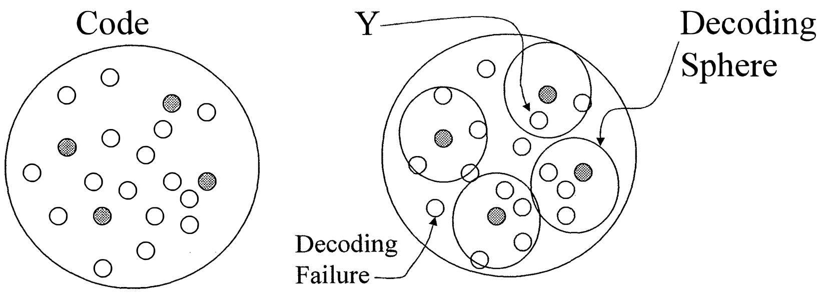
Theorem 3.6 *(The* **MacWilliams Identity***). Let C be an (n, k) linear block code over $\mathbb{F}_q$ with weight enumerator $A(z)$ and let $B(z)$ be the weight enumerator of $C^{\perp}$. Then*

$$B(z) = q^{-k}(1 + (q-1)z)^n A\left(\frac{1-z}{1+(q-1)z}\right), \tag{3.12}$$

or, turning this around algebraically,

$$A(z) = q^{-(n-k)}(1 + (q-1)z)^n B\left(\frac{1-z}{1+(q-1)z}\right). \tag{3.13}$$

Max. Likelihood Decoding



- Encoding: **m→ c**, it is a mapping from a block of message bits to a codeword.
- ML Decoding: ~~make~~ decision in favor of a message index *m* that maximizes Pr{Y|*m*}. $Pr(\underline{Y} \mid \underline{m})$
  - This ~~leads to~~ a minimum distance decoding rule.
  - Minimum distance errors dominate the error performance.

Decoders

- *Complete* decoders

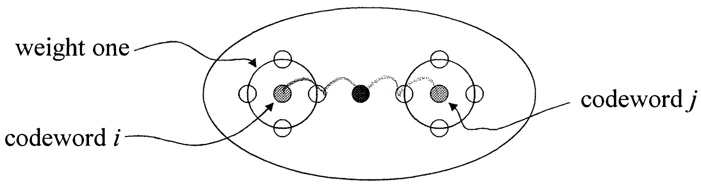
$$\hat{\mathbf{c}} = \min_{\mathbf{c}\in\mathcal{C}} d(\mathbf{r},\mathbf{c})$$

- t-error correcting *bounded-distance decoders*

$$\hat{\mathbf{c}} = \begin{cases} \min\limits_{\mathbf{c}:\, d(\mathbf{r},\mathbf{c})\le t} d(\mathbf{r},\mathbf{c}) \\ \text{failure, if not a single } c \text{ is found} \end{cases}$$

- *Erasure decoding*
  - *Error location is known*

Minimum Distance Decoding and Correctable Errors



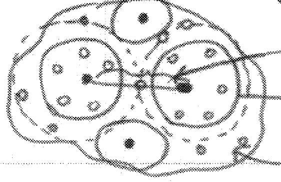
- Note d_{min} = 4. Thus, it can detect all error patterns up to weight 3.
- MLD = minimum distance decoding.
- The blue ball is a decoding failure because it has the same distance with codeword i and codeword j.
- **A code with d_{min} can correct all error patterns of weight <= *floor*((d_{min}- 1)/2)**

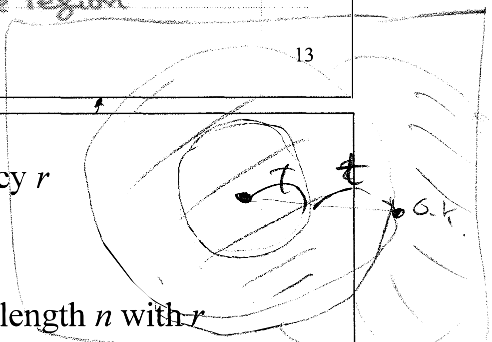
Error Correction and Detection Capability

- A code can correct up to all t errors if
  $t \le \left\lfloor \frac{d_{min}-1}{2} \right\rfloor$
- A code can detect up to all $d_{min}-1$ errors
- A code can correct up to e_1 errors and can detect up to e_2 errors if
  $\begin{cases} e_1 + e_2 = d_{min} - 1 \\ e_1 \le e_2 \end{cases}$

Example) (5, 2) code $d_{min} = n-k+1 = 5-2+1 = 4$

$\left\lfloor \frac{d_{min}-1}{2} \right\rfloor = \left\lfloor \frac{3}{2} \right\rfloor = 1 \ge t$





Upper Bound on Redundancy r
(Gilbert Bound)

- There exists a t-error correcting code of length n with r satisfying $r <= \log_q V_q(n, 2t)$.
  - Consider the pool of n-tuple vectors. There are q^n such vectors.
  - Choose one vector as a codeword, and eliminate all neighboring vectors in the Hamming sphere $V_q(n, 2t)$ from future selection.
  - Proceed until no selection can be made.
  - This insures the code's capability of correcting t errors.
  - $ceil(q^n / V_q(n, 2t))$ is the no. of codewords since overlapping is allowed.
  - $M = \text{ceil}(q^n / V_q(n, 2t)) >= q^n / V_q(n, 2t)$.
  - Redundancy $r := n - \log_q M \le n - (n - \log_q V_q(n, 2t))$.

$= \log_q V_q(n, 2t)$

$\left\lceil \frac{q^n}{V_q(n, 2t)} \right\rceil$

1개 $q^n - V_q(n, 2t)$

2개 $q^n - 2V_q(n, 2t) > 0$

$(M-1)$개 $q^n - (M-1)V_q(n, 2t)$

M개 ≥ 0

최대 갯수는

$= 2.5$

한개라도 word가 남아있으면 codeword간 틈이 있으므로 $M = \left\lceil \frac{q^n}{V_q(n, 2t)} \right\rceil \Leftarrow q^n - MV_q(n, 2t) \le 0$.

Error Correction and Detection Capability of a Code

- A code can correct up to all t-error patterns if

$$t \le \left\lfloor \frac{d_{\min} - 1}{2} \right\rfloor$$

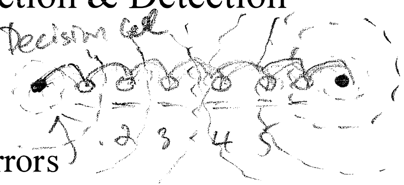
- A code can detect up to all $d_{min} - 1$ error patterns.

- A code can correct up to e_1-error patterns and detect all e_2-error patterns if
  - $e_1 + e_2 \le d_{\min} - 1$

    and
  - $e_1 \le e_2$

Simultaneous Correction & Detection

Decision cell



- Example with $d_{min} = 7$
- Correct 0 error and detect 6 errors e_2
- Correct 1 error and detect 5 errors
- Correct 2 errors and detect 4 errors
- Correct 3 errors and detect 3 errors
- $d_{min} = e_1 + e_2 + 1$ and $e_2 >= e_1$

- Recover up to d_{min} - 1 erasures.

$d=3$

[illegible] Error [illegible] ok Detect [illegible] error [illegible]

a) $d_{min} = 5$

$f = 1 \Rightarrow e = 1$ $\quad\therefore\quad 2(1)+1 < 5$

$2(2)+1 = 5 < 5$ Not hold.

$f = 2 \Rightarrow e = 1$

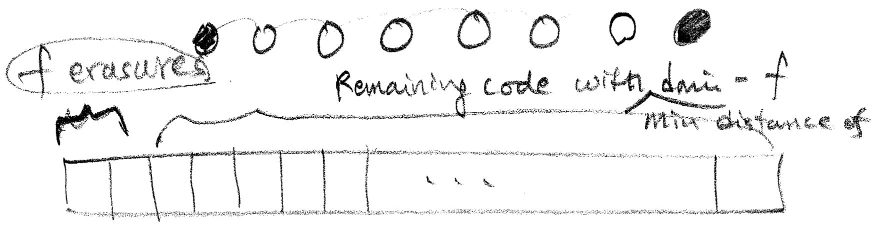
Erasures and Errors

$f = 3 \Rightarrow e = 0$, $f = 4$, $e = 0$

- A code with d_{min} can recover f erasures and e-errors if

$$2e + f < d_{min}.$$

  - With f erasures, the remaining code still has minimum distance of $d_{min} - f$.



Some Block Code Bounds

- How many vectors in n-dimensional space can be chosen as a codeword of a code with d_{min}?
- What is the minimum redundancy required for a t-error correcting q-ary code of length n?
- Only some bounds are available for these questions.

Hamming Sphere

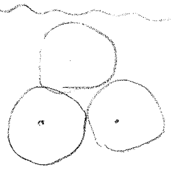
- Consider, received = codeword + error.
- A Hamming sphere of radius t is a set of all possible received words that are t distance away from the codeword.
- Weight of the error <= t
- How many error patterns (or the received words) are there in the sphere (for (n, k) code over GF(q))
  - No. of weight-one errors: n choose 1 x (q-1)
  - No. of weight-two errors: n choose 2 x (q-1)(q-1)
  - No. of weight-three errors: n choose 3 x (q-1)(q-1)(q-1)

$$V_q(n,t) = \sum_{j=0}^{t} \binom{n}{j} (q-1)^j$$

- This is the number of error patterns a t-error correcting code can correct where $t = floor((d_{min}-1)/2)$.

Lower Bound on Redundancy r *(Hamming Bound)*

- A t-error correcting q-ary code of length n must have redundancy $r \geq \log_q V_q(n, t)$.
  - A t-error correcting q-ary code of length n requires a Hamming sphere of size $V_q(n, t)$
  - How many such balls could n-dim space contain?
  - $q^n \geq M\, V_q(n, t)$ where M is no. of codewords
  - $q^n/M \geq V_q(n, t)$ or $q^n / V_q(n,t) \geq M$
  - Redundancy $r := n - \log_q M \geq \log_q V_q(n, t)$.



Perfect Codes

- Block codes that achieve the Hamming bound on redundancy.

- M must be of the form q^k.
- A q-ary (n, k) t-error correcting code is related by

$$\sum_{j=0}^{t} \binom{n}{j} (q-1)^j = q^{n-k}$$

- Example) Hamming, Repetition, Golay codes.

Linear Block Code (n, k) over GF(q)

- Code C is a vector space with dimension k.
- *Linearity*: For all a, $b \in$ GF(q) and **v**, **u** $\in$ C, a**v** $\in$ C and a**v**+b**u** $\in$ C.
  - If **c** is a codeword, **c** + (- **c**) = 0 is a codeword.
  - d_{min} = min d(**v**, **u**) for **v**, **u** $\in$ C is equivalent to

    d_{min} = min w(**v** - **u**)

    = min w(**c**=**v** - **u**, **0**), over all **c** $\in$ C,
  - the minimum weight of non-zero codeword.
- Completely defined by **G** or **H**, where $\mathbf{GH}^T=0$
  - Gaussian Elimination gives systematic **G** and **H**.
- **r = c + e.**
- **Standard array:** all possible **c** as the first row and some **e** as the coset leaders.

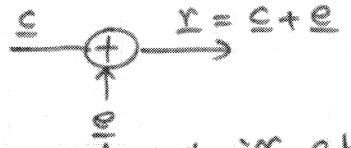
Syndrome

- Given the I/O relation **r** = **c** + **e**, a *syndrome* vector **s** is obtained by

$$\mathbf{s} = \mathbf{r}\mathbf{H}^T = (\mathbf{c}+\mathbf{e})\mathbf{H}^T = \mathbf{e}\mathbf{H}^T$$

  – Non-zero **s** indicates "there is a problem."

Syndrome Decoding



- $\underline{c}$ is a codeword iff $\underline{c}H^T = \underline{0}$
- $\underline{s} = \underline{r}H^T$, is $(n-k)$-tuple vector representing failures in parity checking, and is called <u>syndrome</u>
- Syndrome depend only on error pattern

$$\underline{s} = \underline{r}H^T = (\underline{c}+\underline{e})H^T = \underline{c}H^T + \underline{e}H^T = \underline{e}H^T$$

- Syndrome = Generated parity check + Received parity check

$$\underline{s} = (r_0\, r_1 \cdots r_k, r_{k+1}, \cdots, r_{n-1}) \begin{pmatrix} P \\ I_{n-k} \end{pmatrix} = (r_1 \cdots r_k)P + \underline{(r_{k+1} \cdots r_n)\, I_{n-k}}$$

Syndrome Decoding (2)

(22)

- There are 2^{n-k} syndromes
- Example : (5,2) code

Let $H^T = \begin{bmatrix} P \\ I_3 \end{bmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ and

and let $\underline{r} = (\underbrace{1\,0}_{\text{info bits}}\ \underbrace{1\,1\,1}_{\text{parity check bits}}) \oplus (\underbrace{0\,1\,0\,0\,0}_{\text{error pattern}}) = (\underbrace{1\,1\,1\,1\,1}_{\text{rec. word}})$

$\underline{s} = \underline{r}H^T = (1\,1 \vdots 1\,1\,1)H^T = [1\ 1]\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix} \oplus [1\ 1\ 1]\, I_3$

$= \underbrace{(0\,0\,1)}_{\text{Generated Parity Checks}} \oplus \underbrace{(1\,1\,1)}_{\text{Rec. Parity checks}} = \underbrace{(1\,1\,0)}_{\text{Syndrome}}$

Or $\underline{s} = \underline{e}H^T$

$= (0\,1\,0\,0\,0)H^T = 1\,1\,0 \Leftarrow$ store this in a ~~table~~ table look up.

~~Be~~ Correct error by adding the error pattern to $\underline{r}$

Syndrome Decoding (3)

(23)

- Choose 2^{n-k} most likely error patterns
  - start from smallest weight patterns
- Find all syndromes of most likely error patterns
- Construct a table look-up with 2 columns and 2^{n-k} rows (First syndrom, Second Error Pattern)
- Decoding Steps
  1 – Find syndrome $\underline{s} = \underline{r}H^T$
  2 – Use table (look up) to find the error pattern for the calculated syndrome in 1.
  3 – Add the correctable error pattern to the rec. word

Standard Array

- It is a $[2^{(n-k)} \times 2^k]$ table of words on length n.
- All 2^n words appear once and only once.
- It shows decoding cells.
- The first row lists up all possible 2^k codewords.
- The first column lists up all $2^{(n-k)}$ distinct error patterns **e** that can be recovered.

Standard Array for a (5, 2) LBC

Info bits	00	01	10	11	Syndrome
Codewords	00000	01110	10111	11001	000
Correctable Single Error Patterns	00001	01111	10110	11000	001
	00010	01100	10101	11011	010
	00100	01010	10011	11101	100
	01000	00110	11111	10001	110
	10000	11110	00111	01001	111
Correctable Error Patterns (w=2)	10100	11010	00011	01101	011
	10010	11100	00101	01011	101

$$H^T = \begin{pmatrix} 111 \\ 110 \\ 100 \\ 010 \\ 001 \end{pmatrix}$$

- Construct it row by row
  - Make sure not to select the error pattern which have appeared already
- When choosing the error patterns (1st column), make sure they lead to distinct syndrome.

d_{min} and Detectable Errors

- d_{min} is the minimum Hamming distance between any two codewords in C.
  - The minimum Hamming weight of any non-zero codeword in C
- Received = codeword(i) + error pattern
- When the error pattern itself is a codeword, then the received is a codeword but is not the transmitted codeword (i).
  - This is an undetectable error event. It happens IFF the error pattern is a codeword.
  - Syndrome is zero.
  - Minimum weight of such error patterns is d_{min} .
- An error pattern with weight less than d_{min} can always be detectable.
- **A code with d_{min} can detect all error patterns of weight <= d_{min}-1.**

The Singleton Bound (Linear Codes)

- The minimum distance d_{min} of a linear (n, k) code is bounded from above by $d_{min} <= n - k + 1$.
  - An (n, k) code has a parity matrix which contains $(n - k)$ linearly independent rows.
  - The dimension of row space, and thus that of the column space, is $(n - k)$.
  - Thus, any collection of $(n - k) + 1$ columns of **H** has to be linearly dependent.

Example

Example

$$H = \begin{array}{c} \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix} \\ \quad H^1 \;\; H^2 \;\; H^3 \;\; H^4 \;\; H^5 \end{array}$$

- the column vectors of H span a 3 dim. vector space.
- note $H^1 = H^3 + H^4 + H^5$

  $H^2 = H^3 + H^4$

  $d_{min} = 3$ n-k+1 = 5-2+1 = 4

Hamming Codes (Linear)

- A single-error correcting *perfect* code with $m >= 2$ parity symbols
  - $n = (q^m - 1)/(q - 1)$
  - $k = (q^m - 1)/(q - 1) - m$
  - $n - k = m$
  - $d_{\min} = 3$
  - The simplest Hamming code, m=3.

Example of Hamming Code (7,4)

$$H = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

true in general

- Note every pair of columns of H is independant.
- But, some collection of 3 columns are dependant.

$\Rightarrow d_{min} = 3$

Decoding of Hamming Code

- Compute the syndrome $\mathbf{s} = \mathbf{r}\,\mathbf{H}^T = \mathbf{e}\,\mathbf{H}^T$
- Can correct all error patterns of weight = 1
- For a single error occurred at j-th coordinate, the syndrome is equal to the j-th column of $\mathbf{H}$.
- Thus, decoding steps are
  1. Compute the syndrome.
  2. If zero, then the rec. word is a codeword.
  3. If not equal to zero, examine if any match can be found from the columns of $\mathbf{H}$. Record the column index j.
  4. Complement the j-th bit of the received word.

Example (4)

Decoding of Hamming code

④

$$H = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

$$\underline{r} = (1\ 0\ 1\ 0\ 1\ 1\ 0)$$

$$\underline{s} = \underline{r} H^T = (0\ 1\ 1)$$

$\therefore$ Error correction $\underline{r} + (0\ 0\ 1\ 0\ 0\ 0\ 0)$

$$= \underbrace{(1\ 0\ 0\ 0\ 1\ 1\ 0)}_{\text{Decided codeword}}$$

q-ary Hamming Code with m parity symbols

- Consider q-ary m-tuples
- There are $q^m - 1$ distinct non-zero vectors.
- For each vector **v**, there are $(q-1)$ vectors that are multiples of **v**
  - For all $a \in GF(q)$, $a * (v_1, v_2, v_3, \ldots) = (a*v_1, a*v_2, a*v_3, \ldots)$.
  - Thus, **v** and a**v** are linearly dependent for all $a \in GF(q)$.
- There are $(q^m - 1)/(q-1)$ such sets of multiples.
  - Select one vector from each set as columns of H.

Example (3)

q-ary Hamming Code with m parity symbols

- $q = 4$, $(5, 3)$ code $d_{min} \le 5-3+1 = 2+1 = 3$
- GF(4)

+	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

*	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	3	1
3	0	3	1	2

No two columns are the same.

$$H = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 \\ 1 & 2 & 3 & 0 & 1 \end{pmatrix}$$

$$G = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 2 \\ 0 & 0 & 1 & 1 & 3 \end{pmatrix}$$

(a) $\begin{pmatrix} 1 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \end{pmatrix} = 0$

$\begin{pmatrix} 1 \\ 0 \end{pmatrix} + 3\begin{pmatrix} 0 \\ 1 \end{pmatrix} + \begin{pmatrix} 1 \\ 3 \end{pmatrix} = 0$

$\vdots$

$d_{min} = 3$

Linear Cyclic Code

- A linear cyclic code is a block code that is closed under cyclic shifts.
- Consider the following examples:

0	0	0		0	0	0		0	0	0
0	1	1		1	0	1		1	1	0
1	0	1		0	1	0		0	1	1
1	1	0		1	1	1		1	1	1
Cyclic linear				Linear not cyclic				No cyclic! not linear!		

Usage of $x^n - 1$!

Cyclic shift

Polynomial Representation

- A codeword, $\mathbf{c} = (c_0, c_1, \ldots, c_{n-1})$
- Polynomial: $c(x) = c_0 + c_1 x + c_2 x^2 + \ldots + c_{n-1} x^{n-1}$
- Cyclic right-shift by j is $x^j c(x) \bmod x^n - 1$, denoted as $c(x)^{(j)}$

Very Important.

- Example:
  - $x\, c(x) = c_0 x + c_1 x^2 + \ldots + c_{n-1} x^n \bmod x^n - 1$
    $= c_{n-1} + c_0 x + \ldots + c_{n-2} x^{n-1} \bmod x^n - 1$
  - $x^2 c(x) = c_0 x^2 + c_1 x^3 + \ldots + c_{n-1} x^{n+1} \bmod x^n - 1$
    $= c_{n-1} x^1 + c_0 x^2 + \ldots + c_{n-2} x^n \bmod x^n - 1$
    $= c_{n-2} + c_{n-1} x + \ldots + c_{n-3} x^{n-1} \bmod x^n - 1$

$$\begin{array}{r} c_{n-1} \\ x^n - 1 \;)\; c_n x^n + c_{n-1} x^{n-1} \\ c_n x^n \quad - c_n \\ \hline 0 \end{array}$$

Cyclic shift 해도 codeword.

Then, for any $x^j c(x) \bmod (x^n - 1)$ is a codeword.

For this to happen, a codeword must be a factor of $(x^n - 1)$.

⇒ $(x^n - 1)$ when divided by this, each must reduce to $0 \bmod (x^n - 1)$.

Generator Polynomial $g(x)$

⇒ $c(x)$ must devide $x^n - 1$. & $c(x)^{(j)}$ also must devide $x^n - 1$.

- Every cyclic code has a generator polynomial
  $g(x) = g_0 + g_1 x + \ldots + g_r x^r$
  - Let $g_r = 1$, unique monic polynomial of minimum degree ($g_0 \neq 0$).
  - It is a codeword.
  - For a length n code, $r = n - k <= n - 1$.
- $g(x)$ generates its cyclic code.
  - $m(x)$ is a message polynomial with maximum degree $k - 1$.
  - Then, every $c(x) = m(x)g(x)$ is a codeword.

$c(x) = m(x)g(x)$ & $g(x)$ ⇒ ⇓ $\boxed{g(x)\,h(x) = (x^n - 1)}$

q-ary (n, k) Cyclic Code C

- C is a set of code polynomials.
- C has a *unique monic* polynomial $g(x)$ — the generator polynomial — with minimal degree $r < n$.
  - For every code polynomial $c(x)$ in C, $c(x) = m(x)\, g(x)$.
  - $g(x) \mid (x^n - 1)$ in $\mathrm{GF}(q)[x]$.
- From ~~a~~ the factorization of $(x^n - 1)$, we can select $g(x)$ and $h(x)$ such that $g(x)h(x) = (x^n - 1)$.
- $h(x)$ is a parity polynomial of degree $k = (n - r)$.
- $s(x) = c(x)h(x) = 0 \bmod (x^n - 1)$.

m(x) 와 h(x)의 관계는?

m(k) has degree K-1.

Generator Matrix **G**

- $c(x) = m(x)\, g(x) = (m_0 + m_1 x + \ldots + m_{n-r-1} x^{n-r-1})g(x)$
  $= m_0\, g(x) + m_1\, x\, g(x) + \ldots + m_{n-r-1} x^{n-r-1} g(x)$
  $= [m_0\ m_1 \ldots m_{n-r-1}]\ [g(x);\ x\, g(x);\ \ldots;\ x^{n-r-1}\, g(x)]$
- Then, the generator matrix **G** is

$$\begin{bmatrix} g_0\ g_1 \cdots g_r & & 0 \\ & \cdots & \\ 0 & g_0\ g_1 \cdots g_r & \\ & & g_0\ g_1 \cdots g_r \end{bmatrix}$$

- $\mathbf{c} = \mathbf{mG}$

Parity Matrix **H**

- $\mathbf{s} = \mathbf{c}\,\mathbf{H}^T$
- $\mathbf{c} = (c_0\ c_1\ \ldots\ c_{n-1})$
- **H** matrix is

$$\begin{bmatrix} h_k\ h_{k-1}\ \cdots\ h_0 & & & 0 \\ & \cdots & & \\ & & h_k\ h_{k-1}\ \cdots\ h_0 & \\ 0 & & & h_k\ h_{k-1}\ \cdots\ h_0 \end{bmatrix}$$

- $\mathbf{G}\mathbf{H}^T = 0$

Since $g(x)h(x) \bmod (x^n-1) = 0$, we aim to study factorization of (x^n-1). This enables us to choose $g(x)$ and $h(x)$.

Factoring $x^n - 1$

- When $n = q^m - 1$,
  - All non-zero elements $\alpha \in GF(q^m)$ are the n roots.
  - Separate these non-zero elements into conjugacy classes. $\{\alpha^{q^0}, \alpha^{q^1}, \alpha^{q^2}, \ldots\}$
  - Compute the minimal polynomial for each class $\in GF(q)[x]$.
- When n divides $q^m - 1$,
  - Look for element β with $\mathrm{ord}(\beta)=n$ in $GF(p^m)$
  - We know $1, \beta, \beta^2, \beta^3, \ldots, \beta^{n-1}$ are distinct roots of $x^n - 1$
    - Roots are generated by the powers of β. So, these powers of β are called the primitive n-th roots of unity.
  - see pg. 61, 73 Separate these roots into conjugacy classes and compute the minimal polynomials in $GF(p)[x]$ of the associated classes.

In general

- The cardinality of a conjugacy class is the order of the associated minimal polynomial.
- Number of classes ~~are~~ is the number of minimal polynomials.

* polynomial $(x^n - 1)$ can be uniquely factored into the product of irreducible polynomials (unique)

Just like a composite number can be uniquely factored into ~~the~~ product of primes.

An element with order $(q-1)$ is called primitive in $GF(q)$.

$\alpha^n = 1$ $x^n - 1$

Primitive *n*-th roots of unity

$\phi(n)$ The roots are primitive in ... since their orders are all the same with n.

~~smallest~~

- How do we find them?
- Recall from Galois Field Lectures Totient!
  - If $n \mid (q^m - 1)$, then there are $\phi(n)$ elements of order n in $GF(q^m)$.
- Find the smallest extension field of a ground field
  - $5 \mid (2^4 - 1)$, but not $(2^3 - 1)$, $(2^2 - 1)$, or $(2 - 1)$. Thus, GF(16) is the smallest extension field for finding primitive 5th roots of unity $(x^n - 1)$
  - GF(27) is the smallest extension field of GF(3) for finding primitive 13th roots of unity 3
  - GF(125) is the smallest extension field of GF(5) for finding primitive 31st roots of unity

$m = 4,\ q = 3$ $GF(3^3)$

$\boxed{31 \mid 124 = 5^3 - 1}$ but $31 \nmid 5^2 - 1$.

* $13 \mid 3^3 - 1$, $13 \mid 3^2 - 1$ (No) $13 \mid 3^1 - 1$ (No)

Thus, GF(27) is the smallest ~~extended~~ field of GF(3) when primitive 13th roots of unity can be found.

Factoring $x^5 - 1$ in GF(4)[x]

- Note m=2 is the smallest such that $5 \mid 4^m - 1$.
  - We can find the primitive 5th roots of unity in GF(16).
- All elements of GF(16) can be represented by the powers of a primitive element α.
- Note $\beta:=\alpha^3$ which is an order 5 element.
  - By the definition of order, the powers of β, 1, β, β^2, β^3 and β^4 (β^5 = 1) are distinct. i.e. 1 $= \beta^0$
  - They are the 5 roots of $x^5 - 1$.
- Conjugate classes wrt GF(4)
  - $\{1\} \quad \Rightarrow \quad x+1$
  - $\{\alpha^3, (\alpha^3)^4\} \quad \Rightarrow (x-\alpha^3)(x-\alpha^{12}) = x^2 + \alpha^{10}x + 1$
  - $\{\alpha^6, (\alpha^9)^4 = \alpha^9\} \quad \Rightarrow (x-\alpha^6)(x-\alpha^9) = x^2+(\alpha^9+\alpha^6)x + \alpha^6\alpha^9$
    $= x^2+\alpha^5x+1$

irreducible over GF(4).

$(x^5-1) = (x+1)(x^2+\alpha^{10}x+1)(x^2+\alpha^5x+1)$

~~over~~ in GF(4)[x]

Factoring $x^5 - 1$ in GF(2)[x]

- Conjugate classes wrt GF(2)
  - $\{1\} \quad \Rightarrow \quad x+1$
  - $\{\alpha^3, (\alpha^3)^2, \alpha^{12}, \alpha^{24} = \alpha^9\}$
    $\Rightarrow (x+\alpha^3)(x+\alpha^{12})(x+\alpha^6)(x+\alpha^9) = (x^2+\alpha^{10}x+1)(x^2+\alpha^5x+1)$
    $= x^4 + (\alpha^{10}+\alpha^5)x^3 + (\alpha^5\alpha^{10}+1+1)x^2 + (\alpha^5+\alpha^{10})x + 1$
    $= x^4 + x^3 + x^2 + x + 1$
  - The polynomials are in GF(2)[x]

$(x^5-1) = (x+1)(x^4+x^3+x^2+x+1)$.

~~over~~ in GF(2)[x]

Binary Cyclic Codes of Length 7

- Find $g(x) \mid (x^7 - 1)$ in GF(2)[x].
- $m = 3$ is the smallest such that $7 \mid (2^3 - 1)$.
- The 7[th] roots of unity can be found in GF(8).
  - They are in fact primitive elements.
- Find the conjugacy classes wrt a primitive element α.
  - $\{1\}$ $\Rightarrow$ $(x+1)$
  - $\{\alpha, \alpha^2, \alpha^4\}$ $\Rightarrow$ (x^3+x+1)
  - $\{\alpha^3, \alpha^6, \alpha^{12} = \alpha^5\}$ $\Rightarrow$ (x^3+x^2+1)
- Choose $g(x) = (x+1)(x^3+x+1) = x^4 + x^3 + x^{2} + 1$.
- Then, $h(x) = (x^3+x^2+1)$.

Typo!

$$\begin{array}{r} x^3 + x + 1 \\ x^4 + x^2 + x \\ \hline x^4 + x^3 + x^2 + 1 \end{array}$$

Binary Cyclic Codes of Length 7 (Cont'd)

- $g(x) = x^4 + x^3 + x + 1$
- $\mathbf{G} = \begin{array}{ccccc} g_0 & g_1 & & & \end{array}$ $\begin{bmatrix} 1 & 0 & 1 & 1 & 1 & & \\ & 1 & 0 & 1 & 1 & 1 & \\ & & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$

Typo?

Right!

- $h(x) = x^3+x^2+1$
- $\mathbf{H} = \begin{bmatrix} 1 & 1 & 0 & 1 & & & \\ & 1 & 1 & 0 & 1 & & \\ & & 1 & 1 & 0 & 1 & \\ & & & 1 & 1 & 0 & 1 \end{bmatrix}$
- $m(x) = (1+x)$, $c(x) = m(x)g(x) = 1+x+x^2+x^5 \Rightarrow (1\ 1\ 1\ 0\ 0\ 1\ 0)$

4-ary Cyclic Codes of Length 5

- We have factored $x^5 - 1$ in GF(4)[x].
  $$(x^5 - 1) = (x+1)(x^2 + \beta x + 1)(x^2 + \gamma x + 1)$$
- They are polynomials in GF(4)[x].
- GF(4) = $\{0, 1, \beta = \alpha^5, \gamma = \alpha^{10}\}$
- Recall the table, but we want to use 2 = β and 3 = γ.

+	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

*	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	3	1
3	0	3	1	2

4-ary Cyclic Codes of Length 5 (Cont'd)

- Choose $g(x) = (x+1)(x^2 + \gamma x + 1) = x^3+(1+\gamma)x^2+(1+\gamma)x+1$
  $= x^3+ \beta x^2 + \beta x + 1.$
- Then the corresponding $h(x) = x^2 + \beta x + 1$.
- Thus, they are for (5, 2) 4-ary code.
- $m(x) \in \{a+bx\text{: } a, b \in \text{GF}(4)\}$
  - $c(x) = (\gamma + \beta x)\, g(x) = \beta x^4+(\gamma+ \beta^2)x^3+(\gamma\beta+\beta^2)x^2+(\gamma\beta+\beta)x+\gamma$
    $= \beta x^4+ \quad 0 \quad x^3+ \quad \beta \quad x^2+ \quad \gamma \quad x+\gamma$

  corresponding to codeword = (γ, γ, β, 0, β)
- $\mathbf{G} = \begin{bmatrix} 1 & \beta & \beta & 1 & \\ & 1 & \beta & \beta & 1 \end{bmatrix}$ and $\mathbf{H} = \begin{bmatrix} 1 & \beta & 1 & & \\ & 1 & \beta & 1 & \\ & & 1 & \beta & 1 \end{bmatrix}$

β × ① = β γ γ β 0

+②

β

Systematic Encoding

- Consider (n, k) cyclic code with $g(x)$
- $\mathbf{m} = (m_0, m_1, \ldots, m_{k-1}) \Leftrightarrow m(x)=m_0+m_1x+\ldots+m_{k-1}x^{k-1}$
- $x^{n-k}m(x) = m_0x^{n-k}+m_1x^{n-k+1}+\ldots+m_{k-1}x^{n-1}$
  $\Leftrightarrow \mathbf{m} = (0, 0, \ldots, 0, m_0, m_1, \ldots, m_{k-1})$
- $x^{n-k}m(x) = Q(x)g(x) + d(x)$ where degree$(d(x))$ $<$ degree$(g(x))=n-k$ $=r$
- $c(x) := [x^{n-k}m(x) - d(x)] = Q(x)g(x)$ is a valid code polynomial.
- $\mathbf{c} = [-d_0, -d_1, \ldots, -d_{n-k-1}, m_0, m_1, \ldots, m_{k-1}]$.

Systematic Encoding Rule

- Multiply the message polynomial by x^{n-k}.
- Divide the result by $g(x)$ and get the remainder $d(x)$.
- $\boxed{\text{Set } c(x) = x^{n-k}\, m(x) - d(x).}$

Systematic Encoding of (5,2) 4-ary Code

- $\mathbf{G} = \begin{bmatrix} 1 & \beta & \beta & 1 & 0 \\ \beta & \beta & 1 & 0 & 1 \end{bmatrix}$ and $\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 1 & \beta \\ 0 & 1 & 0 & \beta & \beta \\ 0 & 0 & 1 & \beta & 1 \end{bmatrix}$

Syndrome Error Detection

- **For systematic code**
- Note the codeword $\mathbf{c}=(-d_0, -d_1, \ldots, -d_{n-k-1}, m_0, m_1, \ldots, m_{k-1})$
- The received $\mathbf{r} = (-d'_0, -d'_1, \ldots, -d'_{n-k-1}, m'_0, m'_1, \ldots, m'_{k-1})$
- Compute the estimate of the remainder $\mathbf{d}^*= (-d^*_0, -d^*_1, \ldots, -d^*_{n-k-1})$ using the received message block $\mathbf{m}'$.
- The syndrome is $\mathbf{s} = \mathbf{d}' - \mathbf{d}^*$.

Note $\underline{s} = H^T \underline{r} = [I \; P^T] \begin{bmatrix} \underline{d}' \\ \underline{m}' \end{bmatrix}$

$= \underline{d}' + P^T \underline{m}'$

$\underline{s} = H^T \underline{r} = [I \;\; P^T] \begin{bmatrix} -\underline{d}' \\ \underline{m}' \end{bmatrix} = -\underline{d}' + P^T \underline{m}'$

$(r \times 1)$

Syndrome Error Correction

It can be shown from $\underline{s} = H^T \underline{r}$.

- $s(x) = s_0 + s_1 x + \ldots + s_{r-1} x^{r-1}$ ① multiply $r(x)$ by $h(x)$; take modulo $(x^n - 1)$ ② divide $r(x)$ by $g(x)$ and take the residue.
- $\boxed{s(x) = r(x)h(x) \text{ modulo } (x^n - 1).}$
- Or, it can be obtained from $\boxed{r(x) = a(x)\, g(x) + s(x),}$ degree$(s(x))$ < degree$(g(x)) = n - k$.
- Tabulate the error pattern for each syndrome.
- Or, use the shift register circuits for decoding.

Syndrome Error Computation

- $r(x) = r_0 + r_1 x + \ldots + r_{n-1} x^{n-1}$
- By cyclically shifting the coefficients of $r(x)$ once to the right we have

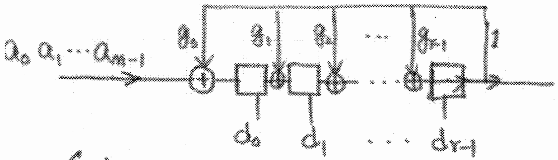
  $r^{(1)}(x) = r_{n-1} + r_0 x + \ldots + r_{n-2} x^{n-1}$ $\quad r_{n-1}$ €
- $x\, r(x) = \quad r_0 x + r_1 x^2 + \ldots + r_{n-1} x^n$
- $r^{(1)}(x) = x r(x) - r_{n-1} (x^n - 1)$
- Express $r(x)$ and $r^{(1)}(x)$ as multiples of $g(x)$ and remainders
- $r^{(1)}(x) = x(a(x)g(x) + s(x)) - r_{n-1}\, g(x)h(x) = b(x)g(x) + \underbrace{d(x)}_{= x s(x)}$
- $xs(x) = [b(x) - xa(x) + r_{n-1} h(x)]g(x) + d(x)$
- Thus, $\underbrace{xs(x) \text{ mod } g(x)}_{s^{(1)}(x)}$ is the syndrome for $r^{(1)}(x)$

Repeat the process n times.
We only need to store one syndrome $\underline{s}$ for an error pattern $\underline{e}$ and all cyclic shifts of $\underline{e}$.

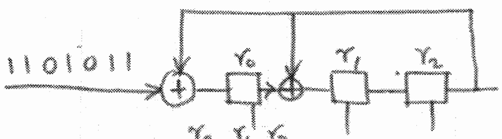
Long Division

$$a(x) = Q(x)\,g(x) + d(x)$$

$$\begin{cases} a(x) = a_0 + a_1 x + \cdots + a_{m-1} x^{m-1} \\ g(x) = g_0 + g_1 x + \cdots + g_{r-1} x^{r-1} + x^r \end{cases}$$



Ex)



$a(x) = 1 + x + x^3 + x^5 + x^6$

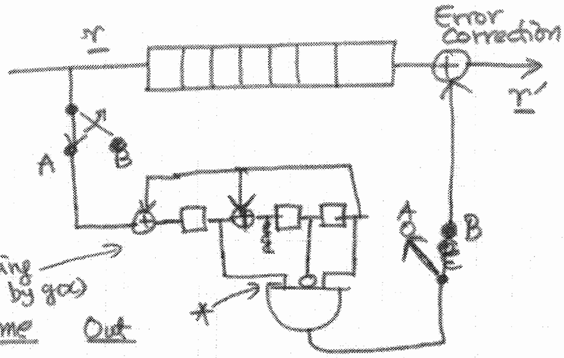
$g(x) = 1 + x + x^3$

Input	r_0	r_1	r_2
1101011	0	0	0
-110101	1	0	0
--11010	1	1	0
---1101	0	1	1
----110	0	1	1
-----11	1	1	1
------1	0	0	1
	0	1	0

$$\begin{array}{r} x^3+x^2+x+1 \\ x^3+x+1 \,\big)\, x^6+x^5+x^3+x+1 \\ x^6+x^4+x^3 \\ \hline x^5+x^4+x+1 \\ x^5+x^3+x^2 \\ \hline x^4+x^3+x^2+x+1 \\ x^4+x^2+x \\ \hline x^3+1 \\ x^3+x+1 \\ \hline x \end{array}$$

Shift Register Decoder for (7,4) code

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



dividing s(x) by g(x)

Rec Word Buffer	Syndrome	Out
1101011	010	1
110101	001	1
11010	110	0
1101	011	1
110	111	0
11	101	0
1	100	1

A: Loading
B: Error Correcting

Summary of Linear Cyclic Codes

What's in Moon Ch3. and Ch4?

- MacWilliams Identity
- Soft-Decision Decoders
- Coding Gain
- Shortening/Extending Block Codes

Midterm

- Oct. 27th, 2010, Wednesday
- One single page cheat sheet
- Coverage
  - Moon Chapters 2, 3, 4, 5, 6

HW #3

- Problem #1:
  Moon P3.3, P3.8, P3.20, P3.26, P4.1, P4.9, P4.11, P4.15
- Problem #2: $g(x) = x^6+3x^5+x^4+x^3+2x^2+2x+1$ is the generator polynomial for a (15, 9) double error correcting code over GF(4)
  - A) is $v(x) = x^{10}+3x^2+x+2$ a codeword?
  - B) Compute the syndrome polynomial of $v(x)$
  - C) How many syndrome polynomials must be tabulated to cover all correctable error patterns?

BCH and Reed Solomon Codes

Ref: Moon Ch 6, Wicker Ch. 8, 9

Brief History

- Algebraic cyclic codes, done mostly in 50s and 60s.
- Code design: Hocquenghem[59], Bose and Chaudhuri[60], Reed and Solomon[60]
  - Guaranteed t-error correcting code.
- Decoder designs in 60s: Peterson, Gorenstein and Zierler, Chien, Forney, Berlekamp, Massey
  - Berlekamp's iterative algorithm and Chien's search algorithm considered most efficient
- Decoder design in 90s: Sudan and Guruswami's list decoder capable of decoding beyond the design distance of the code.
  - Covered in Moon Ch. 7, better to read their papers.

Motivation for Studying BCH and RS Codes

- Most successful algebraic codes.
- What is the current status of these codes?
  - Google, Wikipedia

Coverage in this course

- Code design
- Review of some basic decoders

Question of Distance Measure in GF(p)

- *Can we define a set of positive or negative numbers in a finite field?*
  - No. Just think of Z_p where p is prime. Mod-p are the operations. Under modulo operations, can we think of the notion of positive or negative? O.K. but why?
- Ordered fields: real, rational, complex numbers.
  - A field must satisfy Axioms of Order (*Real Analysis* by Royden)
    - $(x, y \in P) \Rightarrow x + y \in P$ (P is a subset of positive real.)
    - $(x, y \in P) \Rightarrow xy \in P$
    - $(x \in P) \Rightarrow -x \notin P$
    - $(x \in R) \rightarrow (x = 0)$ or $(x \in P)$ or $(-x \in P)$
  - Any system satisfying the field requirement as well as this Axiom is an ordered field.
  - Notion $x > y$ is to mean $y - x \in P$.
- A field can be an ordered field IFF it has characteristic 0.
  $0 < 1, 1+1, 1+1+1, \ldots$
- A finite field cannot be turned into an ordered field, because they do not have characteristic 0.

The BCH bound and BCH codes

- Code design starts with selecting a *design distance D.*
- The BCH bound ensures the minimum distance of a code:
  - Consider a q-ary (n, k) cyclic-code with $g(x)$ for which $GF(q^m)$ is the smallest extension field that contains a *primitive* n-th root of unity α (i.e. $n \mid q^m - 1$).
  - **If $g(x)$ is constructed by a set of D consecutive powers of α, the code defined by g(x) has minimum distance, $d_{min} \geq D + 1$.**
    - **Proof: see next pages**
- Select $D = 2t$. The code can correct all errors up to t-errors.

Generator/Parity Check of BCH codes

- $g(x)$ = LCM of minimal polynomials for the D consecutive powers of α.
  - LCM is the sufficient condition that the roots of $g(x)$ are the D consecutive powers of α.

$= C_0 + C_1 x^1 + C_2 x^2 + \cdots + C_{n-1} x^{n-1}$

- A code polynomial $c(x)$ is a multiple of generator polynomial. ($\Leftrightarrow$ $c(\alpha^b) = c(\alpha^{b+1}) = \ldots = c(\alpha^{b+D-1}) = 0$ for some $b = 1, 2, \ldots$)
  - *Nomenclature: Narrow sense* ($b = 1$) and *primitive* ($n = q^m - 1$)

- Consider $\mathbf{s} = \mathbf{H}\mathbf{c}^T$:

These D eq's are all lin. independent!

$$\begin{pmatrix} 1 & \alpha^b & \alpha^{2b} & \alpha^{3b} & \cdots & \alpha^{(n-1)b} \\ 1 & \alpha^{b+1} & \alpha^{2(b+1)} & \alpha^{3(b+1)} & \cdots & \alpha^{(n-1)(b+1)} \\ 1 & \alpha^{b+2} & \alpha^{2(b+2)} & \alpha^{3(b+2)} & \cdots & \alpha^{(n-1)(b+2)} \\ 1 & \alpha^{b+D-1} & \alpha^{2(b+D-1)} & \alpha^{3(b+D-1)} & \cdots & \alpha^{(n-1)(b+D-1)} \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \cdots \\ c_{(n-1)} \end{pmatrix} = \mathbf{0}$$

Singleton Bound $D_{min} \leq D + 1$ ①

If # of errors smaller than t, all errors can be found & corrected.

The following proof shows that $\boxed{D_{min} > 2t = D.}$

② $D_{min} \geq 2t+1$

From ① + ② $D_{min} = D$.

Proof of the BCH Bound

- Define a support set of indices, $\{i_1, i_2, \ldots, i_w\}$, on which the codeword coefficients are non-zero.
  - Note then that the weight of the codeword is w.

Shows that the $D_{min} \geq 2t+1$.

- Proof by Contradiction: Suppose $w \leq D$, and show it leads to contradiction

$$\begin{pmatrix} \alpha^{i_1 b} & \alpha^{i_2 b} & \cdots & \alpha^{i_w b} \\ \alpha^{i_1(b+1)} & \alpha^{i_2(b+1)} & \cdots & \alpha^{i_w(b+1)} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha^{i_1(b+D-1)} & \alpha^{i_2(b+D-1)} & \cdots & \alpha^{i_w(b+D-1)} \end{pmatrix} \begin{pmatrix} c_{i_1} \\ c_{i_2} \\ \vdots \\ c_{i_w} \end{pmatrix} = (\mathbf{0}_{w\times 1})$$

Vandermonde matrix is full rank!

- This matrix is D x w.
- The w columns should be linearly dependent.
- However, a mathematical fact dictates that the matrix is full rank and thus the columns are linearly independent. (see next page)
- Thus, any weight w should be $w > D$.

Proof of the BCH Bound(2)

- (Since $w \le D$, we can eliminate the rows from below and obtain a square w x w matrix.)
- The equation in the previous page can be rewritten as:

normalize the first row

$$\begin{pmatrix} 1 & 1 & \cdots & 1 \\ \alpha^{i_1} & \alpha^{i_2} & \cdots & \alpha^{i_w} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha^{i_1(w-1)} & \alpha^{i_2(w-1)} & \cdots & \alpha^{i_w(w-1)} \end{pmatrix} \begin{pmatrix} \alpha^{i_1 b} & & & 0 \\ & \alpha^{i_2 b} & & \\ & & \ddots & \\ 0 & & & \alpha^{i_w b} \end{pmatrix} \begin{pmatrix} c_{i_1} \\ c_{i_2} \\ \vdots \\ c_{i_w} \end{pmatrix} = (\mathbf{0}_{w\times 1})$$

- This leads to det(**H**) $\neq 0$:

$$\det(\mathbf{H}) = \alpha^{b(i_1+i_2+\cdots+i_w)} \begin{vmatrix} 1 & 1 & \cdots & 1 \\ \alpha^{i_1} & \alpha^{i_2} & \cdots & \alpha^{i_w} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha^{i_1(w-1)} & \alpha^{i_2(w-1)} & \cdots & \alpha^{i_w(w-1)} \end{vmatrix}$$

$$= \alpha^{b(i_1+i_2+\cdots+i_w)} \prod_{1\le j<k\le w} \left(\alpha^{i_k} - \alpha^{i_j}\right) \neq 0$$

Q.E.D.

Binary BCH codes of length 31
(Primitive BCH codes)

- Let α be a primitive element of order 31 in GF(32).
- Use the add-one tables, and Appendix C and Appendix D to find the conjugacy classes and the minimal polynomials

Exponents of α (Cyclotomic Cosets)	Minimal Polynomials
$C_0 = \{0\}$	$M_{(0)} = x + 1$
$C_1 = \{1, 2, 4, 8, 16\}$	$M_{(1)} = x^5 + x^2 + 1$
$C_3 = \{3, 6, 12, 24, 17\}$	$M_{(3)} = x^5 + x^4 + x^3 + x^2 + 1$
$C_5 = \{5, 10, 20, 9, 18\}$	$M_{(5)} = x^5 + x^4 + x^2 + x^1 + 1$
$C_7 = \{7, 14, 28, 25, 19\}$	$M_{(7)} = x^5 + x^3 + x^2 + x^1 + 1$
$C_{11} = \{11, 22, 13, 26, 21\}$	$M_{(11)} = x^5 + x^4 + x^3 + x^1 + 1$
$C_{15} = \{15, 30, 29, 27, 23\}$	$M_{(15)} = x^5 + x^3 + 1$

$\alpha^{2^0}, \alpha^{2^1}, \alpha^{2^2}, \alpha^{2^3}, \alpha^{2^4}$

2^m 아!

Wicker

Appendix C

Cyclotomic Cosets *modulo* $2^m - 1$

The following is a list of the cyclotomic cosets *modulo* $2^m - 1$ with respect to GF(2). The coset containing the integer x is as follows.

$$\{x \bmod (2^m-1),\ x\cdot 2 \bmod (2^m-1),\ x\cdot 2^2 \bmod (2^m-1), \ldots, x\cdot 2^{d-1} \bmod (2^m-1)\}$$

The cardinality d of the coset must be a divisor of m. In coding theory we are primarily interested in cyclotomic cosets *modulo* $(2^m - 1)$ because they partition the $(2^m - 1)$ powers of a primitive element in GF(2^m) into distinct sets of conjugate elements. By Theorem 3-4 these sets correspond to the binary minimal polynomials for the elements in GF(2^m). For example, let α be primitive (i.e., order 7) in GF(8). Under the heading "*modulo* 7" we see {0}, {1, 2, 4} and {3, 5, 6}—one cyclotomic coset of size 1 and two of size 3. It follows that $m_0(x) = (x + 1)$, $m_1(x) = (x + \alpha)(x + \alpha^2)(x + \alpha^4)$, and $m_3(x) = (x + \alpha^3)(x + \alpha^6)(x + \alpha^5)$ are the three distinct minimal polynomials for elements in GF(8).

modulo 3

{0},
{1, 2}

GF(8) *modulo 7*

{0},
{1, 2, 4}, {3, 5, 6}

modulo 15

{0},
{5, 10},
{1, 2, 4, 8}, {3, 6, 9, 12}, {7, 11, 13, 14}

modulo 31

Roots
$m_1(x)$ $m_3(x)$ $m_5(x)$

{0},
{1, 2, 4, 8, 16}, {3, 6, 12, 17, 24}, {5, 9, 10, 18, 20}, {7, 14, 19, 25, 28}, {11, 13, 21, 22, 26},
{15, 23, 27, 29, 30}

modulo 63

{0},
{21, 42},
{9, 18, 36}, {27, 45, 54},
{1, 2, 4, 8, 16, 32}, {3, 6, 12, 24, 33, 48}, {5, 10, 17, 20, 34, 40},
{7, 14, 28, 35, 49, 56}, {11, 22, 25, 37, 44, 50}, {13, 19, 26, 38, 41, 52},
{15, 30, 39, 51, 57, 60}, {23, 29, 43, 46, 53, 58}, {31, 47, 55, 59, 61, 62}

modulo 127

{0},
{1, 2, 4, 8, 16, 32, 64}, {3, 6, 12, 24, 48, 65, 96}, {5, 10, 20, 33, 40, 66, 80},
{7, 14, 28, 56, 67, 97, 112}, {9, 17, 18, 34, 36, 68, 72}, {11, 22, 44, 49, 69, 88, 98},
{13, 26, 35, 52, 70, 81, 104}, {15, 30, 60, 71, 99, 113, 120}, {19, 25, 38, 50, 73, 76, 100},
{21, 37, 41, 42, 74, 82, 84}, {23, 46, 57, 75, 92, 101, 114}, {27, 51, 54, 77, 89, 102, 108},
{29, 39, 58, 78, 83, 105, 116}, {31, 62, 79, 103, 115, 121, 124}, {43, 45, 53, 85, 86, 90, 106},
{47, 61, 87, 94, 107, 117, 122}, {55, 59, 91, 95, 109, 110, 118},
{63, 95, 111, 119, 123, 125, 126}

modulo 255

{0},
{85, 170},
{17, 34, 68, 136}, {51, 102, 153, 204}, {119, 187, 221, 238},
{1, 2, 4, 8, 16, 32, 64, 128}, {3, 6, 12, 24, 48, 96, 129, 192}, {5, 10, 20, 40, 65, 80, 130, 160},
{7, 14, 28, 56, 112, 131, 193, 224}, {9, 18, 33, 36, 66, 72, 132, 144},
{11, 22, 44, 88, 97, 133, 176, 194}, {13, 26, 52, 67, 104, 134, 161, 208},
{15, 30, 60, 120, 135, 195, 225, 240}, {19, 38, 49, 76, 98, 137, 152, 196},
{21, 42, 69, 81, 84, 138, 162, 168}, {23, 46, 92, 113, 139, 184, 197, 226},
{25, 35, 50, 70, 100, 140, 145, 200}, {27, 54, 99, 108, 141, 177, 198, 216},
{29, 58, 71, 116, 142, 163, 209, 232}, {31, 62, 124, 143, 199, 227, 241, 248},
{37, 41, 73, 74, 82, 146, 148, 164}, {39, 57, 78, 114, 147, 156, 201, 228},
{43, 86, 89, 101, 149, 172, 178, 202}, {45, 75, 90, 105, 150, 165, 180, 210},

Wicker

Appendix D[1]

Minimal Polynomials of Elements in GF(2^m)

The following is a list of the binary minimal polynomials for all elements in binary extension fields from GF(2^2) through GF(2^{10}). The polynomials are denoted here by the exponent of one root and the powers of the polynomials' nonzero terms. For example, the entry "3 (0,2,3)" under the heading "GF(8)" corresponds to $m(x) = 1 + x^2 + x^3$, which has as roots the conjugate elements $\{\alpha^3, \alpha^6, \alpha^5\}$, where α is primitive in GF(8). To conserve space, the minimal polynomials are enumerated using the root with the smallest exponent. A complete listing of the exponents of the elements in the various conjugacy classes (i.e., the cyclotomic cosets) can be found in Appendix C.

GF(4)

1	(0, 1, 2)		

α $\{\alpha, \alpha^2, \alpha^4\}$ $\alpha^3, \alpha^6, (\alpha^3)^4$

GF(8)

1	(0, 1, 3)	3	(0, 2, 3)

GF(16)

1	(0, 1, 4)	3	(0, 1, 2, 3, 4)
5	(0, 1, 2)	7	(0, 3, 4)

GF(32)

$M_1(x) = 1 + x^2 + x^5$ $M_3(x) = 1 + x^2 + x^3 + x^4 + x^5$

1	(0, 2, 5)	3	(0, 2, 3, 4, 5)
5	(0, 1, 2, 4, 5)	7	(0, 1, 2, 3, 5)
11	(0, 1, 3, 4, 5)	15	(0, 3, 5)

[1] The material in this appendix is taken, with permission, from S. Lin and D. J. Costello, Jr., *Error Control Coding: Fundamentals and Applications*, pp. 579–582. Englewood Cliffs, NJ: Prentice Hall, 1983.

GF(64)

1	(0, 1, 6)	3	(0, 1, 2, 4, 6)
5	(0, 1, 2, 5, 6)	7	(0, 3, 6)
9	(0, 2, 3)	11	(0, 2, 3, 5, 6)
13	(0, 1, 3, 4, 6)	15	(0, 2, 4, 5, 6)
21	(0, 1, 2)	23	(0, 1, 4, 5, 6)
27	(0, 1, 3)	31	(0, 5, 6)

GF(128)

1	(0, 3, 7)	3	(0, 1, 2, 3, 7)
5	(0, 2, 3, 4, 7)	7	(0, 1, 2, 4, 5, 6, 7)
9	(0, 1, 2, 3, 4, 5, 7)	11	(0, 2, 4, 6, 7)
13	(0, 1, 7)	15	(0, 1, 2, 3, 5, 6, 7)
19	(0, 1, 2, 6, 7)	21	(0, 2, 5, 6, 7)
23	(0, 6, 7)	27	(0, 1, 4, 6, 7)
29	(0, 1, 3, 5, 7)	31	(0, 4, 5, 6, 7)
43	(0, 1, 2, 5, 7)	47	(0, 3, 4, 5, 7)
55	(0, 2, 3, 4, 5, 6, 7)	63	(0, 4, 7)

GF(256)

1	(0, 2, 3, 4, 8)	3	(0, 1, 2, 4, 5, 6, 8)
5	(0, 1, 4, 5, 6, 7, 8)	7	(0, 3, 5, 6, 8)
9	(0, 2, 3, 4, 5, 7, 8)	11	(0, 1, 2, 5, 6, 7, 8)
13	(0, 1, 3, 5, 8)	15	(0, 1, 2, 4, 6, 7, 8)
17	(0, 1, 4)	19	(0, 2, 5, 6, 8)
21	(0, 1, 3, 7, 8)	23	(0, 1, 5, 6, 8)
25	(0, 1, 3, 4, 8)	27	(0, 1, 2, 3, 4, 5, 8)
29	(0, 2, 3, 7, 8)	31	(0, 2, 3, 5, 8)
37	(0, 1, 2, 3, 4, 6, 8)	39	(0, 3, 4, 5, 6, 7, 8)
43	(0, 1, 6, 7, 8)	45	(0, 3, 4, 5, 8)
47	(0, 3, 5, 7, 8)	51	(0, 1, 2, 3, 4)
53	(0, 1, 2, 7, 8)	55	(0, 4, 5, 7, 8)
59	(0, 2, 3, 6, 8)	61	(0, 1, 2, 3, 6, 7, 8)
63	(0, 2, 3, 4, 6, 7, 8)	85	(0, 1, 2)
87	(0, 1, 5, 7, 8)	91	(0, 2, 4, 5, 6, 7, 8)
95	(0, 1, 2, 3, 4, 7, 8)	111	(0, 1, 3, 4, 5, 6, 8)
119	(0, 3, 4)	127	(0, 4, 5, 6, 8)

GF(512)

1	(0, 4, 9)	3	(0, 3, 4, 6, 9)
5	(0, 4, 5, 8, 9)	7	(0, 3, 4, 7, 9)
9	(0, 1, 4, 8, 9)	11	(0, 2, 3, 5, 9)
13	(0, 1, 2, 4, 5, 6, 9)	15	(0, 5, 6, 8, 9)
17	(0, 1, 3, 4, 6, 7, 9)	19	(0, 2, 7, 8, 9)

Example BCH codes

$D = r = n - k \Rightarrow$

- *Binary BCH codes*
  - $A\ t = 1$ error correcting (31, 26) *primitive* BCH code ($D = 2t = 2$)
  - $A\ t = 2$ error correcting (31, 21) *primitive* BCH code ($D = 2t = 4$)

No, No, No, $k = n - D = 31 - 2 = 29$.

4-ary BCH codes of length 21

- First, find the extension field GF(4^m) which contains a primitive 21st root of unity γ.
  - Such an m is 3. Thus, it's GF(4^3=64).
  - We can let $\gamma = \alpha^3$ where α is a primitive element of GF(64).
- Second, decompose the 21 roots into conjugacy classes (taking it to the powers of 4^d, $d = 0, 1, 2, \ldots$)
  - Find the minimal polynomials associated with each class, note that the coefficients are from the ground field GF(4) = $\{0, 1, \gamma^7 = \beta, \gamma^{14} = \beta^2\}$.

Cosets and Minimal Polynomials for 4-ary BCH of Length 21

Exponents of γ
(Cyclotomic Cosets)
$C_0 = \{0\}$
$C_1 = \{1, 4, 16\}$
$C_2 = \{2, 8, 11\}$
$C_3 = \{3, 12, 6\}$
$C_5 = \{5, 20, 17\}$
$C_7 = \{7\}$
$C_9 = \{9, 15, 18\}$
$C_{10} = \{10, 19, 13\}$
$C_{14} = \{14\}$

Minimal Polynomials

$M_{(0)} = x + 1$
$M_{(1)} = x^3 + \beta^2 x + 1$
$M_{(2)} = x^3 + \beta x + 1$
$M_{(3)} = x^3 + x^2 + 1$
$M_{(5)} = x^3 + \beta^2 x^2 + 1$
$M_{(7)} = x + \beta$
$M_{(9)} = x^3 + x + 1$
$M_{10} = x^3 + \beta x^2 + 1$
$M_{14} = x + \beta^2$

Non binary BCH code examples

- *Non Binary BCH codes*
  - Use Appendix B only
  - 4-ary BCH codes of length 21
    - $t = 1$
    - $t = 2$

Some Design Considerations in BCH codes

- The redundancy r is the number of roots in $g(x)$.
- In t-error correcting BCH codes, having $2t$ consecutive powers in $g(x)$ is the sufficient condition for having

  $d_{\min} \geq 2t + 1.$
- For the same error correction capability, a code with a higher code rate, or with a lower redundancy, is certainly desirable.
- $g(x)$ contains more roots than required number of roots, $r \geq 2t$. Why?
  - The cardinality of conjugacy class is greater than 1.
  - The *extraneous* roots are conjugates of the desired roots.
- One way to deal with this problem is to
  - Choose the starting power exponent d wisely, i.e. α^b, α^{b+1}, …, α^{b+D-1} so that the number of extraneous roots are minimized.

Some Design Considerations in BCH codes (2)

- The larger the cardinality of conjugacy classes, the more $g(x)$ contains extraneous roots.
- Now, let's think about how to reduce the cardinality of conjugacy classes?
- Observation: the cardinality is smaller for
  - *Primitive* codes such that $n = q^m - 1$ for a fixed q-ary symbol alphabet.
  - Codes with the size of code-symbol getting closer to n.
- Reed-Solomon Codes
  - q^m-ary BCH codes of length $q^m - 1$.

Reed Solomon Codes

- RS codes are q^m-ary BCH codes of length $n = q^m - 1$.
  - The primitive n-th roots of unity are in GF(q^m).
  - Let α be such one, and then the powers of α are the n distinct non-zero field elements in GF(n+1).

- Now, consider t-error correcting codes.
- The minimal polynomials for each element in GF(q^m) wrt GF(q^m) are first degree polynomials, i.e.

$$(x - \alpha^s), \text{ for } s = 0, 1, \ldots, n.$$

  - The size of conjugacy classes is always equal to 1!!

- Thus, $g(x)$ is the product of $2t$ first degree polynomials, i.e.

$$g(x) = (x - \alpha^b)(x - \alpha^{b+1}) \ldots (x - \alpha^{b+2t-1})$$

Reed Solomon Codes

- The minimum distance of an (n, k) RS code is $d_{min} = n-k+1$.
  - Achieves the Singleton bound, and thus they are maximum distance separable codes.

Examples of RS codes

- The $t = 2$ error correcting RS code of length 7.
- The $t = 3$ error correcting RS code of length 7.
- The $t = 3$ error correcting RS code of length 63.

$GF(8)$, $g(x) = (x-\alpha)(x-\alpha^2)(x-\alpha^3)(x-\alpha^4) = x^4$

Decoding Outline

- Compute the syndrome
- Determine the error locator polynomial.
- Find the roots of the error locator polynomial.
- Determine the error values. (for non-binary case only)

Decoding

- Note that $g(x)$ is selected to have its zeros the $2t$ consecutive powers of α such that
  $g(\alpha) = g(\alpha^2) = \ldots = g(\alpha^{2t}) = 0$.
- Thus, $c(x) = m(x)g(x) = c_0 + c_1 x + \ldots + c_{n-1} x^{n-1}$ must have the $2t$ consecutive powers of α as zeros.
- $r(x) = c(x) + e(x)$, $e(x) = e_0 + e_1 x + \ldots + e_{n-1} x^{n-1}$.

Syndrome, Error Values, and Error Locators

- The syndrome can be evaluated at each and every $2t$ zero:
  $s(x) = r(x=\alpha^j)$
  where $j = 1, 2, \ldots, 2t$.

- Let's call $S_j = r(x=\alpha^j) = e(\alpha^j) = \sum_{k=0}^{n-1} e_k (\alpha^j)^k$. Note that this can be evaluated using the receive polynomial $r(x)$.

- Assume P errors happened in coordinates $i_1, i_2, \ldots, i_P$, then
  $S_j = \sum_{p=1}^{P} e_{i_p} (\alpha^j)^{i_p}$
  $=$(binary case only) $\sum_{p=1}^{P} (\alpha^j)^{i_p} = \sum_{p=1}^{P} X_p^j$,
  for $j = 1, 2, \ldots, 2t$.

- Note $X_p := (\alpha)^{i_p}$ indicates the coordinate i_p of the p-th error.
  Ex) $e = (0\ 0\ e_{i_1}\ 0\ e_{i_2}\ 0\ 0\ 0) \Rightarrow i_1 = 2, i_2 = 4$.

0 1 2 3 4

Syndrome Equation wrt P Error Locators X_p^j

- The $2t$ syndrome equations are

$$\left.\begin{aligned} S_1 &= e_{i_1}X_1 + e_{i_2}X_2 + \ldots + e_{i_P}X_P \\ S_2 &= e_{i_1}X_1^2 + e_{i_2}X_2^2 + \ldots + e_{i_P}X_P^2 \\ &\vdots \\ S_{2t} &= e_{i_1}X_1^{2t} + e_{i_2}X_2^{2t} + \ldots + e_{i_P}X_P^{2t} \end{aligned}\right\} \quad (1)$$

Binary case
P unknowns
2t equations.
where $P \le t$.

- How do we find the error locations $\{X_p, p=1, \ldots, P\}$?

non Linear equations
⇒ Linear eq.

Error Locator Polynomial Λ(x)

- Consider the error locator polynomial $\Lambda(x)$ that has the P $1/X_p$ as its roots, i.e.

$$\begin{aligned} \Lambda(x) &= \prod_{p=1}^{P} (1 - X_p x) \\ &= \Lambda_P x^P + \Lambda_{P-1} x^{P-1} + \ldots + \Lambda_1 x + \Lambda_0. \end{aligned}$$

- We can express the coefficients Λ_p wrt X_p, i.e.

$$\left.\begin{aligned} \Lambda_0 &= 1 \\ \Lambda_1 &= X_1 + X_2 + \ldots + X_P \\ \Lambda_2 &= X_1X_2 + X_1X_3 + \ldots \\ &\ldots \\ \Lambda_P &= X_1X_2 \ldots X_P \end{aligned}\right\} \quad (2)$$

Newton's Identities (Binary case)

- Using (2), (1) can be rewritten as

$$S_1 + \Lambda_1 = 0$$
$$S_2 + \Lambda_1 S_1 + 2\Lambda_2 = 0$$
$$S_3 + \Lambda_1 S_2 + \Lambda_2 S_1 + 3\Lambda_3 = 0$$
$$S_4 + \Lambda_1 S_3 + \Lambda_2 S_2 + \Lambda_3 S_1 + 4\Lambda_4 = 0$$
$$S_5 + \Lambda_1 S_4 + \Lambda_2 S_3 + \Lambda_3 S_2 + \Lambda_4 S_1 = 0$$
$$\cdots$$
$$S_{2t} + \Lambda_1 S_{2t-1} + \Lambda_2 S_{2t-2} + \Lambda_4 S_{2t-3} + \Lambda_4 S_{2t-4} = 0.$$

$P = 4$ Example

$t \geq P$

- Note that there are $2t$ equations and up to four (P=4) unknowns.

Newton's Identities (Binary case)

- Assuming binary case and $P = t$ errors have occurred, the syndromes and the coefficients of the error locator polynomials are related by the following:

$$S_1 + \Lambda_1 = 0$$
$$S_3 + \Lambda_1 S_2 + \Lambda_2 S_1 + \Lambda_3 = 0$$
$$S_5 + \Lambda_1 S_4 + \Lambda_2 S_3 + \Lambda_3 S_2 + \Lambda_4 S_1 + \Lambda_5 = 0$$
$$\cdots$$
$$S_{2t-1} + \Lambda_1 S_{2t-2} + \ldots + \Lambda_t S_{t-1} = 0.$$

- In binary case, $S_{2j} = S_j^2$.

$t=5$ $\quad (t=5)\times(2t-1) = 5\times 9$

$$5 \quad \begin{bmatrix} 1 & & & & \\ S_2 & S_1 & 1 & & \\ S_4 & S_3 & S_2 & S_1 & 1 \\ S_6 & S_5 & S_4 & S_3 & S_2 \\ S_8 & S_7 & S_6 & S_5 & S_4 \end{bmatrix} \begin{pmatrix} \Lambda_1 \\ \Lambda_2 \\ \Lambda_3 \\ \Lambda_4 \\ \Lambda_5 \end{pmatrix} = \begin{bmatrix} -S_1 \\ -S_3 \\ -S_5 \\ -S_7 \\ -S_9 \end{bmatrix}$$

Where is S_{10}? Why it is not used?

Peterson's Direct Solution Decoding Algorithm (Binary)

- $\mathbf{A}\Lambda =$

$$\underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 & & 0 & 0 \\ S_2 & S_1 & 1 & 0 & \cdots & 0 & 0 \\ S_4 & S_3 & S_2 & S_1 & 1\ \cdot & 0 & 0 \\ \cdots & & & & & & \\ S_{2t-2} & S_{2t-3} & S_{2t-4} & S_{2t-5} & \cdots & S_t & S_{t-1} \end{pmatrix}}_{t \times t \text{ matrix}} \begin{pmatrix} \Lambda_1 \\ \Lambda_2 \\ \Lambda_3 \\ \vdots \\ \Lambda_t \end{pmatrix} = \begin{pmatrix} -S_1 \\ -S_3 \\ -S_5 \\ \vdots \\ -S_{2t-1} \end{pmatrix}$$

- **A** is non-singular (thus the equation has a unique solution) if there are t or t -1 errors in the rec. word. [Peterson]
- If fewer than $t-1$ errors have occurred, **A** is singular.
- Then, eliminate the two bottom rows and the two rightmost columns of **A** and see if the remaining matrix is non-singular.
- Proceed with decoding if non-singular; otherwise eliminate more.
- And so on for even fewer errors.

Chien Search

- Once the error locator polynomial is found, use the Chien Search (a systematic exhaustive search over all elements in GF(2^m)) to find the roots.
- $\Lambda(x) = \prod_{p=1}^{P} (1 - X_p x) = \overbrace{\Lambda_P x^P + \Lambda_{P-1} x^{P-1} + \ldots + \Lambda_1 x}^{A_i(x=\alpha^i)} + 1.$
- Take a primitive element α and find all roots x s.t. $\Lambda(x) = 0$.
- *By definition,* $X_p^{-1} := (\alpha)^{-j_p}$. *Thus, if the CS gives* $X_p^{-1} := \alpha^i$, *then* $X_p = \alpha^{-i}$.

$A_1 = \alpha\Lambda_1 + \alpha^2\Lambda_2 + \cdots + \alpha^P\Lambda_P$

$i = \cancel{0}, 1, 2, \ldots, 2^m - 1.$

$A_2 = \alpha^2\Lambda_1 + (\alpha^2)^2\Lambda_2 + \cdots + (\alpha^2)^P\Lambda_P$

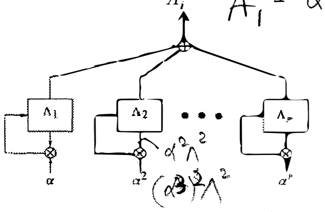


Figure 6.1: Chien search algorithm.

α

Peterson's Direct Solution Decoding for Binary t-Error Correcting BCH Codes

1. Compute the syndromes for $r(x)$: $\{S_j\} = \{r(\alpha^j)\}, j = 1, 2, \ldots, 2t$.
2. Construct the syndrome matrix **A**.
3. Compute the determinant of **A**. If non-zero, go to step 5.
4. If zero, reconstruct a smaller matrix **A** by deleting the two last columns of old **A**. Go to step 3.
5. Solve for Λ and construct ELP $\Lambda(x)$.
6. Find the roots of $\Lambda(x)$.
   - If the roots are not distinct or no roots, then declare decoding failure. Else, go to step 7.
- Complement the bit positions in $r(x)$ indicated by the ELP $\Lambda(x)$. Verify if the resulting corrected word satisfies all $2t$ syndrome equations.
  - If not, declare decoding failure.

Some Simple Cases

- Single Error Correcting
  - $\Lambda_1 = S_1$
- Double Error Correcting
  - $\Lambda_1 = S_1$
  - $\Lambda_2 = (S_3 + S_1^3)/S_1$
- Triple Error Correcting
  - $\Lambda_1 = S_1$
  - $\Lambda_2 = (S_1^2 S_3 + S_5)/(S_1^3 + S_3)$
  - $\Lambda_3 = (S_1^3 + S_3) + S_1\Lambda_2$

Binary Decoding Examples

- Double error correction using the Peterson's algorithm with a code capable of correcting up to $t = 2$ errors.
- Single error correction using Peterson's algorithm with a code capable of correcting up to $t = 3$ errors.

Correction needed!

Observation from the two previous examples

- The first example shows that the algorithm must check the singularity of the largest t x t matrix.
  - ~~For $P = 1$, the 1 x 1 matrix, [1], is non singular.~~
  - But for $P = t = 2$, the 2 x 2 matrix **A** is also ~~non~~ singular.

없거나, 14쪽의 개념 이용!

- The second example shows that the 3 x 3 matrix is singular; finds 1 x 1 matrix non-singular.
- Thus, we must check singularity of the largest t x t matrix **A** anyhow.

Decoding of Non-Binary BCH codes

- Not only the *error locations* but also the *error values* need to be determined.
- Using $2t$ roots, we get up to $2t$ equations, which contain up to t location-unknowns and up to t error-value unknowns.

1. Perterson-Gorenstein-Zierler algorithm
2. Berlekamp-Massey algorithm
   - LFSR based
3. Euclidean algorithm

The PGZ Decoding Algorithm

- Take the error locator polynomial again

$$\Lambda(x) = \prod_{p=1}^{P} (1 - X_p x)$$
$$= \Lambda_P x^P + \Lambda_{P-1} x^{P-1} + \ldots + \Lambda_1 x + 1.$$

- At $x = X_p^{-1}$,

$$\Lambda(X_p^{-1}) = \boxed{\Lambda_P X_p^{-P} + \Lambda_{P-1} X_p^{-P+1} + \ldots + \Lambda_1 X_p^{-1} + 1 = 0.}$$

- Multiply $e_{i_p} X_p^j$ to both sides of the box:

$$\Lambda_P e_{i_p} X_p^{j-P} + \Lambda_{P-1} e_{i_p} X_p^{j-P+1} + \ldots + \Lambda_1 e_{i_p} X_p^{j-1} + e_{i_p} X_p^j = 0.$$

- Repeat for each p and take the sum over all p, using (1)

$$\boxed{\Lambda_P S_{j-P} + \Lambda_{P-1} S_{j-P+1} + \ldots + \Lambda_1 S_{j-1} + S_j = 0} \qquad (3)$$

for $j = P+1, \ldots, 2P$.

The PGZ Decoding Algorithm

- $\mathbf{A'\Lambda} =$

$$\begin{array}{c} t \times t \text{ matrix} \\ \begin{pmatrix} S_1 & S_2 & S_3 & S_4 & \cdots & S_{t-1} & S_t \\ S_2 & S_3 & S_4 & S_5 & \cdots & S_t & S_{t-1} \\ S_3 & S_4 & S_5 & S_6 & \cdots & S_{t+1} & S_t \\ \cdots & & & & \cdots & \cdots & \\ S_t & S_{t+1} & S_{t+2} & S_{t+3} & \cdots & S_{2t-2} & S_{2t-1} \end{pmatrix} \end{array} \begin{pmatrix} \Lambda_t \\ \Lambda_{t-1} \\ \Lambda_{t-2} \\ \cdot \\ \cdot \\ \cdot \\ \Lambda_1 \end{pmatrix} = \begin{pmatrix} -S_{t+1} \\ -S_{t+2} \\ -S_{t+3} \\ \cdot \\ \cdot \\ \cdot \\ -S_{2t} \end{pmatrix}$$

- **A'** is non-singular (thus the equation has a unique solution) if there are t errors in the rec. word. If fewer than t errors occurred, **A'** is singular. [Gor61],[Blahut84]
- Then, eliminate the bottom row and the rightmost column of **A'** and see if the remaining matrix is non-singular (see if $|\mathbf{A'}| \neq 0$).
- Proceed with decoding if non-singular; otherwise eliminate more.
- And so on for even fewer errors.

Error Value Computation

- Once the P error locations are known, the first P syndrome equations can be used to find the error values.
- Note that it is a Vandermonde matrix with P non-zero distinct values.
  - Thus, non singular!

$$\mathbf{Be} = \begin{pmatrix} X_1 & X_2 & \cdots & X_P \\ X_1^2 & X_2^2 & \cdots & X_P^2 \\ \vdots & \vdots & \ddots & \vdots \\ X_1^P & X_2^P & \cdots & X_P^P \end{pmatrix} \begin{pmatrix} e_{i_1} \\ e_{i_2} \\ \vdots \\ e_{i_P} \end{pmatrix} = \begin{pmatrix} S_1 \\ S_2 \\ \vdots \\ S_P \end{pmatrix}$$

The PGZ Decoding Algorithm

[Wicker, pg. 216]

1. Compute the syndromes.
2. Construct the syndrome matrix **A'**.
3. Compute the determinant. If it is non-zero, go to 5.
4. Construct a new syndrome matrix by deleting the rightmost column and the bottom row. Shorten Λ by one coordinate position by deleting Λ_t for the largest remaining t. Go to 3.
5. Solve for Λ and construct $\Lambda(x)$.
6. Find the roots of $\Lambda(x)$. If they are not distinct or $\Lambda(x)$ does not have roots in the desired field, go to 10.
7. Construct the matrix **B** and solve for the error values.
8. Subtract the error values from the values at the appropriate coordinates of the received word.
9. Output the correct word and STOP.
10. Declare a decoding failure and STOP.

Decoding Examples

- Double error correction using (7, 3) RS codes capable of correcting $t = 2$ errors.

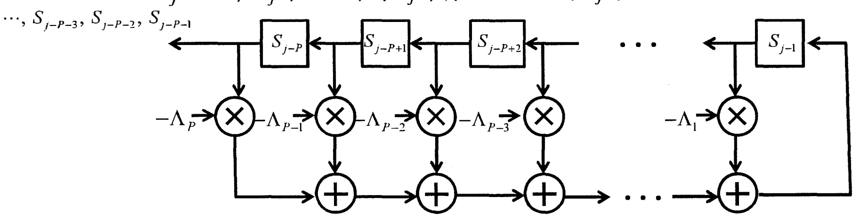
The Berlekamp-Massey Algorithm

- Computationally efficient than the Peterson algorithm.
- It uses eq. (3) where P the number of errors is unknown.
  - It builds the order P error locator polynomial starting from scratch, say order $L = 0$.
  - See if the current polynomial can generate the observed syndrome sequence, starting from the first syndrome, say $k = 1$.
  - When the polynomial is not the correct one, discrepancy between the output of the equation and the observed syndrome will occur.
  - Use the discrepancy to update the polynomial, and continue until the updated polynomial is capable of generating all $2t$ element of the syndrome sequence, $k = 2t$.

The Berlekamp-Massey Algorithm

- Five basic parameters:
  - the indexing variable k
  - the connection polynomial $\Lambda^{(k)}(x) = \Lambda_k x^k + \Lambda_{k-1} x^{k-1} + \ldots + \Lambda_1 x + 1$,
  - the correction polynomial $T(x)$,
  - the discrepancy $\Delta^{(k)}$, and
  - the length L of the shift register.
- The syndrom S_j can be expressed in a recursive manner (Recall (3)):

$$-S_j = \Lambda_P S_{j-P} + \Lambda_{P-1} S_{j-P+1} + \cdots + \Lambda_1 S_{j-1}$$



LFSR Interpretation for the Syndrome

The Berlekamp-Massey Algorithm

1. Compute all $2t$ syndromes for an $r(x)$.
2. Initialize: k=0, $\Lambda^{(k)}(x)=1$, L=0, and $T(x)=x$.
3. Set $k = k + 1$. Compute the discrepancy $\Delta^{(k)}$: $\Delta^{(k)} = S_k - \sum_{i=1}^{L} \Lambda_i^{(k-1)} S_{k-i}$
4. If $\Delta^{(k)}= 0$, go to 8.
5. Modify the connection polynomial: $\Lambda^{(k)}(x) = \Lambda^{(k-1)}(x) - \Delta^{(k)} T(x)$
6. If $2L \geq k$, go to 8.
7. Set $L = k - L$ and $T(x) = \Lambda^{(k-1)}(x) / \Delta^{(k)}$
8. Set $T(x) = x\,T(x)$.
9. If $k< 2t$, go to 3.
10. Determine the roots of $\Lambda(x)=\Lambda^{(2t)}(x)$. If they are distinct and lie in the right field, then determine the error values, correct the corresponding locations in the received word, and STOP.
11. Declare a decoding failure and STOP.

Key Equation for BCH/RS Decoding

$$\Lambda(x)S(x) \equiv \Omega(x) \bmod x^{2t}$$

- We define
  - Syndrome polynomial $S(x) := \sum_{j=0}^{2t-1} S_{j+1} x^j$
  - $S(x) = S_1 + S_2x + \ldots + S_{2t}x^{2t-1} + \ldots$
  - Error value polynomial Ω(x)

$$\begin{aligned}\Omega(x) &:= S(x)\Lambda(x)\\ &= (S_1 + S_2 x + \cdots)(1 + \Lambda_1 x + \Lambda_2 x + \cdots)\\ &=: \Omega_0 + \Omega_1 x + \Omega_2 x^2 + \cdots\end{aligned}$$

Key Equation (2)

- Note that EVP contains the error values e_{i_p}:

$$\Omega(x) := S(x)\Lambda(x) = \left(\sum_{j=0}^{2t-1} S_{j+1}x^j\right)\Lambda(x) \mod x^{2t}$$

$$= \left[\sum_{j=0}^{2t-1}\left(\sum_{p=1}^{P} e_{i_p} X_p^{j+1}\right)x^j\right]\Lambda(x) \mod x^{2t}$$

$$= \left[\sum_{p=1}^{P} e_{i_p} X_p \sum_{j=0}^{2t-1}\left(xX_p\right)^j\right]\Lambda(x) \mod x^{2t}$$

$$= \left[\sum_{p=1}^{P} e_{i_p} X_p \left(\frac{1-\left(xX_p\right)^{2t}}{1-X_p x}\right)\right]\Lambda(x) \mod x^{2t}$$

$$= \sum_{p=1}^{P} e_{i_p} X_p \prod_{l\neq p}(1-X_l x)$$

Use $\Lambda(x) = \prod_{l=1}^{P}(1-X_l x)$

Definition 6.5 Let $f(x) = f_0 + f_1 x + f_2 x^2 + \cdots + f_t x^t$ be a polynomial with coefficients in some field $\mathbb{F}$. The **formal derivative** $f'(x)$ of $f(x)$ is computed using the conventional rules of polynomial differentiation:

$$f'(x) = f_1 + 2f_2 x + 3f_3 x^2 + \cdots + t f_t x^{t-1}, \tag{6.33}$$

where, as usual, mf_i for $m \in \mathbb{Z}$ and $f_i \in \mathbb{F}$ denotes repeated addition:

$$mf_i = \underbrace{f_i + f_i + \cdots + f_i}_{m \text{ summands}}.$$

Theorem 6.16 *(Forney's algorithm) The error values for a Reed-Solomon code are computed by*

$$e_{i_k} = -\frac{\Omega(X_k^{-1})}{\Lambda'(X_k^{-1})}, \tag{6.34}$$

where $\Lambda'(x)$ is the formal derivative of $\Lambda(x)$.

Still needed even for Extend EA

Some Decoding Examples

- Double error correction using (7, 3) RS codes capable of correcting t = 2 errors. (Again with the Berlekamp-Massey algorithm)

The Extended Euclid Algorithm for $\Omega(x)$

∃ a, b such that
Gcd(a, b) = ax + by

- The key equation

$$\Lambda(x)S(x) \equiv \Omega(x) \bmod x^{2t}$$

is equivalent to

$$\Omega(x) = Q(x)x^{2t} + \Lambda(x)S(x)$$

r

for some $Q(x)$.

- Thus, given $\Lambda(x)$ and $S(x)$, we may use the extended EA to find $\Omega(x)$ as well.

Summary

- BCH and Reed Solomon codes have been used in many applications.
  - Cyclic codes, LFSR implementation.
- Reed Solomon codes achieve the Singleton bound.
- In the light of new decoding methods [Sudan, Guruswami, Koetter, etc], Reed Solomon codes are again in the research spot light.

Term Project Idea 1

- **Objective**: Survey the literature with the objective of finding out the state-of-the-arts in Reed-Solomon codes:
  - Read IEEE Transactions on Information Theory papers
  - New encoders and decoders
    - Find and compare the maximum design distance $D = 2t$, the maximum block length n, the rate region k/n, etc. , each codec can practically achieve today.
    - What are the required complexity for each one (how fast the algorithms are?)
    - What are the target application areas for each?
    - Come up with other relevant questions and answers.

Term Project Idea 2

- **Objective**: Show the possibility of using RS codes in a Compressive Sensing system.

- Use an (n>128, k, D>6) Reed Solomon codec in MATLAB. (or whatever programs available in the Internet, MATLAB, or the textbook)
  - Push n and D as large as possible.
  - Note we need a low rate k/n codes (Compression)
- Use it as a Compressive Sensing system
  - Obtain a reasonable size picture, say a GIST logo, from the Internet.
  - Compressively sense the picture using the RS code.
- Compare its performance with the standard CS approaches
  - Use a known CS approach (visit the L_1 magic web-site or the RICE University web-site and download the relevant MATLAB program package)

HW#4

- Moon: P6.1, P6.2, P6.6, P6.11, P6.12, P6.14

References

- [Gor61] D. Gorenstein and N. Zierler, "A class of error correcting codes in pm symbols," Journal of the Society of Industrial and Applied Mathematics, Vol.9, pp. 207-214, June 1961.
- [Blahut84] R.E. Blahut, Theory and Practice of Error Control Codes, Reading, MA: Addison Wesley, 1984.
- [Royden] Royden, *Real Analysis*, 3rd Edition, MacMillan, 1988.
- [Wicker] S.B. Wicker, Error Control Systems for Digital Communications and Storage. Englewood Cliffs, NJ: Prentice Hall, 1995.
- [Moon] T.K. Moon, *Error Correction Coding:mathematical methods and algorithms*. Wiley Interscience, 2005.

Convolutional Codes

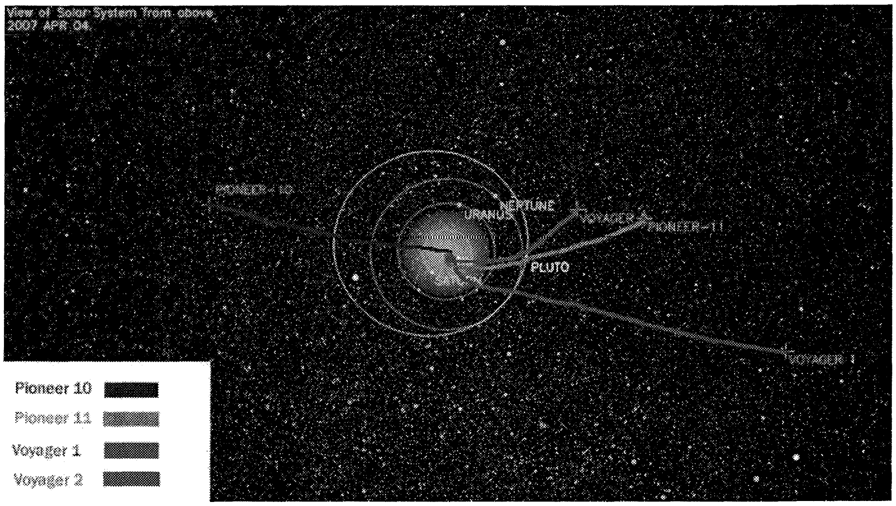
Ref: Moon Ch. 12, Wicker Ch. 11, 12

Agenda

- Brief History of Convolutional Codes
  - Peter Elias [1955], introduction of C codes, instead of block codes, and list decoding, instead of unique decoding.
  - Sequential decoding algorithms in the 60s [Wozencraft, Massey's majority logic, Fano, …]
  - Forney's dissertation, RS code + C code [1965], later adapted in Voyager 1[1977]
- Convolutional Code, What is it?
  - Feedforward form
  - Feedback form
- Maximum Likelihood Sequence Detection
  - Tree Search, Trellis Search (Viterbi Decoding)
- State Diagram, Distance Spectrum, Performance Evaluation
- HW#5

Voyager 1's current location [2007]

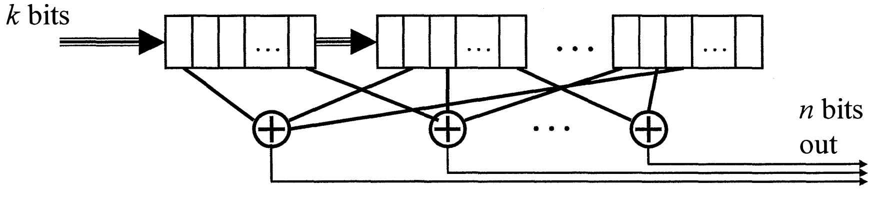
[Wiki2010]



Memoir of Gallager on Peter Elias

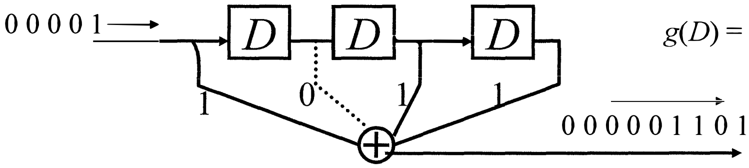
Convolutional Codes

- Code words are generated by $c(D) = m(D)g(D)$
  - D is one unit delay of a shift register circuit
  - $g(D)$ is realized with a linear finite-state shift register
  - The degree of $m(D)$ can be infinite. So can be that of $c(D)$.
- Rate k/n code, (k, n, K) *CC*
  - k information bits get shifted in at each D,
  - goes thru K units of delay, (K is called the *constraint length*)
  - generates n coded bits output at each delay



Convolution (FIR)

$m(D) = 1 \quad K = 3$

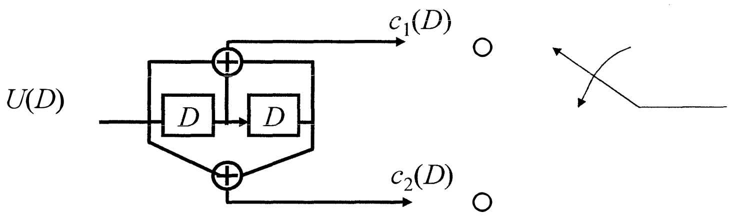


$g(D) = 1+D^2+D^3$

$c(D) = g(D)$

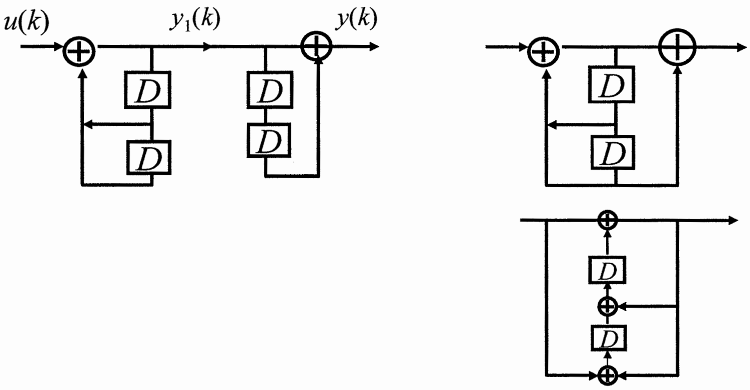
- From systems class, we know the convolution is
  - Shifting the input sequence
  - Multiplying the shifted input term by term with the filter coefficients
  - Add up the product terms
  - Shift the input sequence by one and repeat
- Here the input and the filter coefficients are binary and the summation is module 2 addition

Feedforward Rate 1/2 convolutional encoder



- $g_1(D)-1+D+D^2$, and $g_2(D)=1+D^2$
- $G(D) = [g_1(D)\ g_2(D)]$
- Consider input $U(D) = 1+D^2$
- $C(D) = [g_1(D)U(D) \quad g_2(D)U(D)] = [1+D+D^3+D^4 \quad 1+D^4]$
- C=[11 , 10 , 00 , 10, 11]

Feedback Circuit (IIR)

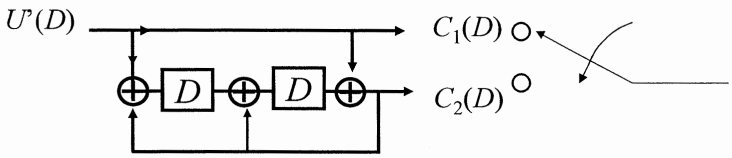


Direct Form II

Transposed Form

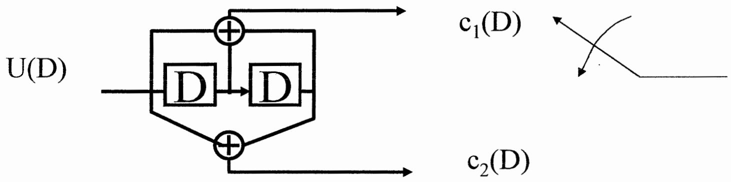
- $y_1(k)=y_1(k\text{-}1)+y_1(k\text{-}2)+u(k)$ $\Rightarrow$ $Y_1(D)=U(D)/(1\text{-}D\text{-}D^2)$
- $y(k) = y_1(k) + y_1(k\text{-}2)$ $\Rightarrow$ $Y(D)=Y_1(D)(1+D^2)$
- $Y(D)=U(D)(1+D^2)/(1\text{-}D\text{-}D^2)$

Recursive Rate 1/2 Convolutional Encoder

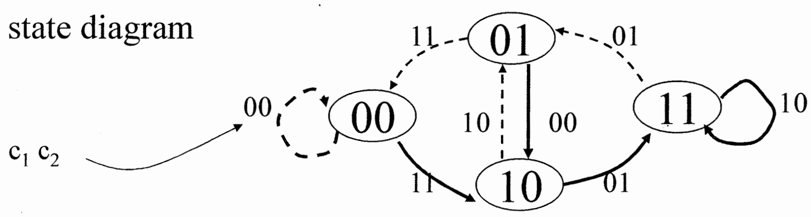


- $G'(D) = [1\ (1+D^2)/(1+D+D^2)]$
- Consider input $U'(D)=(1+D^2)(1+D+D^2)=1+D+D^3+D^4$
- $C_1(D) = U'(D)$, $C_2(D) = (1+D^2)^2 = 1+D^4$
- C=[1 1, 1 0, 0 0, 1 0, 1 1]
- For $U'(D) = U(D)(1+D+D^2)$,
  $U'(D)G'(D)=U(D)G(D)$
- $G'(D)$ and $G(D)$ are equivalent; they generate the same codewords.

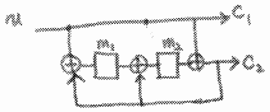
State Diagram of Rate ½ Code



- State can be defined as an ordered set (Memory1, Memory2) = 00, 01,10,11
- The state diagram

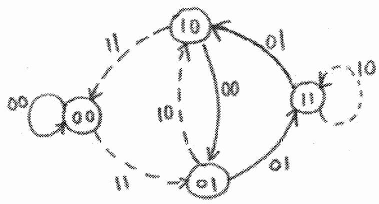


State Diagram for Recursive Form

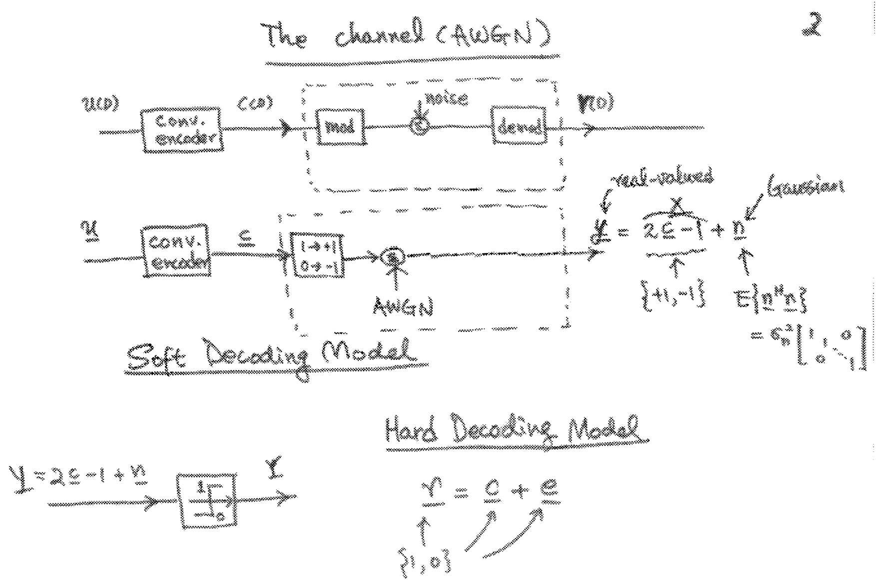


—— 0 input
- - - - 1 input

Current state m_1	m_2	u	output $c_1 = u$	$c_2 = u \oplus m_2$	Next state $\hat{m}_1 = u \oplus c_2$	$\hat{m}_2 = m_1 \oplus c_2$
0	0	0	0	0	0	0
0	0	1	1	1	0	1
0	1	0	0	1	1	1
0	1	1	1	0	1	0
1	0	0	0	0	0	1
1	0	1	1	1	0	0
1	1	0	0	1	1	0
1	1	1	1	0	1	1



Comparing to the feedforward form, we note that the outputs are the exactly the same.



Soft Decoding

- $\mathbf{y} = \mathbf{x} + \mathbf{n}$, $\mathbf{x} = 2\mathbf{c} - 1$, and let N = length of binary vector $\mathbf{u}$.
- One-to-one: $\mathbf{u}$, $\mathbf{c}$, and $\mathbf{x}$
- Consider likelihood function $p(\mathbf{y}|\mathbf{x} = \mathbf{x}_i)$, $i = 0, 1, \ldots, 2^N - 1$

$$\begin{aligned} p(\mathrm{y}|\mathrm{x} = \mathrm{x}_i) &= p(\mathrm{n} = \mathrm{y} - \mathrm{x}_i) \\ &= \frac{1}{(2\pi)^{N/2}|R_n|^{1/2}} exp[-\frac{1}{2}(\mathrm{y} - \mathrm{x}_i)^H R_n^{-1}(\mathrm{y} - \mathrm{x}_i)] \end{aligned}$$

- Assume white noise such that $R_n = \sigma_n^2 I_n$,
  where $\sigma_n^2 = N_0/(2E_s)$

Soft Decoding (2)

- $$p(\mathbf{y}|\mathbf{x}=\mathbf{x}_i) = C \cdot exp[-\frac{1}{2\sigma_n^2}(\mathbf{y}-\mathbf{x}_i)^H(\mathbf{y}-\mathbf{x}_i)]$$
  $$= C \cdot exp[-\frac{1}{2\sigma_n^2}\sum_{j=0}^{N-1}|y_j - x_{ij}|^2]$$

Common constant for all hypothesis $\mathbf{x}_i$

This term is what matters

- $log\, p(\mathbf{y}|\mathbf{x}_i) = -C_1 \cdot \sum_{j=0}^{N-1}|y_j - x_{ij}|^2$

  where C_1 is positive

Soft Decoding (3)

- Thus, we have

$$\begin{aligned}\hat{\mathbf{u}} &= arg\, max_{\mathbf{u}_i \in U}\, p(\mathbf{y}|\mathbf{u}_i) \\ &= arg\, max_{\mathbf{x}_i \in X}\, p(\mathbf{y}|\mathbf{x}_i) \\ &= arg\, max_{\mathbf{x}_i \in X}\, log\, p(\mathbf{y}|\mathbf{x}_i) \\ &= arg\, min_{\mathbf{x}_i \in X} \sum_{j=0}^{N-1}|y_j - x_{ij}|^2\end{aligned}$$

Euclidean distance

Hard Decision Decoding

- When we only have hard decision **r** available, we can obtain

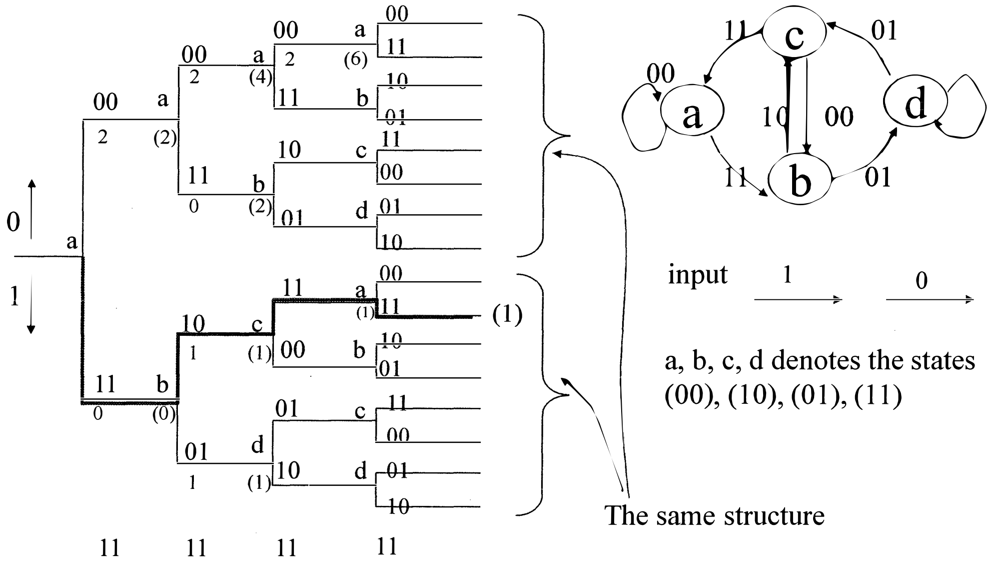
$$\begin{aligned}\hat{\mathbf{u}} &= arg\; max_{\mathbf{u}_i \in U}\; p(\mathbf{r}|\mathbf{u}_i) \\ &= arg\; max_{\mathbf{c}_i \in C}\; p(\mathbf{r}|\mathbf{c}_i) \\ &= arg\; max_{\mathbf{c}_i \in C}\; d_H(\mathbf{r}, \mathbf{c}) \\ &= arg\; min_{\mathbf{c}_i \in C} \sum_{j=0}^{N-1} w_H(r_j - c_{ij})\end{aligned}$$

Hamming Distance

Tree Decoding

1. The node metric at the starting node *a* is zero.
2. Expand the tree, resulting in two branches for one node.
3. At each branch, calculate the distance (Hamming or Euclidean) with the corresponding received symbol. This is called the *branch metric*.
4. Add the branch metric to the node-metric. The result forms so-called the *cumulative metric*. Record the cumulative metric at the node and use it for its offspring paths, in the subsequent tree expansion.

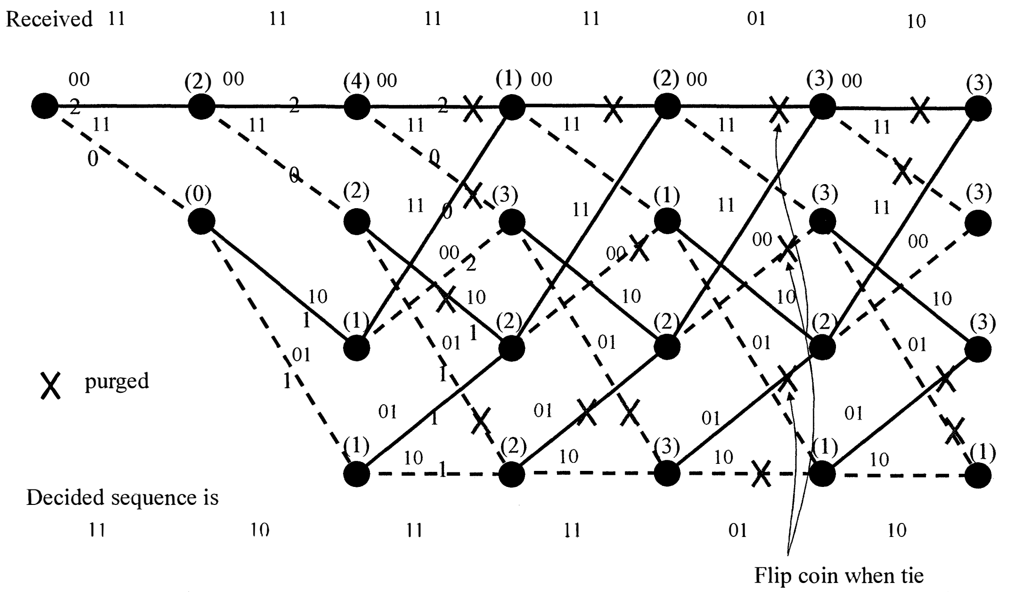
Tree Decoding



Tree Decoding

- Observation: The same tree structure repeats after 3rd expansion
- Consider the cumulative metrics of the two merged paths at node-a, such that a-a-a-a and a-b-c-a. Let's call them σ(1) and σ(2)
- Let's assume another tree search decoder spawning out of node-a starting from the 4-th expansion, and an optimal path a-x-y-z-… with minimal metric path of length (*N*-3) can be found.
- Thus, the overall minimum of the two metrics associated with the two paths, a-a-a-a-x-y-z… and a-b-c-a-x-y-z…, can be determined by σ(1) and σ(2).
- This indicates that when some paths merge to a same state, an *early pruning decision* can be made without loss of optimality.
- After pruning, the decoding on a tree can be done on a *collapsed* tree, a trellis.

Trellis Decoding



Viterbi Algorithm

- The algorithm shown is the Viterbi algorithm.
- Note that the number of states is 2^{N_m}.
- The number of branches is $2^{N_m + 1}$.
- Let $\sigma(j, k)$ be the partial cumulative metric at state j at the k-th trellis section.

1. Set $\sigma(j\text{=a},0)=0$ (Starting at the state a).
2. At time k, compute the partial cumulative metrics for all paths merging to each state.
3. Set $\sigma(j, k)$ equal to the best partial path metric entering the node corresponding to state-j at time t. Break a tie with coin-flipping. Mark the best metric path.
4. At the end of sequence, trace back the marked path for decoding.

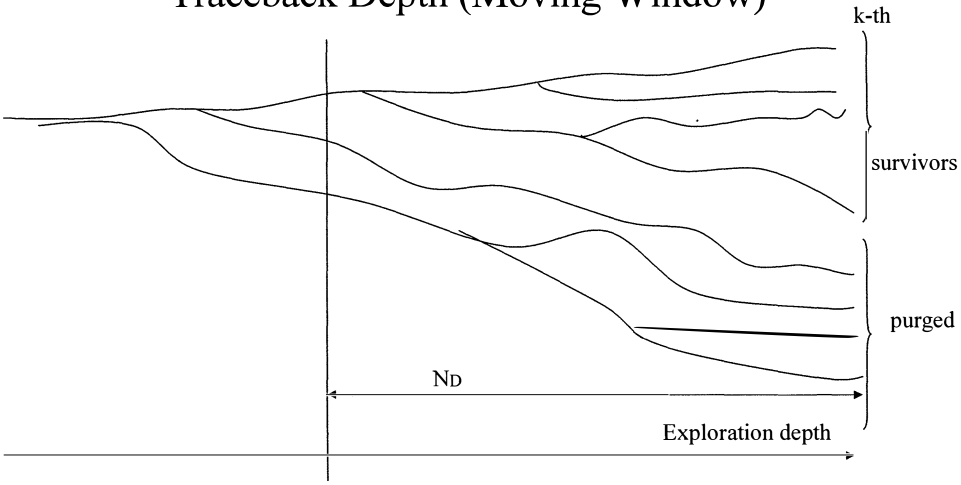
Travelling Salesman Problem

- Given a list of cities and their pairwise distances, the problem is to determine the shortest distance trip route in which each city is visited. (1930)
- An NP-complete problem.
  - Currently many heuristic algorithms are out there, and problems with 10 thousands cities can be solved.

Continuous Viterbi Algorithm

- Making decision at the end of the sequence is optimal in the sense of MLSE:
  - Delay
  - Memory requirement to store the survivor path metrics
- Early decision reduces delay and memory requirement.
- In practice, decisions are made earlier than the end of the sequence.
- The delay required to make this early decision is called the traceback depth N_D.
- If we make $N_D = N$, the length of the sequence, the decision is optimal.
- If we make N_D too small, the decisions will not be reliable.

Traceback Depth (Moving Window)



At the k-th exploration depth, start to make decision on symbols at the (k - N_D)-th epoch that is stored in the current best survivor path

Traceback Depth

Looking at the contents of the survivors after exploring k time epochs

At k-N_D and further back in history, it is highly likely to see a merge in survivors.

Survivors (Input—(u_k u_{k-1} u_{k-2} …))

1 0 1 1 0 0 0 0 0 0 0 1 0 1 0 1 0 1 0 1 1 1 1 1 0 0 0 0 0 1 1 0 1

1 0 1 1 1 0 0 1 0 0 0 1 0 1 0 0 1 1 0 1 1 1 1 1 0 0 0 0 0 1 1 0 1

0 1 0 1 0 0 0 0 1 0 0 1 0 1 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 1 1 0 1

1 0 1 0 0 0 1 0 0 0 0 1 0 1 0 1 1 1 0 1 1 1 1 1 0 0 0 0 0 1 1 0 1

N_D

Good traceback depth can be determined with Viterbi search of minimum free distance

Catastrophic Convolutional Codes

- With a catastrophic convolutional code, a small error in the received code word can cause an unlimited number of errors.
- When does it happen? How do we know ahead of time?
- State diagram contains a loop with non-zero input which generates all zero outputs.

Examples from Wicker

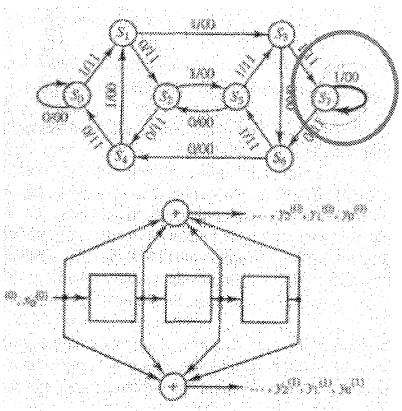


Figure a

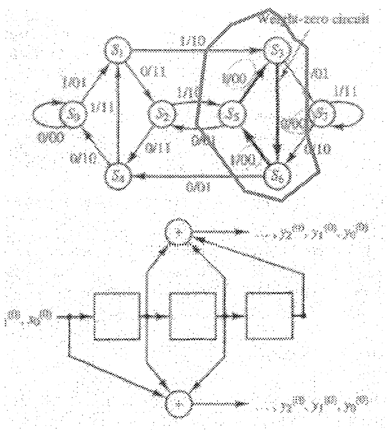


Figure b

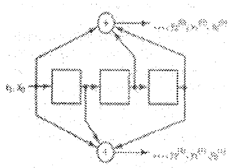
IFF Condition for Non-Catastrophic

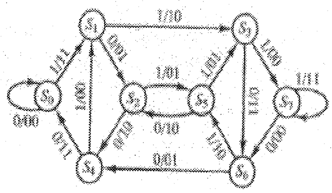
- A rate 1/n code C is not *catastrophic* IFF
  - GCD of all the constituent generator polynomials is equal to D^l for some non negative integer l.
- Example of Figure b,
  - $G^{(0)}=D+D^2+D^3$, $G^{(1)}=1+D+D^2$
  - $GCD(G^{(0)}, G^{(1)}) = 1+D+D^2 \neq D^l$

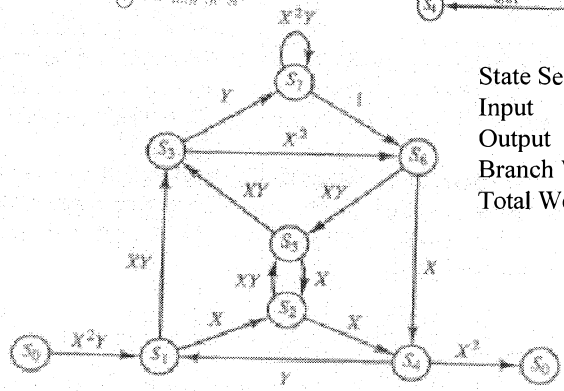
Graphs and Weight Enumerators

- Weight enumerator is the generating function of an encoder graph
- Obtain a state transition graph, starting from the all zero initial state and ending at the all zero final state
- These states are the same states defined in the state diagram
- We are interested in listing out all possible paths such that the state transition can possibly take, departing from the all-zero state and re-merging back to the all-zero state
- The label of branch records the weight of the output and the weight of the input

Example





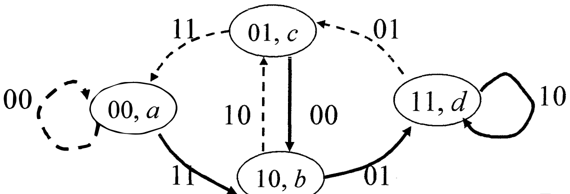


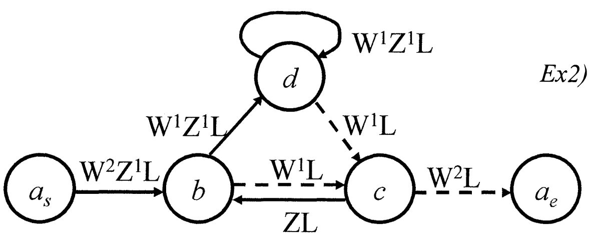
State Seq.	S_0	S_1	S_3	S_6	S_4	S_0
Input	1	1	0	0	0	
Output	11	10	11	01	11	
Branch Weights	X^2Y	XY	X^2	X	X^2	
Total Weights	X^8Y^2					

Some Definitions:
Loop, Forward Loop (starting at S_0)
Nontouching loops (no common state)

Transfer Function of State Diagram

- Consider the feedforward rate ½ code in Lecture-9 again





Ex1) a_s, b, c, a_e: $W^5Z^1L^3$
- *codeword weigths is 5*
- *Input weights is 1*
- *Length of this path is 3*

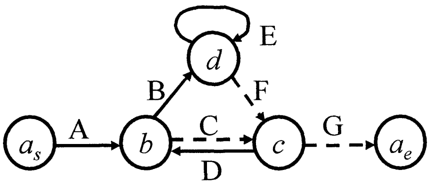
Ex2) a_s, b, d, d, d, c, a_e: $W^8Z^4L^6$

Transfer Function

- Compute weight distributions for all possible state transitions from a_s to a_e
- Define accumulated path label x_i, which is an accumulation of all branch label from the initial state to i, as influenced by all other nodes

Transfer Function from Simultaneous Equations

- Let x_i denote the accumulated path labels of the node-i starting from the initial state as influenced by all others
- $x_b = \mathrm{A} + \mathrm{D}x_c$
  $x_c = \mathrm{C}x_b + \mathrm{F}x_d$
  $x_d = \mathrm{B}x_b + \mathrm{E}x_d$
  $x_{ae} = \mathrm{G}x_c$



$$x_{a_e} = \frac{ACG(1-E)+ABFG}{1-E-D(C-CE+BF)}$$

- We call this the transfer function
  $\mathrm{T(W, Z, L)} := x_{ae}$

$$T(W,Z,L) = \frac{ACG(1-E)+ABFG}{1-E-D(C-CE+BF)}$$

$$= \frac{W^2Z^1L\,W^1L\,W^2L\,(1-WZL)+W^2Z^1LWZLWLW^2L}{1-WZL-ZL(WL-WLWZL+WZLWL)}$$

$$= \frac{W^5Z^1L^3(1-WZL)+W^6Z^2L^4}{1-WZL-WZL^2}$$

Input/Output Weight Enumeration Function (IOWEF)

$$T(W,Z,L) = \frac{W^5 Z^1 L^3 (1 - WZL) + W^6 Z^2 L^4}{1 - WZL - WZL^2}$$

$$\begin{array}{l} \quad -W^5 Z\, L^3 + W^6 Z^2 L^4 + \left(W^7 Z^3 - W^6 Z^2\right) L^5 + \ldots \\ 1 - WZL - WZL^2 \,\big)\; -W^5 Z^1 L^3 + 2W^6 Z^2 L^4 \\ \qquad -W^5 Z^1 L^3 + W^6 Z^2 L^4 + W^6 Z^2 L^5 \\ \qquad W^6 Z^2 L^4 - W^6 Z^2 L^5 \\ \qquad W^6 Z^2 L^4 - W^7 Z^3 L^5 - W^7 Z^3 L^6 \\ \qquad \left(W^7 Z^3 - W^6 Z^2\right) L^5 + W^7 Z^3 L^6 \\ \qquad \ldots \end{array}$$

- We can let L = 1 for IOWEF.

Transition Matrix

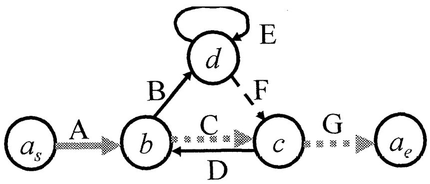
- In matrix notation, we have $\mathbf{x} = \mathbf{T}\mathbf{x} + \mathbf{x_0}$ where $\mathbf{x} := (x_b, x_c, x_d, x_{ae})$, $\mathbf{x_0} = (A, 0, 0, 0)$, and $\mathbf{T} = \begin{pmatrix} 0 & D & 0 & 0 \\ C & 0 & F & 0 \\ B & 0 & E & 0 \\ 0 & G & 0 & 0 \end{pmatrix}$
- $\mathbf{x} = \mathbf{T}\mathbf{x} + \mathbf{x_0} \quad \Rightarrow \quad \mathbf{x} = (1 - \mathbf{T})^{-1}\, \mathbf{x_0}$
- $\mathbf{x} = [\mathbf{I} + \mathbf{T} + \mathbf{T}^2 + \mathbf{T}^3 + \ldots]\, \mathbf{x_0}$
- This tells us about the weights of all the error events
- Get x_e by multiplying with $\mathbf{e}_4 := (0\ 0\ 0\ 1)$
- $x_e = \mathbf{e}_4 \mathbf{x}$

L^3 Error Events

- ACG = $(W^2ZL)(WL)(W^2L) = W^5ZL^3$ is the accumulated path label
- $x_e = \mathbf{e}_4\,\mathbf{x}_0 + \mathbf{e}_4\mathbf{T}\mathbf{x}_0 + \mathbf{e}_4\mathbf{T}^2\mathbf{x}_0 =$

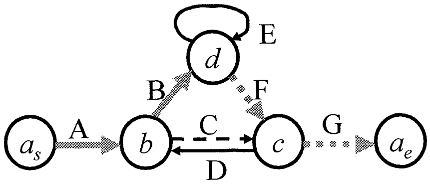
$$= 0 + \mathbf{e}_4 \begin{pmatrix} 0 & D & 0 & 0 \\ C & 0 & F & 0 \\ B & 0 & E & 0 \\ 0 & G & 0 & 0 \end{pmatrix} \begin{pmatrix} A \\ 0 \\ 0 \\ 0 \end{pmatrix} + \mathbf{e}_4 \begin{pmatrix} CD & 0 & FD & 0 \\ FB & CD & FE & 0 \\ EB & BD & EE & 0 \\ GC & 0 & FG & 0 \end{pmatrix} \begin{pmatrix} A \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$$= 0 + 0 + AGC$$



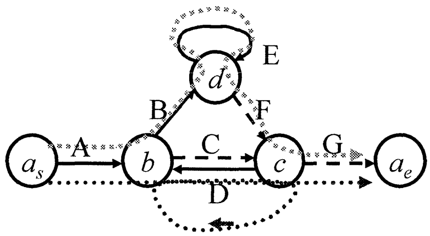
L^4 Error Events

- ABFG = $(W^2ZL)(WZL)(WL)(W^2L) = W^6Z^2L^4$
- $\mathbf{T}^3 = \begin{pmatrix} DFB & CD^2 & DFE & 0 \\ C^2D+FEB & FBD & CFD+FE^2 & 0 \\ BCD+E^2B & EBD & BFD+E^3 & 0 \\ GFB & GCD & GFE & 0 \end{pmatrix}$
- $\mathbf{e}_4\mathbf{T}^3\mathbf{x}_0 = AGFB$



L^5 Error Events

- Two paths: ABEFG +ACDCG = $W^7Z^3L^5 + W^6Z^2L^5$
- $\mathbf{e}_4\mathbf{T}^4\mathbf{x}_0$ = ABEFG + ACDCG



Weights of All Possible Error Events

- Thus, $\mathbf{e}_4\mathbf{x} = \mathbf{e}_4\ [\mathbf{I} + \mathbf{T} + \mathbf{T}^2 + \mathbf{T}^3 + \ldots]\ \mathbf{x}_0$
  $= \mathbf{e}_4\ [\mathbf{T}^2 + \mathbf{T}^3 + \ldots]\ \mathbf{x}_0$
- This describes the weight distribution of the error events, arranged in different lengths.
- The free distance and error rate can be obtained from this.
- The free distance here is obtained from L^3 term.
  - The weight of the codeword is 5, the exponent of W, which corresponds to 1 bit error, the exponent of Z (Why? – the weigths of higher order terms are greater than 5.)

Minimum Free Distance d_{free}

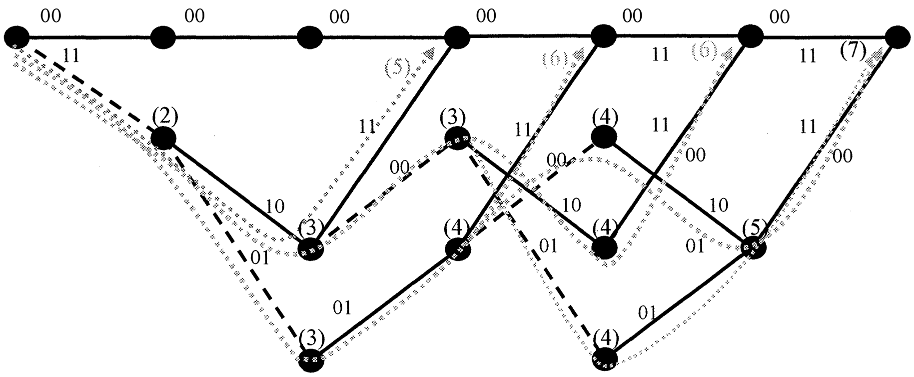
- Most frequent decoding errors are due to wrong decisions made in favor of the nearest neighbors of the transmitted codeword.
  - These are undetected errors because these neighbors are valid codewords.
  - Correct codewords are eliminated.
- The minimum Hamming distance between **all** pairs of convolutional code words

$$d_{free} := \min\{d(\mathbf{c}', \mathbf{c}'') \mid \mathbf{c}' \neq \mathbf{c}''\}$$
$$= \min\{w(\mathbf{c}=\mathbf{c}'-\mathbf{c}'') \mid \mathbf{c} \neq 0\} \qquad \text{linear code}$$

- Thus, we can determine d_{free} by assuming all-zero sequence as the transmitted codeword, and finding out the weights of neighboring allowed sequences
- Thus, increasing d_{free} at a particular constraint length is a desired design criterion (Wicker 11.3: maximal *mimimum free distance* codes).
  - One idea of the trellis code is to use Euclidean distance -- instead of the Hamming distance.

Minimum Free Distance d_{free}

- Assumption: all-zero codeword was transmitted
- Find the distance of an error path from the all-zero word



Determining d_{free} with Viterbi Algorithm

- Consider a single sequence departed from the all-zero path.
- Run the Viterbi algorithm on all the resulting sequences of this single sequence.
- Compute the Hamming distance of the path from the all-zero codeword.
  - The accumulated path metric is the Hamming weight of all the codeword bits along the path.
- After running it for a while, find out the best path among the survivors
  - There is one survivor per state. The survivor at the zero-th state is the minimum metric path. (Why?)
- Not always is a path remerged in the shortest length the d_{free} path.
  - In other words, it is possible to see that the weight of a path re-merged at a length greater than the shortest merged path is smaller than the weight of the shortest path (Will see this in HW).

Traceback Depth

- *Traceback Depth* (TD)?: TD is a number of delay units. After a delay by the amount of TD, the first decoded bit from the Viterbi decoder becomes available.
- How to set the TD right?
  - One approach is explore the trellis starting from the all-zero path. At a certain depth of exploration, all accumulated metrics of survivors will become larger than the free distance. We can set this depth as the traceback depth.
  - Viterbi decoding with a depth set larger than this traceback depth gives small improvement.
  - The other approach is to set the TD to be "7 times the constraint length." This is a heuristic design rule of thumb.

- Will see this in HW.

Determining d_{free} with Transfer Function

- Evaluate the transfer function wrt W (let Z=1, L=1).
  - Use the transfer function in previous example
- Note $A=W^2$, $B=W$, $C=W$, $D=1$, $E=W$, $F=W$, $G=W^2$

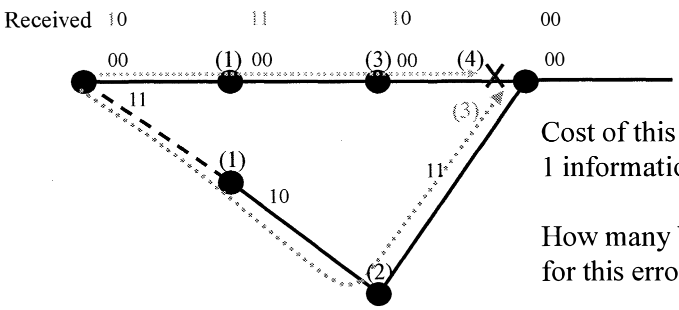
$$T(W, Z=1, L=1) = \frac{W^5 - W^6 + W^6}{1 - W - W + W^2 - W^2} = \frac{W^5}{1-2W}$$

Use $\frac{1}{1-D} = 1 + D + D^2 + D^3 + \cdots$

- We have $T(W) = W^5 (1+2W+4W^2+8W^3+ \ldots)$
  $= W^5 + 2W^6 + 4W^7 + 8W^8 + \ldots$
  - 1 weight five path, 2 weight six paths, 4 weight seven paths, …

Erroneous Decision

- Let's assume all-zero codeword was transmitted
- Noise was so strong that it caused some of bits flipped



Cost of this erroneous decision is 1 information bit

How many bits, need to be flipped for this erroneous event to occur?

Do location of errors matter?

Erroneous Decision (2)

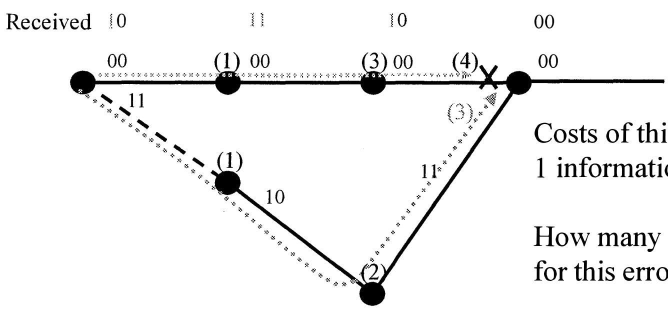
- Transmitted codeword = (00 00 00 00 00 ...)
- Received codeword = (10 11 10 00 00 ...)
- Sequence of d_{free} = (11 10 11 00 00 ...), the nearest neighbor

Need to pay attention on these positions

Don't need to pay attention on this coordinate because the codeword bit is also zero (same as the all-zero codeword)
- Whether it is flipped or not will not make a distinction in making a comparison with the all-zero codeword

Erroneous Decision (BSC)

- Let's assume all-zero codeword was transmitted
- Noise was strong in some coordinates that it caused some of bits flipped



Costs of this erroneous decision is 1 information bit

How many bits, need to be flipped for this erroneous event to occur?

Do location of errors matter?

Erroneous Decision (2) (BSC)

- Transmitted codeword = (00 00 00 00 00 …)
- Received codeword = (10 11 10 00 00 …)
- Sequence of d_{free} neighbor = (11 10 11 00 00 …), the nearest

Need to pay attention on these positions

Don't need to pay attention on this coordinate because the codeword bit is also zero (same as all-zero's)

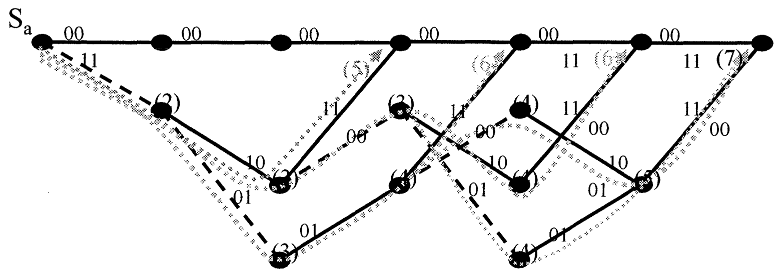
- Whether it is flipped or not will not make a distinction in making a comparison with the all-zero codeword

The Probability of Making Erroneous Decision (BSC)

- What is the probability of receiving a non-zero codeword with the required number of bits (three or more in our previous example) flipped on those critical coordinates?
- This is called the pairwise error probability (a pair between the error path and the all-zero one)

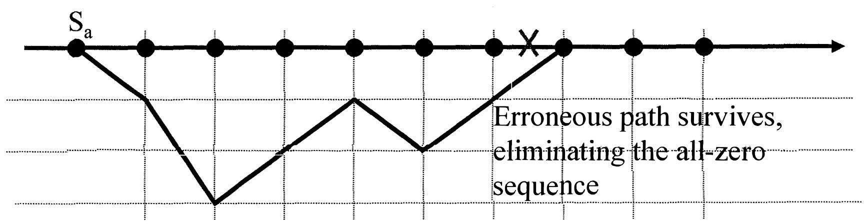
Performance Analysis

- An error path leaves the all-zeros path, say at node S_a, re-enters the node once with smaller partial accumulated path metric than that of the all-zero path
- Pairwise erroneous events are not disjoint. The underlying assumption of the union bound is that the pairwise error events are disjoint events.
- The node error probability $P_e(S_a)$ at node S_a is the probability of the union of all such pairwise error events



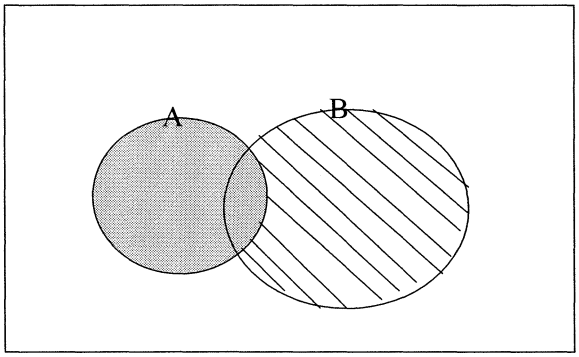
Union Bound

- A pairwise error event E_j: Given all-zero sequence, ML decoder makes a decision in favor of a different path which departed from the all-zero path at S_a, and re-merged to it once at a later time.
- Consider all individual pairwise error (PE) events $\{E_j\}$ that are possible from the node S_a, and treat them as if they are disjoint events.
- Sum up all the PE probabilities.



Basic Fact in Probability

- $\Pr\{A \cup B\} = \Pr\{A\} + \Pr\{A^c \cap B\}$
  $\leq \Pr\{A\} + \Pr\{B\}$



Union Bound (2)

- Let $\{E_1, E_2, \ldots, E_n\}$ be a collection of all the possible erroneous events, the probability of the union of events is less than or equal to the sum of individual probabilities $P(E_j)$, j=1, 2, …
- $\Pr(E_1 \cup E_2 \cup \ldots \cup E_n) \leq \Pr(E_1) + \Pr(E_2) + \ldots + \Pr(E_n)$
- Each E_j denotes a particular pairwise error event
- Using this bound, we can say
- $P_e(S_a) \leq \sum_{\mathbf{c} \in C_a} \Pr\{\mathbf{r}: p(\mathbf{r}|\mathbf{c}) \geq p(\mathbf{r}|\mathbf{0})\}$
  where C_a is the set of all possible error paths diverged at S_a from, and re-merged to, the all-zero path only once.
- We call $\Pr\{\mathbf{r}: p(\mathbf{r}|\mathbf{c}) \geq p(\mathbf{r}|\mathbf{0})\}$ the *pairwise error probability*

Union Bound (3)

- $P_e(S_a) \le \sum_{\mathbf{c} \in C_a} \Pr\{\mathbf{r}: p(\mathbf{r}|\mathbf{c}) \ge p(\mathbf{r}|\mathbf{0})\}$
  $=: \sum_{d=1} n_d P_d$

  where
  - d = weight($\mathbf{c}$),
  - n_d is the number of paths with weight d,
  - $P_d := \Pr\{\mathbf{r}: p(\mathbf{r}|\mathbf{c}) \ge p(\mathbf{r}|\mathbf{0}),\ w(\mathbf{c})=d\}$
    $= \Pr\{\mathbf{r}: \prod_{j=1}^{d} p(r_j|1)/p(r_j|0) \ge 1\}$
- Now let's focus more on the pairwise error probability P_d with weight d.

Indicator Function

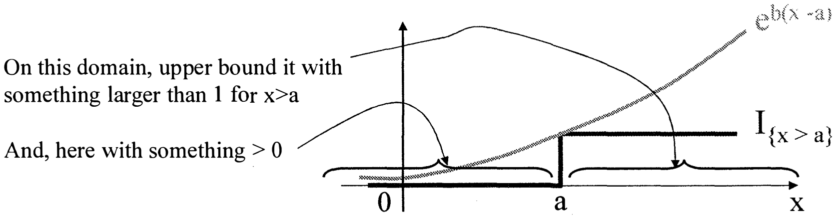
- X: $\Omega \to R$

$$I_{\{X>\delta\}} := \begin{cases} 1 & X > \delta \\ 0 & X \le \delta \end{cases}$$

- Taking expectation
- $E\{I_{(X>\delta)}\} = 1 * \Pr\{X>\delta\} + 0 * \Pr\{X \le \delta\}$

Chernoff Bound

- Let X: $\Omega \rightarrow R$
- We are interested in approximating the tail pdf of X, such that $\Pr\{X > a\}$, for the interval (a, ∞)
- Note that $I_{\{X > a\}} \leq e^{b(X-a)}$ for any real b>0



- Taking the expectation on both sides
- $E\{I_{\{X > a\}}\} = \Pr\{X>a\} \leq E\{e^{b(X-a)}\} = e^{-ba} E\{e^{bX}\}$ for any real b>0
  - We can find the optimal b that achieves the equality closely

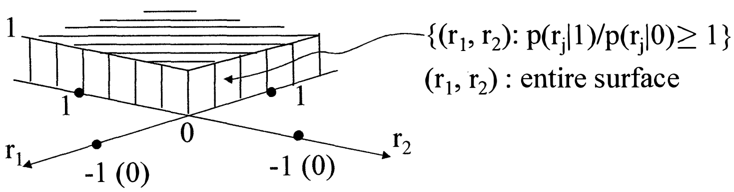
Pairwise Error Probability P_d with weight *d*

- For a path with weight *d*, we have

$$P_d = \Pr\{\mathbf{r}: \prod_{j=1}^{d} p(r_j|1)/p(r_j|0) \geq 1\}$$

- Now consider the following (construct Chernoff bound)

$$I_{\{r'=(r1\ldots rd):\ p(rj|1)/p(rj|0) \geq 1,\ \text{for all } j=1,\ldots,d |\ \mathbf{0}\}}$$
$$\leq \prod_j (p(r_j|1)/p(r_j|0))^s\, I_{r'=(r1,\ \ldots,\ rd\ |\ \mathbf{0})}$$



Pairwise Error Probability P_d with weight *d* (2)

- Taking the expectation on both sides
- LHS = P_d
- RHS = $\Sigma_{(r1,r2,\ldots,rd)} \prod_j p(r_j|0)\,(p(r_j|1)/p(r_j|0))^s$
  $= \Sigma_{(r1,r2,\ldots,rd)} \prod_j p(r_j|1)^s / p(r_j|0)^{1-s}$
- Thus, we have
  $P_d < \prod_{j=1}^{d} \Sigma_{rj}\, p(r_j|1)^s / p(r_j|0)^{1-s}$
- The best *s* which minimizes RHS and thus, making RHS as close as possible to LHS, can be found.
- When the channel is symmetric (AWGN, BSC, …), the best *s* = ½. The bound with *s*=1/2 is called the **Bhattacharyya bound.**

Bhattacharya Bound on Pairwise Error Prob.

- Let **c** = (0 0 1 0 0 1 …) has *d* non-zero coordinates
- Comparing **c** with all-zero codeword, and considering the event that metric of **c** is favorable than all-zero codeword
- Need to consider only the *d* non-zero coordinates
- $P_d \le \prod_{j=1}^{d} [\Sigma_{rj}\, p(r_j|1)^{1/2} / p(r_j|0)^{1/2}] = Q^d$
  where $Q = [\Sigma_{rj}\, p(r_j|1)^{1/2} / p(r_j|0)^{1/2}]$

Union Bound (4)

- $P_e(S_a) \leq \sum_{\mathbf{c} \in \mathbf{C}a} Pr\{\mathbf{r}: p(\mathbf{r}|\mathbf{c}) \geq p(\mathbf{r}|\mathbf{0})\}$
  $=: \sum_{d=1} n_d P_d$
- Note n_d—the number of paths with weight d — can be readily obtained from evaluating the transfer function at $T(W) = T(W, Z=1, L=1)$
- Note that T(W) has the form
  $T(W) = n_1 W + n_2 W^2 + n_3 W^3 + \ldots$
- Finally, the probability of error at node a is

$$P_e(S_a) \leq \sum_{d=1} n_d Q^d = T(W)|_{W=Q}$$

T(W, Z, L=1) for bit error probability P_b

- $T(W,Z) = n_{1,1}WZ + n_{1,2} WZ^2 + n_{1,3}WZ^3 + \ldots + n_{i,j}W^iZ^j + \ldots$
- $\frac{\partial T(W,Z)}{\partial Z} = n_{1,1}WZ + 2 n_{1,2}WZ^2 + 3n_{1,3}WZ^3 + \ldots + jn_{i,j}W^iZ^j + \ldots$
- $\frac{\partial T(W,Z)}{\partial Z}|_{Z=1} = (n_{1,1} + 2n_{1,2} + 3n_{1,3})W + (\ldots)W^2 + \ldots + (\ldots) W^i$
  $= b_1W + b_2W^2 + b_3W^3 + \ldots$

  where $\{b_i\}$ are the total number of nonzero information bits associated with codeword of weight i
- Thus, the average number of bit error rate

$$P_b \leq \frac{1}{k}\frac{\partial T(W,Z)}{\partial Z}|_{W=Q,Z=1}$$

Union Bound BER equation

- $P_b = 1/k \, E[b_i]$
  $= 1/k \sum_i b_i P_i$
  $\leq 1/k \, [\partial T(W,Z)/\partial Z]_{W=Q,\, Z=1}$
- For each trellis-section, k information bits are transmitted
- b_j denotes the number of information bits in error, associated with codeword of weight j
- P_i is the pairwise error probability of weight i

Approximation of P_b

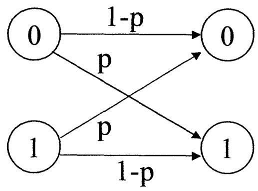
- Obtaining the transfer function is difficult, thus often P_b is approximated by considering the paths whose weights are d_{free}

$$P_b \approx \frac{1}{k} b_{d_{free}} Q^{b_{free}}$$

- Where b_{dfree} is the number of non-zero information bits associated with the codewords of weight d_{free}

Q for Binary Symmetric Channel

- $Q = [\sum_{rj} p(r_j|1)^{1/2} / p(r_j|0)^{1/2}] = 2\sqrt{(1-p)p}$
- Note that $r_j = 0$ or 1
- When $r_j = 1$, $p(1|1)^{1/2}p(1|0)^{1/2} = \sqrt{(1-p)p}$
- When $r_j = 0$, $p(0|1)^{1/2}p(0|0)^{1/2} = \sqrt{p(1-p)}$



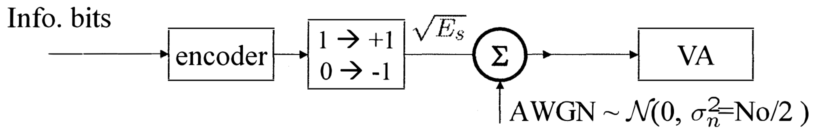
Exact Pairwise Error Probability for BSC

- P_d = Pr{half or more bits in d-coordinates are in error}
- ML decoding makes erroneous decision,
  - For d odd, $(d+1)/2$ or more bits must be in error
    - Ex) For d=5, 3, 4 or 5 bit-flips are needed
  - For d even, $(d/2)+1$ more bits must be in error
    - Ex) For d=6, 4 or 5 bit-flips are needed. When 3 bit-flips, with probability ½, ML decoding will make a wrong decision

Performance over AWGN

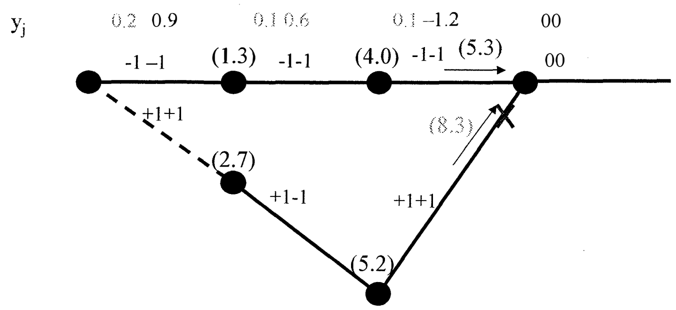
- $y_j = \sqrt{E_s}\,x_j + n_j$, j=1,2,…,
- For a particular pairwise error event with weight *d*:
  - For hard decision decoding, Hamming distance of 1 is counted when x_j=-1 for codeword bit 0 was sent but received was $r_j \geq 0$.
  - For soft-decision decoding, it will be a different story for soft-decision decoding

$$E_b = E_s \cdot \frac{1}{R_c}$$



Performance over AWGN (2)

- Recall from Lecture-9, the ML soft-decision metric is
  $\sum |y_j - x_j|^2$, (Use E_s = 1 for simplicity)
  - We can use the absolute value for metric
- Note what happens when using this Euclidean distance, instead of the Hamming distance



Under hard decision Decoding, all zero path gets eliminated

Under soft-decision decoding, the erroneous path gets eliminated

Performance over AWGN (3)

- On those *d* non-zero coordinates (d=5 in our example), find out the probability of a particular receive signal y_j with which the likelihood of all-zero path is smaller than that of non-zero one
- Since y_j is Gaussian with mean = –1 and variance σ_n^2
- $P_d = \Pr\{ \sum |y_j - x_j(\text{error-path})|^2 < \sum |y_j - x_j(\text{all-zero})|^2\}$
  $= \Pr\{\sum_{j=1}^{d} (|y_j - 1|^2 - |y_j+1|^2) < 0\}$
  $= \Pr\{\sum_{j=1}^{d} y_j > 0\}$
- $Y := \sum_{j=1}^{d} y_j$, each y_j are iid Gaussian rv's
- Gaussian is defined completely with mean and variance
- $E(Y) = \sum_{j=1}^{d} E(y_j) = d*(-1) = -d$
- $Var(Y) = \sum_{j=1}^{d} Var(y_j) = d*\sigma_n^2$

Performance over AWGN (3)

- Let's use $\sigma_n^2 = N_o/2$ and an arbitrary E_b, then we have

$$P_d = Pr(Y > 0) = \frac{1}{\sqrt{\pi d N_0}} \int_0^{\infty} e^{-\frac{|y+d\sqrt{E_s}|^2}{dN_0}} dy = Q\left(\sqrt{\frac{2dE_s}{N_0}}\right)$$

where $Q(x) := \frac{1}{2\pi} \int_x^{\infty} e^{-\frac{t^2}{2}} dt, \quad x \geq 0$

Finally, the upper bound on BER over AWGN

- Make use of a bound $Q(\sqrt{x+y}) \leq Q(\sqrt{x})e^{-y/2}$

$$\begin{aligned}
P_d &= Q(\sqrt{\frac{2dE_s}{N_o}}) \\
&= Q(\sqrt{\frac{2d_{free}E_s}{N_o} + \frac{(d - d_{free}E_s}{N_o}}) \\
&\leq Q(\sqrt{\frac{2d_{free}E_s}{N_o}})e^{-(d-d_{free})E_b/N_o} \\
&= e^{d_{free}E_s/N_o}Q(\sqrt{\frac{d_{free}E_s}{N_o}})e^{-dE_s/N_o}
\end{aligned}$$

$$E_b = E_s \cdot \frac{1}{R_c}$$

- Finally, the BER is

$$P_b \leq \frac{1}{k}e^{d_{free}E_s/N_o}Q(\sqrt{\frac{2d_{free}E_s}{N_o}})\frac{\partial T(W,Z)}{\partial Z}|_{W=e^{-E_s/N_o},Z=1}$$

Genie-Aid Lower Bound on BER over AWGN

- For a lower bound, let's assume an unrealistic scenario in which a magic genie provides the receiver two codewords from which the receiver makes a decision, one is the actual transmitted codeword and the other a codeword with d_{free} distance away from it

$$P_b \geq \frac{1}{k}P_{d_{free}} = \frac{1}{k}Q(\sqrt{\frac{2d_{free}E_s}{N_o}})$$

Tables of Good Conv. Codes

[Moon506]

The program `finddfree` finds d_{free} for a given set of connection coefficients. It has been used to check these results. (Currently implemented only for $k = 1$ codes.)

$R = 1/2$ [251, 197]

L	$g^{(1)}$	$g^{(2)}$	d_{free}
3	5	7	5
4	64	74	6
5	46	72	7
6	65	57	8
7	554	744	10
8	712	476	10
9	561	753	12
10	4734	6624	12
11	4762	7542	14
12	4335	5723	15
13	42554	77304	16
14	43572	56246	16
15	56721	61713	18
16	447254	627324	19
17	716502	514576	20

$R = 1/3$ [251, 197]

L	$g^{(1)}$	$g^{(2)}$	$g^{(3)}$	d_{free}
3	5	7	7	8
4	54	64	74	10
5	52	66	76	12
6	47	53	75	13
7	554	624	764	15
8	452	662	756	16
9	557	663	711	18
10	4474	5724	7154	20
11	4726	5562	6372	22
12	4767	5723	6265	24
13	42554	43364	77304	24
14	43512	73542	76266	26

Tables of Good Conv. Codes

[Moon507]

$R = 1/4$ [251, 197]

L	$g^{(1)}$	$g^{(2)}$	$g^{(3)}$	$g^{(4)}$	d_{free}
3	5	7	7	7	10
4	54	64	64	74	13
5	52	56	66	76	16
6	53	67	71	75	18
7	564	564	634	714	20
8	472	572	626	736	22
9	463	535	733	745	24
10	4474	5724	7154	7254	27
11	4656	4726	5562	6372	29
12	4767	5723	6265	7455	32
13	44624	52374	66754	73534	33
14	42226	46372	73256	73276	36

$R = 2/3$ [254, 172]

L	ν	$g^{(1,1)}$ / $g^{(2,1)}$	$g^{(1,2)}$ / $g^{(2,2)}$	$g^{(1,3)}$ / $g^{(2,3)}$	d_{free}
2	2	6	2	6	3
		2	4	8	
3	3	5	2	6	4
		1	4	7	
3	4	7	1	4	5
		2	5	7	
4	5	60	30	70	6
		14	40	74	
4	6	64	30	64	7
		30	64	74	
5	7	60	34	54	8
		16	46	74	
5	8	64	12	52	8
		26	66	44	
6	9	52	06	74	9
		05	70	53	
6	10	63	15	46	10
		32	65	61	

Tables of Good Conv. Codes

[Moon507]

$R = 3/4$ [254, 172]

L	ν	$g^{(1,1)}$ $g^{(2,1)}$ $g^{(3,1)}$	$g^{(1,2)}$ $g^{(2,2)}$ $g^{(3,2)}$	$g^{(1,3)}$ $g^{(2,3)}$ $g^{(3,3)}$	$g^{(1,4)}$ $g^{(2,4)}$ $g^{(3,4)}$	d_{free}
2	3	4	4	4	4	4
		0	6	2	4	
		0	2	5	5	
3	5	6	2	2	6	5
		1	6	0	7	
		0	2	5	5	
3	6	6	1	0	7	6
		3	4	1	6	
		2	3	7	4	
4	8	70	30	20	40	7
		14	50	00	54	
		04	10	74	40	
4	9	40	14	34	60	8
		04	64	20	70	
		34	00	60	64	

Table 12.2 presents a comparison of d_{free} for systematic and nonsystematic codes (with polynomial generators), showing that nonsystematic codes have generally better distance properties. Results are even more pronounced for longer constraint lengths.

Other Subjects on Convolutional Codes

- Punctured Convolutional Codes

- Suboptimal Decoding Algorithms
  - *M* algorithm
  - *T* algorithm
  - Fano algorithm

Summary

- Convolutional codes have been one of the very successful codes.
- MLSD decoder has been available since Viterbi[67].
- Large d_{free} codes have been found and tabulated.
- Concatenation of Reed Solomon code and Convolutional code [Forney's dissertation 1965]
  - Voyager [1977]
- Still used in many communications systems
  - Cell phones, space crafts, telecom/broadcasting systems
- They are also used as a component in Trellis Codes, Turbo Codes, LDPC Codes, …

HW #5

- #1. (Moon 12.6, 12.7, 12.12, 12.23)
- #2. Draw the rate ½ recursive convolution encoder defined by G(D) = [1 $(1+D+D^2+D^3)/(1+D+D^3)$], and obtain its state diagram
- #3. Consider the rate ½ feedforward convolutional encoder given in the lecture, and assume the soft decoding channel model defined in the lecture — $\mathbf{y} = \mathbf{x} + \mathbf{n}$, where $\mathbf{x}=2\mathbf{c}-1$ and $\mathbf{n}$ is AWGN with $E(\mathbf{n}^T\mathbf{n})$ being a diagonal matrix. Suppose y = (0.9 0.5, 0.2 0.1, 0.2 0.3, 0.2 0.1, –1.0 0.1, 0.9 –0.2). Use the soft-decoding Viterbi Algorithm on the trellis, and find the maximum likelihood codeword. Compare your result with the one obtained using hard decision decoding metric in the lecture.

HW #5

- #4. Consider a rate ½ feedforward convolutional encoder defined by $G(D)=[1+D+D^3 \quad 1+D+D^2+D^3]$.
  - Obtain its state diagram.
  - Draw a completed trellis-section that defines the the encoder.
  - Find the free minimum distance by running Viterbi algorithm on your trellis. (Hint: Start with a single path that departs from the all-zero path. Use the procedure—determining d_{free} with VA—described in the lecture. Show your working of VA on the trellis, as was given in lecture.
  - Based on your results, determine the traceback depth

- #5. Calculate the union bounds for upper and lower bounds of the rate ½ encoder, $G(D)=[1+D+D^2 \quad 1+D^2]$, whose transfer function was obtained in Lecture. Obtain performance curve graphs for soft- and hard-decision bounds. Compare them with the uncoded case.

References

1. P. Elias, "Coding for noisy channels," IRE Conv. Record, Part 4, pp.37-47, 1955.
2. G.D. Forney, "The Viterbi algorithm," Proc. of IEEE, vol.61, no.3, Mar. 1973.

Trellis Codes

Ref: Ungerboeck's 1982 paper

Gottfried Ungerboeck

[Wiki]

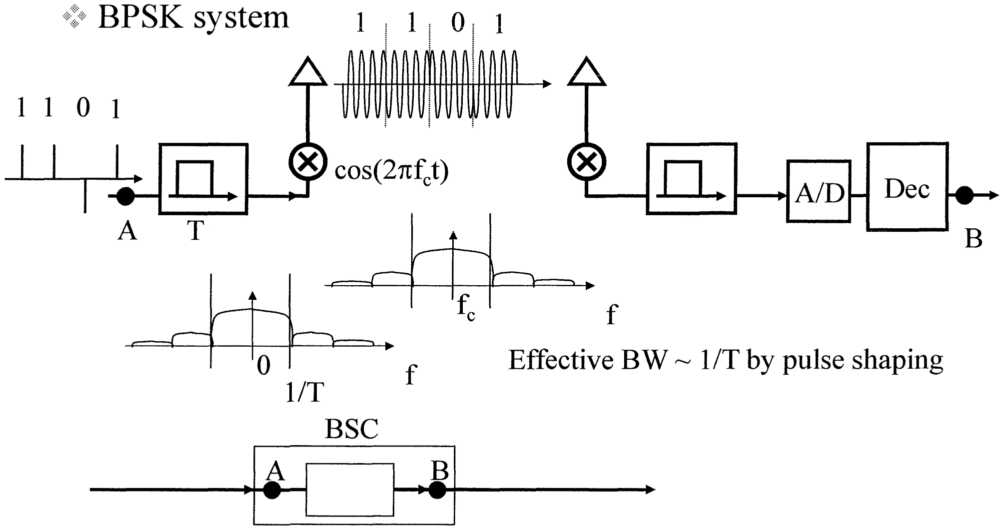
- Born 15 March 1940
- Austrian Communications Engineer.
- Ungerboeck received an electrical engineering degree (with emphasis on telecommunications) from Vienna University of Technology in 1964, and a Ph.D. from the Swiss Federal Institute of Technology, Zurich, in 1970.
- He joined IBM Austria as a systems engineer in 1965, and the IBM Zurich Research Laboratory in 1967.
- Ungerboeck joined Broadcom in 1998 as Technical Director for Communication Systems Research.

Channel Coding with Modulation (Trellis Codes)

- What was proposed?
  - New coding technique to improve detection performance without increasing the bandwidth or sacrificing the data rate.
  - Joint channel coding and modulation.
- How?
  - Design convolutional (trellis) codes which increase the free Euclidean distance, instead of increasing the free Hamming distance.
- Benefits?
  - Achieves coding gain of 3-4 dB with simple codes for 8 PSK and 16 QAM modulations.

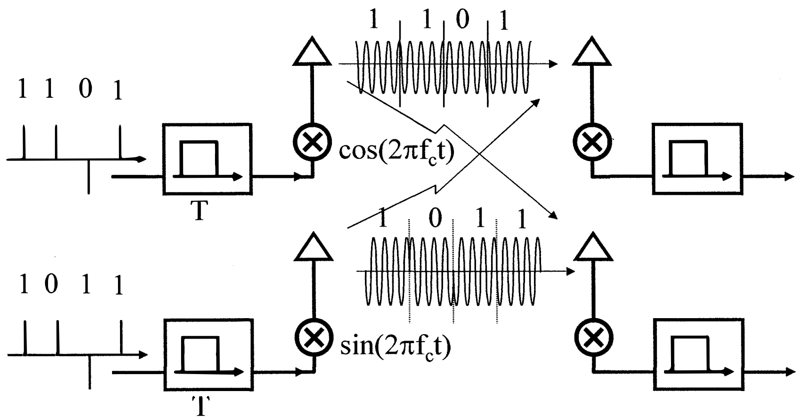
A Little Bit of Review on Modulation Theory (From EE 1473)

- BPSK system

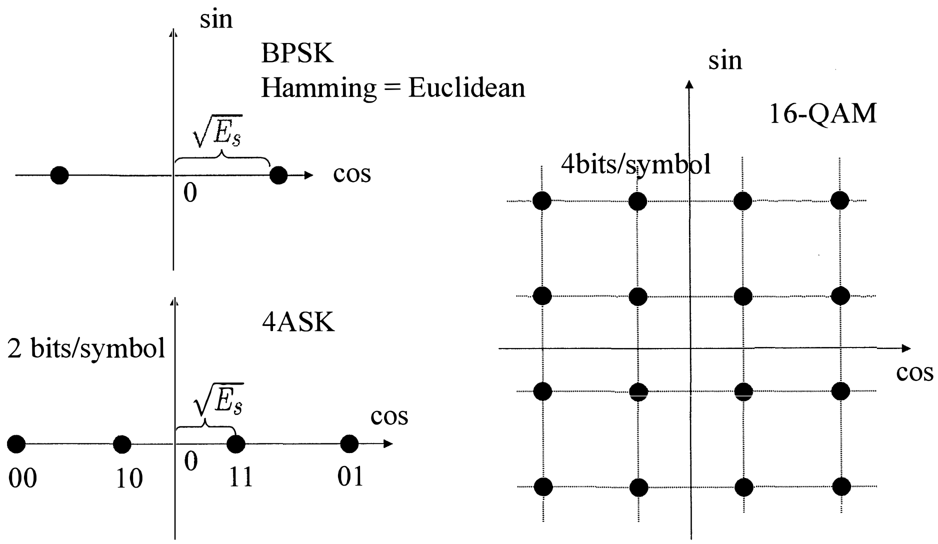


QAM signals
(QASK in Ungerboeck)

- cos and sin are orthogonal to each other. Thus, two independent channels



1-D/2-D Signal Constellation

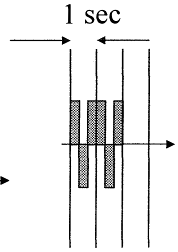


For multi-level signals, Hamming ≠ Euclidean in general

How to provide room for redundancy?

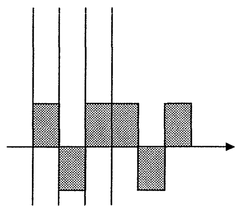
- Increase the bandwidth

Information bit rate of 1 bits/sec → Rate 1/3 code → 3bits/sec



- Or, decrease the data rate

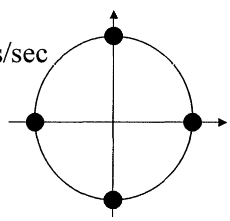
Information bit rate of 1/3 bits/sec → Rate 1/3 code → 1bits/sec



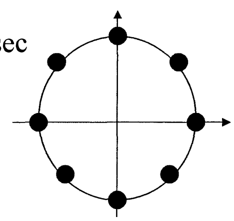
Or, use Trellis Codes

- Trellis codes increase the size of signal constellation to create room for redundancy
  - Example) QPSK ⇒ 8 PSK

2 bits/sec



3 bits/sec

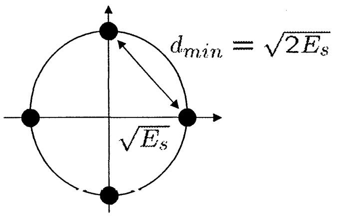


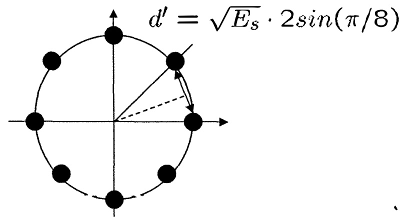
8 PSK

- Anything that we are loosing?

d_{min} is smaller

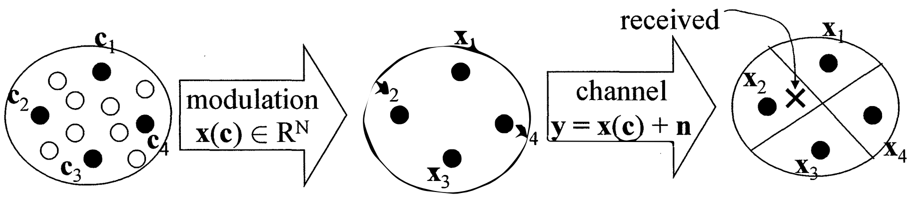
- d_{min} is critical in determining the symbol error probability (or BER) for uncoded transmission (why?)
- For a bigger signal set with equal average symbol energy, d_{min} should be smaller.
- Thus, in order for Trellis code to work, the benefit from coding should surpass the loss of having smaller $d_{min} = d'$





Design Criteria

- Design a trellis structure that maximizes the free Euclidean distance d_{free}
  - For convolutional codes, codes which maximize the Hamming d_{free} were desired
- Ungerboeck noticed what ultimately matters is the Euclidean d_{free} of transmitted symbol sequences $\{\mathbf{x}_j\}$, because they determines the detection performance



Minimum Free Hamming distance was maximized for convolution codes

But, actually the minimum free Euclidean distance should be maximized (Ungerboeck's Design Criterion)

Coded Transmission vs. Uncoded

- Signal constellation {a, b, c, d}
- Uncoded transmission
  - Transmitted sequence: a b c a d b c d a d …
  - Symbols are chosen independently at each epoch
  - If N = length of the sequence, 4^N possible sequences
  - Minimum distance between transmitted sequences = minimum distance in signal constellation
- Coded transmission
  - Transmitted sequence: a b d d c c b . . .,
  - Symbols are chosen dependently upon previous symbols in the sequence.
  - If N = length of the sequence, only $4^{N(k/n)}$ possible sequences.
  - Dependency means the increase in minimum distance between sequences

Thus, we could both loose and gain

- We get smaller d' by going for bigger constellation.
- But, we will eventually get bigger gain by having increased d_{free} of the trellis-coded transmitted sequence.
- Thus, the gain should be $(d_{free})^2/(d_{min})^2$.
- Note that for a fair comparison we should fix the transmitted powers of both uncoded and coded cases to be the same
  - Or use a compensation factor E_s'/E_s where E_s' is the average symbol energy with coding and E_s is that without coding.
- Thus, we can define an overall gain factor

$$\gamma := (d^2_{free}E_s)/(d^2_{min}E_s')$$

How to Introduce Redundancy

- Create some dependency rule
  - Current symbol is dependent upon a few past symbols in the sequence
  - The few past symbols constitute a state

moving window

a a b b c a b d d

Current symbol is dependent upon past three and current input bits

- We can make use of trellis structure (or a state-diagram) for this.

Signal Constellations

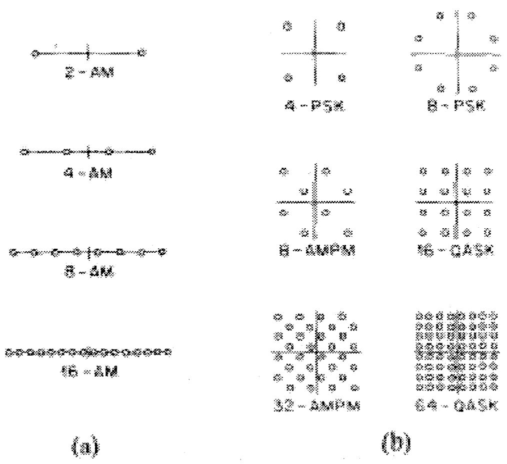


Fig. 1. Channel-signal sets considered in this paper. (a) One-dimensional modulation. (b) Two-dimensional modulation.

Constraint Channel Capacity

- Analysis of constraint channel coding capacity shows that we only need to **double** the size of signal constellation to get the most benefit
  - QPSK ⇒ 8 PSK
  - QPSK ⇒ 16 PSK, a significant diminishing return
- The input constraint: only symbols from a finite alphabet are allowed
  - Thus, the capacity is bounded
  - Constellation with 8 symbols ⇒ the maximum capacity is 3 bits per channel symbol

Channel Capacity Analysis

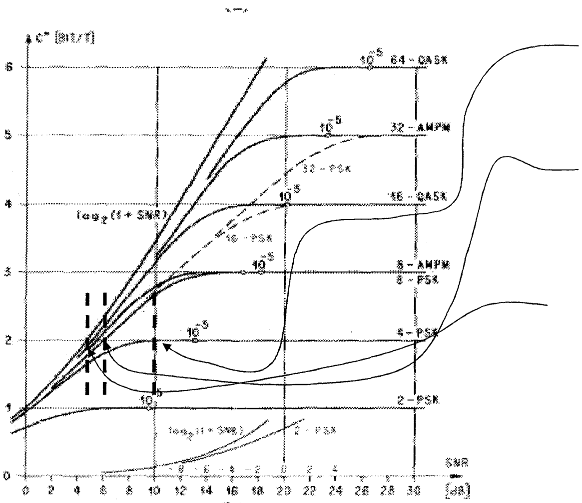


Fig. 2. Channel capacity C^* of bandlimited AWGN channels with discrete-valued input and continuous-valued output. a) One-dimensional modulation. b) Two-dimensional modulation.

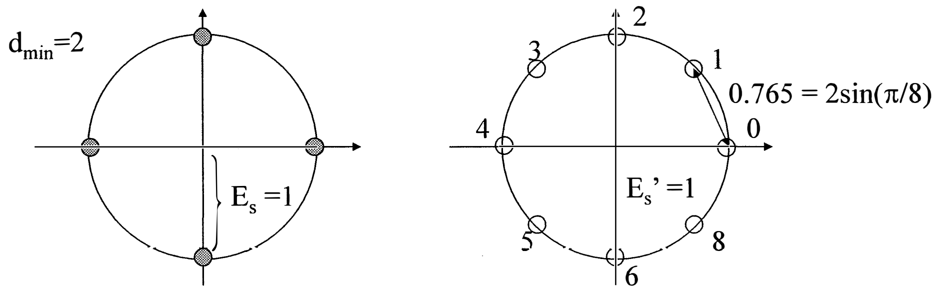
2 bits/T achievable around at SNR = 10.0 dB

If go for 8 PSK but send only 2 bits/sec, error free trans. is achievable at SNR = 5.9 dB

By going for unconstrained, we could gain only 1.2 dB. Thus, most benefit is obtained from going for the twice bigger constellation

Before examples, let's consider

- Take 4 PSK as reference uncoded system, and use 8 PSK for coded system
- The goal is to maximize the min. free ED of the resulting sequences
- Construct a state machine that gives out a symbol from 8 PSK constellation (three coded bits) with two input bits at a single encoding step
- Note in this example, $\gamma=(d_{free})^2/2$, since the energy per symbol is 1 for both



Set Partitioning and Idea Behind

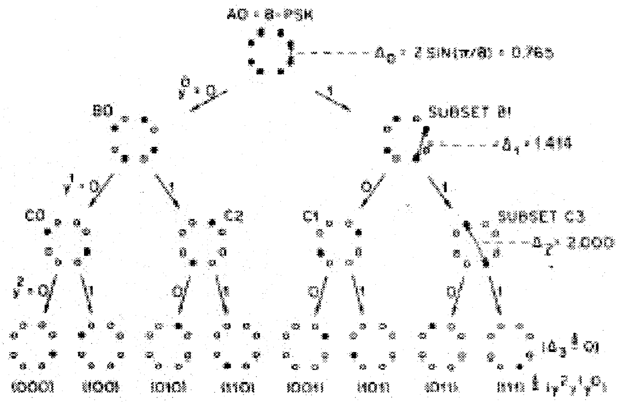


Fig. 4. Partitioning of 8-PSK channel signals into subsets with increasing minimum subset distances ($\Delta_0 < \Delta_1 < \Delta_2$; $E\{|a_n^2|\} = 1$).

Observation

- As going deeper into the partitioning, the minimum distance within a partition increases
  - When an early decision should be made, it would be desirable to make one within a deeper partition, as much as possible
  - For example, *conditioning* upon an event that C0 has occurred, we only need to make a decision between the most separated signal points 0 and 4 whose distance is 2.0
- Now, what about the the likelihood of occurrence of a subset C*i*?
  - These can be assigned to states in trellis such that we can deal with the sequence of partitioned subsets C1 C0 C2 C3 …, instead of symbols
- The more the number of states is, the larger free distance but the more difficult in decoding
- So, let's start with a small trellis-state example

1 Trellis State

- With one state, we cannot afford to create a coded sequence
  - At every trellis-section we have four merged paths, and thus, we must make four decisions—which is the same for uncoded 4 PSK modulation

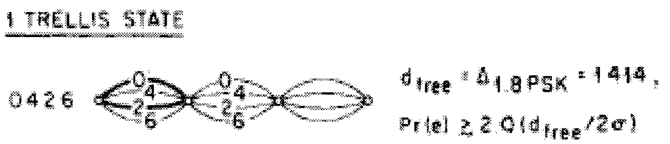
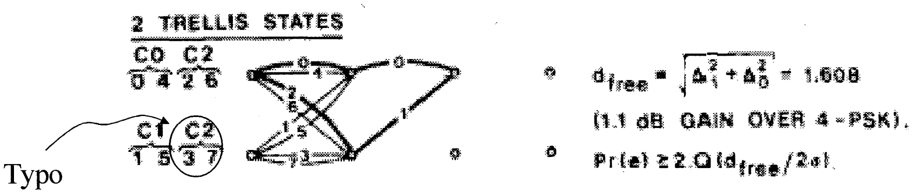


Fig. 6. Uncoded 4-PSK modulation, 2 bit/*T*.

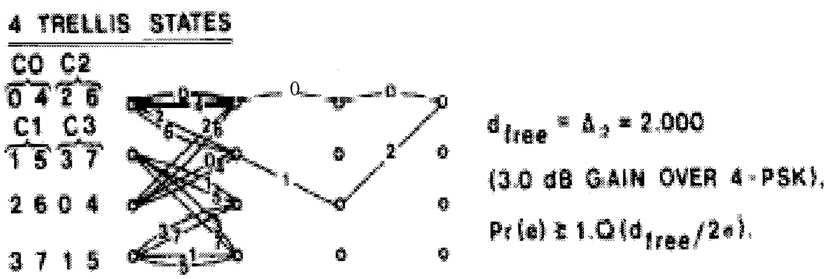
2 Trellis States

- With 2 states, some coding benefit can be achieved, but parallel transitions are required
  - Consider the first state, we need to make four transitions thus need four signals—say 0,4,2,6
  - Assign the maximally separated symbols on parallel transitions, such as 0-4, 2-6, 1-5 and 3-7
  - Similarly for the second state, use signals 1,5,3,7 (Entropy—use signals equally often)
- $(d_{free})^2 = d^2(0,2)+d^2(0,1)=2.0+(0.765)^\wedge 2=2.586$
- $\gamma = (d_{free})^2/2.0 = 2.586/2.0 = 1.293 \quad \Rightarrow \quad$ 1.1 dB SNR gain

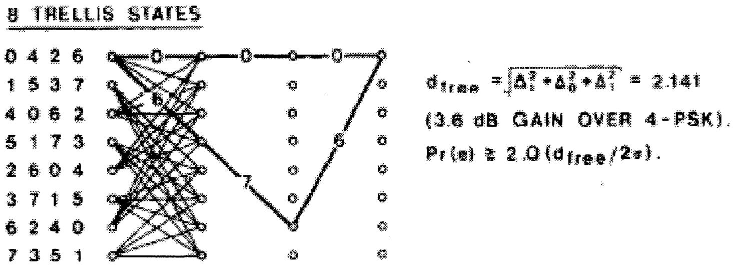


4 Trellis States

- $(d_{free})^2 = d^2(0,4) = 4.0$
- $\gamma = 4.0/2.0 \quad \Rightarrow \quad$ 3.0 dB SNR gain
- Parallel transition was choice, why it was used here?



8 Trellis States



- $(d_{free})^2 = d^2(0,6) + d^2(0,7) + d^2(0,6) = 2 + (0.765)^2 + 2 = 4.568$
- $\gamma = 4.856/2 = 2.293 \quad \Rightarrow$ 3.6 dB SNR gain

16 Trellis States

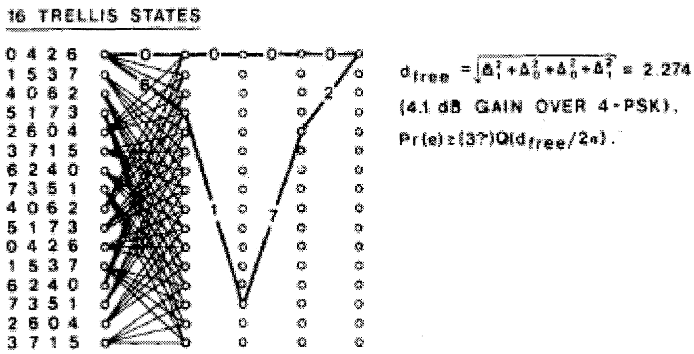


Fig. 7. Coded 8-PSK modulation, 2 bit/T.

16-QAM

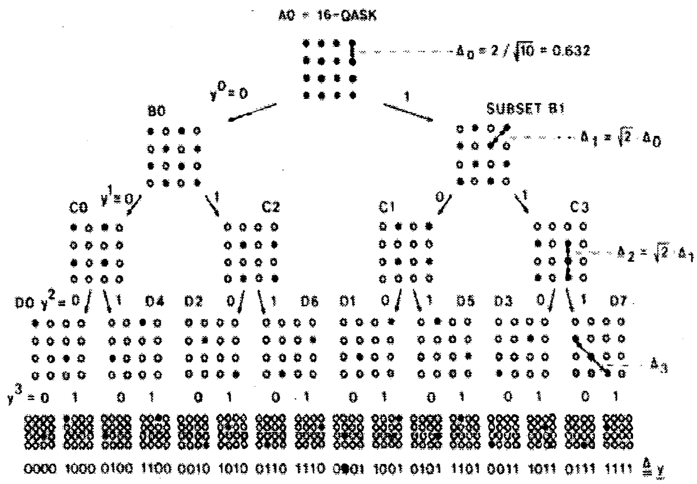


Fig. 5. Partitioning of 16-QASK channel signals into subsets with increasing minimum subset distances ($\Delta_0 < \Delta_1 < \Delta_2 < \Delta_3$; $E(|a_n^2|) = 1$).

8 Trellis States

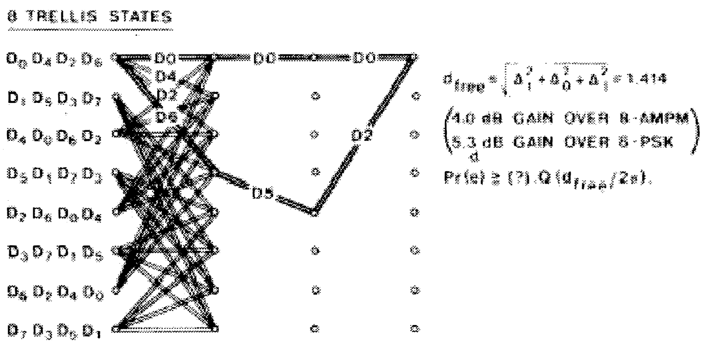


Fig. 8. Coded 16-QASK modulation, 3 bit/T.

16 Trellis States

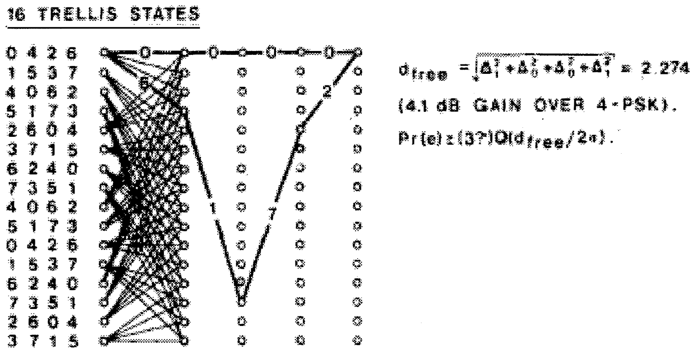


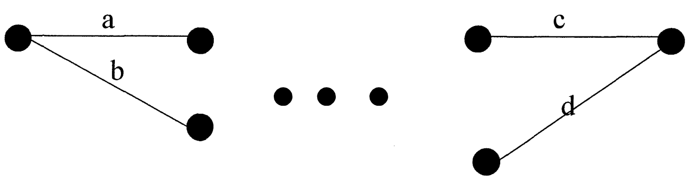
Fig. 7. Coded 8-PSK modulation, 2 bit/T.

Dominant Factors in Determining Free Distance

- "Parallel transitions"

- "adjacent transitions",

$$(d_{free})^2 = d^2(a,b) + \ldots + d^2(c,d)$$



Ungerboeck's Heuristic Rules

- For a given number of states, we are looking for a trellis which gives maximum free Euclidean distance
- Rule#1: Signals are used equally often
- Rule#2: Parallel transitions are assigned to the members of the same partition
- Rule #3: Adjacent transitions are assigned to the members of the next larger partition

How Far Can We Take This?

- Partitioning one dimensional signal sets results in minimum subset distances $d_{j+1} = 2\ d_j$.
- Partitioning two dimensional signal sets results in minimum subset distances $d_{j+1} = \text{sqrt}(2)\ d_j$.
- Partitioning a large signal set, after a few partitions, gives a minimum subset distance that exceeds the free ED that one can ever expect.
- It is sufficient to partition only two or three times.

Codes for 8 PSK Modulation

- As complexity increases (as the number of trellis states increases), the coding gain is larger, but with a limit.
- For example, trellis code with the 2^7 states, the gain factor is 5.0 dB; with 2^9 states, it's 5.7 dB.
  - Recall the capacity results

TABLE II
CODES FOR 8-PSK MODULATION

ν	$\tilde{m}$	$H^0(D)$	$H^1(D)$	$H^2(D)$	d^2_{free}/Δ^2_1	$G^{m=2}_{8PSK/4PSK}$
2	1	5_8	2_8	–	2.000	3.0 dB
3	2	11	02	04	2.293	3.6
4	2	23	04	16	2.586	4.1
5	2	45	16	34	2.879	4.6
6	2	105	036	074	3.000	4.8
7	2	203	014	016	3.172	5.0
8	2	405	250	176	3.465	5.4
9	2	1007	0164	0260	3.758	5.7
$\overset{*}{\Delta}$ 10	2	2003	0164	0770	3.758	5.7

$E\{|a_n^2|\} = 1$;
$\Delta_0(4PSK) = \sqrt{2}$, $\Delta_0(8PSK) = 2\sin(\pi/8)$,
$\Delta_1(8PSK) = \sqrt{2}$, $\Delta_2(8PSK) = 2$

$\Delta_1^2(8PSK)/\Delta_0^2(4PSK) = 1$ (0 dB)

*Search not completed.
ΔNo improvement obtained.

Obtaining an Encoder From the Trellis

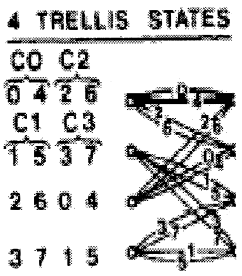
- So far, we have been concerned only with the design of a trellis that gives maximum free ED
- Now, let's think about how to realize the trellis with a state machine
- First, we need a mapping rule that assigns coded bits to the channel signals
  - We may use the natural mapping rule, such that 0→(000), 1→(001), 2→(010), 3→(011), 4→(100), 5→(101), 6→(110), 7→(111), for 8 PSK example
  - Note that these are just a naming convention (different names can do the job as well.)
- Now, we can find the binary state machine (binary convolutional encoder) that does the job,
  - It takes some effort but is do-able.

Obtaining an Encoder From the Trellis (2)

- A state machine produces an output (coded bits) due to a set of input bits (information bits) and the current state (memory), and jumps to a successor state.
- Let's use a convention like this
  - The four transitions from the top to bottom are due to input (00), (01), (10), (11).
  - The Ungerboeck's mapping (natural mapping) is used for naming the 8 PSK signals.
- Based on this, one could come up with a state transition table.
  - Let's take an example of the 4 trellis states code

4 Trellis State Example

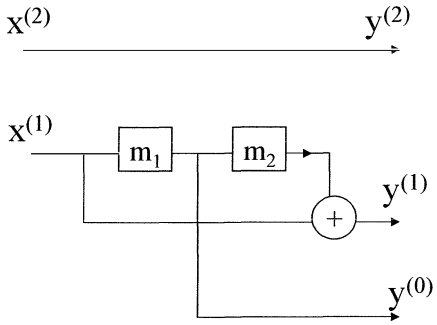
- A Binary representation of the 4 state Trellis Code



Current m_2 m_1	Input $x^{(1)}$ $x^{(2)}$	Output $y^{(2)}$ $y^{(1)}$ $y^{(0)}$	Next m_2 m_1
0 0	0 0	000	0 0
0 0	0 1	100	0 0
0 0	1 0	010	0 1
0 0	1 1	110	0 1
0 1	0 0	001	1 0
0 1	0 1	101	1 0
0 1	1 0	011	1 1
0 1	1 1	111	1 1
1 0	0 0	010	0 0
1 0	0 1	110	0 0
1 0	1 0	000	0 1
1 0	1 1	100	0 1
1 1	0 0	011	1 0
1 1	0 1	111	1 0
1 1	1 0	001	1 1
1 1	1 1	101	1 1

4 Trellis-States Example (2)

- First, give names to the trellis
  - I just chose the *natural* mapping for names of states and branches
- After constructing the table, look for governing relationships, from the current state and the input, to the output, and to the next state.
- Note the following can be obtained from the table
  - $y^{(2)} = x^{(2)}$
  - Input $x^{(1)}$ becomes next state's m_1, and the current state's m_1 becomes the next state's m_2
  - $y^{(1)} = x^{(1)} + m_2$
  - $y^{(0)} = m_1$



Other Examples in the Paper

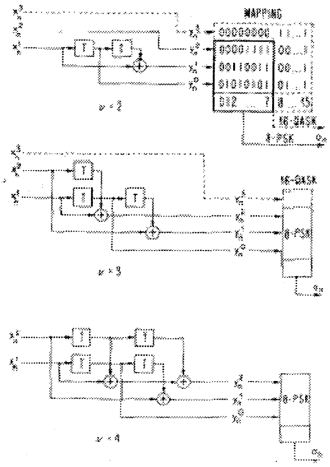
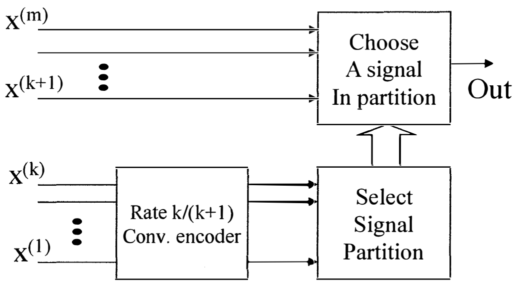


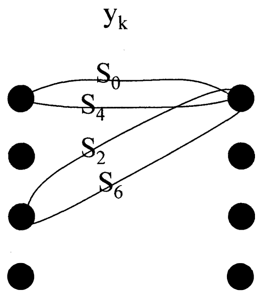
Fig. 9. Realization of 8-PSK and 16-QASK codes by means of minimal convolutional encoders.



No parallel transitions ⇒ no upper part (the selected partition has a single channel signal)

Viterbi Algorithm

- We already know how to perform Viterbi algorithm on trellis, except how to handle the parallel transition.
- On branches where you have parallel transitions, we first need to make a pruning decision among the parallel transitions, and then make another pruning decision among the paths merged

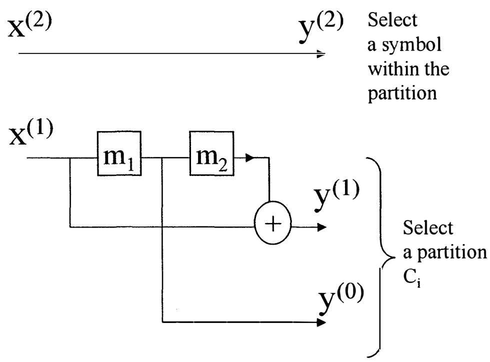


Min1=min$\{|y_k - S_0|^2, |y_k - S_4|^2\}$,
Min2=min$\{|y_k - S_2|^2, |y_k - S_6|^2\}$

Make the final decision among the two paths, in favor of the path having the minimum metric, min{Min1, Min2}

4 Trellis-States Example (2)

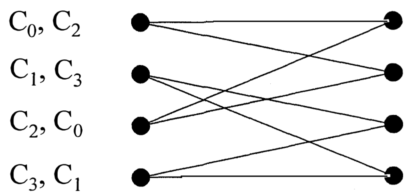
- First, give names to each elements in the trellis
  - Natural mapping for states and branches
- After constructing the table, look for governing relationships, from the current state and input, to the output, and to the next state
- Note the following can be obtained from the table
  - $y^{(2)} = x^{(2)}$
  - Input $x^{(1)}$ becomes next state's m_1, and the current state's m_1 becomes the next state's m_2
  - $y^{(1)} = x^{(1)} + m_2$
  - $y^{(0)} = m_1$



4 Trellis-States Example (3)

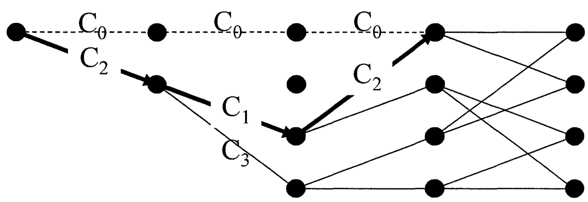
- $x^{(2)} = y^{(2)}$ does not make any influence on the output of the convolutional encoder
- Thus, a trellis can be obtained by excluding the two

Current $m_2\ m_1$	Input $x^{(1)}$	Output $y^{(1)}\ y^{(0)}$	Next $m_2\ m_1$
0 0	0	C_0	0 0
0 0	1	C_2	0 1
0 1	0	C_1	1 0
0 1	1	C_3	1 1
1 0	0	C_2	0 0
1 0	1	C_0	0 1
1 1	0	C_3	1 0
1 1	1	C_1	1 1



Two Types of Distances

- The minimum squared distance within the partition assigned to the parallel transitions (the single signal error event)
  - This is $d^2_{parallel} = 4.0$ for the example of four-states coded 8 PSK
- The minimum free distance of sequences of partition sets
  - The squared distances of any pair of partitions are $d^2(C_0, C_1)=d^2(0,1)$ $=0.585$, $d^2(C_0, C_2) = d^2(0,2)=2.0$, and $d^2(C_0, C_3) = 0.585$
  - $d^2_{sequence} = d^2(C_0, C_2)+d^2(C_0, C_1)+d^2(C_0, C_2) = 4.585 > 4.0$
- $d^2_{free} = \min\{d^2_{parellel},\ d^2_{sequence}\}=4.0$



Other Examples in the Paper

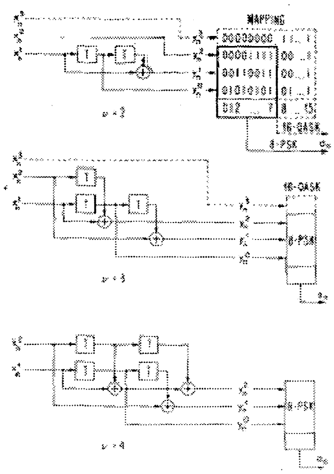
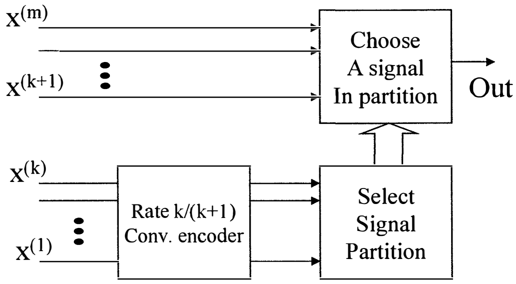


Fig. 9. Realization of 8-PSK and 16-QASK codes by means of minimal convolutional encoders.

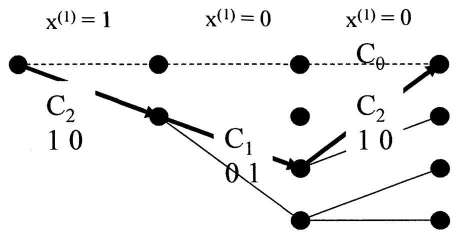


No parallel transitions ⇒ no top part (the selected partition has a single channel signal)

- Example is the 8 state 8 PSK code

Another Way of Obtaining an Encoder, From the Trellis

- Obtain the generator polynomial G(D) directly from the trellis by using the fact that G(D) is simply a collection of impulse responses



- Thus, $G(D) = [1 + D^2, \quad D]$ for $y^{(1)}$ and $y^{(0)}$ respectively

BER for BPSK

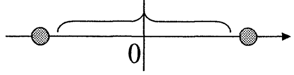
- $y_k = \text{sqrt}(E_s)\, x_k + n_k$, where $x_k \in \{-1, 1\}$ with equally likely

$$
\begin{aligned}
Pr(y_k > 0 | x_k = -1) &= \int_0^{\infty} \frac{1}{\sqrt{2\pi(N_0/2)}} e^{-\frac{(y+\sqrt{E_s})^2}{2(N_0/2)}} dy \\
&= \int_{\sqrt{2E_s/N_0}}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx \\
&= Q(\sqrt{2E_s/N_0}) = Q(d_E/\sqrt{2N_0})
\end{aligned}
$$

- $Q(x) := \frac{1}{2\pi} \int_x^{\infty} e^{-\frac{t^2}{2}} dt, \quad x \geq 0$

$E_b = E_s \cdot (1/R_c)$

$d_E = 2\sqrt{E_s}$



R_c = rate of the code
= number of bits each baud carries

The Complementary Error Function vs. Q(x)

- $erfc(x) := \frac{2}{\pi} \int_x^{\infty} e^{-t^2} dt$

- $Q(x) = \frac{1}{2} erfc(\frac{x}{\sqrt{2}})$

- MATLAB only defines the complementary error function erfc(x)

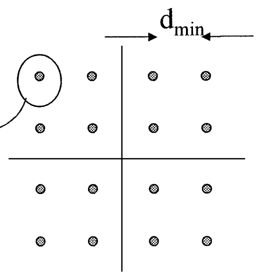
BER for 4 PSK

- We can treat this system as having two independent BPSK systems, one on sine and the other on cosine carrier
- Thus, the BER for 4 PSK is exactly the same as that for BPSK system
- HW#6, Problem #3 is now solved

Approximation of Uncoded Symbol Error Prob. P(e)

- $P(e) \approx \frac{N_{d_{min}}}{2} \mathrm{erfc}(\frac{d_{min}}{2\sqrt{N_0}}) = N_{d_{min}} Q(\frac{d_{min}}{\sqrt{2N_0}})$

- Where d_{min} is the minimum ED of the constellation and N_{dmin} is the average number of minimum distance events per signal point
- Example: 16-QAM signal constellation
  - N_{dmin} = (4*2 + 8*3 + 4*4)/16 = 3



- Need to convert d_{min} in terms of average energy of the signals for P(e) vs. SNR

Performance Analysis (Approximation) of Trellis Code Over AWGN

- At high SNR, the following approximations are good
- For the node error probability

$$P_e \approx N(d_{free})Q\left(\sqrt{\frac{d^2_{free}E_s}{2N_0}}\right)$$

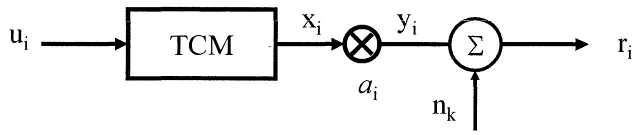
- Where $N(d_{free})$ is the number of sequences that are distance d_{free} from the transmitted sequence.
- For the bit error probability

$$P_b \approx \frac{b_{d_{free}}}{m}Q\left(\sqrt{\frac{d^2_{free}E_s}{2N_0}}\right)$$

- Where b_{dfree} is the total number of information bit errors in the erroneous path with distance d_{free}, m is the number of bits per trellis-section, and E_s is the average energy of signal

Rayleigh Fading Channel

- Now consider TCM signals over the following channel



- The fading a_i is a complex-valued Gaussian—real and imaginary parts are zero mean, variance of γ^2 and mutually independent real valued Gaussian r.v.s
- Let $r := |a_i|$, Rayleigh distribution $f(r) := (r/\gamma^2)\exp(-r^2/2\gamma^2)$
- Note that γ^2/N_0 is an average channel SNR

An Error Event involving L branches

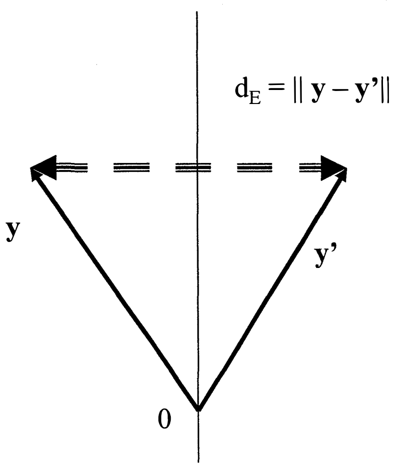
- Consider the following one-to-one mappings, starting from the input bit sequence

  $\mathbf{u} = (u_1, u_2, \ldots) \rightarrow \mathbf{x}=(x_1, x_2, \ldots) \rightarrow \mathbf{y} = (y_1, y_2, \ldots)$
- Now consider an error event in which **u** is the transmitted input bit sequence but the receiver makes a decision in favor of **u'**, which has corresponding **x'** and **y'**
- By making an assumption that there are L different symbols between **x** and **x'**, the squared Euclidean distance between the two sequences **y** and **y'** observed at the receiver is

  $$d^2_E = \|\mathbf{y} - \mathbf{y}'\|^2 = \|\boldsymbol{a}^T\mathbf{x} - \boldsymbol{a}^T\mathbf{x}'\|^2 = \sum_{i=1}^{L} |a_i|^2 \, |x_i - x_i'|^2$$
  $$= \sum_{i=1}^{L} r^2_i \, d^2_i$$

  squared ED between two symbols in the *i*-th coordinate in difference

Pairwise Error Event



$P(e) = Q(d_E/\mathrm{sqrt}(2N_0))$

Pairwise Error Probability (*the channel* ***a*** *is* known)

- $\Pr(\mathbf{u} \rightarrow \mathbf{u}' | \boldsymbol{a}) = Q(d_E/\mathrm{sqrt}(2N_0))$

$$= Q\left(\sqrt{\frac{\sum_{i=1}^{L} r_i^2 d_i^2}{4N_0}}\right)$$

$$\leq \frac{1}{2} exp\left(-\frac{\sum_{i=1}^{L} r_i^2 d_i^2}{4N_0}\right)$$

$$= \frac{1}{2} \prod_{i=1}^{L} exp\left(-\frac{r_i^2 d_i^2}{4N_0}\right)$$

$Q(\sqrt{x+y}) \leq Q(\sqrt{x})e^{-y/2}$, and $Q(0) = 1/2$

Instantaneous SNR

What only matters is the energy of the fading channel, when the channel is known.

Probability of Pairwise Error

- $\Pr\{\mathbf{u} \rightarrow \mathbf{u}'\} = \int_{\mathbf{r}} \Pr(\mathbf{u} \rightarrow \mathbf{u} | \mathbf{r})\, f(\mathbf{r})\, d\mathbf{r}$

$$= \int_{\mathbf{r}} \frac{1}{2} \prod_{i=1}^{L} exp\left(-\frac{r_i^2 d_i^2}{4N_0}\right) \prod_{j=1}^{L} \frac{r_j}{\gamma^2} exp(-r^2/2\gamma^2) dr_j$$

$$= \frac{1}{2} \prod_{i=1}^{L} \int_{r=0}^{\infty} \frac{r}{\gamma^2} exp\left(-\frac{r^2}{2\gamma^2}\left(1 + \frac{\gamma^2 d_i^2}{2N_0}\right)\right) dr$$

$$= \frac{1}{2} \prod_{i=1}^{L} \left(1 + \frac{\gamma^2 d_i^2}{2N_0}\right)^{-1}$$

$$\approx \frac{1}{2} \prod_{i=1}^{L} \left(\frac{\gamma^2 d_i^2}{2N_0}\right)^{-1} \quad \text{when} \frac{\gamma^2 d_i^2}{2N_0} \gg 1 \quad \text{, i.e. high avg. SNR}$$

For high avg. SNR, P(e) ~ avg.SNR^{-L}
Thus, we want to have a large L

Design Criteria for Trellis Code over Fading Channel

- L is called a effective length. It is the number of symbols that are different in a pair of sequences
- The first design criterion of Trellis Codes for fading channel is the maximization of the minimum effective code length *L* in the trellis
  - P(e) ~ SNR^{-L} asymptotically
- The secondary criterion is to maximize the product of the *L* distance terms
  - Spread the distances as evenly as possible among the *L* locations

Developments of Trellis Codes

- Summary of good techniques is given in
  - Introduction to Trellis-Coded Modulation with Applications (1991) by Ezio Biglieri, D. Divsalar, P. McLane, M.K. Simon.
  - Multi-dimensional (Lattice) Trellis Codes
  - Multiple Trellis Coded Modulation
- Extension of the Trellis Code idea to MIMO channels
  - Space-Time Trellis Codes, by Tarokh, Calderbank, Seshadri, 1998.

Summary

- Trellis codes provide coding gain without sacrificing the transmission rate nor the bandwidth.
- They are called Coded Modulation.
  - Channel coding and modulation are done in a joint manner
  - Code design should be done in a way to increase the Euclidean distance, rather than the Hamming distance, for AWGN case.
  - But for fading channel, it's Hamming distance (minimum free distance) again which is more desirable as diversity becomes more important, rather than Euclidean distance.
- Trellis Codes are very useful for spectrum limited applications such as terrestrial communications, personal area networks, wireless LANs.

HW set for Trellis Codes

- Problem #1: Design a rate-2 8 PSK four trellis-states code without any parallel transitions
  - Find the free ED of your code
  - Compare it with that of the best 4 trellis states code
- Problem #2: Reproduce the Channel Capacity results (Fig.2 in Ungerboeck's paper) for m-ary PSK signals (m=2, 4, 8, 16): (Submit the MATLAB program for this, along with your results)

Computer Simulation Assignment (use MATLAB)

- Refer to Wicker pg. 386 or Fig. 16 of Ungerboeck's paper
- Simulate the uncoded 4 PSK system over the AWGN channel
  - $y_k = x_k + n_k$ where $\{n_k\}$ is complex-valued white Gaussian noise process and x_k is the 4 PSK signals, $\{e^{j\pi/4}, e^{j3\pi/4}, e^{j5\pi/4}, e^{j7\pi/4}\}$.
  - Use a gray mapping such as {(00), (01), (11), (10)} for the four signals
  - Obtain the theoretical bit error probability vs. (E_b/No) SNR curves
  - Obtain bit error rates from simulation and compare them with the theoretical curve (Obtain at least 100 errors; for example for bit error rate of 10^{-3} you should at least generate 100*1000 bits)
- Simulate the 8 state 8 PSK Ungerboeck Trellis Codes for the purpose of generating BER curves, and compare them with the BER curves obtained from the uncoded 4 PSK system
  - Use the Ungerboeck mapping for 8 PSK signals
  - Use the Viterbi algorithm for decoding

LDPC Codes

Review on Linear Code

- *Galois-Field (p),*
  - A set with addition and multiplication operations
  - *Closure, Associativity, Identity, Inverses & Commutativity*
    - *With mult. only non-zero elements*
- Vector space with p^n elements on GF(p)
  - Cardinality of a code with rate R, $|\mathcal{C}| = q^{R\,n}$
- G, generator matrix [nR x n] of a code
  - Row-space of G is space of the code
- H, parity check matrix [n(1-R) x n]
  - n(1-R) rows, [n x 1], of H span the null space of the code
- $GH^T = 0$
- Results in n(1-R) number of linear homogeneous parity check equations.
- $rH^T = (c+e)H^T = eH^T = s$
  - Non-zero s indicates "problem"

Gallager's Thesis ('63)

- (n, j, k) low density parity check code.
- Parity check matrix H [$n(1\text{-}R)$ x n] of the code
  - *j,* number of 1's in each column
  - *k,* number of 1's in each row
  - R = 1 - j/k
- Min. distance of typical (n, j, k) code for $j \geq 3$,
  - increases linearly with n for fixed j & k (Pg. 7; Ch. 2)
- Upper bound on prob. of error for BSC with ML Decoding (Ch. 3)
  - P(e) exponentially-decaying func. of block length n, when $R << 1$
- Practical decoding (Ch. 4)
  - Simple or probabilistic
- Generalization (Ch. 5)

Parity Check Matrix on Bipartite Graph

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_9 \end{bmatrix} = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_6 \end{bmatrix}$$

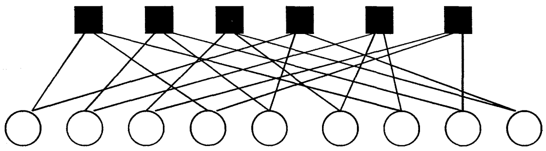
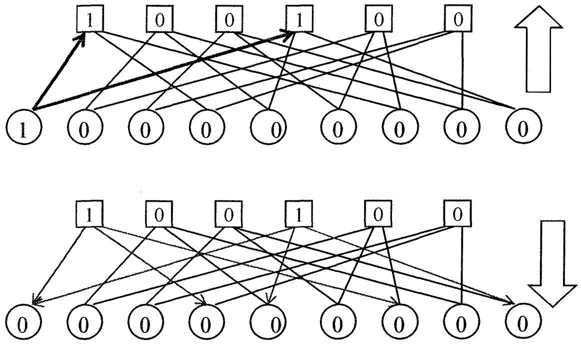


Illustration of Decoding Concept with Simple Hard-Decision Decoder

- Works only for BSC
  - r is binary sequence
- Compute all parity checks
- Change the digit involved in more than a fixed number of unsatisfied parity checks
- Re-compute all parity checks
- Repeat

Simple Decoding Example (j=2, k=3)

- Suppose we have
  r=[1 0 0 0 0 0 0 0 0]
- Violates check nodes (equations) 1 and 4
- Check nodes 1 and 4 send back to error-pattern nodes an instruction to correct
- Error-node 1 corrects it, having two instructions
- Error-nodes 4, 5, 7 & 9 do not correct, since it has only one instruction to correct



Probabilistic Decoding

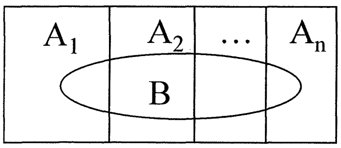
- Bayes' Theorem on The Total Probability
- Total Probability: If A={A_1, A_2, …, A_n} is a partition of S and B is an arbitrary event

$$\Pr\{B\} = \sum^{n}_{i=1}\Pr\{B \cap A_i\} = \sum^{n}_{i=1}\Pr\{B \mid A_i\} \Pr\{A_i\}$$

- Bayes' Theorem: We know

$$Pr\{A_i|B\} = \frac{Pr\{A_i \cap B\}}{Pr\{B\}}$$

- The aposteriori probability is then given by



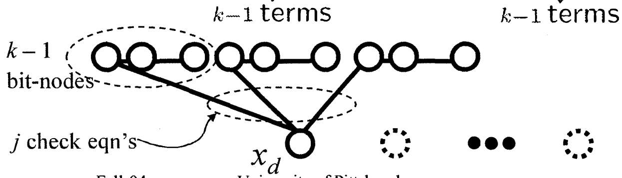
$$Pr(A_i|B) = \frac{Pr(B|A_i)Pr(A_i)}{\sum_{i=1}^{n} Pr(B|A_i)Pr(A_i)}$$

aposteriori

prior

Gallager's Decoding Principle

- A posteriori probability: $Pr(x_d = 1|\mathbf{y}, S)$
  - Event S: All participating check equations are satisfied.
  - Event $\{\mathbf{y}\}$: Observed output of the channel.
- First, let us think about a codeword in a sub-code.
  - A sub-code is a collection of codewords which satisfy all the j parity checks.
  - Each of the j parity check equations involves (k-1) bit nodes.
  - Note a codeword in this sub-code is comprised of j*(k-1)+1 bit nodes.
  - A codeword $\mathbf{c} = (x_d, \underbrace{x^1_{d_1}, x^1_{d_2}, \ldots, x^1_{d_{k-1}}}_{k-1 \text{ terms}}, \ldots, \underbrace{x^j_{d_1}, x^j_{d_2}, \ldots, x^j_{d_{k-1}}}_{k-1 \text{ terms}})$



Gallager's Decoding (2)

- Assumption 1: Digits x_d, x_{d1}, ... are independent
- Assumption 2: $y=(y_d, y_1, y_2, \ldots, y_{j(k-1)})$ and independent channel transition probability

$$P_x(\mathbf{y}) = P_x(y_d)P_x(y_1)\cdots P_x(y_{j(k-1)})$$

Gallager's Lemma 4-1

- Assume m-independent binary digits ($x_1, x_2, \ldots, x_m$)
- Assume ($p_1, p_2, \ldots, p_m$) available, denoting $p_i=\Pr\{x_i = 1\}$
- Then we have $Pr(x_1 \oplus x_2 \oplus \ \ldots \oplus x_m = 1)$

$$= Pr\{\text{even number of 1's}\}$$
$$= \frac{1 + \prod_{l=1}^{m} (1 - 2p_l)}{2}$$

- Hint: consider the two following functions, add/subtract for even/odd number of 1's, then select t = 1

$$\prod_{l=1}^{m} ((1-p_l)\pm p_l\, t) = \prod_{l=1}(1-p_l)\pm\sum_{l=1}^{m} p_l \prod_{j\neq l}^{m}(1-p_j)t + \cdots \pm \sum_{l=1}^{m}(1-p_l)\prod_{j\neq l} p_j + \prod_{l=1}^{m} p_l$$

Ex) m even

Gallager's Decoding Theorem 4.1

- Bayes Theorem:

$$\begin{aligned} Pr(x_d = 1|\mathrm{y}, S) &= \frac{Pr(x_d = 1, \mathrm{y}, S)}{p(\mathrm{y}, S)} \\ &= \frac{Pr(S|x_d = 1, \mathrm{y})\ p(x_d = 1, \mathrm{y})}{p(\mathrm{y}, S)} \\ &= \frac{Pr(S|x_d = 1, \mathrm{y})\ Pr(x_d = 1|\mathrm{y})\ p(\mathrm{y})}{p(\mathrm{y}, S)} \end{aligned}$$

- The ratio is of our interest

$$\frac{Pr(x_d = 0|\mathrm{y}, S)}{Pr(x_d = 1|\mathrm{y}, S)} = \frac{Pr(S|x_d = 0, \mathrm{y})\ Pr(x_d = 0|\mathrm{y})}{Pr(S|x_d = 1, \mathrm{y})\ Pr(x_d = 1|\mathrm{y})}$$

Gallagar's Decoding

- Find the probability that the *first* parity check equation is satisfied, given $x_d = 0$ and the channel output y,

$$
\begin{aligned}
&Pr(0 \oplus x^1_{d_1} \oplus x^1_{d_2} \oplus \ \dots \oplus x^1_{d_{k-1}} = 0 \,|x_d = 0, \mathrm{y}) \\
&= Pr(\text{even 1's in the rest } (k-1) \text{ bit-nodes} \,|\, \mathrm{y}) \\
&= Pr(\text{ all zero } |\, \mathrm{y}) + Pr(\text{ two 1's } |\, \mathrm{y}) + \cdots \\
&= Pr(x^1_{d_1} = 0, x^1_{d_2} = 0, \dots, x^1_{d_{k-1}} = 0 \,|\, \mathrm{y}) + \\
&\quad Pr(x^1_{d_1} = 1, x^1_{d_2} = 1, \dots, x^1_{d_{k-1}} = 0 \,|\, \mathrm{y}) + \cdots \\
&= \prod_{l=1}^{k-1} (1 - Pr(x^1_{d_l} = 1 \,|\, \mathrm{y})) + \\
&\quad \prod_{l=1}^{2} Pr(x^1_{d_l} = 1 \,|\, \mathrm{y}) \prod_{l=3}^{k-1} (1 - \overbrace{Pr(x^1_{d_l} = 1 \,|\mathrm{y})}^{p_{1l} :=}) + \cdots \\
&= \frac{1 + \prod_{l=1}^{k-1}(1 - 2p_{1l})}{2}
\end{aligned}
$$

assuming independence

Gallager's Decoding Theorem 4.1

- Now, find the probability that the *first* parity check equation is satisfied, given $x_d = 1$ and the channel output y,

$$
\begin{aligned}
&Pr(1 \oplus x^1_{d_1} \oplus x^1_{d_2} \oplus \ \dots \oplus x^1_{d_{k-1}} = 0 \,|x_d = 1, \mathrm{y}) \\
&= Pr(\text{odd 1's in the rest } (k-1) \text{ bit-nodes} \,|\, \mathrm{y}) \\
&= Pr(\text{ single 1 } |\, \mathrm{y}) + Pr(\text{ three 1's } |\, \mathrm{y}) + \cdots \\
&= Pr(x^1_{d_1} = 1, x^1_{d_2} = 0, \dots, x^1_{d_{k-1}} = 0 \,|\, \mathrm{y}) + \cdots \\
&\quad Pr(x^1_{d_1} = 1, x^1_{d_2} = 1, x^1_{d_3} = 1, \dots, x^1_{d_{k-1}} = 0 \,|\, \mathrm{y}) + \cdots \\
&= Pr(x^1_{d_1} = 1 |\, \mathrm{y}) \prod_{l=2}^{k-1} (1 - Pr(x^1_{d_l} = 1 \,|\, \mathrm{y})) + \cdots \\
&\quad \prod_{l=1}^{3} Pr(x^1_{d_l} = 1 \,|\, \mathrm{y}) \prod_{l=4}^{k-1} (1 - \overbrace{Pr(x^1_{d_l} = 1 \,|\mathrm{y})}^{p_{1l} :=}) + \cdots \\
&= \frac{1 - \prod_{l=1}^{k-1}(1 - 2p_{1l})}{2}
\end{aligned}
$$

Again, assuming independence

Showing only a single term of the same kind

Note the negative sign

Gallager's Decoding Theorem 4.1

- Let S_i : = {the *i*-th check is satisfied}
- Then, $S = S_1$ and S_2 and … and S_j
- Thus, we have

$$\begin{aligned} Pr(S|x_d = 0, \mathrm{y}) &= \prod_{i=1}^{j} Pr(S_i|x_d = 0, \mathrm{y}) \\ &= \prod_{i=1}^{j} \frac{1 + \prod_{l=1}^{k-1}(1 - 2p_{il})}{2} \end{aligned}$$

- Similarly, we have

$$Pr(S|x_d = 1, \mathrm{y}) = \prod_{i=1}^{j} \frac{1 - \prod_{l=1}^{k-1}(1 - 2p_{il})}{2}$$

The Decoding Theorem 4.1

- Now summarizing the decoding theorem, we have the equation (4.1)

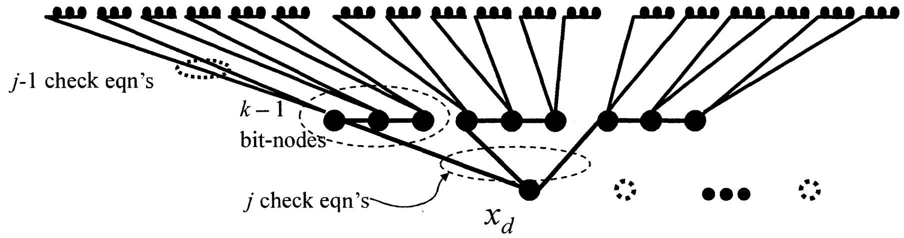
$$\begin{aligned} \frac{Pr(x_d = 0|\mathrm{y}, S)}{Pr(x_d = 1|\mathrm{y}, S)} &= \frac{1 - Pr(x_d = 1|\ \mathrm{y})}{\underbrace{Pr(x_d = 1|\ \mathrm{y})}_{p_d :=}} \prod_{i=1}^{j} \frac{1 + \prod_{l=1}^{k-1}(1 - 2p_{il})}{1 - \prod_{l=1}^{k-1}(1 - 2p_{il})} \\ &= \frac{1 - p_d}{p_d} \prod_{i=1}^{j} \frac{1 + \prod_{l=1}^{k-1}(1 - 2p_{il})}{1 - \prod_{l=1}^{k-1}(1 - 2p_{il})} \end{aligned}$$

- Note p_d and p_{il}, i=1, 2, …, j, l = 1, 2, …, *k*-1 are posterior probabilities of having digit "1" at the particular location given the complete output y

$$\begin{aligned} p_d &:= Pr(x_d = 1 \mid \mathrm{y}) \\ p_{il} &:= Pr(x^i_{d_l} = 1 \mid \mathrm{y}) \end{aligned}$$

Iterative Decoding

- We can consider the sub-code scenario for each of the $j^*(k\text{-}1)$ first tier bit nodes
- Now, we have (j - 1) check equations for each bit node (why?)
  - Each check equation checks (k – 1) bit nodes



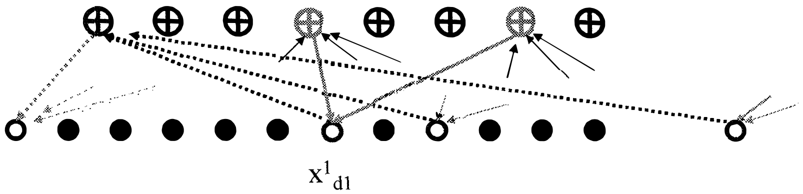
Iterative Decoding (2)

- Second or higher tier probability calculation

$$\frac{Pr(x_d^1=0|\mathrm{y},S)}{Pr(x_{d_1}^1=1|\mathrm{y},S)} = \frac{1-Pr(x_{d_1}^1=1|\ \mathrm{y})}{\underbrace{Pr(x_{d_1}^1=1|\ \mathrm{y})}_{p_{11}:=}} \prod_{i=2}^{j} \frac{1+\prod_{l=1}^{k-1}(1-2p_{il})}{1-\prod_{l=1}^{k-1}(1-2p_{il})}$$

$$= \frac{1-p_{11}}{p_{11}} \prod_{i=2}^{j} \frac{1+\prod_{l=1}^{k-1}(1-2p_{il})}{1-\prod_{l=1}^{k-1}(1-2p_{il})}$$

j - 1 terms



Iterative Decoding (3)

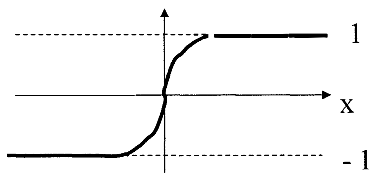
- Assume the probability calculation is started off from the last tier of the tree, and coming down toward the first tier of the tree
- After a number of second tier calculations — (j-1) check equations and (k-1) bit nodes for each check — we assume we are close to the origin of the tree, and make the first tier calculation — j check equations with (k-1) bit nodes in each (given by Theorem 4.1)

The Start of Iteration

- $y_k = \sqrt{E_s}(2x_k-1) + n_k$, where n_k is AWGN for k=1, 2, …, N
- Obtain the likelihood probability $p(y_k \mid x_k)$.
- This log likelihood probability is used to start the iteration.

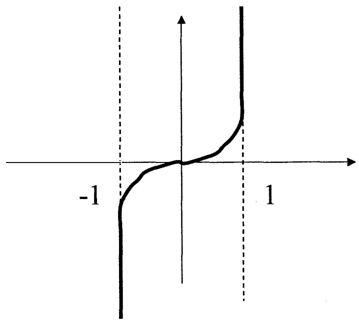
tanh and tanh^{-1} function

- Note $tanh(\frac{x}{2}) = \frac{e^x-1}{e^x+1}$



- Note

$$\begin{aligned} tanh^{-1}(x) &= \frac{1}{2}log\frac{1+x}{1-x} \\ &= sign(x)\frac{1}{2}log\frac{1+|x|}{1-|x|} \end{aligned}$$



Hyperbolic Tangent, tanh

- $$\begin{aligned} (1-2p_i) &= \frac{1-p_i-p_i}{1-p_i+p_i} \\ &= \frac{1-\frac{p_i}{1-p_i}}{1+\frac{p_i}{1-p_i}} \\ &= \frac{1-e^{log\frac{p_i}{1-p_i}}}{1+e^{log\frac{p_i}{1-p_i}}} \\ &= -tanh(\frac{LR(p_i)}{2}) \end{aligned}$$

where we defined $LR(p_i) := log\frac{p_i}{1-p_i}$

Log Ratio Algorithm

- Take the log of the ratio of the posteriors

$$log\frac{Pr(x_d = 1|\mathrm{y}, S)}{Pr(x_d = 0|\mathrm{y}, S)} = log\frac{p_d}{1-p_d} + \sum_{i=1}^{j} log\frac{1-\prod_{l=1}^{k-1}(1-2p_{il})}{1+\prod_{l=1}^{k-1}(1-2p_{il})}$$

- Using $tanh(\frac{x}{2}) = \frac{e^x-1}{e^x+1}$, the summand of the second term is

$$log\frac{1-(-1)^{k-1}\prod_{l=1}^{k-1} tanh(\frac{LR(p_{il})}{2})}{1+(-1)^{k-1}\prod_{l=1}^{k-1} tanh(\frac{LR(p_{il})}{2})} = log\frac{1+(-1)^{k}\prod_{l=1}^{k-1} tanh(\frac{LR(p_{il})}{2})}{1-(-1)^{k}\prod_{l=1}^{k-1} tanh(\frac{LR(p_{il})}{2})}$$

- Making use of $tanh^{-1}(x) = \frac{1}{2}log\frac{1+x}{1-x}$, it becomes

$$\sum_{i=1}^{j} 2\,tanh^{-1}((-1)^k \prod_{l=1}^{k-1} tanh(\frac{LR(p_{il})}{2}))$$

$$= \sum_{i=1}^{j} (-1)^k\, 2\,tanh^{-1}(\prod_{l=1}^{k-1} tanh(\frac{LR(p_{il})}{2}))$$

Making use of $tanh^{-1}$ being odd function

Product of Real Numbers

- $\prod_i \alpha_i = [\prod_i sign(\alpha_i)] \cdot exp(\sum_i log(|\alpha_i|))$

- $a\ b$ = sign(a) sign(b) exp(log($|a|$)) exp(log(|b|))

- $\prod_{l=1}^{k-1} tanh(\frac{\alpha(p_{il})}{2})$

$$= [\prod_{l=1}^{k-1} sign(LR(p_{il}))] \cdot exp(\sum_{l=1}^{k-1} log(tanh(\frac{|LR(p_{il})|}{2}))$$

$$f(x) := -\log(\tanh(x/2)) = log\frac{e^x+1}{e^x-1}$$

- Use the identity of product of real numbers to get rid of product

$$\sum_{i=1}^{j} \quad (-1)^k\, 2\; tanh^{-1}\Big(\prod_{l=1}^{k-1} tanh\Big(\frac{LR(p_{il})}{2}\Big)\Big)$$

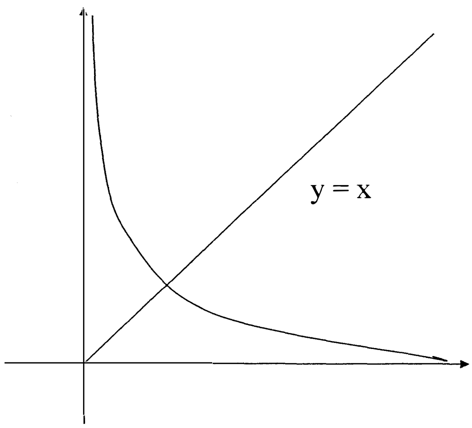
$$= \sum_{i=1}^{j}(-1)^k\Big[\prod_{l=1}^{k-1} sign(LR(p_{il}))\Big]\, 2\; tanh^{-1}\Big[exp\Big(\sum_{l=1}^{k-1} log\Big(tanh\Big(\frac{|LR(p_{il})|}{2}\Big)\Big)\Big)\Big]$$

$$= \sum_{i=1}^{j}\underbrace{\Big[\prod_{l=1}^{k-1} sign(LR(p_{il}))\Big]\cdot f^{-1}\Big(\sum_{l=1}^{k-1} f(|LR(p_{il})|)\Big)}_{\text{Information generated by the i-th check node}}$$

Log ratio: info. from bit nodes

$$f^{-1}(x) = f(x)$$

- Symmetric wrt y=x

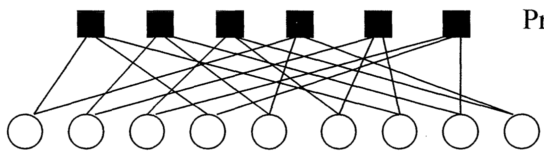


Finally, the Log Ratio Algorithm

- Note the ratio here is Pr(x=1)/Pr(x=0), which is the inverse of the ratio used in Gallagar's thesis
- With the following definitions
  - $LR(p_d) := log\frac{p_d}{1-p_d} \quad LR(p_{il}) := log\frac{p_{il}}{1-p_{il}}$
  - $LR(p'_d) := log\frac{Pr(x_d=1|S,\mathrm{y})}{Pr(x_d=0|S,\mathrm{y})}$
- Theorem 4.1 becomes

$$LR(p'_d) := LR(p_d)+\sum_{i=1}^{j}[\prod_{l=1}^{k-1} sign(LR(p_{il}))]f[\sum_{l=1}^{k-1} f(|LR(p_{il})|)]$$

Number of Edges in Bipartite Graph

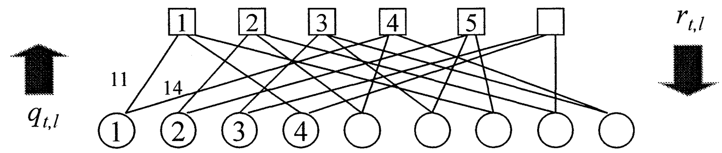


Pr(S_i|x_d=1,y)

Pr(x_d=1|S,y)

- There are n bit nodes and L check equations
  - Then there are $E=n*j = L*k$ edges
  - The total number of messages flowing from bit to check, and also from check to bit, is E

Let's Label the Edges



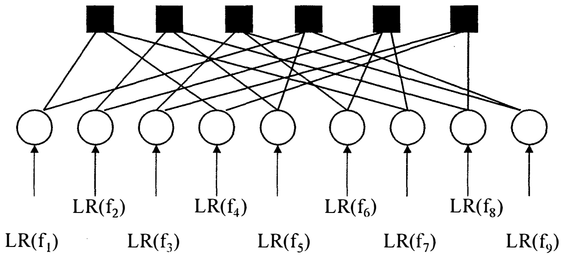
- Starting from the left of the graph
  - For the edges connecting bit-1 to check-1 and check-4, let's name them to be '11' and '14' respectively
  - For the edges connecting bit-2 to check-2 and check-5, let's name them to be '22' and '25' respectively
  - And so on
- Now, let's define $q_{t,l}$ the message from the bit-t to check-l
- And, define $r_{t,l}$ the message from the check-l to bit-t

Likelihoods as Input

- $y_t = (2x_t - 1) + n_t$, where n_t is AWGN for t=1, 2, …, n where n_t is $\mathcal{N}(0, N_0/(2E_s))$ with $E_s=E_b*R$
- Obtain the likelihood function $p(y_t|x_t)$
- The log likelihoods are used to start the iteration
- Let's denote the likelihood functions
  - $f_t(1)=p(y_t|x_t=1)$ and $f_t(0)=p(y_t|x_t=0)$
- The Log Ratio of Likelihood Probability is

$$LR(f_t)=\log(f_t(1)/f_t(0))=(4E_s/N_0)y_t$$

In the beginning, we have LLR(f_k)

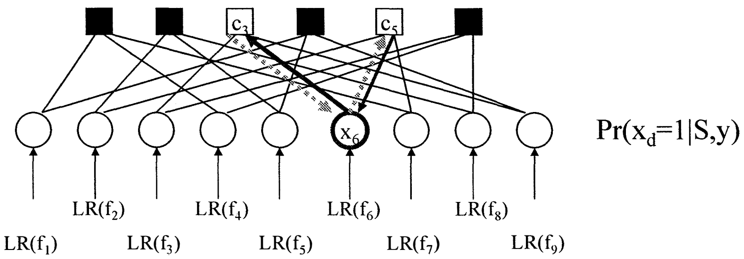


- The log likelihood ratios (or $\{f_t(1)\}$) are the input to the message passing decoder, to start the iterative decoding

Until the last iteration

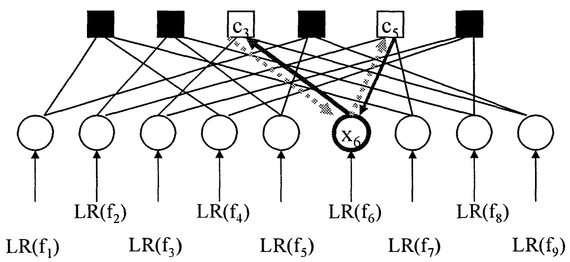
- Do the second tier calculation
- Each bit node generates *j* bit-to-check messages
  - Each bit-to-check message is generated by checking *j*-1 check equations, excluding the check equation to which the message flows
- Each check node generates *k* check-to-bit messages
  - Each check-to-bit message is generated by utilizing *k*-1 posteriors, excluding the edge connecting to the bit node to which the message flows

Bit-to-Check Message $q_{t,l}$



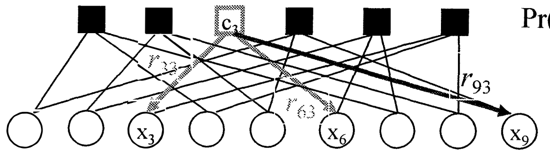
- The message to check node c_3,
  - $q_{63}(1)$:=$\Pr(x_6=1|S_5,y) = \Pr(x_6=1|y)\Pr(S_5|x_6=1,y) = f_6(1)\, r_{65}(1)$
- The message to check node c_5,
  - $q_{65}(1)$:=$\Pr(x_6=1|S_3,y) = \Pr(x_6=1|y)\Pr(S_3|x_6=1,y) = f_6(1)\, r_{63}(1)$
- Similarly for $q_{63}(0)$ and $q_{65}(0)$
- In terms of log ratio, $LR(q_{63}) = LR(f_6)+LR(r_{65})$ and $LR(q_{65}) = LR(f_6)+LR(r_{63})$

Log Ratio Bit-to-Check Messages $LR(q_{t,l})$



- $LR(q_{63}) = LR(f_6)+LR(r_{65})$
- $LR(q_{65}) = LR(f_6)+LR(r_{63})$

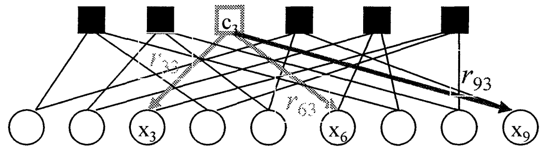
Check-to-Bit $r_{t,l}$



Pr(S_3 |x_d=1,y)

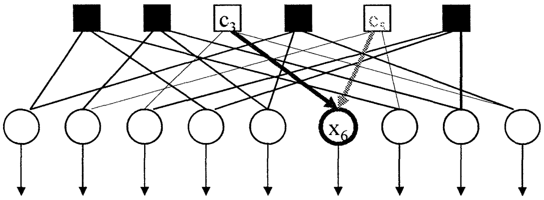
- Check-to-Bit message is $r_{d3}(1)$:=Pr(S_3 |x_d=1,y), d=3, 6, 9
- The message to bit node x_3 is $r_{33}(1) = \frac{1-(1-2q_{63}(1))(1-2q_{93}(1))}{2}$
- The message to bit node x_6 is $r_{63}(1) = \frac{1-(1-2q_{33}(1))(1-2q_{93}(1))}{2}$
- The message to bit node x_9 is $r_{93}(1) = \frac{1-(1-2q_{33}(1))(1-2p_{63}(1))}{2}$
- Similarly for $r_{d3}(0)$:= Pr(S_3 |x_d=0,y), d=3, 6, 9
  - For example, $r_{33}(0) = \frac{1+(1-2q_{63}(0))(1-2q_{93}(0))}{2}$

Log Ratio Check-to-Bit Messages LR($r_{t,l}$)



- g(x) := tanh(-x/2)
- The check-to-bit messages in log ratio
  - $LR(r_{33}) = g^{-1}[g(LR(q_{63})) \cdot g(LR(q_{93}))]$
  - $LR(r_{63}) = g^{-1}[g(LR(q_{33})) \cdot g(LR(q_{93}))]$
  - $LR(r_{93}) = g^{-1}[g(LR(q_{33})) \cdot g(LR(q_{93}))]$

At the last iteration



- Do the first tier calculation (Theorem 4.1)
  $\Pr(x_6=1|S_3,S_5,y) = \Pr(x_6=1|y)\,\Pr(S_3\,|x_6=1,y)\,\Pr(S_5\,|x_6=1,y)$
  - $p_6(1) = f_6(1)\,r_{63}(1)\,r_{65}(1)$
  - $p_6(0) = f_6(0)\,r_{63}(0)\,r_{65}(0)$
- $LR(p_6) = LR(f_6) + LR(r_{63}) + LR(r_{65})$

Parity Check Matrix on Bipartite Graph
(n=9, j=2, k=3) example

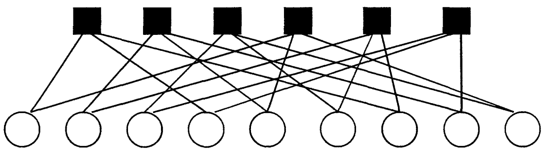
$$\begin{array}{c} \xrightarrow{\quad t \quad} \\ l \downarrow \begin{bmatrix} 1&0&0&1&0&0&1&0&0\\ 0&1&0&0&1&0&0&1&0\\ 0&0&1&0&0&1&0&0&1\\ 1&0&0&0&1&0&0&0&1\\ 0&1&0&0&0&1&1&0&0\\ 0&0&1&1&0&0&0&1&0 \end{bmatrix} \end{array} \begin{bmatrix} x_1\\ x_2\\ \vdots\\ x_9 \end{bmatrix} = \begin{bmatrix} s_1\\ s_2\\ \vdots\\ s_6 \end{bmatrix}$$

Q1(m,t): Row in a column

t	1	2	3	4	5	6	7	8	9
m=1	1	2	3	1	2	3	1	2	3
m=2	4	5	6	6	4	5	5	6	4

$\}\ j$

L the number of checks



Q2(m, l): Column in a row

l	1	2	3	4	5	6
m=1	1	2	3	1	2	3
m=2	4	5	6	5	6	4
m=3	7	8	9	9	7	8

$\}\ k$

Summary of Decoding (Log Ratio)
(n, j, k) code with $g(x) := \tanh(-x/2)$

- Initialize:
  - $LR(f_t)=(4E_s/N_0)y_t$,
  - $LR(r_{tl})=0$, $t=1, 2, \ldots, n$ and $l=1,2,\ldots, k$
- Iteration:
  - Bit-to-Check messages: $LR(q_{t,Q1(m,t)})$, $t=1,2,\ldots, n$; $m=1,2, \ldots, j$

    $LR(q_{t,Q1(m,t)}) = LR(f_t) + \sum_{m' \neq m} LR(r_{t,Q1(m',t)})$
  - Check-to-Bit messages: $LR(r_{Q2(m,l),l})$, $l=1,2,\ldots,L$; $m=1,2,\ldots, k$

    $LR(r_{Q2(m,l),l}) = g^{-1}[\prod_{m' \neq m} g(LR(q_{Q2(m',l),l}))]$
- Output:
  - $LR(p_t) = LR(f_t) + \sum_m LR(r_{t,Q1(m,t)})$
- Decision:
  - *if* $LR(p_t) > 0$ $x_t = 1$; *else* $x_t = 0$

Summary of Decoding (Log Ratio)
(n, j, k) code with $f(x) := -\log(\tanh(x/2)) = \log[(e^x + 1)/(e^x - 1)]$

- Initialize:
  - $LR(f_t)=(4E_s/N_0)y_t$,
  - $LR(r_{tl})=0$, $t=1, 2, \ldots, n$ and $l=1,2,\ldots, k$
- Iteration:
  - Bit-to-Check messages: $LR(q_{t,Q1(m,t)})$, $t=1,2,\ldots, n$; $m=1,2, \ldots, j$

    $LR(q_{t,Q1(m,t)}) = LR(f_t) + \sum_{m' \neq m} LR(r_{t,Q1(m',t)})$
  - Check-to-Bit messages: $LR(r_{Q2(m,l),l})$, $l=1,2,\ldots,L$; $m=1,2,\ldots, k$

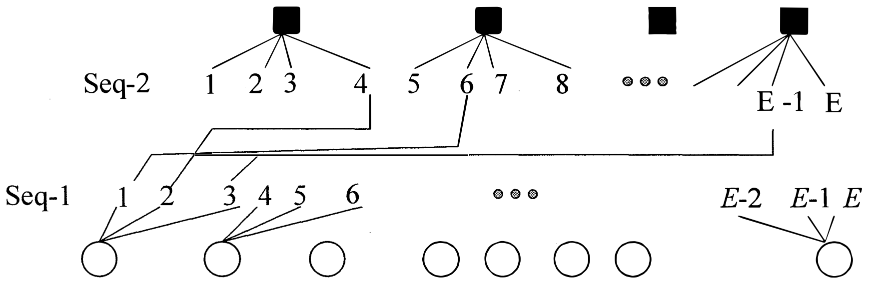
    $LR(r_{Q2(m,l),l}) = \prod_{m' \neq m} \mathrm{sgn}(LR(q_{Q2(m',l),l})) \times f[\sum_{m' \neq m} f(|LR(q_{Q2(m',l),l})|)]\ (-1)^k$
- Output:
  - $LR(p_t) = LR(f_t) + \sum_m LR(r_{t,Q1(m,t)})$
- Decision:
  - *if* $LR(p_t) > 0$ $x_t = 1$; *else* $x_t = 0$

Generation of the **H** matrix

- There are numerous ways to generate the H matrix
- Any solution that satisfies the two constraints will do the job (Some bad choice will increase the code rate)

1. *j* number of ones in each column
2. *k* number of ones in each row

Using s-random interleaver to generate the parity matrix H: (n, j=3, k=4) example



- We know there are $E=n*j = L*k$ edges
  - Sequence each set of edges
  - Find a random interleaved (e.g. s-random interleaver: $s>k$) sequence π
  - π(Seq-1) = (6 4 E-1 …)
  - Make the connections

Generate the **H** matrix directly

- Randomly select the *j* row positions to place one in each column, while making sure that the number of ones in that particular row is not greater than *k*
- Equivalently, we can construct the Q1(m,t) matrix or Q2(m,l) matrix

Example with (n=9, j=2, k=3)

- Let's generate Q1(m,t) matrix
  - Rows={1,2,3,4,5,6}
- At t=1, randomly select two numbers from Rows
  - Suppose the two were 2, 5
- At t=2, select two numbers again from Rows
  - Suppose they were 2, 4
- At t=3, select two numbers again from Rows
  - Suppose it were 2, 3
  - Remove '2' from Rows
- At t=4, select two numbers from Rows-{2}
- And so on

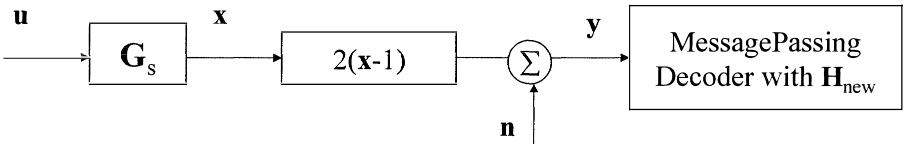
Another Way

- Refer to Gallagar's Thesis pg. 13 or his paper

Gaussian Elimination on **H**

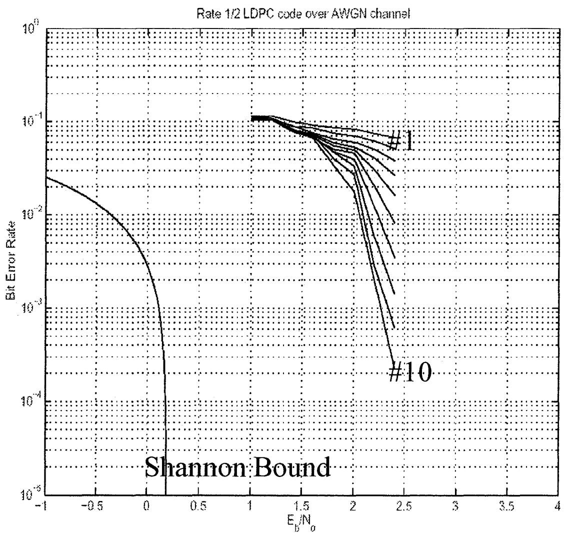
- Once **H** is found, perform Gaussian Elimination on **H** and find a systematic form of **H**, $\mathbf{H}_s$
  - Keep track of the column exchanges made
- Do the same set of column exchanges on **H** and obtain new **H** matrix $\mathbf{H}_{new}$
- Find $\mathbf{G}_s$ using the relationship $\mathbf{G}_s\mathbf{H}_s^T = \mathbf{0}$
- Use $\mathbf{G}_s$ in encoding the data
- Design the message passing decoder according to $\mathbf{H}_{new}$ (Why?)
  - $\mathbf{G}_s\mathbf{H}_{new}^T = \mathbf{0}$

Finally, we have



- Noise is AWGN with mean 0 and variance $N_0/(2E_s)$

Results of the rate ½ LDPC code, N=4096



Try to obtain BER=10^{-5}

Turbo Codes

1

Agenda

- The Turbo Codes
- Forward-Backward Algorithm (BCJR algorithm)
- Soft Input Soft Output (SISO) Module
- Log Domain Algorithm
  - The Max Operation

2

Papers for turbo codes for class notes

- Bahl, Cocke, Jelinek, and Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," IT, Mar. 1974.
- Berrou, Glavieux, and Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo codes," ICC, 1993.
- Hagenauer, "Iterative decoding of binary block and convolutional codes," IT, Mar.1996.
- Benedetto and Montorsi, "Unveiling turbo codes: some results on parallel concatenated coding schemes," IT, Mar. 1996.
- Benedetto and Montorsi, "Design of parallel concatenated convolutional codes," TC, May 1996.
- Benedetto, Divsalar, Montorsi and Pollara, "Serial concatenation of interleaved codes: performance analysis, design, and iterative decoding," IT, May, 1998.

Papers for Gallager codes for class

- Gallager's Thesis
- Richardson and Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," IT, Feb. 2001.
- Richardson, Shokrollahi, and Urbanke, "Design of capacity approaching LDPC codes," IT, Feb. 2001.
- S.Y. Chung, T.J. Richardson, and R. Urbanke, "Analysis of Sum-Product Decoding of Low-Density Parity-Check codes using a Gaussian Approximation," IEEE Trans. on IT, Feb. 2001.
  - Communication Letter: LDPC code 0.0045 dB of the Shannon limit.
- Brink, S. Ten, "Convergence of iterative decoding," Electronics Letters, 1999.

Term Project Idea

- Simulation
  - A turbo transceiver system over AWGN channels or (MIMO) fading channels.
  - Code design using the EXIT chart or Density Evolution.
  - Papers helpful for this
    - Brink, Kramer, Ashikmin, “Design of low-density parity-check codes for modualtion and detection,” TC, April, 2004.
- Some papers that are interesting for a summary presentation:
  - Chen, Xu, Djurdjevic and Lin, “Near-Shannon-Limit Quasi-Cyclic LDPC codes,” TC, July, 2004.
  - Djurdjevic, Xu, Abdel-Ghaffar, Lin, “A class of LDPC codes constructed based on Reed-Solomon codes with two information symbols,” CL, July, 2003.
  - Soft decision decoding of Reed-Solomon codes
    - Guruswami and Sudan, “Improved decoding of Reed-Solomon and Algebraic-Geometry codes,” IT, Sept., 1999.
    - Koetter’s 2003 paper (Late UIUC prof)
    - Narayanan (Texas A&M)

The Original Turbo Codes

- Berrou, Glavieux, and Thitimajshima, “Near Shannon limit error correcting coding and decoding: Turbo Codes,” ICC 1993.

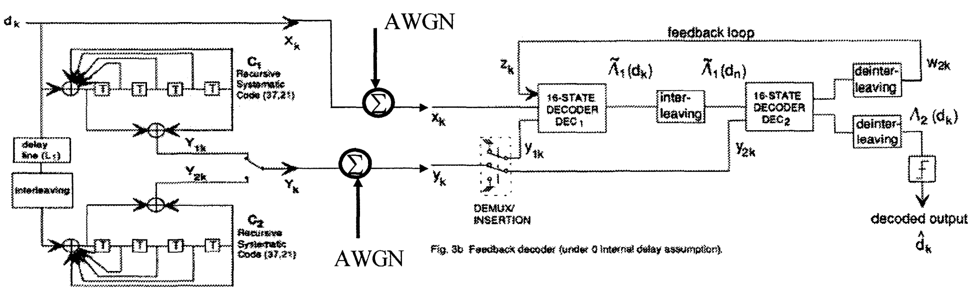


Fig. 2 Recursive Systematic codes with parallel concatenation.

Fig. 3b Feedback decoder (under 0 internal delay assumption).

The notation given here will not be used further in my lecture notes

New Ingredients

- New encoder –randomness reinforced with the use of random interleaver
  - Two state machines combined with a random interleaver
  - The use of random interleaver dramatically increases the *effective constraint length* of the overall code
- New decoder – iterative decoding by exchanging soft-metric among a number of state machines
  - Local optimization using a Maximum A Posteriori (MAP) algorithm
  - Exchange the local optimization results among constituent decoders across de-interleavers—iterative decoding

Dramatic Performance Results

- The interleaver size was 65536.
- Shannon Limit on rate ½ code is 0.187 dB (0 dB if unconstrained).
- The BER curves obtained show $P_b = 10^{-5}$ was achieved at 0.7 dB, which is only about 0.513 dB away from the limit.

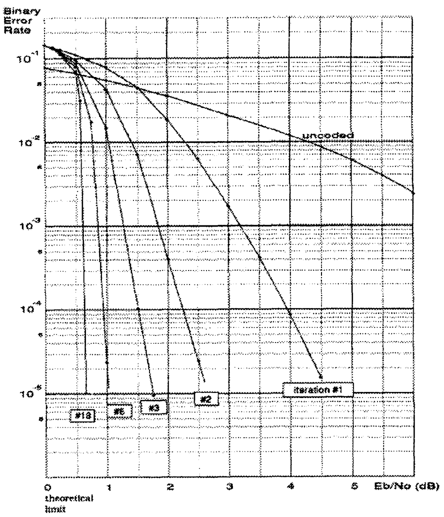
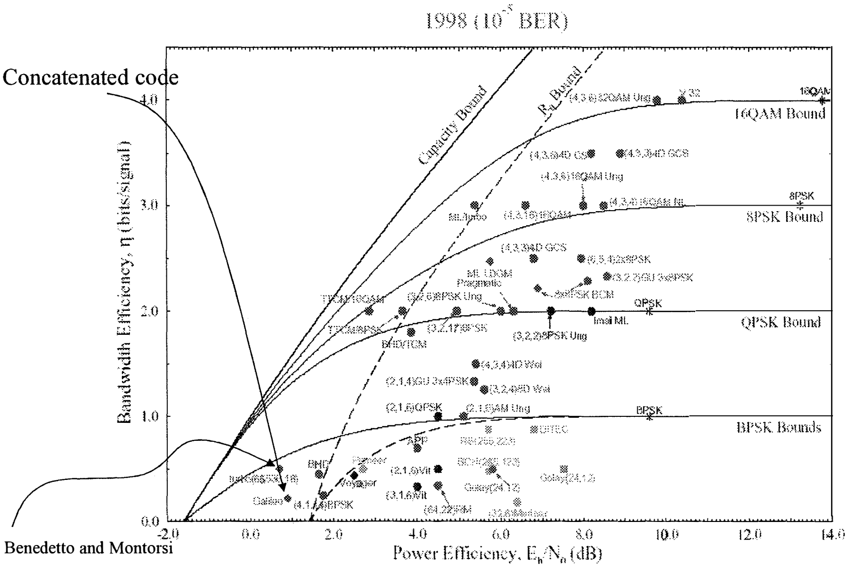


Fig. 5 Binary error rate given by iterative decoding (p=1,....18) of code of fig. 2 (rate:1/2); interleaving 256x256.

Coding Achievements [Costello'98]



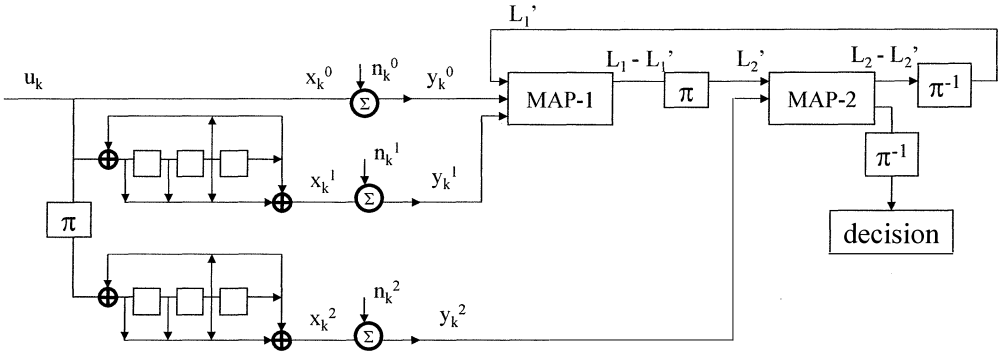
The Turbo Decoder

- Takes the likelihood probability, on input being +1 or –1, sequence from the sequence of received signals.
- Generates *a posteriori* probability, on input being +1 or –1, at each of the decoder assigned to a particular constituent encoder.
- Note the following

  Pr{X=+1|Y} = Pr{X=+1,Y}/p{Y} = p{Y|X=+1}Pr{X=+1}/p{Y}
- Thus, the posterior can be generated by the product of likelihood and the prior.
- Get the priors from the other constituent decoders.

  (This is the Basic Idea)

Rate 1/3 Turbo Code Example



The Turbo Decoder (2)

- Now denote the following vector variables
  - $\mathbf{y^0}$: the sequence (block) received for uncoded transmission
  - $\mathbf{y^1}$: the received sequence for the first constituent conv. encoder
  - $\mathbf{y^2}$: the received sequence for the second one
- Note that all of the three sequence contains useful information about the transmitted information sequence **u**
  - $\mathbf{y^0} = (2\mathbf{u} - \mathbf{1}) + \mathbf{n^0}$
  - $\mathbf{y^1} = 2f(\mathbf{u})-1 + \mathbf{n^1}$
  - $\mathbf{y^0} = 2f(\pi\mathbf{u})-1 + \mathbf{n^2}$ where f(*) denotes the encoding operation of the recursive convolutional encoder on input sequence **u** or the interleaved $\pi\mathbf{u}$ (one-to-one mapping)
- It is trivial to generate a soft metric on each input bit u_k from the first equation (i.e. the likelihood probability). But how about from the second and the third equations? It is not trivial because of encoding operation.

Maximum A Posteriori Prob. On a Bit

- We are interested in calculating $\Pr\{u_k = 1|\mathbf{y}\}$ or $\Pr\{u_k = 0|\,\mathbf{y}\}$, and choosing the bit which gives a bigger measure
  - It is the MAP criterion based on the entire observation sequence **y.**
  - Note that the maximum likelihood sequence detection (VA) is also based upon the entire sequence.

MAP vs. MLSD

- Note the following relationship bet MAP and MLSD:

$$\Pr(\mathbf{u} \mid \mathbf{y}) = p(\mathbf{y}, \mathbf{u})/p(\mathbf{y})$$
$$= p(\mathbf{y} \mid \mathbf{u})\Pr(\mathbf{u})/p(\mathbf{y})$$

- Usually we don't need to consider p(**y**) because it's the same for all competing candidates.
- Thus, we have $\Pr(\mathbf{u} \mid \mathbf{y}) \propto p(\mathbf{y} \mid \mathbf{u}) \Pr(\mathbf{u})$.
  - When Pr(**u**) equally likely, no prior information.
- In addition, if we know the entire sequence **u**, we know the encoded sequence **x**. Thus, we have

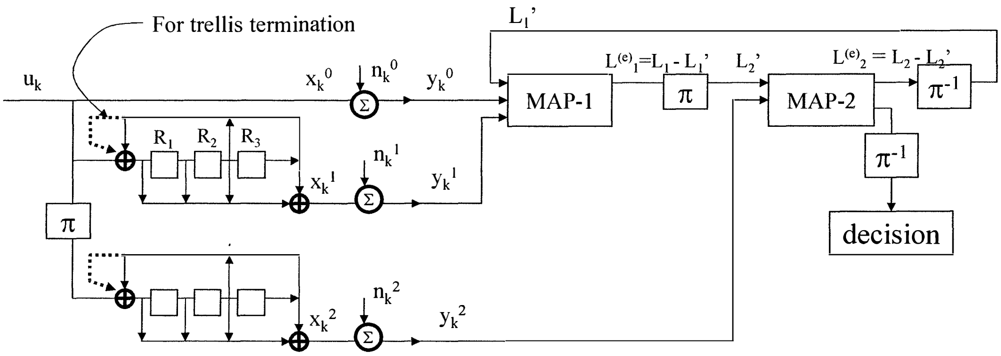
$$\Pr(\mathbf{u} \mid \mathbf{y}) = \Pr(\mathbf{x} \mid \mathbf{y}) \propto p(\mathbf{y} \mid \mathbf{x})\Pr(\mathbf{u})$$

Sum Product Algorithm

- $\Pr(\mathbf{u} \mid \mathbf{y}) = \Pr(\mathbf{x} \mid \mathbf{y}) \propto p(\mathbf{y} \mid \mathbf{x})\Pr(\mathbf{u})$
- Now consider $\Pr(u_k=1 \mid \mathbf{y})$

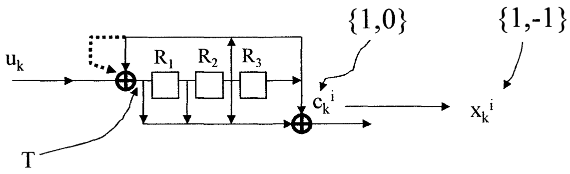
$$
\begin{aligned}
Pr(u_k = 1|\mathbf{y}) &\propto \sum_{\mathbf{x}:u_k=1} Pr(\mathbf{x}|\mathbf{y}) \\
&\propto \sum_{\mathbf{x}:u_k=1} p(\mathbf{y}|\mathbf{x})Pr(\mathbf{u}) \\
&\propto \sum_{\mathbf{x}:u_k=1} \prod_j p(y_j|x_j)Pr(u_j)
\end{aligned}
$$

Consider our 1/3 turbo code example



- First construct forward and backward progression tables
- Second draw the trellis
- Apply the BCJR algorithm on the trellis

Encoder Operation



- $T = u_k + R_2 + R_3 \bmod 2$
- $x_k = T + R_1 + R_2 + R_3$

The Encoder Operation Table

u_k	R_1	R_2	R_3	T $R_2 \oplus R_3 \oplus u_k$	next state (T, R_1, R_2)		parity bit $T \oplus R_1 \oplus R_2 \oplus R_3$
0	0	0	0	0	(0 0 0)	0	0
0	0	0	1	1	(1 0 0)	4	0
0	0	1	0	1	(1 0 1)	5	0
0	0	1	1	0	(0 0 1)	1	0
0	1	0	0	0	(0 1 0)	2	1
0	1	0	1	1	(1 1 0)	6	1
0	1	1	0	1	(1 1 1)	7	1
0	1	1	1	0	(0 1 1)	3	1
1	0	0	0	1	(1 0 0)	4	1
1	0	0	1	0	(0 0 0)	0	1
1	0	1	0	0	(0 0 1)	1	1
1	0	1	1	1	(1 0 1)	5	1
1	1	0	0	1	(1 1 0)	6	0
1	1	0	1	0	(0 1 0)	2	0
1	1	1	0	0	(0 1 1)	3	0
1	1	1	1	1	(1 1 1)	7	0

The Forward and Backward Tables

Table

u_k	c_k	Forward S_{k-1}	S_k
0	0	0	0
1	1		4
1	1	1	0
0	0		4
1	1	2	1
0	0		5
0	0	3	1
1	1		5
0	1	4	2
1	0		6
1	0	5	2
0	0		6
1	0	6	3
0	1		7
0	1	7	3
[illegible]	[illegible]		7

Backward S_k	S_{k-1}	u_k	c_k
0	0	0	0
	1	1	1
1	2	1	1
	3	0	0
2	4	0	1
	5	1	0
3	6	1	0
	7	0	1
4	0	1	1
	1	0	0
5	2	0	0
	3	1	1
6	4	1	0
	5	0	1
7	6	0	1
	7	1	0

Terminology & Assumptions

- Consider the block size N (the size of the interleaver)
- Thus, $\mathbf{u}=(u_1, u_2, u_3, \ldots, u_N)$ is a binary sequence of 1's and 0's.
- Let $c=(c_1', c_2', \ldots, c_N')$ is an encoded sequence of 1's and 0's.
- Let $\mathbf{x} = (x_1, x_2, \ldots, x_N) = 2\mathbf{c} - 1$ is channel symbols of +1 and –1.
- Let $\mathbf{y} = (y_1, y_2, \ldots, y_N) = \mathbf{x} + \mathbf{n}$, be the observation of $\mathbf{x}$ over an AWGN channel.
- *Start* at the all zero state and *end* the trellis at the all zero state by enforcing the *termination rule* at the end of the block.

Another Assumption for the first decoder

- Let $\mathbf{y} := (\mathbf{y}^0, \mathbf{y}^1)$ and proceed for algorithm development
- We can do the same with $\mathbf{y} :=(\mathbf{y}^0, \mathbf{y}^2)$ for the decoding operation of the second constituent decoder.
- Note that in both decoder, $\mathbf{y}^0$ is used in common.

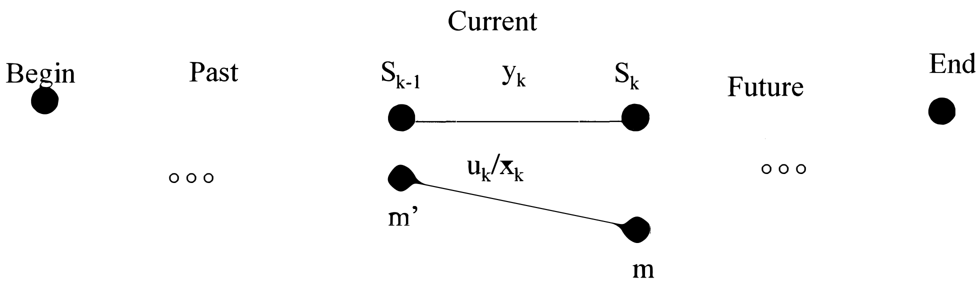
Maximum A Posterior probability on input bit u_k

- Consider the following log ratio

$$L_1(u_k) = log\frac{P(u_k=1|\mathrm{y})}{P(u_k=0|\mathrm{y})} = log\frac{\sum_{(S_{k-1}=m',S_k=m):u_k=1} P(m',m,\mathrm{y})}{\sum_{(S_{k-1}=m',S_k=m):u_k=0} P(m',m,\mathrm{y})}$$

  - The sign gives a hard decision
    - If $L_1(u_k) > 0$, $u_k=1$ with a higher probability
    - Else if $L_1(u_k) <= 0$, $u_k = 0$.
  - The magnitude gives the *reliability of the decision*
    - the larger the more confident
    - For example, if $L_1(u_k) = +\infty$, then we can say $u_k = 1$ with an infinite confidence (absolutely sure).

Consider a state transition at time k-1



- $\Pr(u_k = 1 | \mathbf{y}) \propto p(u_k = 1, \mathbf{y})$
- Think about the event $\{u_k = 1\}$
  - There are 8 edges from a current state S_{k-1} to a state S_k defined by input $u_k = 1$.
  - Note that they are all disjoint events.
- Thus, we have $Pr(u_k = 1, \mathrm{y}) \propto$
  $\sum_{(m',m):u_k=1} p(S_{k-1} = m', S_k = m, \mathrm{y})$

The Markov Property

- Given the current state, the probability on a future event does not depend on the past.
- Pr{future | current state, past} = Pr{future | current state}

$p(S_{k-1} = m', S_k = m, \mathbf{y})$

$p(m', m, \mathbf{y}) = p(S_{k-1}=m', S_k=m, y_k, y_{1:k-1}, y_{k+1:N})$

$= p(S_k = m, y_k, y_{k:N} \mid S_{k-1}=m', y_{1:k-1})\, p(S_{k-1}=m', y_{1:k-1})$

Use the Markov Property

$= p(S_k = m, y_k, y_{k+1:N} \mid S_{k-1}=m')\, p(S_{k-1}=m', y_{1:k-1})$

Conditional Probability

$= p(y_{k+1:N} \mid S_k = m, S_{k-1}=m', y_k)\, p(S_k = m, y_k \mid S_{k-1}=m')\, p(S_{k-1}=m', y_{1:k-1})$

Markov Property, Again

$= p(y_{k+1:N} \mid S_k = m)\ \ p(S_k = m, y_k \mid S_{k-1}=m')\ \ p(S_{k-1}=m', y_{1:k-1})$

By definition

$= \beta_k(m)\, \gamma_k(m',m)\, \alpha_{k-1}(m')$

$\alpha_{k-1}(m') := p(S_{k-1}=m', y_{1:k-1})$

- $\alpha_{k-1}(m') := p(S_{k-1} = m', y_{1:k-1})$

Total Probability Theorem

$$= \sum_{S_{k-2}=m''} p(S_{k-2} = m'', S_{k-1} = m', y_{1:k-1})$$

$p(S_{k-2} = m'', S_{k-1}=m', y_{k-1}, y_{1:k-2})$

$= p(S_{k-2}=m'', y_{1:k-2})\, p(S_{k-1}=m', y_{k-1} \mid S_{k-2}=m'', \cancel{y_{1:k-2}})$

$$= \sum_{S_{k-2}=m''} \alpha_{k-2}(m'')\gamma_{k-1}(m'', m')$$

Forward Algorithm

$\beta_k(m) := p(y_{k:N} \mid S_k = m)$

- $\beta_k(m) := p(y_{k+1:N} \mid S_k = m)$
  $= \sum_{m^*} p(y_{k+1}, y_{k+2:N}, S_{k+1}=m^* \mid S_k=m)$
  $= \sum_{m^*} p(y_{k+2:N} \mid S_k=m, y_{k+1}, S_{k+1}=m^*)\, p(y_{k+1}, S_{k+1}=m^* \mid S_k = m)$
  $= \sum_{m^*} p(y_{k+2:N} \mid S_{k+1}=m^*)\, p(y_{k+1}, S_{k+1}=m^* \mid S_k = m)$
  $= \sum_{m^*} \beta_{k+1}(m^*)\, \gamma_{k+1}(m, m^*)$

Backward Algorithm

The Kernel, $\gamma_k(m',m) := p(S_k = m, y_k \mid S_{k-1}=m')$

- For transitions ($S_{k-1} = m'$, $S_k = m$) with input $u_k = 1$
- $p(S_k = m, u_k = 1, y_k \mid S_{k-1}=m')$
  $= p(y_k \mid S_{k-1}=m', S_k=m, u_k=1)\, \Pr(S_k=m, u_k=1 \mid S_{k-1}= m')$

  $= \Pr(u_k=1 \mid S_k=m, S_{k-1}= m')\, \Pr(S_k=m \mid S_{k-1}=m')$

$= p(y_k \mid S_{k-1}=m', S_k=m, u_k=1)\, \Pr(u_k=1 \mid S_k=m, S_{k-1}= m')\, \Pr(S_k=m \mid S_{k-1}=m')$

Likelihood Prob.	Legitimate Transition or not? 1 or 0	Trans. Prob. By $u_k=1$

Thus, $\gamma_k(m',m) := p(S_k = m, y_k \mid S_{k-1}=m') = p(y_k \mid x_k)\, \Pr(u_k=1)$

The Kernel (2)

- The likelihood Probability is the same as in VA
  - $\{S_{k-1}=m', S_k=m, u_k=1\}$ is a particular edge, and thus it determines the associated channel symbol x_k assigned for the edge
  - Since $y_k = x_k + n_k$, we can calculate the likelihood probability knowing n_k is $N(0, \sigma^2)$ where $\sigma^2 = N_0/(2E_s)$.
  - $p(y_k \mid x_k) = p(n_k = y_k - x_k) \sim \exp(-E_s |y_k - x_k|^2/N_0)$
- Now recall that we have defined $\mathbf{y}=(\mathbf{y}^0, \mathbf{y}^1)$ for the first decoder:
  - $y^j_k = x^j_k + n^j_k$, j=0, 1 for the two independent channels.
- The log likelihood probability

$$p(\mathbf{y}_k \mid x_k) = p(n^0_k)p(n^1_k)$$
$$\sim \exp(-E_s |y^0_k - x^0_k|^2/N_0) \exp(-E_s |y^1_k - x^1_k|^2/N_0)$$

The Kernel (3)

- The second term is 1 if the transition is an allowed transition, 0 if not.
- The third term is *the prior probability* of the transition triggered by input $u_k = 1$.
- Thus, the product of the two terms is $\Pr(u_k=1)$.
  - In turbo decoding, we replace this *prior* with the *extrinsic* information forwarded from the other decoder -- the posterior probability $\Pr(u_k=1 \mid \mathbf{y}^2)$.
  - In the first iteration, we don't have any prior information about u_k.
  - From the second iteration and on, we will get some message from the second decoder, we will make use of that information; vise versa at the other decoder.

Log Ratio Convention
(for all probabilities--priors, likelihood, posteriors)

- In general, with Pr(x=+1)+Pr(x=-1)=1, we have

$$
\begin{aligned}
Pr(x=+1) &= \frac{e^L}{1+e^L} \\
&= \underbrace{\left(\frac{e^{-L/2}}{1+e^{-L/2}}\right)}\cdot e^{xL/2} = Ae^{xL/2}
\end{aligned}
$$

where L = log Pr(x=+1)/Pr(x=-1)

Making use of the Log Ratios
(for both Likelihood/Posteriors)

$E_s = R E_b$

- Recall $x_k := 2u_k - 1$.
- use $y_k = \sqrt{E_s}\, x_k + n_k$ with $\mathcal{N}(0, N_0/2)$
  or $y_k = x_k + n_k$ with $\mathcal{N}(0, N_0/(2E_s))$
- Let's use the second one and for the three independent channels we can define the log likelihood ratios for j=0,1,2

$$
\begin{aligned}
L^j &:= log\frac{p(y_k^j|x_k^j=+1)}{p(y_k^j|x_k^j=-1)} \\
&= log\frac{exp(-E_s|y_k^j-1|^2/N_0)}{exp(-E_s|y_k^j+1|^2/N_0)} \\
&= \frac{4E_s}{N_0}y_k^j \\
&= L_c \cdot y_k^j
\end{aligned}
$$

Computation of $\gamma_k(m',m)$ for $x^0_k = +1$ or -1 ($u_k = 1$ or 0)

- $\gamma_k(m',m) = p(\mathbf{y}_k \mid x_k) \quad Pr(u_k = 1/0)$

$$\propto [\exp(.5x^0_k L_c y^0_k + .5 L_c y^1_k x^1_k) \exp(.5 x^0_k L_1'(u_k))$$

$$= \exp(.5 x^0_k [L_c y^0_k + L_1'(u_k)]) * \exp(.5 L_c y^1_k x^1_k)$$

$\gamma^{(e)}_k(m',m)$

Reliability information provided by the channel

Extrinsic information computed and forwarded from the other constituent decoder

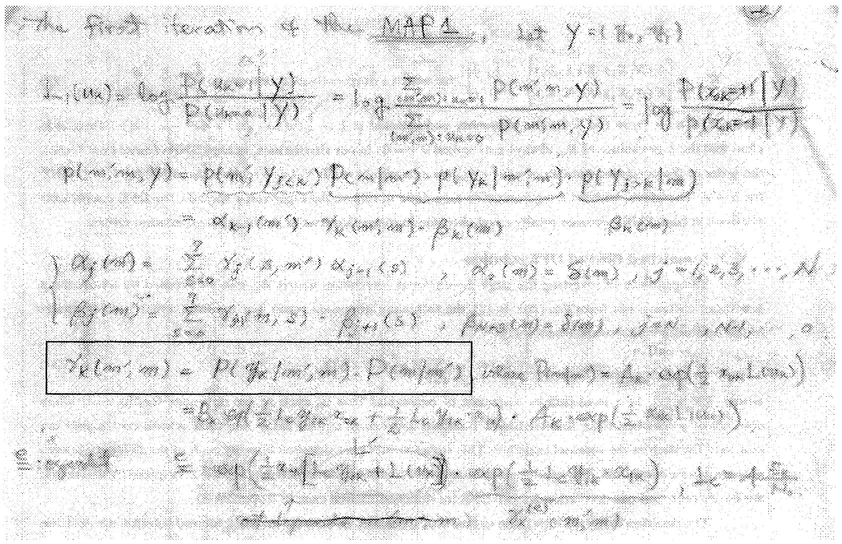
Forward-Backward algorithm

$$
\begin{aligned}
L_1(u_k) &= L_1(x_j^0) \\
&= log \frac{exp(.5(L_c y_k^0 + L_1'(u_k))}{exp(-.5(L_c y_k^0 + L_1'(u_k))} \\
&+ log \frac{\sum_{(m',m):u_k=1} \alpha_{k-1}(m')\gamma_k^{(e)}(m',m)\beta_k(m)}{\sum_{(m',m):u_k=0} \alpha_{k-1}(m')\gamma_k^{(e)}(m',m)\beta_k(m)}
\end{aligned}
$$

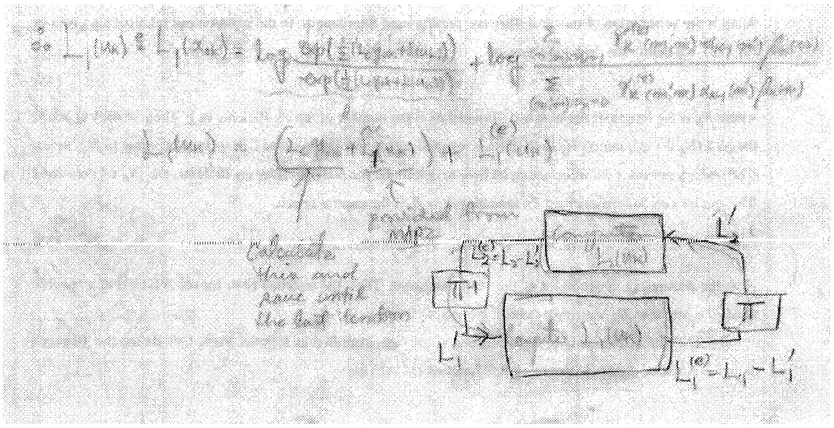
- $L_1(u_k) = L_c y^0_k + L_1'(u_k) + L_1^{(e)}(u_k)$, the posterior log ratio

Extrinsic information generated by the present decoder; this only needs to be forwarded to the other decoder

Looks complicated, but it can be compactly written in a single page



Summary (2)



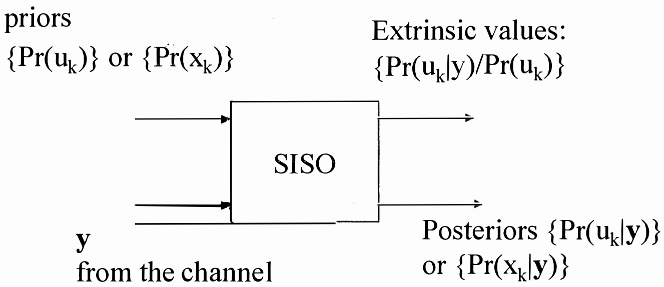
Normalization During Forward/Backward

- Normalize α's and β's
  - $\sum_m \alpha_k(m) = 1$
  - $\sum_m \beta_k(m) = 1$
  - This is necessary because we ignored the coefficient terms when calculating of γ

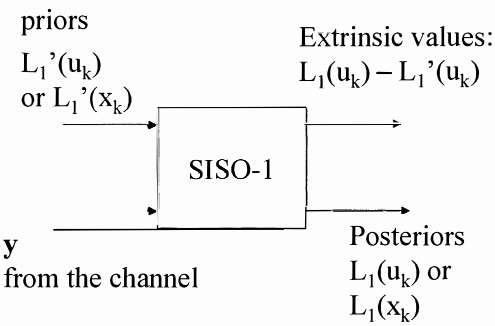
We could have calculated $\Pr(x^1_k = +1/-1 \mid \mathbf{y})$

- In this case, we are calculating the probability of events $\{x^1_k = +1\}$ or $\{x^1_k = -1\}$ given $\mathbf{y}$
-
$$\begin{aligned} L_1(x_k^1) &= log\frac{exp(.5(L_c y_k^0 + L_1'(x_k^1))}{exp(-.5(L_c y_k^0 + L_1'(x_k^1))} \\ &+ log\frac{\sum_{(m',m):x_k^1=1} \alpha_{k-1}(m')\gamma_k^{(e)}(m',m)\beta_k(m)}{\sum_{(m',m):x_k^1=0} \alpha_{k-1}(m')\gamma_k^{(e)}(m',m)\beta_k(m)} \end{aligned}$$

We have Soft-Input Soft-Output Module



SISO Module



Max* operation

- $ab \Rightarrow \log(ab) = \log(a) + \log(b) = A + B$
- $a+b \Rightarrow \log(a+b)$

$$
\begin{aligned}
\log(a+b) &= \log(\exp(\log(a)) + \exp(\log(b)) \\
&= \log(\exp(A) + \exp(B)) \\
&= \max(A, B) + \log[1+\exp(-|A-B|)] \\
&=: \max{}^*(A,B)
\end{aligned}
$$

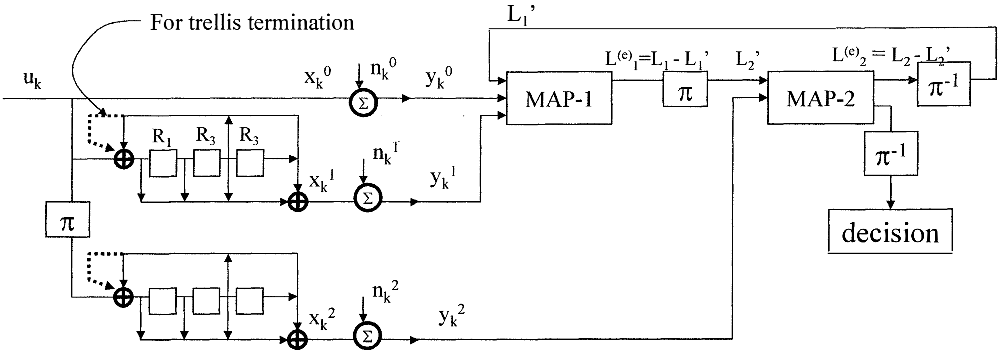
Look-up table

- Approximation: $\max{}^*(A, B) \approx \max(A, B)$
  - Good when high SNR

Avoiding the computation of exponentials and multiplications (Logarithm of probabilities)

- Taking log of in calculating α, β
- The max* operation

Consider our 1/3 turbo code example

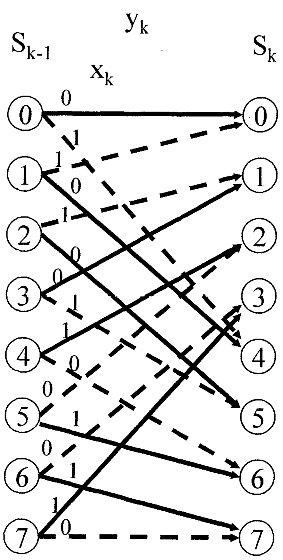


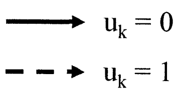
- First construct forward and backward progression tables
- Second draw the trellis
- Apply the BCJR algorithm on the trellis

Again, the first step is to have the trellis

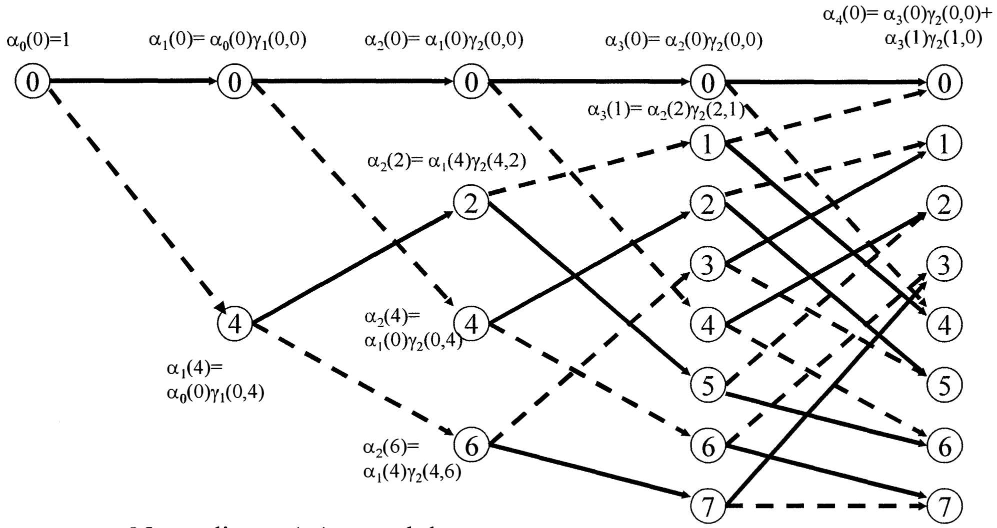
We are interested in calculating $\Pr\{u_k = 1 | \mathbf{y}\}$ or $\Pr\{x_k = 1 | \mathbf{y}\}$.

$\Sigma_{(m',m):\, uk=1} \Pr\{m', m | \mathbf{y}\}$
or
$\Sigma_{(m',m):\, xk=1} \Pr\{m', m | \mathbf{y}\}$



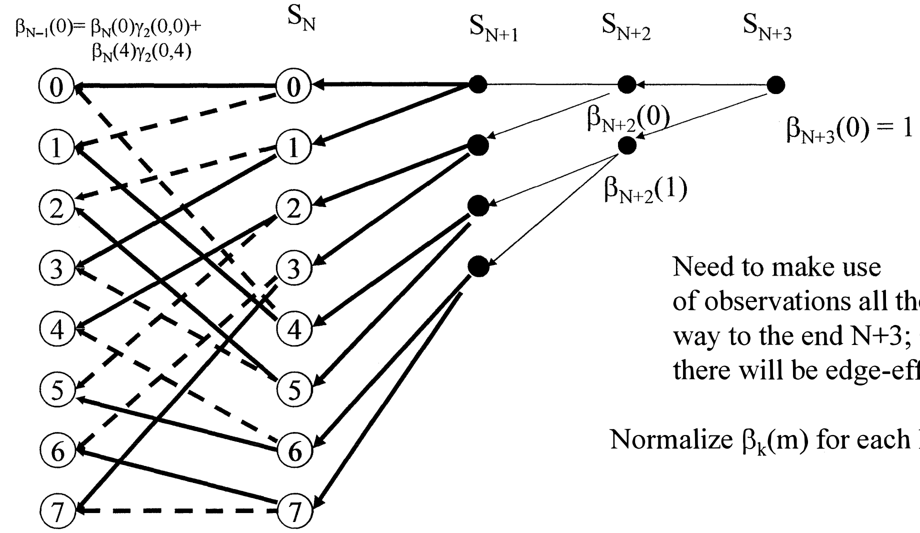


Then, Perform Forward Progression



Normalize $\alpha_k(m)$ at each k

Then, Backward Progression



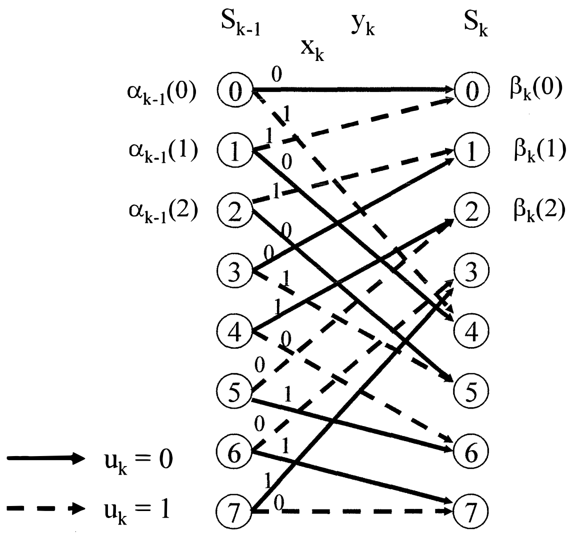
Need to make use of observations all the way to the end N+3; O.W. there will be edge-effect.

Normalize $\beta_k(m)$ for each k

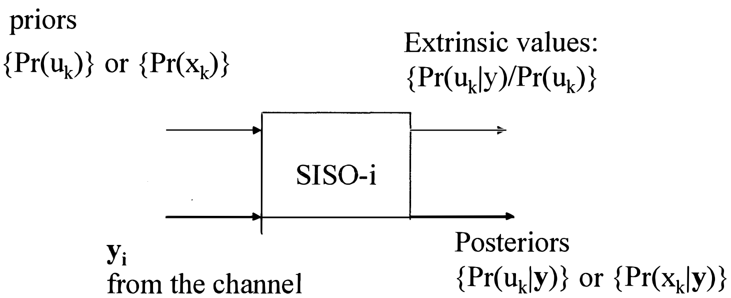
Now with $\alpha_{k-1}(m')$ and $\beta_k(m)$ known, the posterior at any *k*-th trellis-section can be computed

Thus, we can now calculate
$\Pr\{u_k = 1 | \mathbf{y}\}$ or
$\Pr\{x_k = 1 | \mathbf{y}\}$.

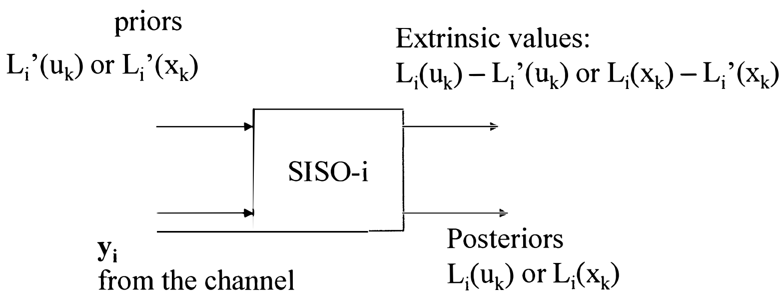
$\sum_{(m',m):\, uk=1} \alpha_{k-1}(m')\, \gamma_k(m',m)\beta_k(m)$
or
$\sum_{(m',m):\, xk=1} \alpha_{k-1}(m')\, \gamma_k(m',m)\beta_k(m)$



We have Soft-Input Soft-Output Module

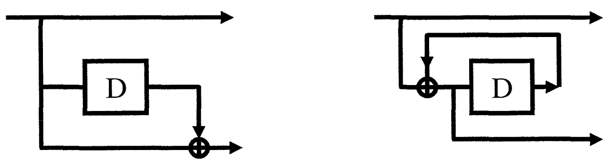


SISO Module (Log Ratios)



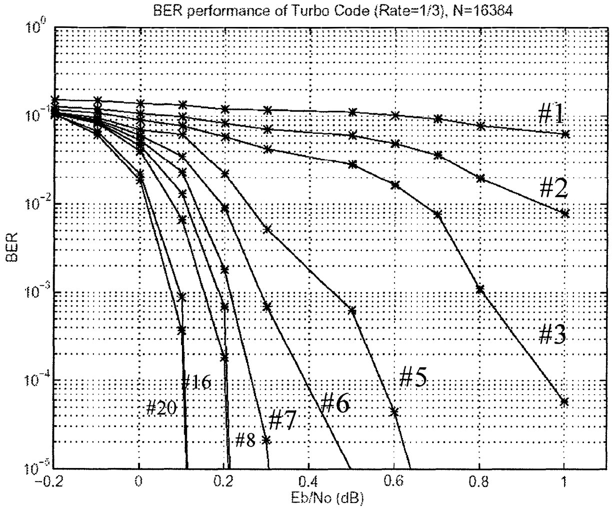
Components of the Turbo Encoder (Why it works so well?)

- The role of Recursive Encoder?
  - The constituent convolutional encoders of Turbo code must be recursive (why?)
- The role of Random Interleaver?
- For illustration, let's consider two simple convolutional encoders—feedfoward and recursive.

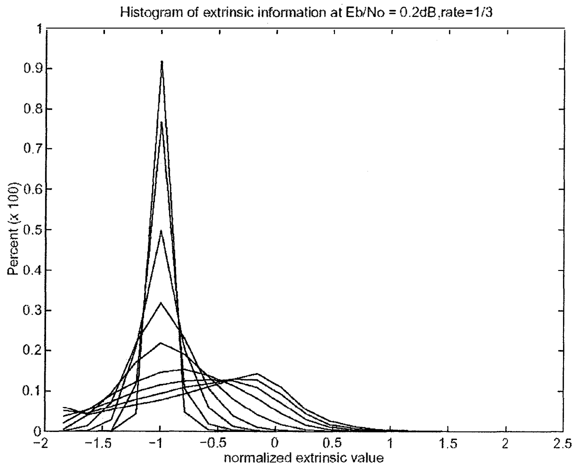


(a) Feedforward (a) Recursive

The Rate 1/3 Turbo Code

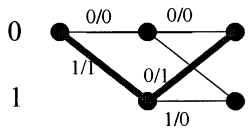


The Rate 1/3 Turbo Code

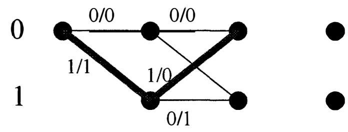


All zero word
-- Log ratio of the extrinsic is normalized.
-- Does each histogram look like a Gaussian?
-- Density Evolution

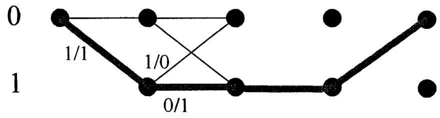
Consider the Trellises of (a) and (b)



Weakest error event is with input weight 1 having output weight 2



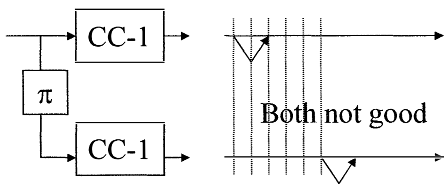
Weakest error event is not with input 1, but with input weight 2 having output weight 1



With another 1, it returns to all-zero path. Otherwise, it never goes back to all-zero — accumulating a large amount of distance metric.

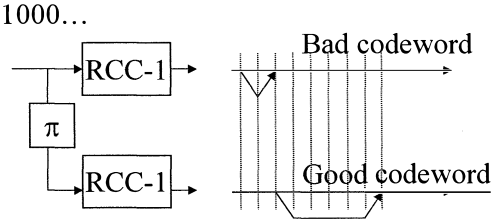
Watch What Happens When used in Turbo Encoder

10000...



000001000...

11000...



00100001000...

Output weight 5

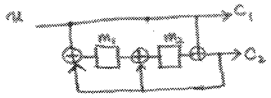
the free distance of the weakest error event (2, 2)

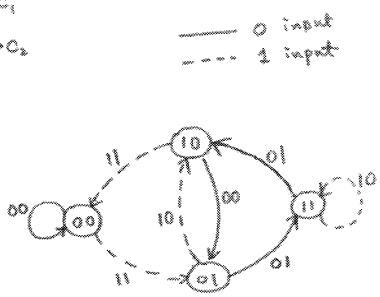
the free distance of the weakest error event (1, 5)

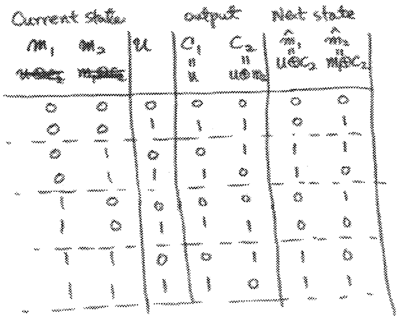
The weakest error event input to one encoder is interleaved and becomes an event strong against errors when presented to the other encoder

Another Example, Recursive CC [Conv'l codes]

State Diagram for Recursive Form



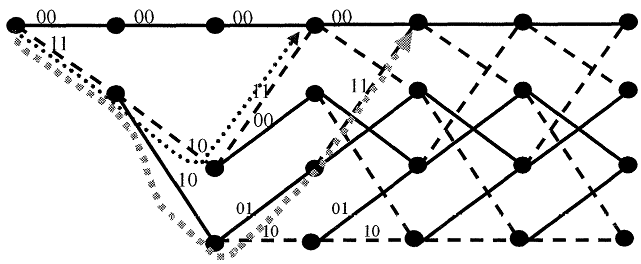




Current state m_1	m_2	u	output $c_1 = u$	$c_2 = u \oplus m_2$	Next state $\hat{m}_1 = u \oplus c_2$	$\hat{m}_2 = m_1 \oplus c_2$
0	0	0	0	0	0	0
0	0	1	1	1	0	1
0	1	0	0	1	1	1
0	1	1	1	0	1	0
1	0	0	0	0	0	1
1	0	1	1	1	0	0
1	1	0	0	1	1	0
1	1	1	1	0	1	1

Comparing to the feedforward form, we note that the outputs are the exactly the same.

The Weakest Events



- The input for the weakest error event is 100100000…, again input weight 2 (output weight is 6)
  - 10000… results in infinite output weight sequence (Strong event)
- The next one is 1000010000….

s-random interleaver [Divsalar]

- Good random interleaver must spread any adjacent bits as far as possible.
  - spread-random interleaver

1 2 3 4 5 6 … i j k l m n … N

s=5

$\pi(1)$ $\pi(2)$ $\pi(3)$ … $\pi(i)$ $\pi(j)$ $\pi(k)$ … $\pi(N)$

3 9 45 20 …

- How to construct a s-random interleaver?
  - Randomly select a number from N integers $\{1, 2, \ldots, N\}$ without replacement
  - The n-th selection $\pi(n)$ will be
    - Put back into the pool if $|\pi(n) - \pi(j)| < s$, for all j, $|n-j| < s$
    - Accepted otherwise
  - Repeat until exhausted
  - Choose s<sqrt(N/2)

s-random interleaver

- There are many ways to generate s-random interleaver
- One way is to use the procedure in previous page with the following heuristic auxiliary rule
- Now, consider a random selection close to the end of sequence, say at the $(N - m)$-th selection such that there are m numbers left in the pool
- What should we do when this set of m numbers do not support the selection rule
  - Ex) Residual set at (N-5)={1,2,3,4,5} with s=5
  - No matter how we select we cannot satisfy the rule
- Go back to (N-100)-th selection, and re-select numbers from that point and on, hoping not to run into such a problematic residual set for the rest of 100 selections

Analytical Performance Bounds using Weight Enumeration Function [Benedetto/Montorsi, '96]

- Recall the transfer function from convolutional codes.
- Note their definition is a little bit different from ours, given at the section on convolutional codes.
- In order to avoid unnecessary confusion when we read the paper, we will follow Benedetto's notation in this section.

Input Output Weight Enumerating Function

- Consider a block code (n, k, $d_{\min}$) C with code rate R_c = k/n.
- Define the input-output weight enumerating function (IOWEF)

$$A^C(W, H) := \sum_{w,h} A^C_{w,h} W^w H^h$$

where $A^C_{w,h}$ is the number of codewords with output weight h and input weigh w.

- Example: (1 0 0) $\Rightarrow$ (1 0 0 1 0 1), input weight 1 and output weight 3, for (n=6, k=3) code.

Union Bound on Bit Error Probability

- Using the knowledge on IOWEF, we can obtain

$$
\begin{aligned}
P_b(e) &\leq \sum_{h=d_{min}}^{n} \sum_{w=1}^{k} \frac{w}{k} A^C_{w,h} Q(\sqrt{2R_c h E_b/N_0}) \\
&= \sum_{h=d_{min}}^{n} D_h Q(\sqrt{2R_c h E_b/N_0})
\end{aligned}
$$

where $D_h := \frac{1}{k}\sum_{w=1}^{k} w A^C_{w,h}$ is *multiplicity* of codewords with weight *h* (not the number of codewords with weight *h*), and E_b/N_0 is the SNR per bit.

Multiplicity of codewords of weight *h*

- It is the total number of nonzero information bits associated with codewords of weight *h,* divided by *k.*

Example

- (3, 2) code C
- $A^C\{0,0\}=1$, $A^C\{1,2\}=2$, and$A^C\{2,2\}=1$
- IOWEF is

  $A^C(W,H) = 1+2WH^2+W^2H^2$

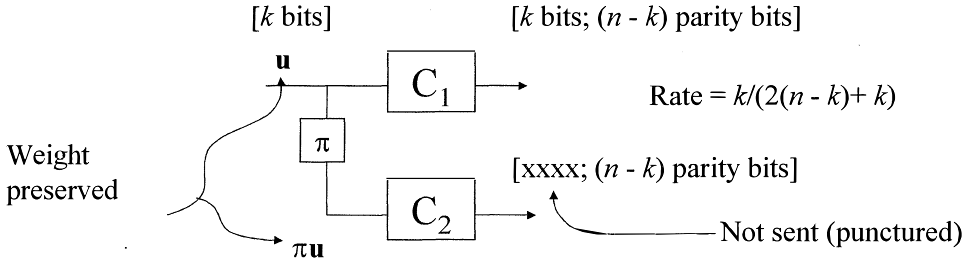
  $= 1+(2W+W^2)H^2$
- Multiplicity
  - $D_2 = (1/2)\,[2*1 + 1*2] = 2$

codewords	w	h
0 0 0	0	0
0 1 1	1	2
1 0 1	1	2
1 1 0	2	2

- $$\begin{aligned} P_b(e) &= \frac{1}{2}[2\cdot 1\cdot Q(\sqrt{2\cdot\frac{2}{3}\cdot 2E_b/N_0}) + 1\cdot 2\cdot Q(\sqrt{2\cdot\frac{2}{3}\cdot 2E_b/N_0})] \\ &= Q(\sqrt{\frac{8}{3}E_b/N_0}) \end{aligned}$$

Parallel Concatenated Block Codes

- Consider parallel concatenation of (n, k) block codes

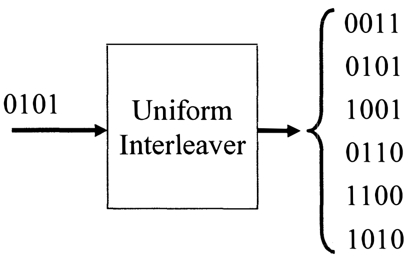


- We know the input-redundancy weight coefficients (IRWC) for the two block codes

  $\{A^{C_1}_{w,h_1}\}, \{A^{C_2}_{w,h_2}\}$

Analysis using Uniform Interleaver

- For the analysis, imagine an abstract interleaver
  - A probabilistic device which maps an input with weight w into $\binom{k}{w}$ permutations, with equal probability $p=1/\binom{k}{w}$.
- Calculate an averaged performance using Uniform Interleaver
  - The performance of a particular interleaver will perform always better than, or at least equal to, the average performance.



Computation of IRWC for Turbo Code

- IRWC $\{A^{C_P}_{w,h}\}$ can be calculated using the properties of the Uniform Interleaver.
  - Maps a single input block of weight w at C_1 into $\binom{k}{w}$ permutations as input to C_2.
- Thus, the number of codewords of output weights h_1, h_2 associated with input word of weight w is defined as

$$A^{C_P}_{w,h_1,h_2} = \frac{A^{C_1}_{w,h_1} \times A^{C_2}_{w,h_2}}{\binom{k}{w}}$$

IRWEF for Turbo Codes

- IRWC $A^{C_P}_{w,h} = \sum_{\{h_1,h_2:h_1+h_2=h\}} A^{C_P}_{w,h_1,h_2}$

- Having the input-output weight coefficient defined, we can obtain the input-output weight enumerating function:

$$A^{C_P}(W,H) = \sum_{w,h} A^{C_P}_{w,h} W^w H^h$$

Notation (the same as BM's paper)

- Consider only the systematic code from now on.
- Example: (3, 2) code C
  - $A^C_{0,0}=1$, $A^C_{1,1}=2$, and$A^C_{2,0}=1$
  - IOWEF is
    $A^C(W,Z) = 1+2WZ+W^2$

codewords	w	j
0 0 0	0	0
0 1 1	1	1
1 0 1	1	1
1 1 0	2	0

- The weight enumerating function is
  $B^C(H) = \sum_{h=0}^{n} B_h H^h$
  where B_h is the number of codewords with weight h.
- The IRWEF vs. the conditional WEF $A^C_w(Z)$
  $A^C(W,Z) = \sum_w \sum_j A_{w,j} W^w Z^j$
  $= \sum_w W^w \sum_j A_{w,j} Z^j$
  $= \sum_w W^w A^C_w(Z)$

Conditional WEF

- Or, the CWEF can be obtained from the derivatives of $A^C(W, Z)$ in the following ways:

$$A_w^C(Z) = \sum_j A_{w,j} Z^j = \frac{1}{w!} \cdot \left. \frac{\partial^w A^C(W,Z)}{\partial W^w} \right|_{W=0}$$

Union Bound

- We have already used the weight enumerating functions in calculating the union bounds for convolutional codes [refer to lectures on Conv. Codes]
- Let's briefly review the union bounds using BM's notation
- First, consider

$$\begin{aligned} \cancel{W} \frac{\partial A^C(W,Z)}{\partial W} &= \sum_{w,j} w A_{w,j} W^w Z^j \\ &= \sum_m [\sum_{m=w+j} w A_{w,j}] H^{w+j} \end{aligned}$$

Let W=Z=H

Total number of info-bits associated with codewords with weight m=w+j.

$$P_m = Q(\frac{D_E}{\sqrt{2N_0}}) = Q(\sqrt{\frac{4E_s m}{2N_0}}) = Q(\sqrt{\frac{2mE_s}{N_0}}) = .5erfc(\sqrt{\frac{mEs}{N_0}})$$

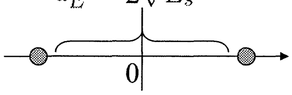
BER for BPSK

- $y_k = \sqrt{E_s}\, x_k + n_k$, where $x_k \in \{-1, 1\}$ with equally likely

$$
\begin{aligned}
Pr(y_k > 0 | x_k = -1) &= \int_0^\infty \frac{1}{\sqrt{2\pi(N_0/2)}} e^{-\frac{(y+\sqrt{E_s})^2}{2(N_0/2)}} dy \\
&= \int_{\sqrt{2E_s/N_0}}^\infty \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx \\
&= Q(\sqrt{2E_s/N_0}) = Q(d_E/\sqrt{2N_0})
\end{aligned}
$$

- $Q(x) := \frac{1}{2\pi} \int_x^\infty e^{-\frac{t^2}{2}} dt, \quad x \geq 0$

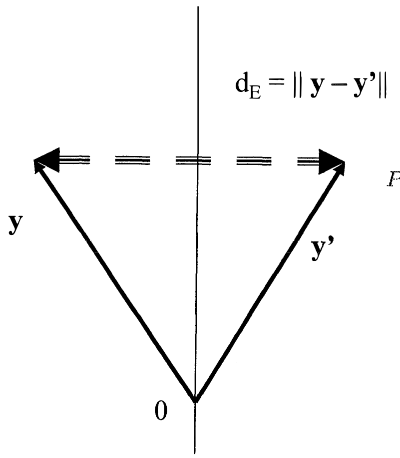
$d_E = 2\sqrt{E_s}$



$E_b = E_s \cdot (1/R_c)$

R_c = rate of the code
= number of bits
each baud carries

Pairwise Error Event in m-dimensional vector space



$\mathbf{y} = \sqrt{E_s}\mathbf{x} + \mathbf{n}$
$\{1, -1\}$

$\mathbf{n} = (n_1, n_2, \ldots, n_m)$
iid $\mathcal{N}(0, N_0/2)$

$$
\begin{aligned}
P_m &= Q(\frac{d_E}{\sqrt{2N_0}}) = Q(\sqrt{\frac{4E_s m}{2N_0}}) = Q(\sqrt{\frac{2mE_s}{N_0}}) \\
&= .5erfc(\sqrt{\frac{mEs}{N_0}})
\end{aligned}
$$

The Complementary Error Function vs. Q(x)

- $erfc(x) := \frac{2}{\pi} \int_x^{\infty} e^{-t^2} dt$
- $Q(x) = \frac{1}{2} erfc(\frac{x}{\sqrt{2}})$
- MATLAB only defines the complementary error function erfc(x)

Probability of Bit Error (Union Bound)

$$P_b(e) \le \frac{W}{k} \frac{\partial A^C(W,Z)}{\partial W}\bigg|_{W=Z=e^{-R_c E_b/N_0}}$$
$$= \sum_{w=1}^{k} \frac{w}{k} W^w A_w^C(Z)\bigg|_{W=Z=e^{-R_c E_b/N_0}}$$
$$= \sum_m D_m H^m \bigg|_{H=e^{-R_c E_b/N_0}}$$

m starting from d_{free}

with

$$D_m \triangleq \sum_{j+w=m} \frac{w}{k} A_{w,j}$$

Multiplicity of the code

where R_c is the code rate.

Traditional approaches → max. d_{free}

In Turbo code → Reduces $D_{d_{free}}$

Approximation

- Using only a finite number of first terms, we have an approximation

$$P_b(e) \approx \frac{1}{2}\sum_m D_m \operatorname{erfc}\left(\sqrt{m\frac{R_c E_b}{N_0}}\right). \qquad (6)$$

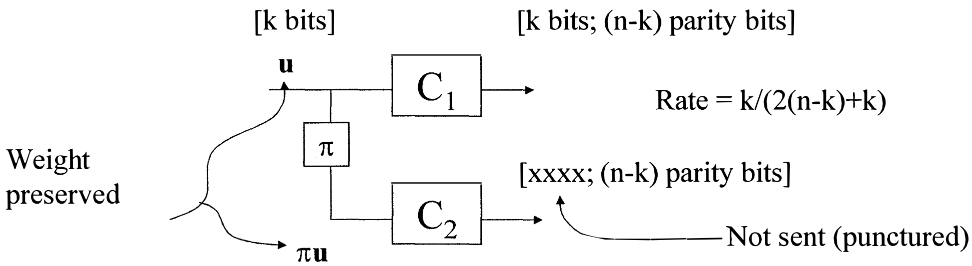
starting from $m=d_{free}$,

Where are we now?

- We have reviewed how to calculate the union bound on (systematic) block code (or similarly on truncated convolutional codes)
- Now let's consider turbo encoder whose constituent encoders are systematic block codes (truncated convolutional codcs)

Parallel Concatenated Block Codes

- Consider parallel concatenation of (n, k) block codes

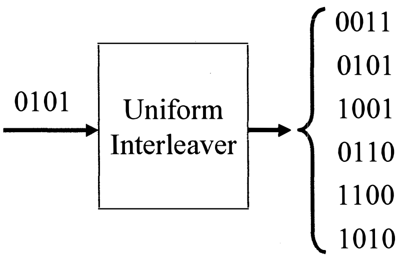


- The input-redundancy weight coefficients (IOWC) for the two block codes or Conditional WEFs are known to us:

$$\{A^{C_1}_{w,j}\}, \{A^{C_2}_{w,j}\} \quad \text{or} \quad \{A^{C_1}_{w}(Z)\}, \{A^{C_2}_{w}(Z)\}$$

Analysis using Uniform Interleaver

- For the analysis, imagine an abstract interleaver
  - A probabilistic device which maps an input with weight *w* into $\binom{k}{w}$ permutations, with equal probability p=1/$\binom{k}{w}$
- Calculate an averaged performance using Uniform Interleaver
  - There exist at least one interleaver the performance of which is better than, or at least equal to, to that of the average performance



Each word with weight *w* has $\binom{k}{w}$ matches using UI

WEFs of PCBC

- Conditional WEF of Turbo code

$$A_w^{C_P}(Z) = \frac{A_w^{C_1}(Z) \times A_w^{C_2}(Z)}{\binom{k}{w}}$$

- Example: (7, 4) Hamming code
- $A^C(W,Z)=1+W(3Z^2+Z^3)+W^2(3Z+3Z^2)+W^3(1+3Z)+W^4Z^3$

$$\binom{4}{1} = 4 \qquad \binom{4}{2} = 6 \qquad \binom{4}{3} = 4$$

- $A^C_1(Z)A^C_1(Z)=(3Z^2+Z^3)^2=Z^4(9+6Z+Z^2)$

$$\binom{4}{1} = 4$$

Due to the operation of UI, the number of codewords with weight w increase by $\binom{k}{w}$

IRWEF of Turbo Code

- Once we know conditional WEF, we know IRWEF

$$A^{C_P}(W,Z) = \sum_{w=1}^{k} W^w A_w^{C_P}(Z), \qquad (8)$$

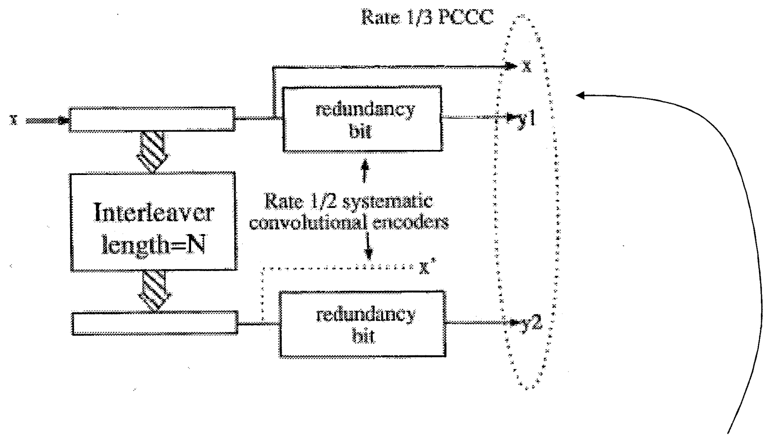
Examples

- IRWEF of the PCBC can be obtained as:

$$\begin{aligned}A^{C_P}(W,Z) = 1 + W(2.25Z^4 + 1.5Z^5 + 0.25Z^6) + \\ + W^2(1.5Z^2 + 3Z^3 + 1.5Z^4) + \\ + W^3(0.25 + 1.5Z + 2.25Z^2) + W^4Z^6. \quad (9)\end{aligned}$$

- Example-1: $A^{C_{p}}{}_1 = A^{C_1}{}_1(Z) \times A^{C_2}{}_1(Z)/4 = Z^4(9+6Z+Z^2)/4$
$= Z^4(2.25 + 1.5Z + 0.25Z^2)$
- Example-2: $A^{C_{i}}{}_2 = (3Z+3Z^2) = Z(3 + 3Z)$.
$A^{C_{p}}{}_2 \times A^{C_{p}}{}_2/36 = Z^2(9+6Z + 9Z)/6$
$= Z^2(1.5+Z + 1.5Z^2)$

Consider Rate 1/3 PCCC
[See Benedetto/Montorsi TC'96 paper now]



Consider rate ½ equivalent (2N, N-ν) block code

Equivalent Rate ½ (2N, N-ν) Block Code

- The codewords are all sequences of length 2N of the convolutional codes, starting from and ending at the zero state.
  - Concatenation of error events of the convolutional codes.

n-error events in the block

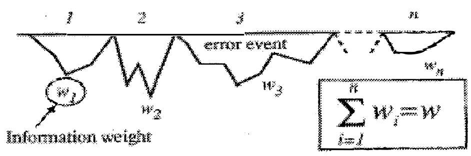


Fig. 2. Example of a sequence belonging to $A(w, Z, n)$.

- Parity check EF generated by n-error events with total weight w

  $A(w, Z, n) := \sum_j A_{wjn} Z^j$

  The number of codewords with input weight *w*, parity weight *j*, and number of concatenated error events *n*

The Conditional WEF

- It can be approximated by (i.e., $A^C{}_w(Z)$)

$$A^C(w, Z) \sim \sum_{n=1}^{n_{max}} \binom{N}{n} A(w, Z, n) \qquad (4)$$

where

- n_{max} is the largest number of error events generated by a weight *w* information sequence, is a function of *w, and*
- We neglected the length of the error event, assuming N >> ν.

CWEF for PCCC

- CWEF is $$A^{C_P}(w, Z) \sim \sum_{n_1=1}^{n_{max}} \sum_{n_2=1}^{n_{max}} \frac{\binom{N}{n_1}\binom{N}{n_2}}{\binom{N}{w}} \cdot A(w, Z, n_1)A(w, Z, n_2).$$
- Use the approximation $\binom{N}{n} \sim \frac{N^n}{n!}$

$$A^{C_P}(w, Z) \sim \sum_{n_1=1}^{n_{max}} \sum_{n_2=1}^{n_{max}} \frac{w!}{n_1! \cdot n_2!} N^{n_1+n_2-w} \cdot A(w, Z, n_1)A(w, Z, n_2) \qquad (5)$$

- For large N, it can be approximated by terms $n_1=n_2=n_{max}$

$$A^{C_P}(w, Z) \sim \frac{w!}{n_{max}!^2} N^{2n_{max}-w}[A(w, Z, n_{max})]^2.$$

Substitute into this and get the final result

$$P_b(e) \le \sum_{w=1}^{N} \frac{w}{N} W^w A^{C_P}(w, Z) \Big|_{W=Z=e^{-R_c E_b/N_0}} \quad (1)$$

Asymptotic Bounds on BER

$$P_b(e) \stackrel{\sim}{\le} \sum_{w=w_{\min}}^{N} w \cdot \frac{w!}{n_{\max}!^2} N^{2n_{\max}-w-1} \cdot W^w [A(w, Z, n_{\max})]^2 \big|_{W=Z=e^{-R_c E_b/N_0}} \quad (7)$$

where $w_{\min}$ denotes the minimum information weight in the error events of the CC.

- For a large *interleaver gain*
  - Make the exponent of N, $2n_{max}$-w-1, as negative as possible.
- Note the term with w_{min} is the dominant one.

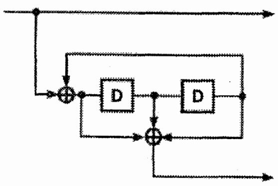
Feedforward conv. code does not work as constituent codes

- w_{min} is 1 for feedforward convolution code (or a block code)
  - For $w=w_{min}=1$, the max. number of error events $n_{max}= 1$.
  - Thus, $2n_{max}-w-1 = 2-2=0$.
- There is no interleaving gain.

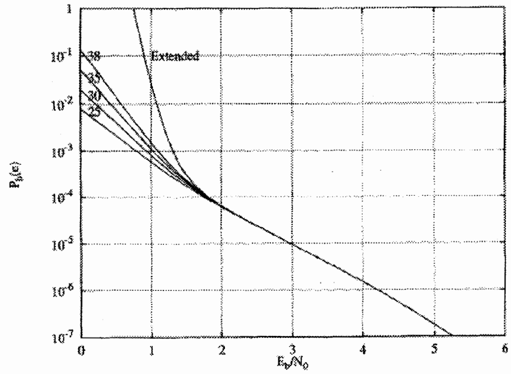
How about Recursive Conv. Code as constituent codes

- w_{min} is 2 for feedback convolution code.
  - For $w=w_{min}=2$, the max. number of error events $n_{max}= 1$
  - Thus, $2n_{max}-w-1 = 2-2-1=-1$.
- There is interleaving gain of 1/N.

Approximation by Truncation D_m, m<M.

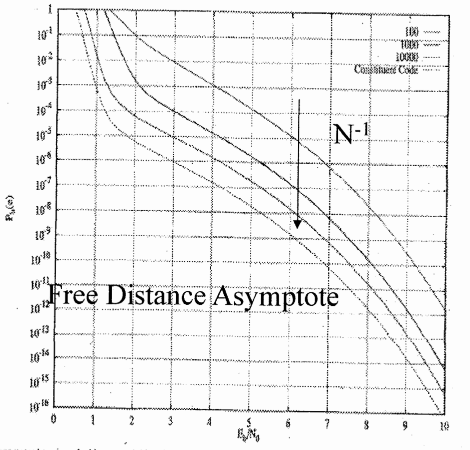


$d_{free}=5$



Interleaving Gain

$$P_b(e) = \sum_m D_m H^m \Big|_{H=e^{-R_c E_b/N_0}}$$



Average upper bounds to the bit error probability for the PCCC of Example 7 with uniform interleavers of lengths 100, 1000, 10000.

d^T_{free} is increase from 5 to 8

TABLE V
COEFFICIENTS D_m FOR THE EVALUATION OF THE BIT ERROR PROBABILITY OF THE PCCC OF EXAMPLE 7 WITH INTERLEAVERS LENGTHS 100, 1000, 10000

Hamming distance	N		
	100	1000	10000
8	3.8900E-02	3.9881E-03	3.9988E-04
9	7.6590E-02	7.9605E-03	7.9960E-04
10	0.1136	1.1918E-02	1.1991E-03
11	0.1508	1.5861E-02	1.5985E-03
12	0.1986	1.9887E-02	1.9987E-03
13	0.2756	2.4188E-02	2.4917E-03
14	0.4079	2.9048E-02	2.8102E-03
15	0.6292	3.4846E-02	3.2281E-03
16	1.197	6.5768E-02	6.8575E-03
17	2.359	0.1457	1.3897E-02
18	4.383	0.2984	2.8543E-02
19	7.599	0.5472	5.2989E-02
20	12.58	0.9171	8.9441E 02
21	20.46	1.437	0.1403
22	33.31	2.144	0.2052
23	54.65	3.090	0.2957
24	91.23	4.465	0.4177
25	154.9	6.716	0.6133
26	265.5	10.67	0.9577
27	455.6	17.65	1.574
28	779.0	29.61	2.646
29	1327.	49.31	4.430
30	2257.	80.57	7.267
31	3842.	128.6	11.60
32	6556.	201.3	18.04
33	11221	311.5	27.57
34	19261	481.2	41.88
35	33143	748.8	63.94

Role of Number of States in CC

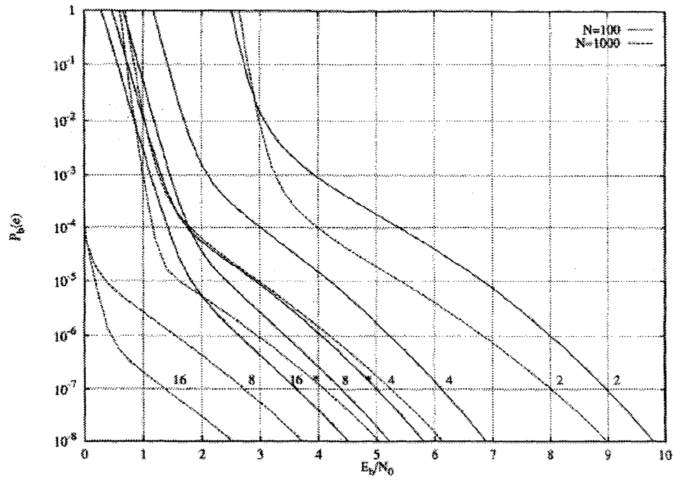
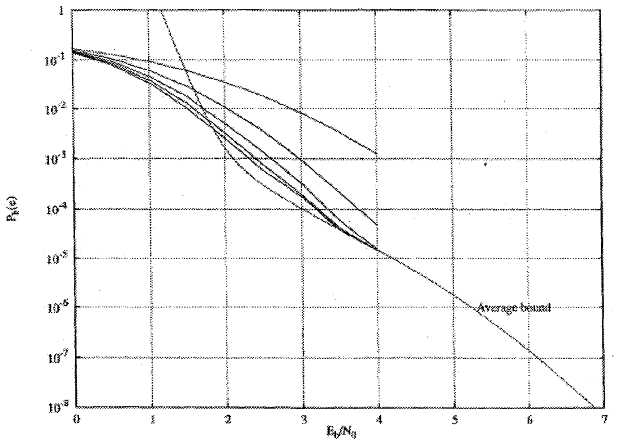


Fig. 15. Average upper bounds to the bit error probability for a PCCC using as CC's two recursive convolutional encoders with 2,4,8, and 16 states and uniform interleavers of length $N = 100$ (continuous curves) and $N = 1000$ (dashed curves).

Comparison with Simulation



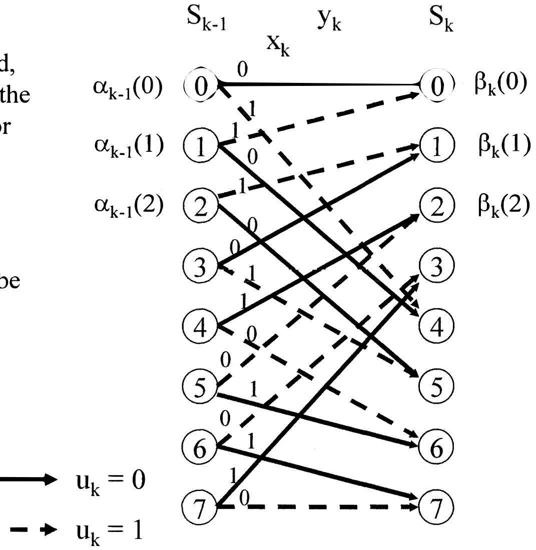
Summary

- All constituent encoders must be *recursive* convolutional codes.
- The effective free distance of constituent encoders must be maximized.

Recall, the posteriors of both input and output can be calculated

- Once $\alpha_{k-1}(m')$ and $\beta_k(m)$ calculated, we can compute the posteriors for the input and the output $\Pr\{u_k = 1|\mathbf{y}\}$ or $\Pr\{x_k = 1|\mathbf{y}\}$.
- $\gamma_{k-1}(m',m)=\Pr(y_k|m',x_k)\Pr(m|m')$
- Note that the priors $\Pr(m|m')$, can be defined either by $\Pr(x_k)$ or $\Pr(u_k)$

$\sum_{(m',m):\, uk=1} \alpha_{k-1}(m')\, \gamma_k(m',m)\beta_k(m)$
or
$\sum_{(m',m):\, xk=1} \alpha_{k-1}(m')\, \gamma_k(m',m)\beta_k(m)$



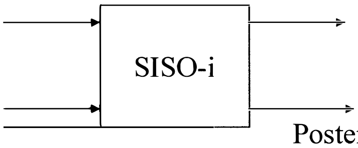
Soft-Input Soft-Output Module

priors

$\{Pr(u_k)\}$ or $\{Pr(x_k)\}$

Extrinsic values:

$\{Pr(u_k|\mathbf{y}_i)/Pr(u_k)\}$ or $\{Pr(x_k|\mathbf{y}_i)/Pr(x_k)\}$



$\mathbf{y}_i$
from the channel

Posteriors

$\{Pr(u_k|\mathbf{y}_i)\}$ or $\{Pr(x_k|\mathbf{y}_i)\}$

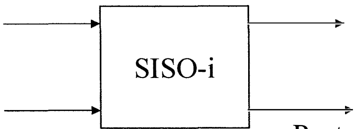
SISO Module (Log Ratios)

priors

$L_i'(u_k)$ or $L_i'(x_k)$

Extrinsic values:

$L_i(u_k) - L_i'(u_k)$ or $L_i(x_k) - L_i'(x_k)$

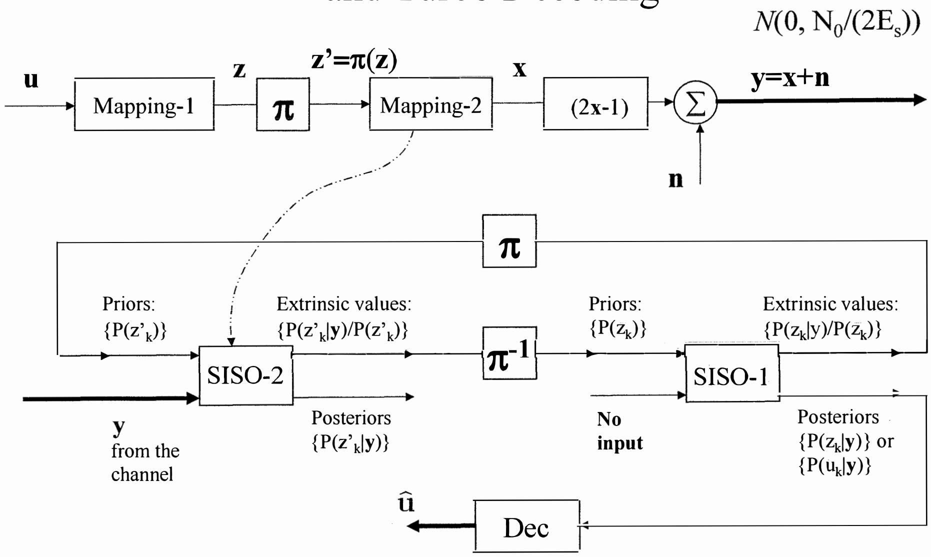


$\mathbf{y}_i$
from the channel

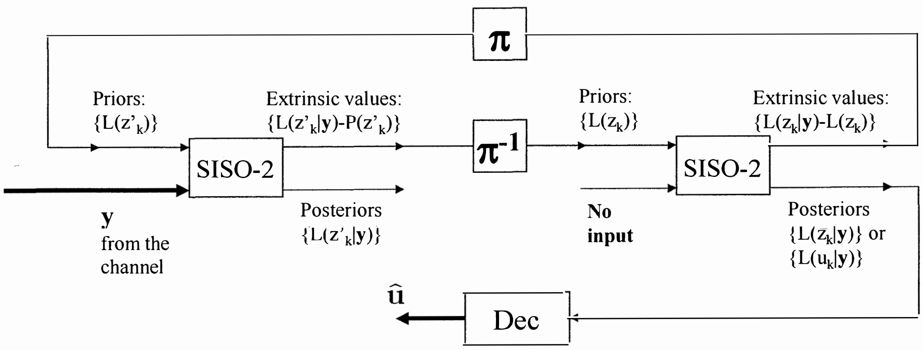
Posteriors

$L_i(u_k)$ or $L_i(x_k)$

Serial Concatenation of Mapping Machines and Turbo Decoding



Serially Concatenation of Mapping Machines and Turbo Decoding (2)

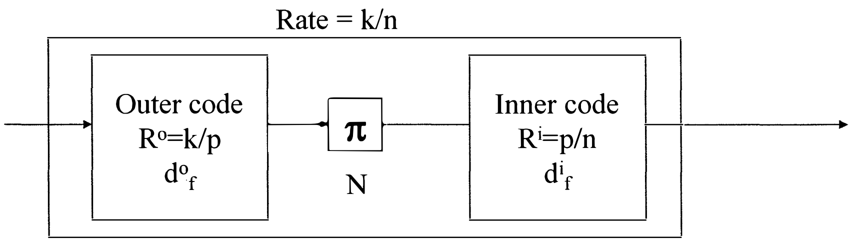


- Where L denotes the log ratio of the probabilities. For example, $L(z_k|\mathrm{y}) := log\frac{Pr(z_k=1|\mathrm{y})}{Pr(z_k=0|\mathrm{y})}$

Application of Iterative Turbo Decoding Principle

- Serially Concatenated Convolutional Codes
  - Benedetto et al, TIT '98
- Iterative Equalization and Decoding
  - One of the earlier one is the paper by Anastasopoulos and Chugg, Asilomar Conf. In 1997
- Bit-Interleaved Coded Modulation with Iterative Decoding
  - Li and Ritcey, TC 2002
- Iterative Mutilevel Demodulation and Decoding
  - Stephan Brink
- Iterative channel estimation and decoding/equalization

Serial Concatenation of Convolutional Codes



- Serially concatenated convolution code (n, k, N) where N is the length of the interleaver (Assumed to be a multiple of p)
- Example: k=1, p=2, n=3, N = 200: Rate 1/3 code
  - The input to outer code is length 200, and the output of the inner code is 300

Union Bound Analysis on SCCC

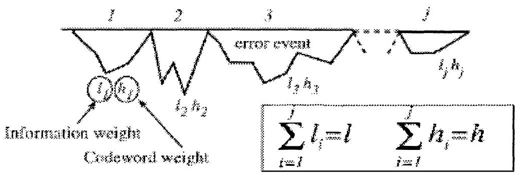


Fig. 7. The meaning of the coefficients $A_{l,h,j}$.

- $A_{l,h,j}$: The number of codewords with j error events, where each error event is with input weight l and the output weight h
- Let n^M is the maximum number of error events, then the number of codewords with input weigh l and the output weight h

$$A^C_{l,h} \le \sum_{j=1}^{n_M} \binom{N/p}{j} A_{l,h,j}$$

No. of trellis sections

Uniform Interleaver Analysis

- The coefficients of conditional WEFs for outer and inner code are

$$A^{C_o}_{w,l} \le \sum_{n^o=1}^{n^o_M} \binom{N/p}{n^o} A^o_{w,l,n^o}$$

$$A^{C_i}_{l,h} \le \sum_{n^i=1}^{n^i_M} \binom{N/p}{n^i} A^i_{l,h,n^i}.$$

Why bounds? -- neglecting the length of error events

- The coefficients of conditional WEF of Serially concatenated block code (Uniform Interleaver Argument)

$$A^{C_S}_{w,h} \le \sum_{l=d_f^o}^{N} \sum_{n^o=1}^{n^o_M} \sum_{n^i=1}^{n^i_M} \frac{\binom{N/p}{n^o}\binom{N/p}{n^i}}{\binom{N}{l}} A^o_{w,l,n^o} A^i_{l,h,n^i}$$

Bit Error Probability

- Use a bound for $\binom{N}{l} > \frac{(N-l+1)^l}{l!} > \frac{N^l}{p^l l!}$.
- Thus, the coefficient of conditional WEF of the SCCC

$$A^{C_s}_{w,h} \le \sum_{l=d_f^o}^{N} \sum_{n^o=1}^{n_M^o} \sum_{n^i=1}^{n_M^i} N^{n^o+n^i-l} \cdot \frac{l!}{p^{n^o+n^i} n^o! n^i!} A^o_{w,l,n^o} A^i_{l,h,n^i}.$$

Examine this further

- Finally,

$$P_b(e) \le \sum_{h=h_m}^{N/R_c^i} e^{-hR_c E_b/N_0} \sum_{w=w_m^o}^{NR_c^o} \sum_{l=d_f^o}^{N} \sum_{n^o=1}^{n_M^o} \sum_{n^i=1}^{n_M^i} N^{n^o+n^i-l-1} \cdot \frac{l!}{p^{n^o+n^i-1} n^o! n^i!} \frac{w}{k} A^o_{w,l,n^o} A^i_{l,h,n^i}. \qquad (18)$$

The Exponent of N

- $\alpha := n^o + n^i - l - 1$
- At high SNR, the first term—the smallest output weight term—will dominate.
- In the paper, the analysis was carried out on this first term and obtain the following result.

Bit Error Probability using Union Bound (2)

- When d^o_f is even,

$$P_b(e) \overset{\sim}{\leq} B_{\text{even}} N^{-d^o_f/2} \exp\left[-\frac{d^o_f d^i_{f,\text{eff}}}{2} R_c E_b/N_0\right] \qquad (31)$$

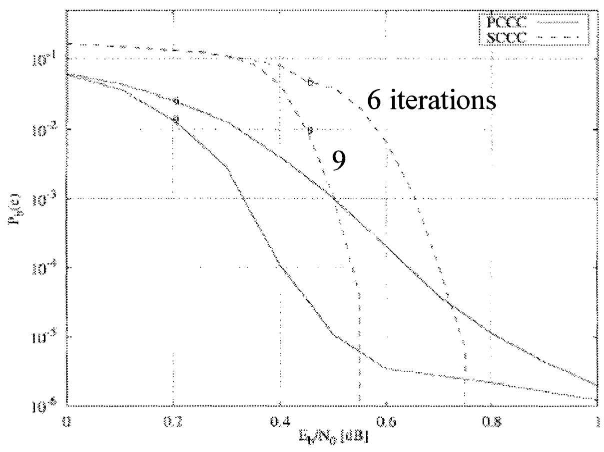
- When d^o_f is odd

$$P_b(e) \overset{\sim}{\leq} B_{\text{odd}} N^{-\frac{d^o_f+1}{2}} \cdot \exp\left\{-\left[\frac{(d^o_f-3)d^i_{f,\text{eff}}}{2} + h_m^{(3)}\right] R_c E_b/N_0\right\} \qquad (34)$$

Summary of Major Results of SCCC

- The inner decoder must be recursive.
- The outer decoder can be either recursive or feedforward.
- The interleaver gain is N^{-d^o_f/2} for even values of d^o_f and N^{-(d^o_f+1/2)} for odd values of d^o_f
  - Choose an outer decoder with a large, possibly odd, free distance

Different Behavior of Convergence



HW#6

- Turbo Code Problems
  - P14.1, P14.2,
- Trellis Code Problems
  - P13.4
  - Problem #1: Design a rate-2 8 PSK four trellis-states code without any parallel transitions
    - Find the free ED of your code
    - Compare it with that of the best 4 trellis states code
  - Problem #2: Reproduce the Channel Capacity results (Fig.2 in Ungerboeck's paper) for m-ary PSK signals (m=2, 4, 8, 16): (Submit the MATLAB program for this, along with your results)

Density Evolution

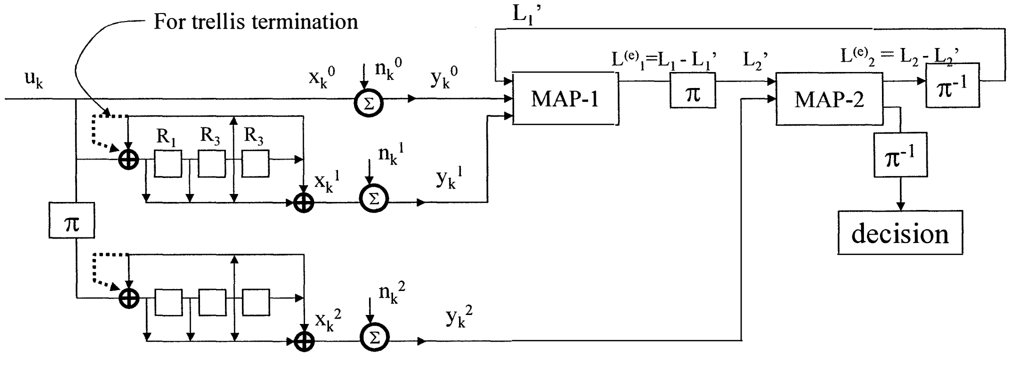
Agenda

- Density Evolution: Turbo Code
  - Refer to Divsalar et al's TMO Progress Report 42-144 (See this paper in the course web-page) and Stephan Ten Brink's paper (Trans. On Comm. Oct. 2001)
- SISO Module for LDPC Decoding
- Coded Modulation over ISI Channel

Reference Papers

- Gallager's Decoding Analysis (Section 4.3 his Thesis)
- Two kinds of methods to analyze the code and design a better code
  - Use EXIT charts (Ten Brink)
  - Use Density Evolution (Richardson)
  - Originally, Gallager's threshold based analysis
- **Design of low-density parity-check codes for modulation and detection**
  ten Brink, S.; Kramer, G.; Ashikhmin, A.; Communications, IEEE Transactions on, Volume: 52, Issue: 4 , April 2004, Pages:670 – 678.
- **The capacity of low-density parity-check codes under message-passing decoding**
  Richardson, T.J.; Urbanke, R.L.;
  Information Theory, IEEE Transactions on , Volume: 47 , Issue: 2, Feb 2001, Pages:599 – 618.
- **Design of capacity-approaching irregular low-density parity-check codes**
  Richardson, T.J.; Shokrollahi, M.A.; Urbanke, R.L.;
  Information Theory, IEEE Transactions on , Volume: 47 , Issue: 2, Feb 2001, Pages:619 – 637.

Consider our 1/3 turbo code example again

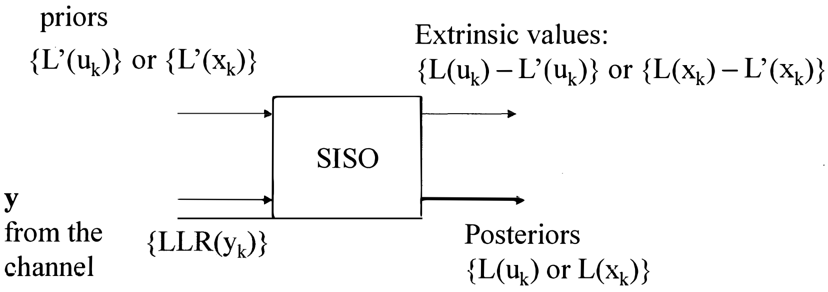


- Assume all zero input bits
  - All three are all zero codewords $\{x_k^0 = 0\}, \{x_k^1 = 0\}, \{x_k^2 = 0\}$
  - All –1's transmitted

Mapping Rule

- Let's use the same mapping rule we have been using
  - Codeword bit 1 is +1 and codeword bit 0 is –1
  - Not very convenient for the purpose of describing the density evolution, but let's use it for the sake of keeping the notation straight
- All zero codeword transmitted → a sequence of all –1's transmitted → Negative log ratios of extrinsic values, likelihood values, and posteriors are favorable.
- All three blocks are sequences of –1's.

The Property of Log Ratio SISO Signals



- $LLR(y_k)$ is Gaussian.
- Posterior LR=$LLR(y^0_k)$+Prior LR+Extrinsic LR.
- Simulation shows the histogram of extrinsic LR values does look Gaussian.
  - Posterior LR is Gaussian
  - Prior LR is Gaussian (Must be independent from y_k)

Property of Log Ratio SISO Signals

- First, let's take a look at the log likelihood ratio
  - y = x + n : suppose y is a realization of random variable Y associated with the binary r.v. $X \in \{-1,1\}$ and the Gaussian r.v. N, with mean 0 and variance $N_0/(2E_s)$
  - $LLR(y) = \log[p(y|x=1)/p(y|x=-1)] = (4E_s/N_0)\, y$
- Given x = -1, we can define

  $L=(4E_s/N_0)*Y = (4E_s/N_0)*(-1+N)$

  $= \mu_L + (-\mu_L)* N$
- $\mu_L = -4E_s/N_0$
- $\sigma_L^2 := Var(L) = (4E_s/N_0)^2 * (N_0/(2E_s)) = 8E_s/N_0$
- $\sigma_L^2 = 2\,|\mu_L|$
  - The variance of log ratio value is twice the absolute value of mean.

Normalized Log Ratio

- Let $Z := L/\mu_L$
  - $Var(Z) = (1/\mu_L)^2\, 2|\mu_L| = 2/|\mu_L|$ =Var(noise)
  - $E\{Z\}=1$
- With μ_L increase to infinity, var(Z) goes to zero or SNR=1/var(Z) goes to infinity
- Higher SNR (Smaller variance) means
  - The decision is getting more and more reliable.
- Recall the density evolution of the rate 1/3 turbo code in previous lecture.

Consistent Densities

- We observed that LLR(y) = $|\mu_L|$ y
  - *Consistent* probability density
  - The variance of the density is twice the mean
- Not only the likelihoods, but also those of the extrinsic are *consistent* Gaussian.
  - Again, the variance is twice the mean.
- We only need to keep track of a single parameter, either the mean or the variance.

Computation of γ_k(m',m) for x^0_k =+1 or –1 (u_k = 1 or 0) [Turbo-Code Lecture]

- $\gamma_k(m',m) = \Pr(y_k \mid x_k)\ \Pr(u_k=1/0)$

$$\propto [\exp(.5x^0_k\, L_c\, y^0_k + .5\, L_c\, y^1_k\, x^1_k)\, \exp(.5\, x^0_k\, L_1'(u_k))$$

$$= \exp(.5\, x^0_k\, [L_c\, y^0_k + L_1'(u_k)]) * \exp(.5\, L_c\, y^1_k\, x^1_k)$$

Reliability information provided by the systematic portion

Reliability information provided by the parity portion

Extrinsic information computed and forwarded from the other constituent decoder

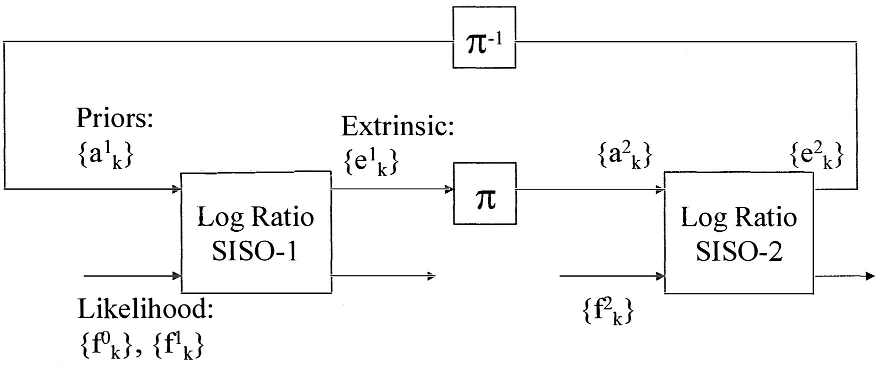
Forward-Backward algorithm [Turbo-Code Lecture]

$$
\begin{aligned}
L_1(u_k) &= L_1(x_j^0) \\
&= log \frac{exp(.5(L_c y_k^0 + L_1'(u_k))}{exp(-.5(L_c y_k^0 + L_1'(u_k))} \\
&+ log \frac{\sum_{(m',m):u_k=1} \alpha_{k-1}(m')\gamma_k^{(c)}(m',m)\beta_k(m)}{\sum_{(m',m):u_k=0} \alpha_{k-1}(m')\gamma_k^{(e)}(m',m)\beta_k(m)}
\end{aligned}
$$

- $L_1(u_k) = L_c\, y^0{}_k + L_1'(u_k) + L_1{}^{(e)}(u_k)$, the posterior log ratio

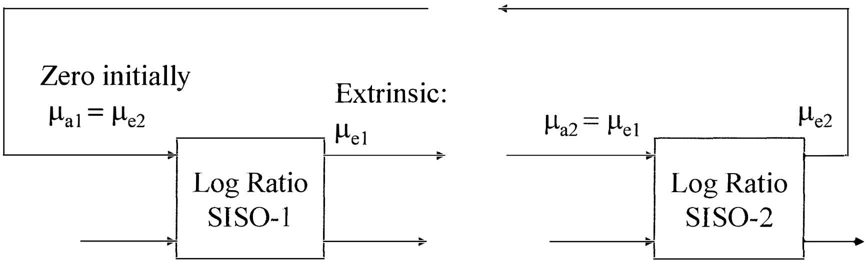
Extrinsic information generated by the present decoder; this only needs to be forwarded to the other decoder

Investigation of Iterative Convergence



- Extrinsic output (*Consistent* Gaussian) is the prior input to the other.
  - Only need to keep track of a single parameter, either the mean or the variance (or SNR = mean^2/var = $\mu^2/2\mu = \mu/2$).

Investigation of Iterative Convergence



- Investigate the input and output relationship of an individual SISO module.
- First SISO-1 starts with the likelihood inputs only.

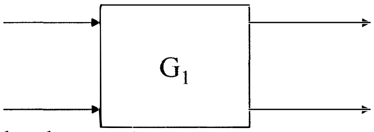
Monte Carlo Simulation

- To determine the noise figure, or input/output transfer function of SNR, of the SISO modules.
- G1:= function($SNR1_{in}$)= $SNR1_{out}$, defined for SISO-1.
- G2:= function($SNR2_{in}$)= $SNR2_{out}$, defined for SISO-2.
- Generate the sample input priors with a given input SNR, and measure the SNR of the output extrinsic values.
- Generate the samples independently from each other for different SNRs and different modules.

Monte Carlo Simulation
For SISO-1

Generate the priors $\{a^1_k\}$
with mean μ_a
$a^1_k = \mu_a + \text{sqrt}(2|\mu_a|)*w_k$ ← $\mathcal{N}(0, 1)$

Calculate the mean μ_e of $\{e^1_k\}$



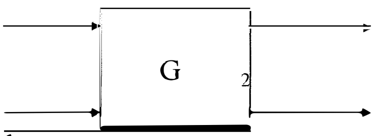
Generate the log likelihood values
$\{f^0_k\}$ and $\{f^1_k\}$ due to all –1's sequences
Note both can be generated with a single information E_b/N_0

- With a fixed E_b/N_0 we can obtain input/output relationship between μ_a and μ_e:
  - Obtain $\{y^i_k=-1+ n^i_k\}$ from Gaussian samples $\{n_k\}$ with zero mean and variance $N_0/(2E_s)$.
  - Obtain the log likelihood ratios from $f^i_k=L_c*y^i_k$,
  - $E_s = E_b*\text{Rate}$.

Monte Carlo Simulation (2)
For SISO-2

Generate the priors $\{a^2_k\}$
with mean μ_a
$a^2_k = \mu_a + \text{sqrt}(2|\mu_a|)*w_k$

Calculate the mean μ_e of $\{e^2_k\}$

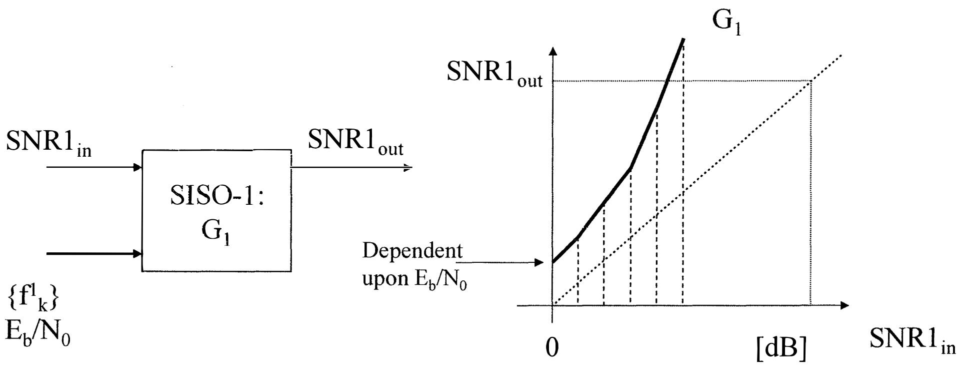


Generate the log-likelihoods
$\{f^2_k\}$ due to all –1's sequences
Note this also can be generated with information E_b/N_0

Monte Carlo Simulation (3)

- Both inputs, likelihood and priors, are *consistent* Gaussians which can be defined by any single parameter from
  - Mean
  - Variance = 2 Mean
  - SNR = $\text{Mean}^2/\text{Variance} = \text{Mean}/2$
- At the first iteration
  - Priors are all 0s
  - Only likelihood ratios are used
- Upon obtaining the output sequence (extrinsic log ratios), calculate the mean (and thus the output SNR).
- Obtain a set of such input-output pairs
  - For example, try SNR_{in} = 0, 1, 2, 3, …, 10 dB.

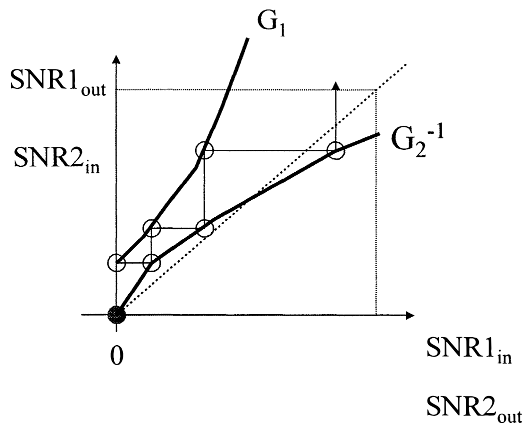
Monte Carlo Simulation (4)



- Obtain the input/output transfer function G1
- Increase input mean μ_a and calculate the output mean
- The first SISO starts with 0 dB SNR_{in} (no prior information)

Monte Carlo Simulation (5)

- Draw G_2^{-1} on the same graph
- Watch the iterations on the graph
- In the beginning $SNR2_{in}=0$



From Divsalar et al's paper

The paper Available at Old class web-Site.

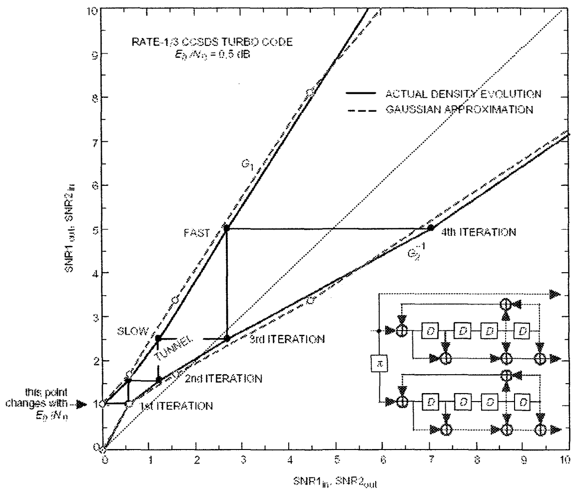
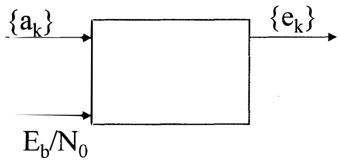


Fig. 7. Iterations and convergence of a turbo decoder.

Mutual Information Between Information Bit and Extrinsic Information [Ref: Ten Brink]

- Investigate the transfer function of mutual information
- $I_A := I(X; a)$.
- $I_E := I(X; e)$.
- Obtain the distributions from the histograms of samples of $\{e_k\}$.
- Assume consistent Gaussian for $\{a_k\}$.
- If iteration be successful, these two measures will converge to an identical measure.



The Same Monte Carlo Simulation

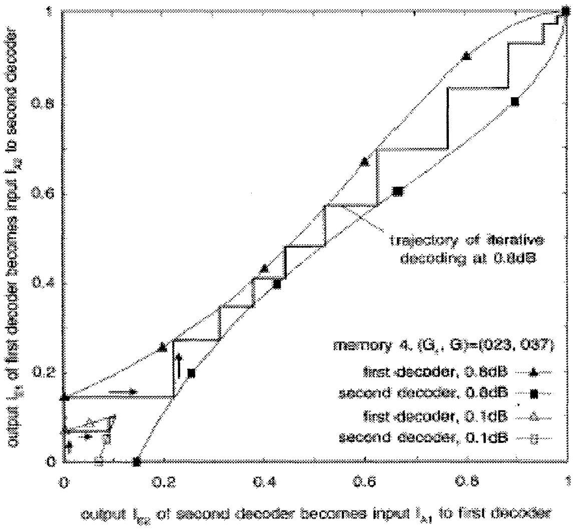
Approach is the same

- generate the *consistent* Gaussian samples $\{a_k\}$ and
- calculate the histogram of extrinsic output $\{e_k\}$ (more complex than calculating only the mean, but more interesting and robust)

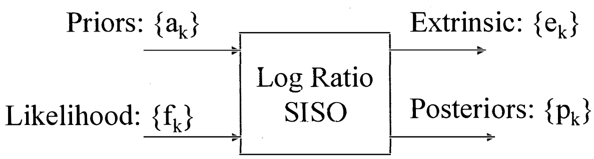
$$I_A = \frac{1}{2} \cdot \sum_{x=-1,1} \int_{-\infty}^{+\infty} p_A(\xi|X=x) \times \mathrm{ld} \frac{2 \cdot p_A(\xi|X=x)}{p_A(\xi|X=-1) + p_A(\xi|X=1)} d\xi \quad (12)$$

$$I_E = \frac{1}{2} \cdot \sum_{x=-1,1} \int_{-\infty}^{+\infty} p_E(\xi|X=x) \times \mathrm{ld} \frac{2 \cdot p_E(\xi|X=x)}{p_E(\xi|X=-1) + p_E(\xi|X=1)} d\xi \quad (19)$$

Extrinsic Information Transfer Chart

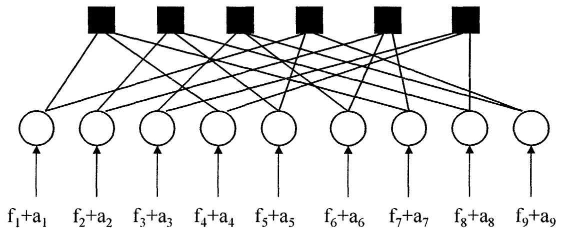


SISO Module for LDPC Decoding



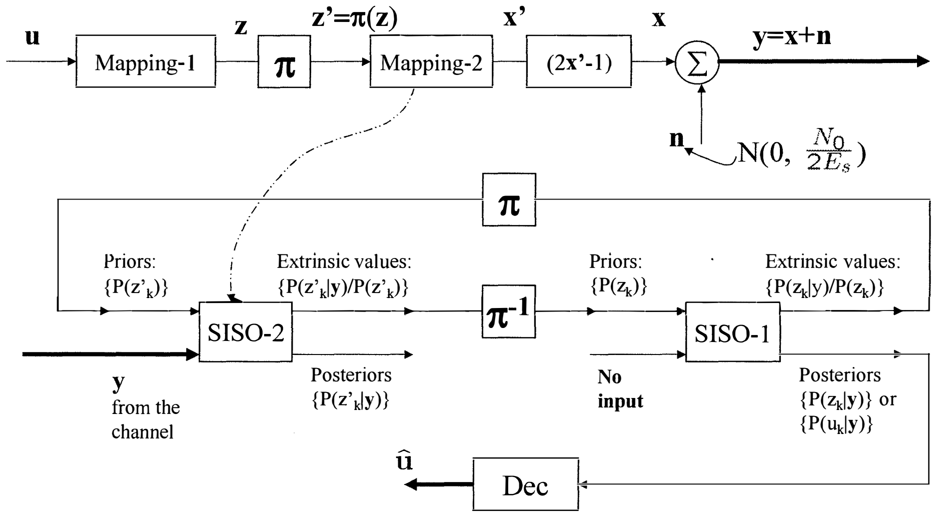
- The posterior is $p_k = (f_k + a_k) + e_k$.
  - The posterior is obtained from message-passing algorithm on the bipartite graph; ideally this should be determined by making use of the entire information presented at the bit nodes such that $\{f_k\}_{k=1:N}$ and $\{a_k\}_{k=1:N}$
- Unless a systematic code is used, the posteriors are available only for codeword bits—we do not have them for information bits.
  - From the posteriors we get the best codeword—but not the information sequence

Start the message passing with $f_k + a_k$

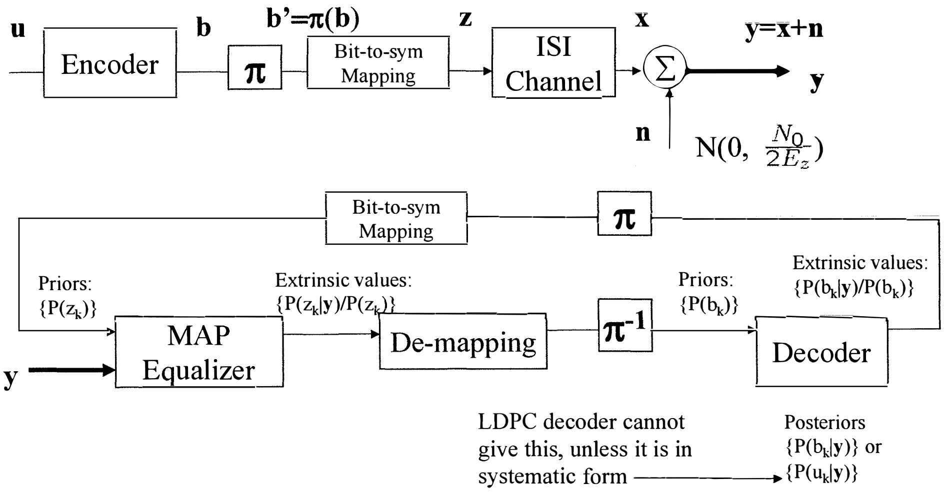


- Start the message passing iterations with the likelihoods and the priors (or whatever one that you have)
- At the end of iterations, we get the posteriors $\{p_k\}$

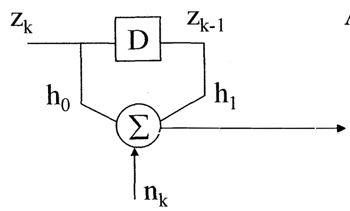
Serial Concatenation of Mapping Machines and Turbo Decoding



Coded Modulation over ISI Channel and Turbo Equalization and Decoding



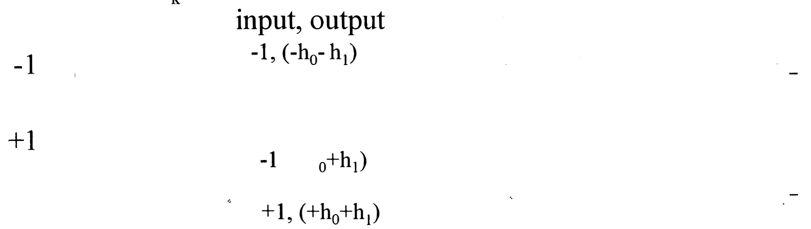
Example of ISI Channel with 1 Memory



Assume $z_k \in \{-1, 1\}$

input state

$y_k=(z_k h_0+z_{k-1}h_1)+n_k = x_k + n_k$



- Four possible clean channel outputs x_k at four branches
- By comparing with y_k, likelihood probability can be computed at each edge

Agenda

- Notation
- Cycle Free Graphs
- Gallager's Decoding Analysis (Section 4.3 of his thesis)
  - Threshold Phenomenon and Calculation
  - Irregular LDPC codes (Luby et al)
  - Density Evolution on Bipartite Graph
  - Richardson/Urbanke, SYChung, etc.

Notation on Reliability L_c

- Is it $L_c = 4E_s/N_0$ or $L_c = 4\sqrt{E_s}/N_0$? – It depends on notation I used
- I have used two channel models without explicitly mentioning which model I was using
- Basically, for both models SNR = $2E_s/N_0$ is the same

1. $y_k = x_k + n_k$ with $\mathcal{N}(0, N_0/(2E_s))$ $\rightarrow$ $LR(f_k) = 4E_s/N_0\, y_k$
2. $y_k = \sqrt{E_s}\, x_k + n_k$ with $\mathcal{N}(0, N_0/2)$ $\rightarrow$ $LR(f_k) = 4\sqrt{E_s}/N_0\, y_k$

- Thus, eventually both approaches give the same results with the respective definition of y_k

Notation (2)

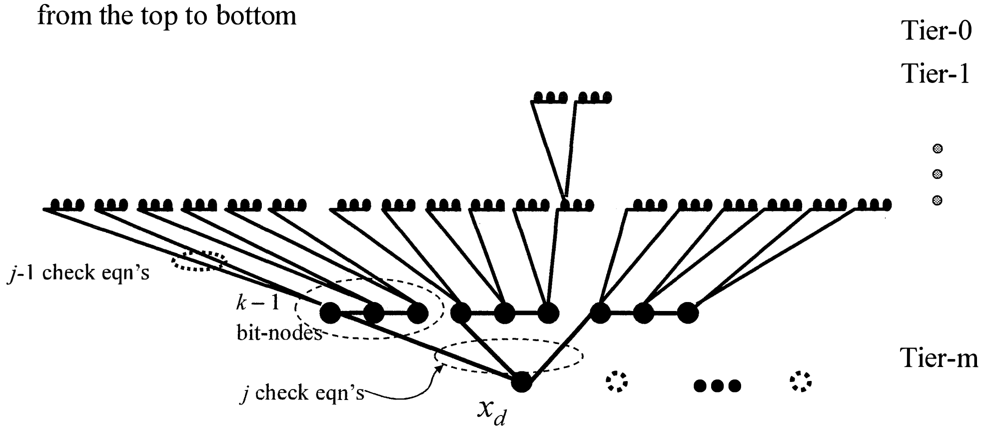
- I meant to say $\Pr\{Y=y\} = \lim_{dy \Rightarrow 0} \Pr\{Y \in (y, y+dy]\}/dy$
  $= \lim_{dy \Rightarrow 0} [\Pr(Y \leq y+dy) - \Pr(Y \leq y)]/dy$
- It's the probability density function when it exists, such that
  - $p(y) := \lim_{dy \Rightarrow 0}[F_Y(y+dy) - F_Y(y)]/dy$
- From now on, I will use the notation p(y) to denote a pdf of random variable Y for continuous random variable

Threshold Phenomenon

- There is a certain threshold value associated with a (n, j, k) LDPC code
- When SNR is greater than the threshold, the bit error probability can be made arbitrarily small as the block length tends to infinity
- When SNR is less than the threshold, the bit error probability is greater than a positive bit error probability, regardless of the block length

Cycle Free Tree

- Unlike our previous notation, lets' now have the tier-index start from the top to bottom

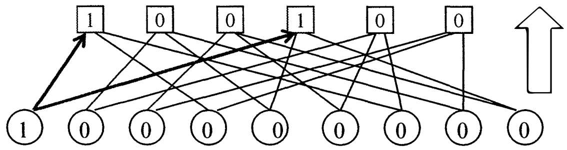


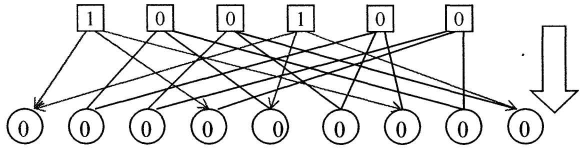
Number of Independent Tiers *m*
(n, j, k) code

- Let *m* be the total number of tiers
- The total number of independent digits at the 0-th tier = 1+j*(k-1)*[(j-1)*(k-1)]$^{m-1}$
  - The last tier = j*(k-1)
  - The others = (j-1)*(k-1)
- $n > 1 + j*(k-1)*[(j-1)*(k-1)]^{m-1} > [(j-1)*(k-1)]^m$
- $m < \log(n)/\log((j-1)*(k-1))$

Recall, Simple Decoding Example (n=9, j=2, k=3)

- Suppose we have
  r=[1 0 0 0 0 0 0 0 0]
- Recall our simple example, using the majority rule, the first error gets corrected.



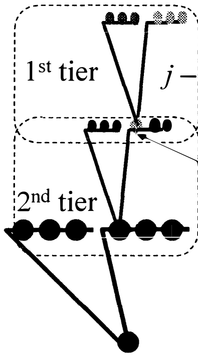


Assumptions/Approach With (j=3) example

- Consider the BSC with cross-over probability p_0.
- Consider the hard decision decoding:
  - If both checks are unsatisfied, change the digit at the first tier.
  - With the changed digit, perform the second tier, and so on.
- The error probability of the hard decision decoding should be an upper bound to that of the probabilistic decoding.

Analysis with The Simple Decoding (j=3, k=4)

- Suppose we start the iteration with an error occurred at a bit node, which happens with probability p_0.
- The first tier calculation involves the first and the second sets of digits (bit nodes).
- The 2^{nd} tier calculation involves the second and the third sets of digits (bit nodes).

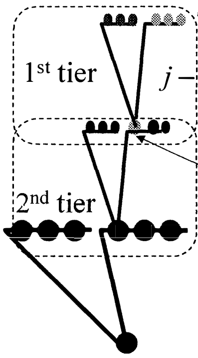


- Now consider the red digit is received in error.
- Each of the two checks constraining the digit is violated when there are even number of errors in (k-1) digits.

Analysis with The Simple Decoding (j=3, k=4)

- A parity check constraining that digit will be unsatisfied iff an even number of errors in the rest (k-1) digits, and the probability of this event is

  $0.5(1+(1 - 2p_0)^{k-1})$
- An error will be corrected when both checks are unsatisfied.



Analysis with The Simple Decoding (j=3, k=4) (2)

- Thus, the probability that a digit is received in error at the first tier, and then corrected after the first iteration is

$$p_0[0.5(1 + (1 - 2p_0)^{k-1})]^2$$

Error in the digit in the second set

Even number of errors in (k - 1) digits

Both checks

Analysis with The Simple Decoding (j=3, k=4) (3)

- Now consider the situation when the probability of a digit is received correctly, but changed due to both checks violated

$$(1- p_0)[0.5(1 - (1 - 2p_0)^{k-1})]^2$$

Received correctly

Odd number of errors in (k-1) bits

Both checks

The Probability of Bit Error in the Second/Higher Sets (j=3)

- The second tier calculations will be similarly done with the bit error probability on the second and the third sets.
- A bit error probability at the second set is determined by
  - $p_1 = p_0(1 - [0.5(1 + (1-2p_0)^{k-1})]^2) + (1- p_0)[0.5(1 - (1-2p_0)^{k-1})]^2$

    {Error occurred, but not flipped} OR {Error not occurred, but flipped}

- A bit error in the third set is again p_0.
- At the end of 2nd tier calculation, a bit error in the third set is determined by
  - $p_2 = p_0(1 - [0.5(1 + (1-2p_1)^{k-1})]^2) + (1- p_0)[0.5(1 - (1-2p_1)^{k-1})]^2$

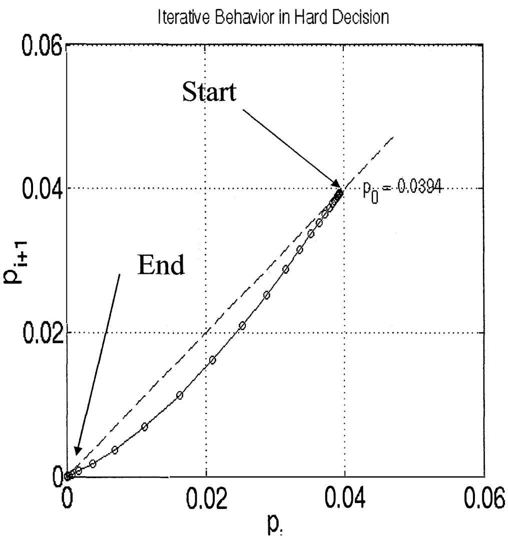
By Induction (for j=3)

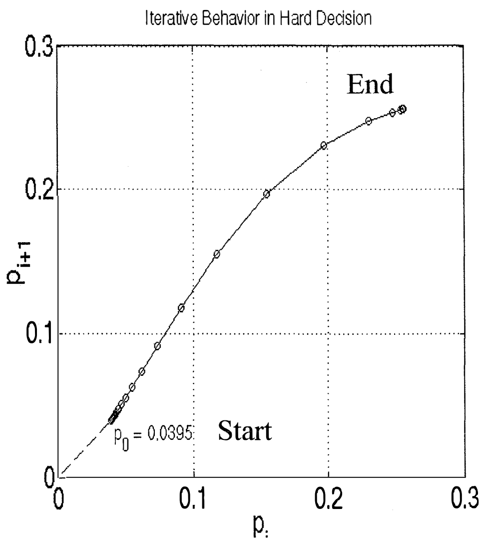
- The error probability of a bit in the (i+1)-th set, obtained at the end of the i-th tier calculation is
  - $p_i = p_0(1 - [0.5(1 + (1-2p_{i-1})^{k-1})]^2) + (1- p_0)[0.5(1 - (1-2p_{i-1})^{k-1})]^2$

Convergence Behavior

- $\{p_i\}$ converges to a number $0 \leq c < 1$
- We want to find $p_{max} := \max p_0$ such that c is arbitrarily small.
- If $p_0 \leq p_{max}$, then $\{p_i\}$ converges to zero.
- If $p_0 > p_{max}$, then $\{p_i\}$ converges to a non-zero positive constant < 1.

$\{p_j\}$ converges to zero if $p_0 \leq p_{max} = 0.0394$ for j=3, k=6





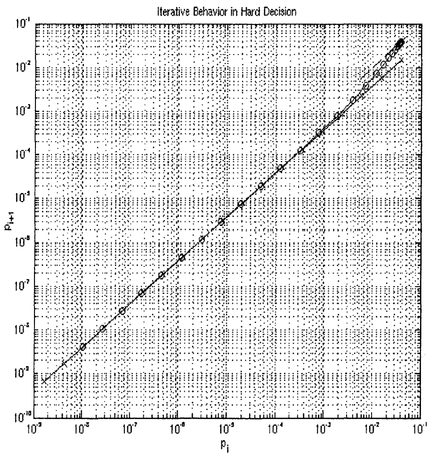
Examples

- Table in the left lists of maximum p_0 resulting in $p_{100} < 1e\text{-}6$.
- Compare the rate ½ codes
  - j = 4 is the best.
- As the rate decreases, p_{max} increases.

j	k	Rate	Maximum p_0
3	6	1/2	0.039
3	5	2/5	0.061
3	4	1/4	0.106
4	8	1/2	0.051
4	6	1/3	0.074
4	5	1/5	0.095
5	10	1/2	0.041
5	8	3/8	0.056
5	6	1/6	0.086

Approximation (j=3)

- $p_{i+1} = p_i\, 2(k-1)\, p_0$
- $p_i = C_1[2(k-1)]^i$



How about when j $\geq$ 4

- More than 3 checks per digit.
- Rule: A digit is changed when b or more checks were violated (Determine optimized b that minimizes p_i).
-

Error; l among (j-1) checks are violated

$$p_{i+1} = p_0\{1 - \sum_{l=b}^{j-1} \binom{j-1}{l} [\frac{1+(1-2p_i)^{k-1}}{2}]^l [\frac{1-(1-2p_i)^{k-1}}{2}]^{j-1-l}\}$$
$$+ (1-p_0) \sum_{l=b}^{j-1} \binom{j-1}{l} [\frac{1-(1-2p_i)^{k-1}}{2}]^l [\frac{1+(1-2p_i)^{k-1}}{2}]^{j-1-l}$$

Minimize the error probability with optimum b

- The solution to this minimization is the smallest integer b for which

$$\frac{1-p_0}{p_0} \leq [\frac{1+(1-2p_i)^{k-1}}{1-(1-2p_i)^{k-1}}]^{2b-j+1}$$

As p_i decreases, b decreases

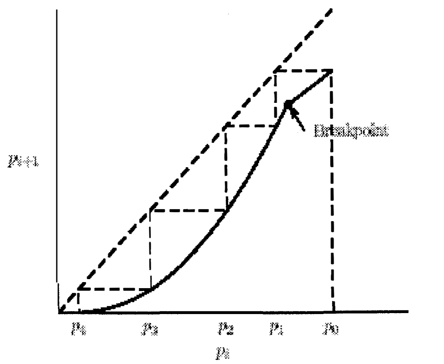
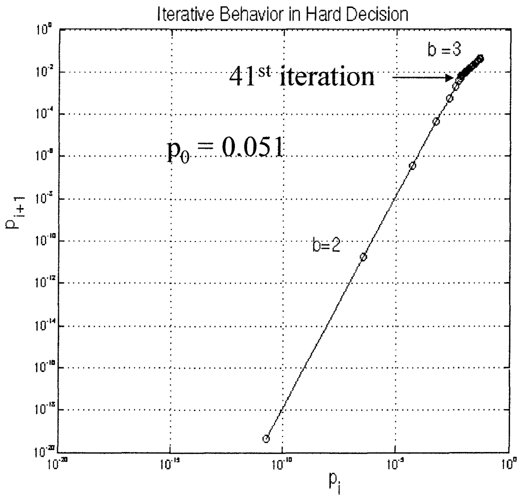


Figure 4.5: Behavior of decoding iterations for $j > 3$.

Change of Slope (j=4, k=8)

Change of slope occurs at 41st iteration



Improved LDPC codes using Irregular Graphs (Take this with a risk)

- The number of total edges should be the same.
- At a bit node, more checks more reliable message it can generate (from our examples, not true in general, only from j=3 to j=4).
- At a check node, a less number of bit nodes means the more valuable message it can generate and pass it to the associated bit nodes.
- Competing requirements
  - Irregular structure provides more flexibility, leading to a better performance.

From Experiments (take this with a risk)

- Higher degree bit nodes -- connecting to a more number of check nodes -- tend to lead quicker correction capability.
- These higher degree nodes provide better information to associated check nodes.
- These checks subsequently provide reliable check to lower degree bit nodes.
- With irregular LDPC codes, the LDPC codes was shown to give performance better than the turbo codes.

Threshold for AWGN Channel with Probabilistic Decoding

- If we know the distribution of the i-th log ratio, we can define the error probability.
- Use the consistent Gaussian density evolution
  - Only needs to know the mean (or the variance)
- Use convolution in time is multiplication in frequency domain
  - Convolution of pdfs is multiplication in characteristic functions (Fourier transform of pdfs)
- Calculate the mean values for each and every tiers, and find the probability of error at the last tier.

Agenda

- Density Evolution on the LDPC code graph

Log Ratio Algorithm

- Take the log of the ratio of the posteriors

$$log \frac{Pr(x_d = 1|\mathbf{y}, S)}{Pr(x_d = 0|\mathbf{y}, S)} = log \frac{p_d}{1-p_d} + \sum_{i=1}^{j} log \frac{1 - \prod_{l=1}^{k-1}(1-2p_{il})}{1 + \prod_{l=1}^{k-1}(1-2p_{il})}$$

- Using $tanh(\frac{x}{2}) = \frac{e^x - 1}{e^x + 1}$, the summand of the second term is

$$log \frac{1-(-1)^{k-1}\prod_{l=1}^{k-1} tanh(\frac{LR(p_{il})}{2})}{1+(-1)^{k-1}\prod_{l=1}^{k-1} tanh(\frac{LR(p_{il})}{2})} = log \frac{1+(-1)^{k}\prod_{l=1}^{k-1} tanh(\frac{LR(p_{il})}{2})}{1-(-1)^{k}\prod_{l=1}^{k-1} tanh(\frac{LR(p_{il})}{2})}$$

- Making use of $tanh^{-1}(x) = \frac{1}{2} log \frac{1+x}{1-x}$, it becomes

$$\sum_{i=1}^{j} 2\, tanh^{-1}((-1)^k \prod_{l=1}^{k-1} tanh(\frac{LR(p_{il})}{2}))$$

$$= \sum_{i=1}^{j} (-1)^k\, 2\, tanh^{-1}(\prod_{l=1}^{k-1} \cancel{tanh}(\frac{LR(p_{il})}{2}))$$

Making use of $tanh^{-1}$ being odd function

Product of Real Numbers

- $\prod_i \alpha_i = [\prod_i sign(\alpha_i)] \cdot exp(\sum_i log(|\alpha_i|))$
- $a\ b$ = sign(a) sign(b) exp(log(|a|)) exp(log(|b|))
- $\prod_{l=1}^{k-1} tanh(\frac{\alpha(p_{il})}{2})$

$$= [\prod_{l=1}^{k-1} sign(LR(p_{il}))] \cdot exp(\sum_{l=1}^{k-1} log(tanh(\frac{|LR(p_{il})|}{2}))$$

f(x) := -log(tanh(x/2)) = $log\frac{e^x+1}{e^x-1}$

- Use the identity of product of real numbers to get rid of product

$$\sum_{i=1}^{j} (-1)^k\, 2\, tanh^{-1}(\prod_{l=1}^{k-1} tanh(\frac{LR(p_{il})}{2}))$$

$$= \sum_{i=1}^{j} (-1)^k [\prod_{l=1}^{k-1} sign(LR(p_{il}))]\, 2\, tanh^{-1}[exp(\sum_{l=1}^{k-1} log(tanh(\frac{|LR(p_{il})|}{2}))]$$

$$= \sum_{i=1}^{j} \underbrace{[\prod_{l=1}^{k-1} sign(LR(p_{il}))] \cdot f^{-1}(\sum_{l=1}^{k-1} f(|LR(p_{il})|)}_{\text{Information generated by the i-th check node}}$$

Log ratio: info. from bit nodes

Getting Rid of $(-1)^k$ Term

- We can get rid of $(-1)^k$ term in the right side of Theorem 4.1 by defining

  $f(x) := -\log(\tanh(x/2)$, $\underline{x \geq 0}$

Finally, the Log Ratio Algorithm

- Note the ratio here is Pr(x=1)/Pr(x=0), which is the inverse of the ratio used in Gallagar's thesis
- With the following definitions
  - $LR(p_d) := log\frac{p_d}{1-p_d}$ $\quad LR(p_{il}) := log\frac{p_{il}}{1-p_{il}}$
  - $LR(p'_d) := log\frac{Pr(x_d=1|S,\mathrm{y})}{Pr(x_d=0|S,\mathrm{y})}$
- Theorem 4.1 becomes

$$LR(p'_d) := LR(p_d) + \sum_{i=1}^{j}[\prod_{l=1}^{k-1} sign(LR(p_{il}))] f[\sum_{l=1}^{k-1} f(|LR(p_{il})|)]$$

Algorithm with Summations
(n, j, k) code with $f(x) := -\log(\tanh(x/2))U(x)$

- Initialize:
  - $LR(f_t)=(4E_s/N_0)y_t$,
  - $LR(r_{tl})=0$, $t=1, 2, \ldots, n$ and $l=1,2,\ldots, k$
- Iteration:
  - Bit-to-Check messages: $LR(q_{t,Q1(m,t)})$, $t=1,2,\ldots, n$; $m=1,2, \ldots, j$

    $LR(q_{t,Q1(m,t)}) = LR(f_t) + \sum_{m' \neq m} LR(r_{t,Q1(m',t)})$
  - Check-to-Bit messages: $LR(r_{Q2(m,l),l})$, $l=1,2,\ldots,L$; $m=1,2,\ldots, k$

    $LR(r_{Q2(m,l),l}) = [\prod_{m' \neq m} \text{sign}(LR(q_{Q2(m',l),l}))]$

    $\cdot [f^{-1}[\sum_{m' \neq m} f(|LR(q_{Q2(m',l),l})|)]$
- Output:
  - $LR(p_t) = LR(f_t) + \sum_m LR(r_{t,Q1(m,t)})$

Distribution of a RV defined by the Sum of Random Variables

- Consider a random variable Z which is defined to be the sum of two independent random variables X and Y.
- Given the distributions of X and Y, say with pdf (or pmf), $p_X(x)$ and $p_Y(y)$, we can find the distribution of Z.
- $p(Z=X+Y=z) = E_X\{p(Y=z-X)\}=\int p_Y(z-x)\, p_X(x)\, dx$
- *Convolution* in one domain is *multiplication* in the other domain [Fourier transform].
- A characteristic function of a random variable Z is defined as

  $E\{e^{jwZ}\}$.
- Thus, we have $E\{e^{jw(X+Y)}\}= E\{e^{jwX}\}\, E\{e^{jwY}\}$.

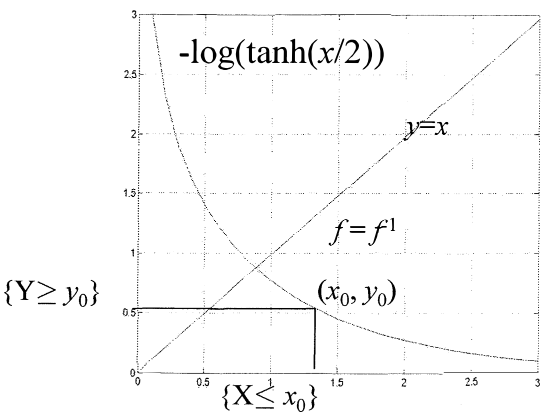
Density Evolution

- Assumption of independence: all the log-ratio variables in the algorithm are independent random variables.
- We want to be able to determine the distributions of all involved random variables as the iteration proceeds.
- Bit-to-check message
  - $A=B+(C_1+C_2+\ldots+C_{j-1})$
  - Apply the Fourier Transform to the distributions of rv's B, C_1, …, C_{k-1}, and get the distribution of A by taking the inverse Fourier Transform of the $F(p(B)) \cdot F(p(C_1)) \cdot F(p(C_1)) \cdot \ldots \cdot F(p(C_{j-1}))$
- Check-to-Bit message
  - $C = h^{-1}[h(A_1) + h(A_2) + \ldots + h(A_{k-1})]$
  - h: $R \to \{-1, 1\} \times \{z: z \geq 0\}$, such that (Sign $\times$ Magnitude)
  - Summation of functions of random variables
  - Apply the Fourier Transform to the distributions of $h(A_j)$

$h(z)$

- $h(x) := (s(x):=\text{sign}(x),\ f(x):=-\log(\tanh(x/2))),\ x \in R$

Monotone decreasing



How Do We Obtain the Distr. of h(A)?

- We have the histogram $p(x)$ of A—the distribution of A.
- How do we obtain the distribution of h(A)?

The distribution of h in terms of $F_X(x)$

- $h(x) := (S{:=}1_{\{x>0\}} - 1_{\{x\leq 0\}},\ M{:=}1_{\{x\geq 0\}} f(x) + 1_{\{x\leq 0\}} f(-x))$
- Thus, the domain of h is $\mathcal{R}$, and the ranges are $\{-1, +1\}$ and $[0, +\infty]$.
- Now consider the distribution of $\Pr\{S{=}s, M(x) \leq y\}$.
- $\Pr\{S(X){=}s, M(X) \leq y\} = \Pr\{M(X) \leq y, X > 0\} 1_{s=+1}$

  $+ \Pr\{M(X) \leq y , X \leq 0\} 1_{s=-1}$
- $\Pr\{M(x) \leq y, X > 0\} = \Pr\{f(X) \leq y,\ X > 0\} = \Pr\{X \geq f^{1}(y), X > 0\}$

  $= \Pr\{X \geq f^{-1}(y)\}$

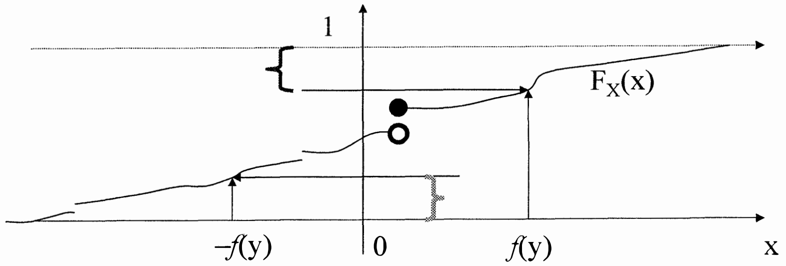
  $= 1 - F^{--}_X(f(y))$
- $\Pr\{M(x) \leq y, X \leq 0\} = \Pr\{f(-X) \leq y, X \leq 0\}$

  $= \Pr\{X \leq -f^{1}(y), X \leq 0\}$

  $= F_X(-f(y))$

Thus, we have

- $H(s, y) := \Pr\{S{=}s, M(x) \le y\} = [1 - F^{--}_X(f(y))]\ 1_{s=+1} + F_X(-f(y))1_{s=-1}$
  $= H^1\ 1_{s=+1} + H^2\ 1_{s=-1}$
- Check if it is a legitimate distribution:
  - $\Pr\{S{=}{+}1, y{=}\infty\} + \Pr\{S{=}{-}1, y{=}\infty\}{=}1$?
  - As $y \Rightarrow \infty$, $f(y) = 0$.



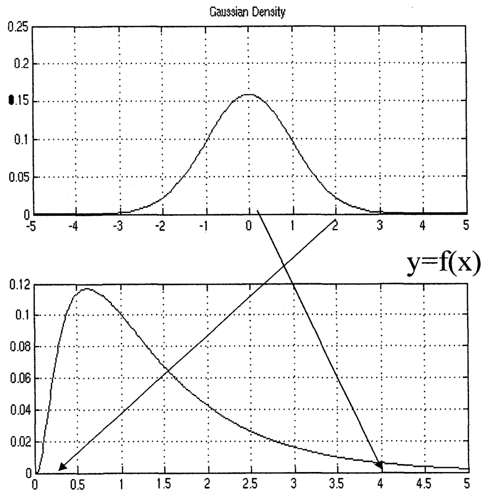
Take the derivative to get the density

- $H(s, y) := \Pr\{S{=}s, M(x) \le y\} = [1 - F^{--}_X(f(y))]\ 1_{s=+1} + F_X(-f(y))\ 1_{s=-1}$
  $= H^1\ 1_{s=+1} + H^2\ 1_{s=-1}$
- $dH(s, y)/dy = [1 - F^{--}_X(f(y))]/dy\ 1_{s=+1} + F_X(-f(y))/dy\ 1_{s=-1}$
- First,
  $d/dy\, f(y) = d/dy\ [(e^y+1)/(e^y - 1)] = -2/(e^y - e^{-y}) = -1/\sinh(y)$
- $[1 - F^{--}_X(f(y))]/dy$: (For this, let's assume continuous $F_X(x)$)
- $-[d/df(y)\ F_X(f(y))] \cdot [d/dy\ f(y))] = -p_X(f(y)) \cdot (-\ \ /\sinh(y))$
  $= p_X(f(y))/\sinh(y)$
- $F_X(-f(y))/dy = [d/d(-f(y))\ F_X(-f(y))] \cdot [d/dy\ (-f(y))]$
  $= p_X(-f(y))/\sinh(y)$
- Therefore,
- $p_H(s, y) = p_X(f(y))/\sinh(y)\ 1_{s=+1} + p_X(-f(y))/\sinh(y)\ 1_{s=-1}$

How about inverse function H^{-1} from (S=s, M=y) to x

- Domain: ($s \in \{-1,1\}$, $M=y \in [0, \infty]$)
- Range: $x=sy \in [-\infty, +\infty]$
- $H^{-1}(x) = H^1(f(x))\, 1_{\{x>0\}} + H^2(f(-x))\, 1_{\{x \le 0\}}$

Gaussian Density



p(x)

x=2.0, y=0.27
x=0.1, y=3.00

p(S=+1, y)

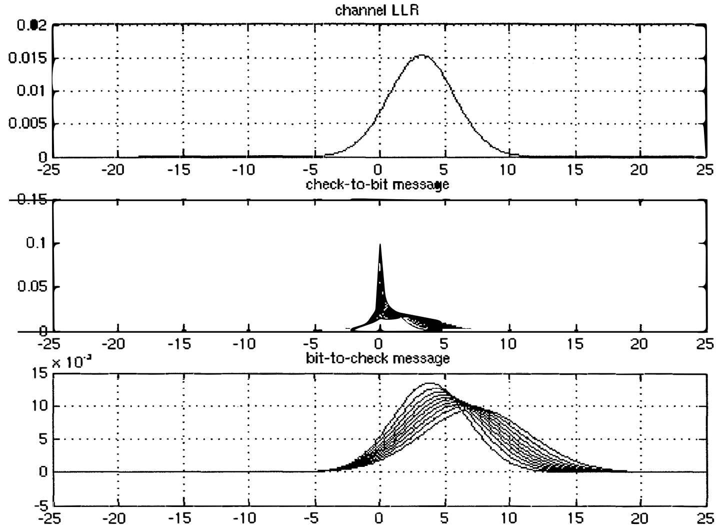
Convolution of Two Distributions

- $H_1 * H_2 = [H_1^1 * H_2^1 + H_1^2 * H_2^2]\, 1_{s=+1} + [H_1^1 * H_2^2 + H_1^2 * H_2^1]\, 1_{s=-1}$
- We can make use of the densities, dH
  - The derivative operator commutes with summation and integrals

The Algorithm

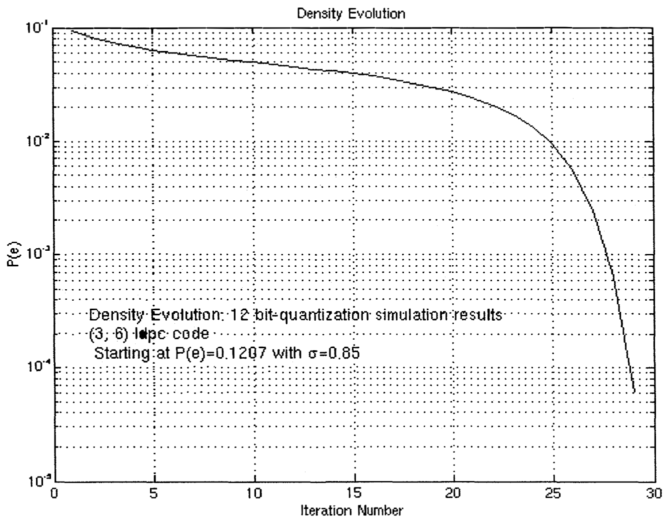
- F here implies the Fourier Transform
- $F(p(A))=F(p(B)) \cdot F(p(C_1)) \cdot F(p(C_1)) \cdot \ldots \cdot F(p(C_{j-1}))$
- $C_i = h^{-1}[h(A_1) + h(A_2) + \ldots + h(A_{k-1})]$
- The distribution of C can be obtained from
  $p(c) = dH^{-1}[F^{-1}\{F(dH_1)\cdot F(dH_2) \cdot \ldots \cdot F(dH_{k-1})\}]$

Graphical Illustration of Density Evolution Algorithm



Using The idea Given In SY Chung (Comm. Letter 01')

P(e) vs. Iteration Number in Density Evolution



Density Evolution

- Making use of Gaussian approximation (Sae Young Chung and Forney, IT 2001)
  - Looks like the fastest DE algorithm
  - Recent one by Jin and Richardson: not clearly written
- These are only for fast calculations

Summary

- Density evolution can determine the threshold.
- Density evolution idea is currently used in many areas
  - Coded Modulation for MIMO channel
  - Compressive sensing
  - Joint equalization and decoding

The Final Exam

- Dec. 15th : 10:30am – 12:30pm.
- Coverage: Entire course materials

The Term Project
(Due by Dec. 17th)

- Choose a topic of your own and submit a paragraph (less than 10 sentences) by next class
  - Topic sentence of your term project
  - Objective : what are you aiming to achieve in your project.
  - Expected results
  - Tasks: what you need to do to

- List of possible topics (simulation and verification)
  - Turbo codes
  - Reed Solomon codes
  - Correlation model (Markov chain) + LDPC codes
  - Trellis codes

- Survey paper
  - Recent advances in Reed Solomon codes (soft decoding, polynomial fitting based issues)
  - Recent advances in LDPC codes (design, decoding issues)

Fountain Codes: Viewed from the perspectives of Shannon and Gallager

Tutorial Session: 4:20pm – 5:50pm
대한전자공학회
제주 그랜드 호텔
6/17/2010

Aim of this Tutorial

- Motivated by the success of Fountain codes for internet application
- Review a few key ideas of Shannon and Gallager leading to the creation of Fountain codes.
- Possible new research directions.

Agenda

- Shannon's Channel Coding Theorem(1948)
  - Typical set
  - The idea of fan
- Gallager's Thesis 1962
  - LDPC codes
  - MAP decoder
- LDPC codes over BEC
  - Irregular LDPC codes (1998)
  - Fountain codes (1998)

Communications System Model

- Input/Output Relation of A Noisy Communication Channel

$$Y = X + N$$

- Use the channel n times.

Entropy

- Shannon introduced Entropy as Measure of Uncertainty
  - Entropy : H(X)
  - Conditional Entropy: H(X|Y) or H(Y|X)

- When channel is noisy, the conditional entropies are non-zero.

- Reliable communication is possible over a noisy channel iff the transmission rate is smaller than H(X) – H(X|Y).
  - Showed it is possible to find a code so that

$$P(e) \rightarrow 0 \text{ as long as rate} < H(X) - H(X|Y)$$

Meaning of Entropy

- Uncertainty = Amount of Information = The number of bits needed.

- An information source with large uncertainty produces a large **amount of information**.
  - Weather forecast in LA vs. Weather forecast in Pittsburgh

- A channel with strong noise causes large **ambiguity**.

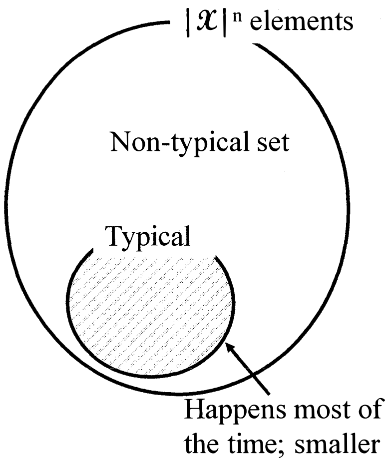
Entropy and the Law of Large Number

- Let X be a 1-0 Bernoulli with 1(or error) appearing with prob. 1/10.
  - H(X) = 0.47 from previous page.
- Consider a sequence of X's of length n, $(X_1, X_2, \ldots, X_n)$.
- For a large n, due to the LLN, the set of sequences can be divided into two exclusive sets.
  - A *typical* set of sequences which occur in real experiment
  - An *atypical* set of sequences which almost never occur
- Shannon noted that the size of the typical set is $2^{nH(X)}$.

Typical Set

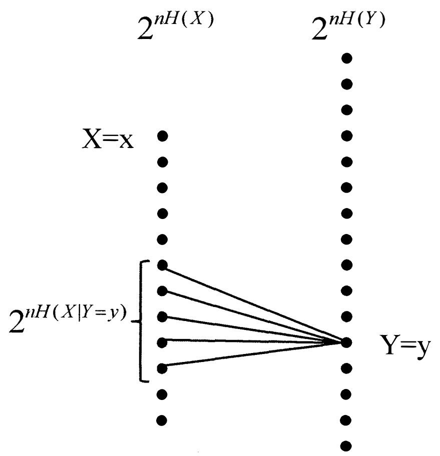
- Consider binary sequences of length 100 (n = 100).
- The number of 1s you see typically is 10.
  - Typical sequence happens with probability close to 1.
  - Non typical sequence happens very rarely. (Law of large numbers)
    - A chance to see the all 1 sequence?
- The size of typical set is

$$2^{nH(X)}$$



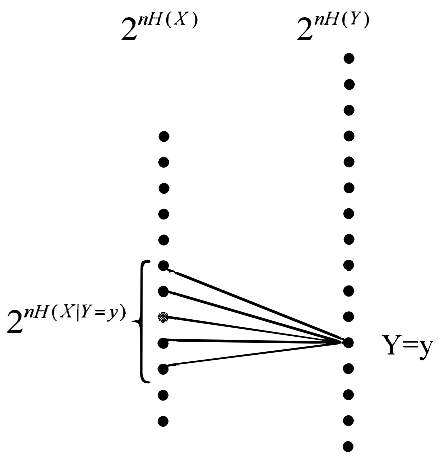
Shannon's Key Idea: P(e) in Random Codebook Construction

- Let's select the message set(a codebook) *randomly*.
- And, see if we can make P(e) very small.
- Given a fan of size $2^{nH(X|Y=y)}$, decoding error occurs if there are more than one messages.
  - See the analysis in the following page



Shannon's Key Idea: P(e) in Random Codebook Construction (2)

- Steps:
  - Select the first message (the red dot) and send.
  - With probability close to 1, we get the typical output *y*.
  - Randomly select the rest of the messages.
  - Consider the *fan* of *y* and find out the probability of decoding error.
  - Decoding error occurs when any one of the other $2^{nR} - 1$ messages is selected inside the fan.
- So, let's obtain the decoding error probability P(e).



P(e) in Random Codebook Construction (3)

- $$P(e) = 1 - \left(1 - \frac{2^{nH(X|Y)}}{2^{nH(X)}}\right)^{2^{nR}-1}$$

  The probability that a message selection is made within the fan

  $$\le 1 - \left(1 - 2^{-n[H(X)-H(X|Y)]}\right)^{2^{nR}}$$

  $$\approx 1 - \left(1 - 2^{nR}2^{-n[H(X)-H(X|Y)]}\right)$$

  $$= 2^{-n[I(X;Y)-R]}$$

- Thus, if R is chosen slightly smaller than I(X; Y), P(e) decreases to zero as n increases.
  - Now we maximize I(X; Y) by selecting the best input distribution, and obtain the capacity, $C = \max_{p(x)} I(X;Y)$.

- Note that the Shannon's capacity theorem is proved!

"*Information*" Channel Capacity

- $C = \max_{p(x)} I(X;Y)$
  - The maximum is taken over all input distr. p(x)

- I(X; Y) is the mutual information between X and Y
  - I(X;Y) = H(X) – H(X|Y),

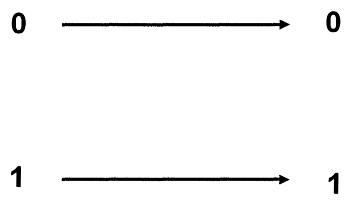
    ---- maximum input-size which causes no equivocation on X given an output Y

    = H(Y) – H(Y|X)

    ---- maximum output-size which causes no uncertainty on Y given an input X
  - H(X) = Amount of information that can be carried by X
  - H(X|Y) = Amount of ambiguity caused by channel noise

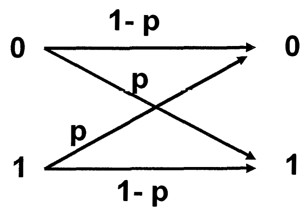
Noiseless Binary Channel

- $C = \max_{p(x)} I(X; Y) = ?$



Binary Symmetric Channel

- Transmitted bits are flipped with prob. p.
- $I(X; Y) = H(Y) - H(Y|X)$
  $= H(Y) - \sum p(x) H(Y|X=x)$
  $= H(Y) - H(p)$
  $\leq 1 - H(p)$



- Equality with uniform X due to symmetry

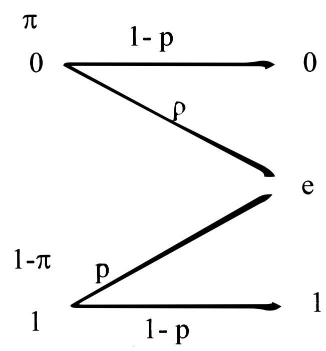
- $C = 1 - H(p)$ bits

Binary Erasure Channel

- A transmitted bit gets lost (no decision) with probability p.

- $C = \max I(X ; Y)$
  $= \max_\pi H(X) - H(X|Y)$
  $= \max_\pi H(\pi) - pH(X|Y = e)$
  $= \max_\pi H(\pi) - p\,H(\pi)$
  $= 1 - p$

- Capacity is achieved if $\pi = 1/2$.



Capacity of the AWGN channel

- P = Signal Energy per Channel Use
- N = Noise Energy per Channel Use
- The channel C is then given by

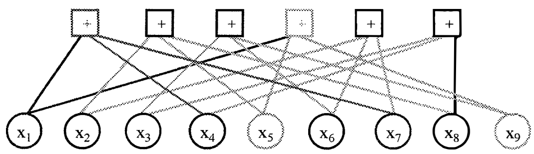
$$C = 0.5 \log2(1 + SNR)$$

Gallager's Thesis ('62)

- (n, d_v, d_c) low density parity check code.
- Parity check matrix H $[n(1\text{-}R) \times n]$ of the code has
  - d_v number of 1's in each column
  - d_c number of 1's in each row
  - The code rate is $R = 1 - d_v / d_c$
- Min. distance of a typical (n, d_v, d_c) code for $d_v \geq 3$
  - increases linearly with n for fixed j & k.
- Practical decoding methods exist
  - Simple or probabilistic

Parity Check Matrix on Bipartite Graph

$$\begin{bmatrix} 1&0&0&1&0&0&1&0&0 \\ 0&1&0&0&1&0&0&1&0 \\ 0&0&1&0&0&1&0&0&1 \\ 1&0&0&0&1&0&0&0&1 \\ 0&1&0&0&0&1&1&0&0 \\ 0&0&1&1&0&0&0&1&0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_9 \end{bmatrix} = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_6 \end{bmatrix}$$



$$\begin{aligned} R &= 1 - d_v / d_c \\ &= 1\text{-}2/3 \\ &= 1/3 \end{aligned}$$

Probabilistic Decoding

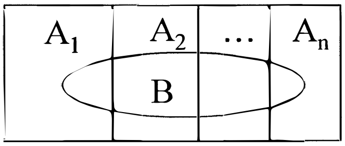
- Total Probability: If $A=\{A_1, A_2, \ldots, A_n\}$ is a partition of S and B is an arbitrary event

$$\Pr\{B\} = \Sigma^{n}_{i=1}\Pr\{B \cap A_i\} = \Sigma^{n}_{i=1}\Pr\{B \mid A_i\}\Pr\{A_i\}$$

- Bayes' Theorem: We know

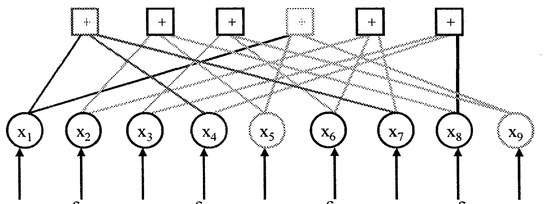
$$Pr\{A_i|B\} = \frac{Pr\{A_i \cap B\}}{Pr\{B\}}$$

- The posterior is The likelihood x the prior



$$Pr(A_i|B) = \frac{Pr(B|A_i)Pr(A_i)}{\sum_{i=1}^{n} Pr(B|A_i)Pr(A_i)}$$

The Iterative Decoding Theorem
The General Case (1)



General Decoding Theorem

1. Get the input probabilities from the channel output $y_k = (2x_k - 1) + n_k,\ n_k \sim \mathcal{N}(0, N_0/(2E_s))$.

   Let $f_k \triangleq \ln \frac{p(y_k \mid x_k = 1)}{p(y_k \mid x_k = 0)}$ which is the input to the iterative decoder

2. Improve the input via the log ratio of posteriors $p_k \triangleq \ln \frac{\mathrm{P}(x_1 = 1 \mid \mathbf{y}, S)}{\mathrm{P}(x_1 = 0 \mid \mathbf{y}, S)}$ for all k.

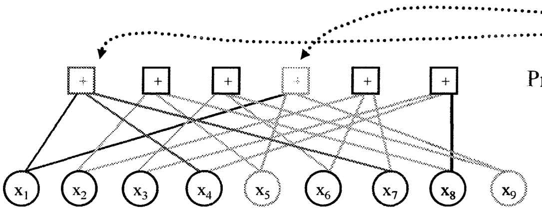
Bayes' Theorem

$$
\begin{aligned}
Pr(x_d = 1|\mathbf{y}, S) &= \frac{Pr(x_d = 1, \mathbf{y}, S)}{p(\mathbf{y}, S)} \\
&= \frac{Pr(S|x_d = 1, \mathbf{y})\ p(x_d = 1, \mathbf{y})}{p(\mathbf{y}, S)} \\
&= \frac{Pr(S|x_d = 1, \mathbf{y})\ Pr(x_d = 1|\mathbf{y})\ p(\mathbf{y})}{p(\mathbf{y}, S)}
\end{aligned}
$$

- The ratio of posteriors is of our interest

$$\frac{P(x_1 = 1 \mid \mathbf{y}, S)}{P(x_1 = 0 \mid \mathbf{y}, S)} = \frac{P(S \mid x_1 = 1, \mathbf{y}) P(x_1 = 1 \mid \mathbf{y})}{P(S \mid x_1 = 0, \mathbf{y}) P(x_1 = 1 \mid \mathbf{y})}$$

The Iterative Decoding Theorem: The General Case (2)



Pr(S|x$_1$=0) = Pr(S$_1$|x$_1$=0) Pr(S$_4$|x$_1$=0)

Pr(x$_1$=0| S)   9 bit nodes

Use f_1

General Decoding Theorem, $k = 1$ case.

$$p_1 \triangleq \ln \frac{\mathrm{P}(x_1 = 1 \mid \mathbf{y}, S)}{\mathrm{P}(x_1 = 0 \mid \mathbf{y}, S)} = \ln \frac{P(S \mid x_1 = 1, \mathbf{y})}{P(S \mid x_1 = 0, \mathbf{y})} + \ln \frac{P(x_1 = 1 \mid \mathbf{y})}{P(x_1 = 0 \mid \mathbf{y})}$$

Similarly, we can update all p_k's.

Given $p_{k,1} \triangleq \mathrm{P}(x_k = 1 \mid \mathbf{y}, S) = \dfrac{e^{p_k}}{e^{p_k} + 1}$, we can find $P(S \mid x_1 = 1, \mathbf{y})$ and $P(S \mid x_1 = 0, \mathbf{y})$:

$$
\begin{aligned}
P(S \mid x_1 = 1, \mathbf{y}) &= \Pr\{\text{odd \# of 1s in } x_4 \text{ and } x_7\} \times \Pr\{\text{odd \# of 1s in } x_5 \text{ and } x_9\} \\
&= \{p_{4,1}(1 - p_{7,1}) + (1 - p_{4,1}) p_{7,1}\} \times \{p_{5,1}(1 - p_{9,1}) + (1 - p_{5,1}) p_{9,1}\}
\end{aligned}
$$

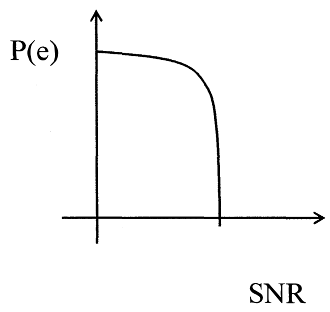
This is the general decoding theorem of Gallager.

The Iterative Decoding Theorem: The General Case (3)

- The log ratio p_k has two parts
  - The sign of p_k ~ + / -
  - The magnitude $|p_k|$ ~ reliability of the sign
- When the channel is BSC(p),
  - The magnitude $|p_k|$ is infinite
  - The sign is in error with probability p.
- When the channel is BEC(p),
  - The magnitude is 0 (erasure) and infinite (sure).
  - There are three kinds of signs, 0, 1 or erasure.

Threshold Phenomenon for AWGN channel

- There is a certain threshold value associated with a (n, d_v, d_c) LDPC code.



LDPC Codes/Decoding over BSC(p_0)

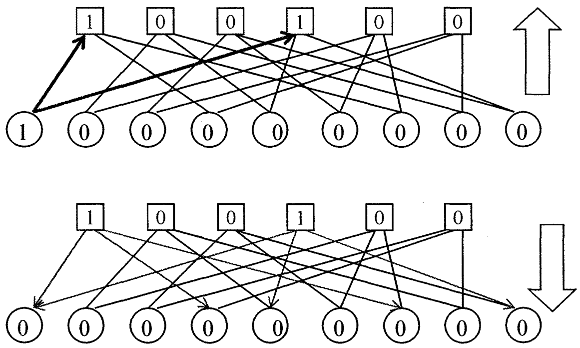
- Decoding for Binary Symmetric Channel (p_0)
  - The probability of bit error: p_0
- Majority Rule Decoding
- Threshold Effect
- Density Evolution

Shannon's Capacity Theorem

- $C = 1 - H(p_0)$ and $R < C$ for $P(e) \sim 0$.
- Use the BSC channel *n* times
  - n: the block length
  - k: the message length
  - r: the number of parities
  - $n = k + r$
  - $R = k/n$
- Then, $nR<nC$ says we should let
  - $k = n - r < n - nH(p_0)$ or
  - $r > nH(p_0)$

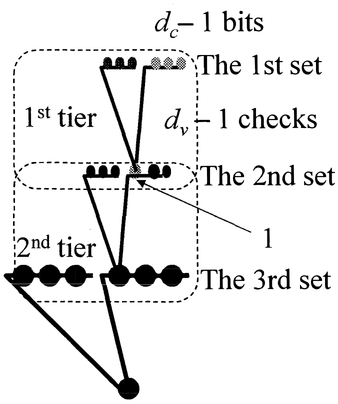
Recall, Simple Decoding Example (n=9, d_v=2, d_c=3)

- Suppose we have
  r=[1 0 0 0 0 0 0 0 0]
- Majority Rule decoding:
  - If more than or equal to m checks are violated, flip the bit.
- In this example, let m = 2. Then, we note, the first error gets corrected.
- Let's consider more realistic cases.



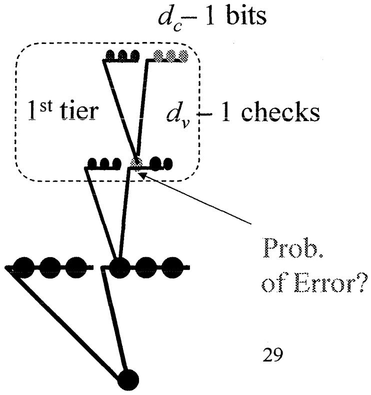
Density Evolution (d_v=3, d_c=4)

- BSC with cross-over probability p_0.
- DE is the evolution of error prob. as iteration increases.
- Majority Rule decoding:
  - If both checks are unsatisfied, change the bit.
- Assume decoding on a tree (cycle free graph) which gives the effect of infinite length n



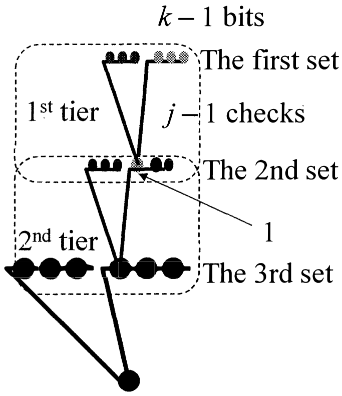
Density Evolution

- Suppose we start the decoding iteration with the input of bit error probability p_0.
- Our aim is to see if the bit error probability gets smaller as iterations proceed.
  - The first tier calculation involves the first and the second sets of nodes.
  - The 2nd tier calculation involves the second and the third sets of digits (bit nodes).
  - And so on.
- Each tier calculation in the tree implies a decoding iteration in the graph.



Density Evolution

- Now suppose that the red node is in error.
- Then, each check constraining the red would be violated if there were even number of errors in the (k-1) digits in the first set.
- Thc probability of such an event is $0.5(1+(1 - 2p_0)^{k-1})$
  - Note $((1- p_o) + p_o t)^{k-1} + ((1- p_o) - p_o t)^{k-1}$, evaluated at $t = 1$, will give the probability of even 1s.
- The error at the red node will be corrected when both checks are unsatisfied.



Density Evolution (2)

- Thus, the probability that a digit is received in error at the first tier, and then corrected after the first iteration is

$$p_0[\underbrace{0.5(1 + (1 - 2p_0)^{k-1})}_{\text{Even number of errors in } (k-1) \text{ digits}}]^2$$

The red node was on error with this prob.

Both checks

- Then, the probability that the red remains in error is

$$p_0\{1- [0.5(1 + (1 - 2p_0)^{k-1})]^2\}$$

Density Evolution (3)

- Now consider the situation when the probability of a digit is received correctly, but changed due to both checks violated

$$(1- p_0)[0.5(1 - (1 - 2p_0)^{k-1})]^2$$

Received correctly

Odd number of errors in (k-1) bits

Both checks

Density Evolution (4)

- Now, let's put them together
- A bit error probability at the second set is determined by
  - $p_1 = p_0(1 - [0.5(1 + (1\text{-}2p_0)^{k-1})]^2) + (1\text{-}\ p_0)[0.5(1 - (1\text{-}2p_0)^{k-1})]^2$

    {Error occurred & not corrected} OR {No error & flipped}

- A bit error in the third set is again p_0 .
- At the end of 2^{nd} tier calculation, a bit error in the third set is determined by
  - $p_2 = p_0(1 - [0.5(1 + (1\text{-}2p_1)^{k-1})]^2) + (1\text{-}\ p_0)[0.5(1 - (1\text{-}2p_1)^{k-1})]^2$

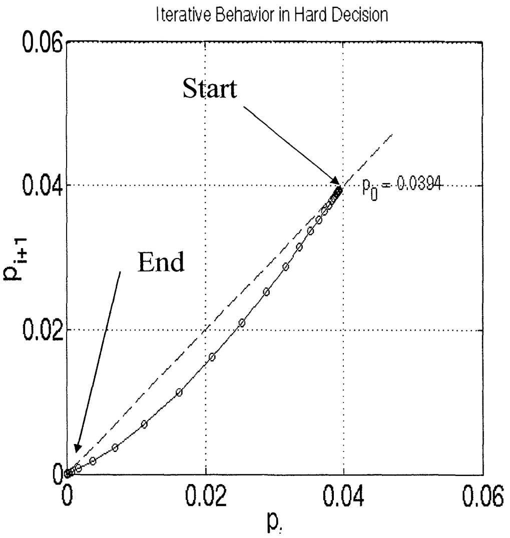
Density Evolution (5)

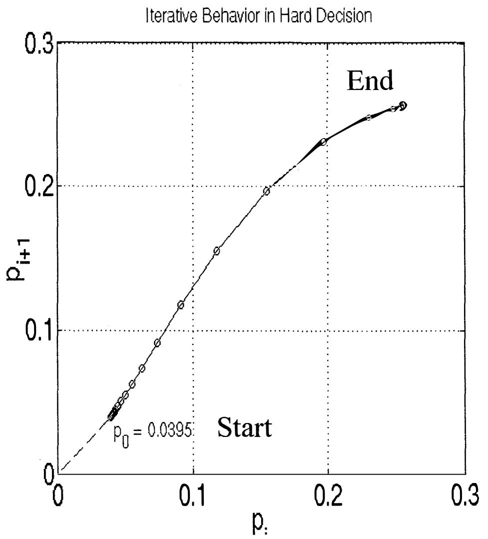
- The error probability of a bit in the (i+1)-th set, obtained at the end of the i-th tier calculation is
  - $p_i = p_0(1 - [0.5(1 + (1\text{-}2p_{i-1})^{k-1})]^2) + (1\text{-}\ p_0)[0.5(1 - (1\text{-}2p_{i-1})^{k-1})]^2$

Threshold Behavior

- $\{p_i\}$ converges to a number $0 \le c < 1$
- We want to find $p_{max} := \max p_0$ such that c is arbitrarily small.
- If $p_0 \le p_{max}$, then $\{p_i\}$ converges to zero.
- If $p_0 > p_{max}$, then $\{p_i\}$ converges to a non-zero positive constant < 1.

$\{p_j\}$ converges to zero if $p_0 \le p_{max} = 0.0394$ for $d_v=3$, $d_c=6$





Examples

- Table in the left lists of maximum p_0 resulting in $p_{100} < 1e\text{-}6$.
- Compare the rate ½ codes
  - $d_v = 4$ is the best.
- As the rate decreases, p_{max} increases.

d_v	d_c	Rate	Maximum p_0
3	6	1/2	0.039
3	5	2/5	0.061
3	4	1/4	0.106
4	8	1/2	0.051
4	6	1/3	0.074
4	5	1/5	0.095
5	10	1/2	0.041
5	8	3/8	0.056
5	6	1/6	0.086

Usage of the Given Analysis

- Application of DE to other channels
  - Density evolution for AWGN case
  - Density evolution for BEC
- Applications to
  - Code Design (Code ensemble search)
  - Decoder Design (Change *m* as iteration proceeds)

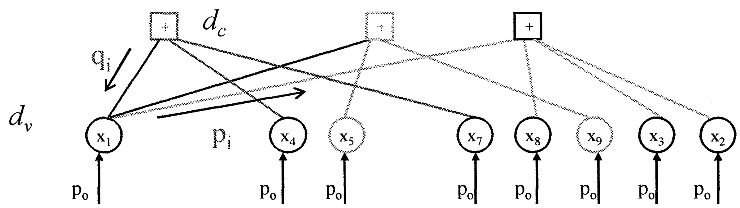
DE for BEC(p_o)

- When the channel is BEC(p_o)
  - The magnitude is 0 (erasure) and infinite (sure).
  - There are three kinds of signs, 0, 1 or erasure.

- A check with two erasures are useless (no information)

- A bit node with any check with non erasures is deterministic.

Density Evolution for BEC(p_o)



- Let p_i, q_i denote the probability of erasure, for b2c and c2b directions, respectively at round i.
- b2c: For an erasure b2c output, all inputs to the bit node should be erasures.
  $$p_{i+1} = p_o \cdot q_i^{d_v - 1}$$
- c2b: If any input to the check node is an erasure, then the output is an erasure too. $q_i = 1 - (1 - p_i)^{d_c - 1}$

- DE: $p_{i+1} = p_o \cdot \left(1 - (1 - p_i)^{d_c - 1}\right)^{d_v - 1}$

An Optimal Code Ensemble Search

- There is a threshold value p_o^* for an ensemble of LDPC codes with fixed d_v and d_c.
  - If $p_o < p_o^*$, p_i converges to zero.
- They have used it to find the best LDPC code ensembles.
  - Application to irregular LDPC code:
    - Use auxiliary poly's.

$$\lambda(x) = \sum_{i=1}^{d_v^{\max}} \lambda_i x^{i-1}$$

Fraction of b-nodes with degree i

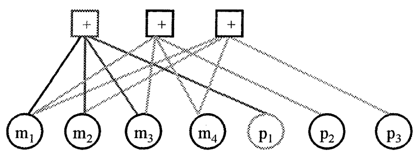
$$\rho(x) = \sum_{i=1}^{d_c^{\max}} \rho_i x^{i-1}$$

Fraction of c-nodes with degree i

- DE is then $p_{i+1} = p \cdot \lambda(1-\rho(1-p_i))$

Fountain Code

- A pre-cursor to fountain code I intend to use is an LDPC code in systematic form.
  - Any LDPC code has its systematic form via Gaussian elimination on H.



Message part Parity part

$$Hx = \left(\underbrace{\begin{matrix}1&1&1&0\\1&0&1&1\\1&1&0&1\end{matrix}}_{G}\;\underbrace{\begin{matrix}1&0&0\\0&1&0\\0&0&1\end{matrix}}_{I}\right)\begin{pmatrix}\mathbf{m}\\ \mathbf{p}\end{pmatrix} = 0$$

$$\Downarrow$$

$$G\,\mathbf{m} = \mathbf{p}$$

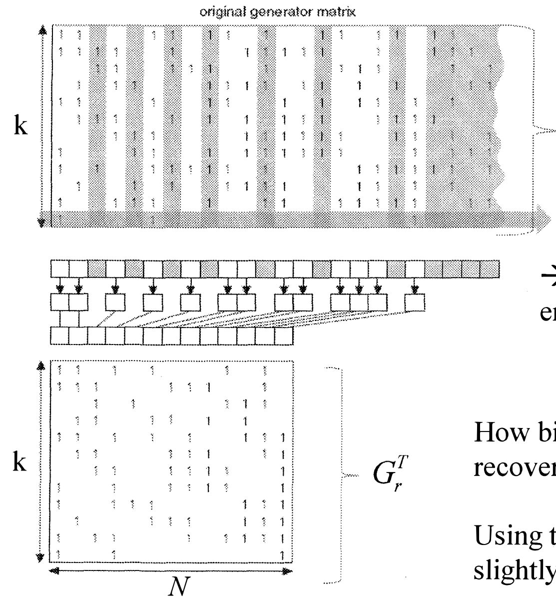
Fountain Code (2)

- $\mathbf{p} = G\mathbf{m}$ where G is a [r x k] binary matrix of 1/0s.
- Can we send the parity vector **p** over a BEC(p_o) and expect to recover **m**?
  - Some p-bits are erased with prob p_o.
  - With the non erased p-bits, we can construct $\mathbf{p} = G_r\mathbf{m}$ at Rx.
  - Yes, as long as G_r has at least k independent rows, G_r is full rank.
  - Thus, $r >= k$.
  - Now, the question is to find the redundancy rate r such that P(e) is very small.
- Assume G is constructed randomly.

Shannon's Capacity Theorem

- $C = 1 - p_0$ and $R < C$ for $P(e) \sim 0$.
- Use the BEC(p_0) channel n times
  - Let n the block length
  - Let k be the message length
  - Let r be the number of parities
  - $n = k + r$
  - $R = k/n$
- Then, nR<nC says we should let
  - $k = n - r < n - np_0$ or
  - $r > np_0$

Illustration of Fountain Codes



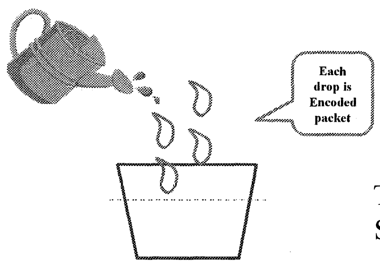
→ The grey p-bits are erased in the channel.

How big N has to be, for recovery of $\mathbf{m}$?

Using the LLN, N only needs to be slightly larger than k.

Random Fountain Codes

- Simple concept of fountain codes



Total size of source file : k x l
Size of each drop : l

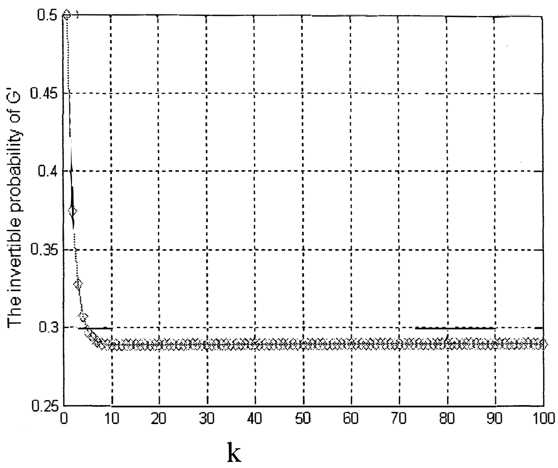
Anyone who wants to receive the source file, should hold a bucket under the fountain and collect $k + E$ drops.

Random Fountain Codes - Decoding

- In order to find the inverse G_r^{-1}, we need at least k independent rows in G_r.

- if $N=k$, what is the probability that a random k x k binary matrix, **G’,** is invertible ?

Random Fountain Codes: P(e)

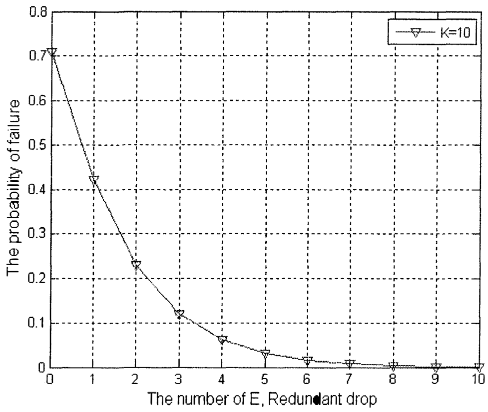


It converges at P=0.289.

→ What if N is slight greater than K?

Random Fountain Codes: P(e)

If we add a few redundancy bits, N = k + E, P → 0.



Random Fountain Codes

- As k increases, we can show that, a small fraction of E/k is suffice for near perfect decoding.
- However, the random fountain codes incur high decoding and encoding cost.

Random Fountain Codes

- Strength

  1) Rateless

  2) The number of received symbols, N, determined on the fly.

- Weakness

  High decoding cost because Gaussian Elimination is used for G_r inverse.

  - Decoding Cost : k^3 per one symbol

Practical Fountain Codes

- Luby Transform code
  - The first practical fountain code
  - Uses a sparse graph
  - encoding and decoding costs are low
- Raptor code
  - LDPC code + LT code
  - Linear encoding and decoding cost
  - Most practical
  - Will not be discussed today, due to time constraint

Luby Transform codes

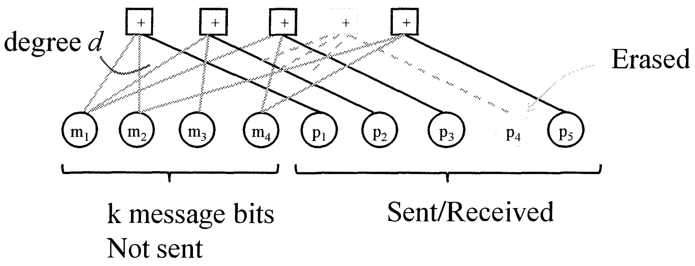
- LT code does encoding/decoding, **p** = G**m**, on a sparse G.

Encoding algorithm

1) Generate a number *d* from a given degree distribution for a check.
2) Choose *d* input symbols at random and connect them to the check
3) Repeat for each check

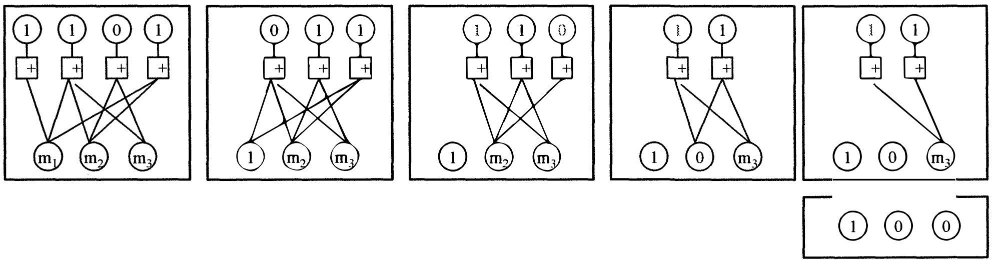
Luby Transform codes

- Luby Transform code on a sparse graph.
- Message bits are not sent; only parity bits are.
- Decode using the message passing algorithm.
- To ensure at least k + E bits be received at Rx, we need to send $N > (k + E)/p_o$ p-bits at Tx.



Luby Transform codes: Decoding

- Decoding algorithm is nothing but the MP algorithm.
- Apply the received p-bits and simplify the graph.
- Decoding progresses if there exists at least one degree-one node at a stage; otherwise, the algorithm gets stuck.



→ How to guarantee the all input symbols are covered to the graph?

→ How to guarantee the existence of degree one node at each stage?

Luby-Transform codes: Degree Distribution

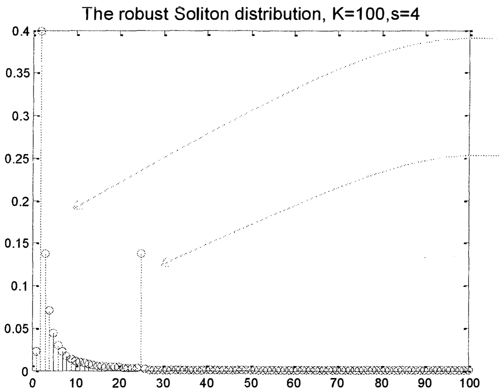
- It needs to strike a balance between the two objectives.

- The degree distribution should provide
  - Coverage: each message bit should be checked at least once by a p-bit.
    - Some connections should be **dense** for this.
  - At least one degree-1 node each stage
    - Connections should be **sparse** for this.

Luby-Transform codes: Degree Distribution (2)

- Luby et. al found the "Soliton distribution."

Fraction of checks



degree

Mostly low degree checks

Add a few high degree checks to ensure coverage

Summary

- Reviewed a few key results of Shannon and Gallager leading to Fountain Codes

References

1. [Shannon48] C.E. Shannon's 1948 paper
2. [Gallager62] R. Gallager, Low Density Parity-Check Codes. Cambridge, MA: MIT Press, 1962.
3. [Luby01] M. Luby et. al, "Efficient erasure correcting codes," IEEE Tran. IT, 2001.
4. [Shok01] A. Shokrollahi and R. Urbanke, "Finite length analysis of a certain class of LDPC codes," IEEE Trans. IT, 2001.
5. [MacKay05] D.J.C. MacKay, "Fountain Codes," IEE Proceedings, 2005.

Erdal Arıkan

Polar Coding

ISIT 2012 Tutorial

June 27, 2012

Preface

These notes on polar coding are prepared for a tutorial to be given at ISIT 2012. The notes are based on the author's paper "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," published in the July 2009 issue of the IEEE Transactions on Information Theory. The 2009 paper has been updated to cover two major advances that took place since the publication of that paper: exponential error bounds for polar codes and an efficient algorithm for constructing polar codes. Both of these topics are now an integral part of the core theory of polar coding. In its present form, these notes present the basic theory of polarization and polar coding in a fairly complete manner. There have been many more important advances in polar coding in the few years since the subject appeared: non-binary polarization, source polarization, multi-terminal polarization, polarization under memory, quantum polar coding, to name some. Also a large number of papers exist now on practical aspects of polar coding and their potential for applications. These subjects are not covered in these notes since the goal has been to present the basic theory within the confines of a three-hour tutorial.

Ankara, *E. Arıkan*
June 2012

Contents

Chapter 0
Preliminaries and Notation

Abstract This chapter gathers the notation and some basic facts that are used throughout.

0.1 Notation

We denote random variables (RVs) by upper-case letters, such as X, Y, and their realizations (sample values) by the corresponding lower-case letters, such as x, y. For X a RV, P_X denotes the probability assignment on X. For a joint ensemble of RVs (X,Y), $P_{X,Y}$ denotes the joint probability assignment. We use the standard notation $I(X;Y)$, $I(X;Y|Z)$ to denote the mutual information and its conditional form, respectively.

We use the notation a_1^N as shorthand for denoting a row vector $(a_1,\dots,a_N)$. Given such a vector a_1^N, we write a_i^j, $1 \le i,j \le N$, to denote the subvector $(a_i,\dots,a_j)$; if $j < i$, a_i^j is regarded as void. Given a_1^N and $\mathscr{A} \subset \{1,\dots,N\}$, we write $a_{\mathscr{A}}$ to denote the subvector $(a_i : i \in \mathscr{A})$. We write $a_{1,o}^j$ to denote the subvector with odd indices $(a_k : 1 \le k \le j;\ k \text{ odd})$. We write $a_{1,e}^j$ to denote the subvector with even indices $(a_k : 1 \le k \le j;\ k \text{ even})$. For example, for $a_1^5 = (5,4,6,2,1)$, we have $a_2^4 = (4,6,2)$, $a_{1,e}^5 = (4,2)$, $a_{1,o}^4 = (5,6)$. The notation 0_1^N is used to denote the all-zero vector.

Code constructions in these notes will be carried out in vector spaces over the binary field GF(2). Unless specified otherwise, all vectors, matrices, and operations on them will be over GF(2). In particular, for a_1^N, b_1^N vectors over GF(2), we write $a_1^N \oplus b_1^N$ to denote their componentwise mod-2 sum. The Kronecker product of an m-by-n matrix $A = [A_{ij}]$ and an r-by-s matrix $B = [B_{ij}]$ is defined as

$$A \otimes B = \begin{bmatrix} A_{11}B & \cdots & A_{1n}B \\ \vdots & \ddots & \vdots \\ A_{m1}B & \cdots & A_{mn}B \end{bmatrix},$$

which is an mr-by-ns matrix. The Kronecker power $A^{\otimes n}$ is defined as $A \otimes A^{\otimes(n-1)}$ for all $n \geq 1$. We will follow the convention that $A^{\otimes 0} \stackrel{\Delta}{=} [1]$.

We write $|\mathscr{A}|$ to denote the number of elements in a set $\mathscr{A}$. We write $1_{\mathscr{A}}$ to denote the indicator function of a set $\mathscr{A}$; thus, $1_{\mathscr{A}}(x)$ equals 1 if $x \in \mathscr{A}$ and 0 otherwise.

We use the standard Landau notation $O(N)$, $o(N)$, $\omega(N)$ to denote the asymptotic behavior of functions.

Throughout log will denote logarithm to the base 2. The unit for channel capacities and code rates will be *bits*.

0.2 Binary Channels and Symmetric Capacity

We write $W : \mathscr{X} \to \mathscr{Y}$ to denote a generic binary-input discrete memoryless channel (B-DMC) with input alphabet $\mathscr{X}$, output alphabet $\mathscr{Y}$, and transition probabilities $W(y|x)$, $x \in \mathscr{X}$, $y \in \mathscr{Y}$. The input alphabet $\mathscr{X}$ will always be $\{0,1\}$, the output alphabet and the transition probabilities may be arbitrary. We write W^N to denote the channel corresponding to N uses of W; thus, $W^N : \mathscr{X}^N \to \mathscr{Y}^N$ with $W^N(y_1^N \mid x_1^N) = \prod_{i=1}^N W(y_i \mid x_i)$.

The symmetric capacity of a B-DMC W is defined as

$$I(W) \stackrel{\Delta}{=} \sum_{y\in\mathscr{Y}} \sum_{x\in\mathscr{X}} \frac{1}{2} W(y|x) \log \frac{W(y|x)}{\frac{1}{2}W(y|0) + \frac{1}{2}W(y|1)}$$

Since we use base-2 logarithms, $I(W)$ takes values in $[0,1]$ and is measured in bits.

The symmetric capacity $I(W)$ is the highest rate at which reliable communication is possible across W using the inputs of W with equal frequency. It equals the Shannon capacity when W is a *symmetric* channel, i.e., a channel for which there exists a permutation π of the output alphabet $\mathscr{Y}$ such that (i) $\pi^{-1} = \pi$ and (ii) $W(y|1) = W(\pi(y)|0)$ for all $y \in \mathscr{Y}$.

The binary symmetric channel (BSC) and the binary erasure channel (BEC) are examples of symmetric channels. A BSC is a B-DMC W with $\mathscr{Y} = \{0,1\}$, $W(0|0) = W(1|1)$, and $W(1|0) = W(0|1)$. A B-DMC W is called a BEC if for each $y \in \mathscr{Y}$, either $W(y|0)W(y|1) = 0$ or $W(y|0) = W(y|1)$. In the latter case, y is said to be an *erasure* symbol. The sum of $W(y|0)$ over all erasure symbols y is called the erasure probability of the BEC.

0.3 Channel Bhattacharyya parameter: A measure of reliability

The Bhattacharyya parameter of a B-DMC W is defined as

$$Z(W) \stackrel{\Delta}{=} \sum_{y\in\mathscr{Y}} \sqrt{W(y|0)W(y|1)}.$$

The Bhattacharyya parameter $Z(W)$ is an upper bound on the probability of MAP decision error when W is used only once to transmit a single bit, a-priori equally likely to be 0 or 1. Hence, $Z(W)$ serves as a measures of *reliability* for W. It is easy to see that $Z(W)$ takes values in $[0,1]$.

Intuitively, one would expect that $I(W) \approx 1$ iff $Z(W) \approx 0$, and $I(W) \approx 0$ iff $Z(W) \approx 1$. The following bounds make this precise.

Proposition 1 *For any B-DMC W, we have*

$$I(W) \geq \log \frac{2}{1+Z(W)}, \tag{0.1}$$

$$I(W) \leq \sqrt{1-Z(W)^2}. \tag{0.2}$$

Furthermore,

$$I(W)+Z(W) \geq 1 \tag{0.3}$$

with equality iff W is a BEC.

Proof of inequality (0.1)*:*
This is proved easily by noting that

$$\log \frac{2}{1+Z(W)}$$

actually equals the channel parameter denoted by $E_0(1,Q)$ by Gallager [6, Section 5.6] with Q taken as the uniform input distribution. (This parameter may be called the *symmetric cutoff rate* of the channel.) It is well known (and shown in the same section of [6]) that $I(W) \geq E_0(1,Q)$. This proves (0.1).

Proof of inequality (0.2)*:*
For any B-DMC $W : \mathscr{X} \to \mathscr{Y}$, define

$$d(W) \stackrel{\Delta}{=} \frac{1}{2} \sum_{y \in \mathscr{Y}} |W(y|0) - W(y|1)|.$$

This is the variational distance between the two distributions $W(y|0)$ and $W(y|1)$ over $y \in \mathscr{Y}$.

Lemma 1 *For any B-DMC W, $I(W) \leq d(W)$.*

Proof. Let W be an arbitrary B-DMC with output alphabet $\mathscr{Y} = \{1,\dots,n\}$ and put $P_i = W(i|0)$, $Q_i = W(i|1)$, $i = 1,\dots,n$. By definition,

$$I(W) = \sum_{i=1}^{n} \frac{1}{2} \left[P_i \log \frac{P_i}{\frac{1}{2}P_i + \frac{1}{2}Q_i} + Q_i \log \frac{Q_i}{\frac{1}{2}P_i + \frac{1}{2}Q_i} \right].$$

The ith bracketed term under the summation is given by

$$f(x) \stackrel{\Delta}{=} x \log \frac{x}{x+\delta} + (x+2\delta)\log\frac{x+2\delta}{x+\delta}$$

where $x = \min\{P_i, Q_i\}$ and $\delta = \frac{1}{2}|P_i - Q_i|$. We now consider maximizing $f(x)$ over $0 \leq x \leq 1-2\delta$. We compute

$$\frac{df}{dx} = \frac{1}{2}\log\frac{\sqrt{x(x+2\delta)}}{(x+\delta)}$$

and recognize that $\sqrt{x(x+2\delta)}$ and $(x+\delta)$ are, respectively, the geometric and arithmetic means of the numbers x and $(x+2\delta)$. So, $df/dx \leq 0$ and $f(x)$ is maximized at $x=0$, giving the inequality $f(x) \leq 2\delta$. Using this in the expression for $I(W)$, we obtain the claim of the lemma,

$$I(W) \leq \sum_{i=1} \frac{1}{2}|P_i - Q_i| = d(W).$$

Lemma 2 *For any B-DMC W, $d(W) \leq \sqrt{1-Z(W)^2}$.*

Proof. Let W be an arbitrary B-DMC with output alphabet $\mathscr{Y} = \{1,\dots,n\}$ and put $P_i = W(i|0)$, $Q_i = W(i|1)$, $i = 1,\dots,n$. Let $\delta_i \stackrel{\Delta}{=} \frac{1}{2}|P_i - Q_i|$, $\delta \stackrel{\Delta}{=} d(W) = \sum_{i=1}^n \delta_i$, and $R_i \stackrel{\Delta}{=} (P_i + Q_i)/2$. Then, we have $Z(W) = \sum_{i=1}^n \sqrt{(R_i-\delta_i)(R_i+\delta_i)}$. Clearly, $Z(W)$ is upper-bounded by the maximum of $\sum_{i=1}^n \sqrt{R_i^2 - \delta_i^2}$ over $\{\delta_i\}$ subject to the constraints that $0 \leq \delta_i \leq R_i$, $i = 1,\dots,n$, and $\sum_{i=1}^n \delta_i = \delta$. To carry out this maximization, we compute the partial derivatives of $Z(W)$ with respect to δ_i,

$$\frac{\partial Z}{\partial \delta_i} = -\frac{\delta_i}{\sqrt{R_i^2 - \delta_i^2}}, \qquad \frac{\partial^2 Z}{\partial \delta_i^2} = -\frac{R_i^2}{{}^{3/2}\sqrt{R_i^2 - \delta_i^2}},$$

and observe that $Z(W)$ is a decreasing, concave function of δ_i for each i, within the range $0 \leq \delta_i \leq R_i$. The maximum occurs at the solution of the set of equations $\partial Z/\partial \delta_i = k$, all i, where k is a constant, i.e., at $\delta_i = R_i\sqrt{k^2/(1+k^2)}$. Using the constraint $\sum_i \delta_i = \delta$ and the fact that $\sum_{i=1}^n R_i = 1$, we find $\sqrt{k^2/(1+k^2)} = \delta$. So, the maximum occurs at $\delta_i = \delta R_i$ and has the value $\sum_{i=1}^n \sqrt{R_i^2 - \delta^2 R_i^2} = \sqrt{1-\delta^2}$. We have thus shown that $Z(W) \leq \sqrt{1-d(W)^2}$, which is equivalent to $d(W) \leq \sqrt{1-Z(W)^2}$.

From the above two lemmas, the proof of (0.2) is immediate.

Proof of inequality (0.3)*:* We defer this proof until Chapter 3 where it will follow as a simple corollary to the results there.

It can be seen that inequality 0.3 is stronger than inequality 0.1 and will prove useful later on. The weaker inequality (0.1) is sufficient to develop the polarization results for the time being.

Chapter 1
Overview of Results

Abstract Shannon proved the achievability part of his noisy channel coding theorem using a random-coding argument which showed the existence of capacity-achieving code sequences without exhibiting any specific sequence [15]. Polar codes are an explicit construction that provably achieves channel capacity with low-complexity encoding, decoding, and code construction algorithms. This chapter gives an overview of channel polarization and polar coding.

1.1 Channel polarization

Channel polarization is a transformation by which one manufactures out of N independent copies of a given B-DMC W a second set of N channels $\{W_N^{(i)} : 1 \leq i \leq N\}$ such that, as N becomes large, the symmetric capacity terms $\{I(W_N^{(i)})\}$ tend towards 0 or 1 for all but a vanishing fraction of indices i. The channel polarization operation consists of a channel combining phase and a channel splitting phase.

1.1.1 Channel combining

This phase combines copies of a given B-DMC W in a recursive manner to produce a vector channel $W_N : \mathscr{X}^N \to \mathscr{Y}^N$, where N can be any power of two, $N = 2^n$, $n \geq 0$. The recursion begins at the 0-th level ($n = 0$) with only one copy of W and we set $W_1 \triangleq W$. The first level ($n = 1$) of the recursion combines two independent copies of W_1 as shown in Fig. 1 and obtains the channel $W_2 : \mathscr{X}^2 \to \mathscr{Y}^2$ with the transition probabilities

$$W_2(y_1,y_2|u_1,u_2) = W(y_1|u_1 \oplus u_2)W(y_2|u_2). \tag{1.1}$$

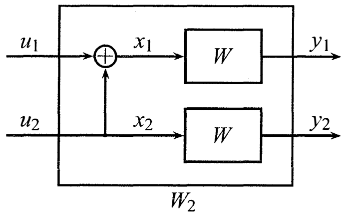


Fig. 1.1 The channel W_2.

The next level of the recursion is shown in Fig. 2 where two independent copies of W_2 are combined to create the channel $W_4 : \mathscr{X}^4 \to \mathscr{Y}^4$ with transition probabilities $W_4(y_1^4|u_1^4) = W_2(y_1^2|u_1 \oplus u_2, u_3 \oplus u_4)W_2(y_3^4|u_2, u_4)$.

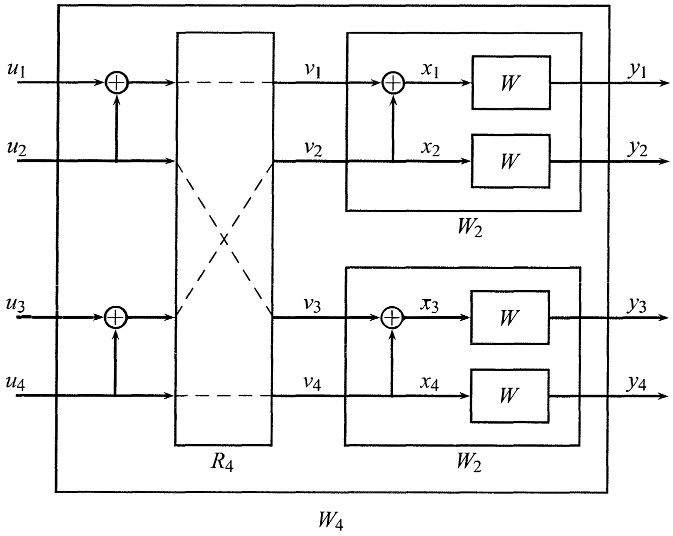


Fig. 1.2 The channel W_4 and its relation to W_2 and W.

In Fig. 2, R_4 is the permutation operation that maps an input (s_1, s_2, s_3, s_4) to $v_1^4 = (s_1, s_3, s_2, s_4)$. The mapping $u_1^4 \mapsto x_1^4$ from the input of W_4 to the input of W^4 can be written as $x_1^4 = u_1^4 G_4$ with $G_4 = \begin{bmatrix} 1&0&0&0 \\ 1&0&1&0 \\ 1&1&0&0 \\ 1&1&1&1 \end{bmatrix}$. Thus, we have the relation $W_4(y_1^4|u_1^4) = W^4(y_1^4|u_1^4 G_4)$ between the transition probabilities of W_4 and those of W^4.

The general form of the recursion is shown in Fig. 3 where two independent copies of $W_{N/2}$ are combined to produce the channel W_N. The input vector u_1^N to W_N is first transformed into s_1^N so that $s_{2i-1} = u_{2i-1} \oplus u_{2i}$ and $s_{2i} = u_{2i}$ for $1 \le i \le$

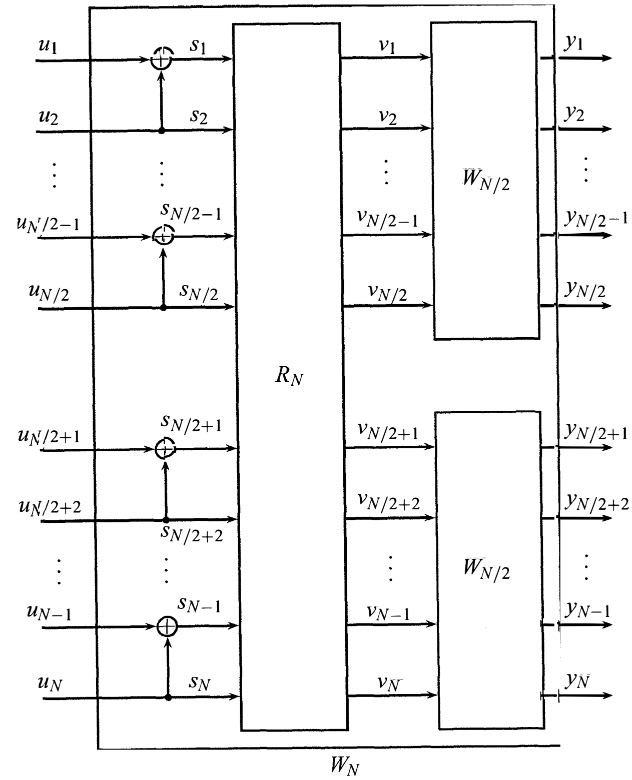


Fig. 1.3 Recursive construction of W_N from two copies of $W_{N/2}$.

$N/2$. The operator R_N in the figure is a permutation, known as the *reverse shuffle* operation, and acts on its input s_1^N to produce $v_1^N = (s_1, s_3, \ldots, s_{N-1}, s_2, s_4, \ldots, s_N)$, which becomes the input to the two copies of $W_{N/2}$ as shown in the figure.

We observe that the mapping $u_1^N \mapsto v_1^N$ is linear over GF(2). It follows by induction that the overall mapping $u_1^N \mapsto x_1^N$, from the input of the synthesized channel W_N to the input of the underlying raw channels W^N, is also linear and may be represented by a matrix G_N so that $x_1^N = u_1^N G_N$. We call G_N the *generator matrix* of size N. The transition probabilities of the two channels W_N and W^N are related by

$$W_N(y_1^N | u_1^N) = W^N(y_1^N | u_1^N G_N) \tag{1.2}$$

for all $y_1^N \in \mathscr{Y}^N$, $u_1^N \in \mathscr{X}^N$. We will show in Sect. 5.1 that G_N equals $B_N F^{\otimes n}$ for any $N = 2^n$, $n \geq 0$, where B_N is a permutation matrix known as *bit-reversal* and $F \triangleq \left[\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix}\right]$. Note that the channel combining operation is fully specified by the matrix F. Also note that G_N and $F^{\otimes n}$ have the same set of rows, but in a different (bit-reversed) order; we will discuss this topic more fully in Sect. 5.1.

1.1.2 Channel splitting

Having synthesized the vector channel W_N out of W^N, the next step of channel polarization is to split W_N back into a set of N binary-input coordinate channels $W_N^{(i)} : \mathscr{X} \to \mathscr{Y}^N \times \mathscr{X}^{i-1}$, $1 \le i \le N$, defined by the transition probabilities

$$W_N^{(i)}(y_1^N, u_1^{i-1}|u_i) \overset{\Delta}{=} \sum_{u_{i+1}^N \in \mathscr{X}^{N-i}} \frac{1}{2^{N-1}} W_N(y_1^N|u_1^N), \tag{1.3}$$

where (y_1^N, u_1^{i-1}) denotes the output of $W_N^{(i)}$ and u_i its input.

To gain an intuitive understanding of the channels $\{W_N^{(i)}\}$, consider a genie-aided successive cancellation decoder in which the ith decision element estimates u_i after observing y_1^N and the *past* channel inputs u_1^{i-1} (supplied correctly by the genie regardless of any decision errors at earlier stages). If u_1^N is a-priori uniform on $\mathscr{X}^N$, then $W_N^{(i)}$ is the effective channel seen by the ith decision element in this scenario.

1.1.3 Channel polarization

Theorem 1 *For any B-DMC W, the channels $\{W_N^{(i)}\}$ polarize in the sense that, for any fixed $\delta \in (0,1)$, as N goes to infinity through powers of two, the fraction of indices $i \in \{1,\dots,N\}$ for which $I(W_N^{(i)}) \in (1-\delta, 1]$ goes to $I(W)$ and the fraction for which $I(W_N^{(i)}) \in [0,\delta)$ goes to $1 - I(W)$.*

This theorem is proved in Sect. 3.3.

The polarization effect is illustrated in Fig. 4 for W a BEC with erasure probability $\varepsilon = 0.5$. The numbers $\{I(W_N^{(i)})\}$ have been computed using the recursive relations

$$\begin{aligned} I(W_N^{(2i-1)}) &= I(W_{N/2}^{(i)})^2, \\ I(W_N^{(2i)}) &= 2I(W_{N/2}^{(i)}) - I(W_{N/2}^{(i)})^2, \end{aligned} \tag{1.4}$$

with $I(W_1^{(1)}) = 1 - \varepsilon$. This recursion is valid only for BECs and it is proved in Sect. 2.2. Figure 4 shows that $I(W^{(i)})$ tends to be near 0 for small i and near 1 for large i. However, $I(W_N^{(i)})$ shows an erratic behavior for an intermediate range of i.

For general B-DMCs, the calculation of $I(W_N^{(i)})$ with sufficient degree of precision is an important problem for constructing polar codes. This issue is discussed in Sect. 5.3.

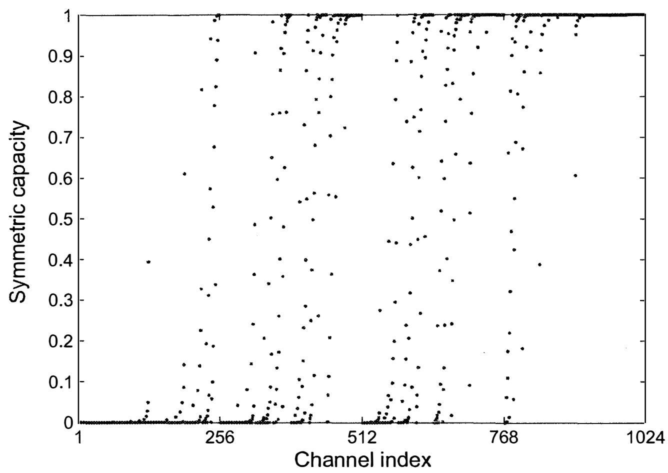


Fig. 1.4 Plot of $I(W_N^{(i)})$ vs. $i = 1,\ldots,N = 2^{10}$ for a BEC with $\varepsilon = 0.5$.

1.1.4 Rate of polarization

For proving coding theorems, the speed with which the polarization effect takes hold as a function of N is important. Our main result in this regard is given in terms of the parameters

$$Z(W_N^{(i)}) = \sum_{y_1^N \in \mathscr{Y}^N} \sum_{u_1^{i-1} \in \mathscr{X}^{i-1}} \sqrt{W_N^{(i)}(y_1^N, u_1^{i-1} \mid 0)\, W_N^{(i)}(y_1^N, u_1^{i-1} \mid 1)}. \tag{1.5}$$

Theorem 2 *Let W be a B-DMC. For any fixed rate $R < I(W)$ and constant $\beta < \frac{1}{2}$, there exists a sequence of sets $\{\mathscr{A}_N\}$ such that $\mathscr{A}_N \subset \{1,\ldots,N\}$, $|\mathscr{A}_N| \geq NR$, and*

$$\sum_{i \in \mathscr{A}_N} Z(W_N^{(i)}) = o(2^{-N^\beta}). \tag{1.6}$$

Conversely, if $R > 0$ and $\beta > \frac{1}{2}$, then for any sequence of sets $\{\mathscr{A}_N\}$ with $\mathscr{A}_N \subset \{1,\ldots,N\}$, $|\mathscr{A}_N| \geq NR$, we have

$$\max\{Z(W_N^{(i)}) : i \in \mathscr{A}_N\} = \omega(2^{-N^\beta}). \tag{1.7}$$

This theorem is proved in Chapter 3.

We stated the polarization result in Theorem 2 in terms $\{Z(W_N^{(i)})\}$ rather than $\{I(W_N^{(i)})\}$ because this form is better suited to the coding results that we will develop. A rate of polarization result in terms of $\{I(W_N^{(i)})\}$ can be obtained from Theorem 2 with the help of Prop. 1.

1.2 Polar coding

Polar coding is a method that takes advantage of the polarization effect to construct codes that achieve the symmetric channel capacity $I(W)$. The basic idea of polar coding is to create a coding system where one can access each coordinate channel $W_N^{(i)}$ individually and send data only through those for which $Z(W_N^{(i)})$ is near 0.

1.2.1 G_N-coset codes

We first describe a class of block codes that contain polar codes—the codes of main interest—as a special case. The block-lengths N for this class are restricted to powers of two, $N = 2^n$ for some $n \geq 0$. For a given N, each code in the class is encoded in the same manner, namely,

$$x_1^N = u_1^N G_N \tag{1.8}$$

where G_N is the generator matrix of order N, defined above. For $\mathscr{A}$ an arbitrary subset of $\{1,\ldots,N\}$, we may write (1.8) as

$$x_1^N = u_{\mathscr{A}} G_N(\mathscr{A}) \oplus u_{\mathscr{A}^c} G_N(\mathscr{A}^c) \tag{1.9}$$

where $G_N(\mathscr{A})$ denotes the submatrix of G_N formed by the rows with indices in $\mathscr{A}$.

If we now fix $\mathscr{A}$ and $u_{\mathscr{A}^c}$, but leave $u_{\mathscr{A}}$ as a free variable, we obtain a mapping from source blocks $u_{\mathscr{A}}$ to codeword blocks x_1^N. This mapping is a *coset code*: it is a coset of the linear block code with generator matrix $G_N(\mathscr{A})$, with the coset determined by the fixed vector $u_{\mathscr{A}^c} G_N(\mathscr{A}^c)$. We will refer to this class of codes collectively as *G_N-coset codes*. Individual G_N-coset codes will be identified by a parameter vector $(N, K, \mathscr{A}, u_{\mathscr{A}^c})$, where K is the code dimension and specifies the size of $\mathscr{A}$.[1] The ratio K/N is called the *code rate*. We will refer to $\mathscr{A}$ as the *information set* and to $u_{\mathscr{A}^c} \in \mathscr{X}^{N-K}$ as *frozen* bits or vector.

For example, the $(4,2,\{2,4\},(1,0))$ code has the encoder mapping

$$\begin{aligned} x_1^4 &= u_1^4 G_4 \\ &= (u_2, u_4) \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} + (1,0) \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}. \end{aligned} \tag{1.10}$$

[1] We include the redundant parameter K in the parameter set because often we consider an ensemble of codes with K fixed and $\mathscr{A}$ free.

For a source block $(u_2,u_4) = (1,1)$, the coded block is $x_1^4 = (1,1,0,1)$.

Polar codes will be specified shortly by giving a particular rule for the selection of the information set $\mathscr{A}$.

1.2.2 A successive cancellation decoder

Consider a G_N-coset code with parameter $(N,K,\mathscr{A},u_{\mathscr{A}^c})$. Let u_1^N be encoded into a codeword x_1^N, let x_1^N be sent over the channel W^N, and let a channel output y_1^N be received. The decoder's task is to generate an estimate $\hat{u}_1^N$ of u_1^N, given knowledge of $\mathscr{A}$, $u_{\mathscr{A}^c}$, and y_1^N. Since the decoder can avoid errors in the frozen part by setting $\hat{u}_{\mathscr{A}^c} = u_{\mathscr{A}^c}$, the real decoding task is to generate an estimate $\hat{u}_{\mathscr{A}}$ of $u_{\mathscr{A}}$.

The coding results in this paper will be given with respect to a specific successive cancellation (SC) decoder, unless some other decoder is mentioned. Given any $(N,K,\mathscr{A},u_{\mathscr{A}^c})$ G_N-coset code, we will use a SC decoder that generates its decision $\hat{u}_1^N$ by computing

$$\hat{u}_i \stackrel{\Delta}{=} \begin{cases} u_i, & \text{if } i \in \mathscr{A}^c \\ h_i(y_1^N, \hat{u}_1^{i-1}), & \text{if } i \in \mathscr{A} \end{cases} \tag{1.11}$$

in the order i from 1 to N, where $h_i : \mathscr{Y}^N \times \mathscr{X}^{i-1} \to \mathscr{X}$, $i \in \mathscr{A}$, are *decision functions* defined as

$$h_i(y_1^N, \hat{u}_1^{i-1}) \stackrel{\Delta}{=} \begin{cases} 0, & \text{if } \frac{W_N^{(i)}(y_1^N, \hat{u}_1^{i-1}|0)}{W_N^{(i)}(y_1^N, \hat{u}_1^{i-1}|1)} \geq 1 \\ 1, & \text{otherwise} \end{cases} \tag{1.12}$$

for all $y_1^N \in \mathscr{Y}^N$, $\hat{u}_1^{i-1} \in \mathscr{X}^{i-1}$. We will say that a decoder *block error* occurred if $\hat{u}_1^N \neq u_1^N$ or equivalently if $\hat{u}_{\mathscr{A}} \neq u_{\mathscr{A}}$.

The decision functions $\{h_i\}$ defined above resemble ML decision functions but are not exactly so, because they treat the *future* frozen bits $(u_j : j > i, j \in \mathscr{A}^c)$ as RVs, rather than as known bits. In exchange for this suboptimality, $\{h_i\}$ can be computed efficiently using recursive formulas, as we will show in Sect. 2.1. Apart from algorithmic efficiency, the recursive structure of the decision functions is important because it renders the performance analysis of the decoder tractable. Fortunately, the loss in performance due to not using true ML decision functions happens to be negligible: $I(W)$ is still achievable.

1.2.3 Code performance

The notation $P_e(N,K,\mathscr{A},u_{\mathscr{A}^c})$ will denote the probability of block error for a $(N,K,\mathscr{A},u_{\mathscr{A}^c})$ code, assuming that each data vector $u_{\mathscr{A}} \in \mathscr{X}^K$ is sent with probability 2^{-K} and decoding is done by the above SC decoder. More precisely,

$$P_e(N,K,\mathscr{A},u_{\mathscr{A}^c}) \stackrel{\Delta}{=} \sum_{u_{\mathscr{A}} \in \mathscr{X}^K} \frac{1}{2^K} \sum_{y_1^N \in \mathscr{Y}^N : \hat{u}_1^N(y_1^N) \neq u_1^N} W_N(y_1^N | u_1^N).$$

The average of $P_e(N,K,\mathscr{A},u_{\mathscr{A}^c})$ over all choices for $u_{\mathscr{A}^c}$ will be denoted by $P_e(N,K,\mathscr{A})$:

$$P_e(N,K,\mathscr{A}) \stackrel{\Delta}{=} \sum_{u_{\mathscr{A}^c} \in \mathscr{X}^{N-K}} \frac{1}{2^{N-K}} P_e(N,K,\mathscr{A},u_{\mathscr{A}^c}).$$

A key bound on block error probability under SC decoding is the following.

Proposition 2 *For any B-DMC W and any choice of the parameters $(N,K,\mathscr{A})$,*

$$P_e(N,K,\mathscr{A}) \leq \sum_{i \in \mathscr{A}} Z(W_N^{(i)}). \tag{1.13}$$

Hence, for each $(N,K,\mathscr{A})$, there exists a frozen vector $u_{\mathscr{A}^c}$ such that

$$P_e(N,K,\mathscr{A},u_{\mathscr{A}^c}) \leq \sum_{i \in \mathscr{A}} Z(W_N^{(i)}). \tag{1.14}$$

This is proved in Sect. 4.3. This result suggests choosing $\mathscr{A}$ from among all K-subsets of $\{1,\ldots,N\}$ so as to minimize the RHS of (1.13). This idea leads to the definition of polar codes.

1.2.4 Polar codes

Given a B-DMC W, a G_N-coset code with parameter $(N,K,\mathscr{A},u_{\mathscr{A}^c})$ will be called a *polar code* for W if the information set $\mathscr{A}$ is chosen as a K-element subset of $\{1,\ldots,N\}$ such that $Z(W_N^{(i)}) \leq Z(W_N^{(j)})$ for all $i \in \mathscr{A}$, $j \in \mathscr{A}^c$.

Polar codes are channel-specific designs: a polar code for one channel may not be a polar code for another. The main result of this paper will be to show that polar coding achieves the symmetric capacity $I(W)$ of any given B-DMC W.

An alternative rule for polar code definition would be to specify $\mathscr{A}$ as a K-element subset of $\{1,\ldots,N\}$ such that $I(W_N^{(i)}) \geq I(W_N^{(j)})$ for all $i \in \mathscr{A}$, $j \in \mathscr{A}^c$. This alternative rule would also achieve $I(W)$. However, the rule based on the Bhattacharyya parameters has the advantage of being connected with an explicit bound on block error probability.

The polar code definition does not specify how the frozen vector $u_{\mathscr{A}^c}$ is to be chosen; it may be chosen at will. This degree of freedom in the choice of $u_{\mathscr{A}^c}$ simplifies the performance analysis of polar codes by allowing averaging over an ensemble. However, it is not for analytical convenience alone that we do not specify a precise rule for selecting $u_{\mathscr{A}^c}$, but also because it appears that the code performance is relatively insensitive to that choice. In fact, we prove in Sect. 4.6 that, for symmetric channels, any choice for $u_{\mathscr{A}^c}$ is as good as any other.

1.2.5 Coding theorems

Fix a B-DMC W and a number $R \geq 0$. Let $P_e(N,R)$ be defined as $P_e(N, \lfloor NR \rfloor, \mathscr{A})$ with $\mathscr{A}$ selected in accordance with the polar coding rule for W. Thus, $P_e(N,R)$ is the probability of block error under SC decoding for polar coding over W with block-length N and rate R, averaged over all choices for the frozen bits $u_{\mathscr{A}^c}$. The main coding result of this paper is the following:

Theorem 3 *For polar coding on a B-DMC W at any fixed rate $R < I(W)$, and any fixed $\beta < \frac{1}{2}$,*

$$P_e(N,R) = o(2^{-N^\beta}). \tag{1.15}$$

This theorem follows as an easy corollary to Theorem 2 and the bound (1.13), as we show in Sect. 4.3. For symmetric channels, we have the following stronger version of Theorem 3.

Theorem 4 *For any symmetric B-DMC W, any fixed $\beta < \frac{1}{2}$, and any fixed $R < I(W)$, consider any sequence of G_N-coset codes $(N,K,\mathscr{A},u_{\mathscr{A}^c})$ with N increasing to infinity, $K = \lfloor NR \rfloor$, $\mathscr{A}$ chosen in accordance with the polar coding rule for W, and $u_{\mathscr{A}^c}$ fixed arbitrarily. The block error probability under successive cancellation decoding satisfies*

$$P_e(N,K,\mathscr{A},u_{\mathscr{A}^c}) = o(2^{-N^\beta}). \tag{1.16}$$

This is proved in Sect. 4.6. Note that for symmetric channels $I(W)$ equals the Shannon capacity of W.

1.2.6 A numerical example

The above results establish that polar codes achieve the symmetric capacity asymptotically. It is of interest to understand how quickly the polarization effect takes hold and what performance can be expected of polar codes under SC decoding in the non-asymptotic regime. To shed some light on this question, we give here a numerical example.

Let W be a BEC with erasure probability 1/2. For the BEC, there are exact formulas for computing the parameters $Z(W_N^{(i)})$, unlike other channels where this is a difficult problem. Figure 7 shows the rate vs. reliability trade-off for W using polar codes with block-lengths $N \in \{2^{10}, 2^{15}, 2^{20}\}$. This figure is obtained by using codes whose information sets are of the form $\mathscr{A}(\eta) \stackrel{\Delta}{=} \{i \in \{1,\dots,N\} : Z(W_N^{(i)}) < \eta\}$, where $0 \leq \eta \leq 1$ is a variable threshold parameter. There are two sets of three curves in the plot. The solid lines are plots of $R(\eta) \stackrel{\Delta}{=} |\mathscr{A}(\eta)|/N$ vs. $B(\eta) \stackrel{\Delta}{=} \sum_{i \in \mathscr{A}(\eta)} Z(W_N^{(i)})$. The dashed lines are plots of $R(\eta)$ vs. $L(\eta) \stackrel{\Delta}{=} \max_{i \in \mathscr{A}(\eta)} \{Z(W_N^{(i)})\}$. The parameter η is varied over a subset of $[0,1]$ to obtain the curves.

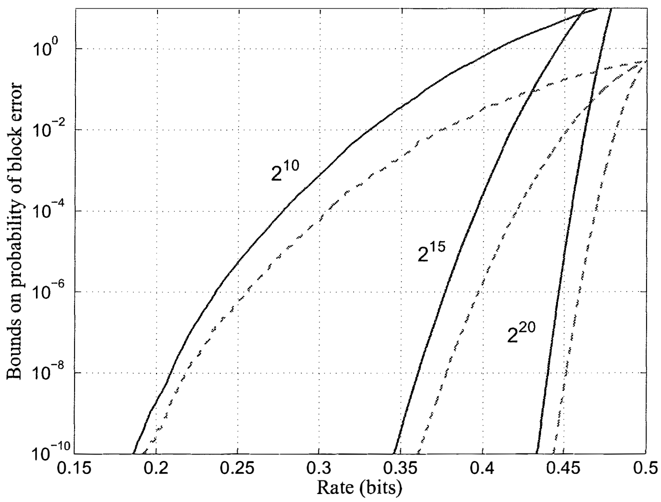


Fig. 1.5 Rate vs. reliability for polar coding and SC decoding at block-lengths 2^{10}, 2^{15}, and 2^{20} on a BEC with erasure probability $1/2$.

The parameter $R(\eta)$ corresponds to the code rate. The significance of $B(\eta)$ is also clear: it is an upper-bound on $P_e(\eta)$, the probability of block-error for polar coding at rate $R(\eta)$ under SC decoding. The parameter $L(\eta)$ is intended to serve as a lower bound to $P_e(\eta)$.

This example provides some empirical evidence that polar coding achieves channel capacity as the block-length is increased—a fact that will be established by exact proofs in the following. The example also shows that the rate of polarization is quite slow, limiting the practical impact of polar codes.

1.2.7 Complexity

An important issue about polar coding is the complexity of encoding, decoding, and code construction. The recursive structure of the channel polarization construction leads to low-complexity encoding and decoding algorithms for the class of G_N-coset codes, and in particular, for polar codes. The computational model we use in stating the following complexity results is a single CPU with a random access memory.

Theorem 5 *For the class of G_N-coset codes, the complexity of encoding and the complexity of successive cancellation decoding are both $O(N \log N)$ as functions of code block-length N.*

This theorem is proved in Sections 5.1 and 5.2. Notice that the complexity bounds in Theorem 5 are independent of the code rate and the way the frozen vector is chosen. The bounds hold even at rates above $I(W)$, but clearly this has no practical significance.

In general, no exact method is known for polar code construction that is of polynomial complexity. One exception is the case of a BEC for which we have a polar code construction algorithm with complexity $O(N)$. However, there exist approximation algorithms for constructing polar codes that have proven effective for practical purposes. These algorithms and their complexity will be discussed in Sect. 5.3.

1.3 Relations to Reed-Muller codes

Polar coding has much in common with Reed-Muller (RM) coding [11], [14]. According to one construction of RM codes, for any $N = 2^n$, $n \geq 0$, and $0 \leq K \leq N$, an RM code with block-length N and dimension K, denoted $\mathrm{RM}(N,K)$, is defined as a linear code whose generator matrix $G_{RM}(N,K)$ is obtained by deleting $(N-K)$ of the rows of $F^{\otimes n}$ so that none of the deleted rows has a larger Hamming weight (number of 1s in that row) than any of the remaining K rows. For instance,

$$G_{RM}(4,4) = F^{\otimes 2} = \begin{bmatrix} 1&0&0&0 \\ 1&1&0&0 \\ 1&0&1&0 \\ 1&1&1&1 \end{bmatrix}$$

and

$$G_{RM}(4,2) = \begin{bmatrix} 1&0&1&0 \\ 1&1&1&1 \end{bmatrix}.$$

This construction brings out the similarities between RM codes and polar codes. Since G_N and $F^{\otimes n}$ have the same set of rows for any $N = 2^n$, it is clear that RM codes belong to the class of G_N-coset codes. For example, $\mathrm{RM}(4,2)$ is the G_4-coset code with parameter $(4,2,\{2,4\},(0,0))$. So, RM coding and polar coding may be regarded as two alternative rules for selecting the information set $\mathscr{A}$ of a G_N-coset code of a given size (N,K). Unlike polar coding, RM coding selects the information set in a channel-independent manner; it is not as fine-tuned to the channel polarization phenomenon as polar coding is. It is shown in [1] that, at least for the class of BECs, the RM rule for information set selection leads to asymptotically unreliable codes under SC decoding. So, polar coding goes beyond RM coding in a non-trivial manner by paying closer attention to channel polarization. However, it is an open question whether RM codes fail to achieve channel capacity under ML decoding.

Another connection to existing work can be established by noting that polar codes are multi-level $|u|u+v|$ codes, which are a class of codes originating from Plotkin's method for code combining [13]. This connection is not surprising in view of the fact that RM codes are also multi-level $|u|u+v|$ codes [9, pp. 114-125]. However, unlike typical multi-level code constructions where one begins with specific small codes to build larger ones, in polar coding the multi-level code is obtained by expurgating rows of a full-order generator matrix, G_N, with respect to a channel-specific criterion. The special structure of G_N ensures that, no matter how expurgation is done, the resulting code is a multi-level $|u|u+v|$ code. In essence, polar coding enjoys the freedom to pick a multi-level code from an ensemble of such codes so as to suit the channel at hand, while conventional approaches to multi-level coding do not have this degree of flexibility.

1.4 Outline of the rest of notes

The rest of the notes is organized as follows. Chapter 2 examines the basic channel combining and splitting operation in detail, in particular, the recursive nature of that transform. In Chapter 3, we develop the main polarization result. In Chapter 4, we investigate the performance of polar codes and complete the proofs of polar coding theorems. Chapter 5 we discuss the complexity of the polar coding algorithms.

Chapter 2
Channel Transformation

Abstract This chapter describes the basic channel transformation operation and investigates the way $I(W)$ and $Z(W)$ get modified under this basic transformation. The basic transformation shows the first traces of polarization. The asymptotic analysis of polarization is left to the next chapter.

2.1 Recursive channel transformations

We have defined a blockwise channel combining and splitting operation by (1.2) and (1.3) which transformed N independent copies of W into $W_N^{(1)}, \ldots, W_N^{(N)}$. The goal in this section is to show that this blockwise channel transformation can be broken recursively into single-step channel transformations.

We say that a pair of binary-input channels $W' : \mathscr{X} \to \tilde{\mathscr{Y}}$ and $W'' : \mathscr{X} \to \tilde{\mathscr{Y}} \times \mathscr{X}$ are obtained by a single-step transformation of two independent copies of a binary-input channel $W : \mathscr{X} \to \mathscr{Y}$ and write

$$(W, W) \mapsto (W', W'')$$

iff there exists a one-to-one mapping $f : \mathscr{Y}^2 \to \tilde{\mathscr{Y}}$ such that

$$W'(f(y_1, y_2)|u_1) = \sum_{u_2'} \frac{1}{2} W(y_1|u_1 \oplus u_2') W(y_2|u_2'), \tag{2.1}$$

$$W''(f(y_1, y_2), u_1|u_2) = \frac{1}{2} W(y_1|u_1 \oplus u_2) W(y_2|u_2) \tag{2.2}$$

for all $u_1, u_2 \in \mathscr{X}$, $y_1, y_2 \in \mathscr{Y}$.

According to this, we can write $(W, W) \mapsto (W_2^{(1)}, W_2^{(2)})$ for any given B-DMC W because

$$
\begin{aligned}
W_2^{(1)}(y_1^2|u_1) &\stackrel{\Delta}{=} \sum_{u_2} \frac{1}{2} W_2(y_1^2|u_1^2) \\
&= \sum_{u_2} \frac{1}{2} W(y_1|u_1 \oplus u_2) W(y_2|u_2), \qquad (2.3) \\
W_2^{(2)}(y_1^2, u_1|u_2) &\stackrel{\Delta}{=} \frac{1}{2} W_2(y_1^2|u_1^2) \\
&= \frac{1}{2} W(y_1|u_1 \oplus u_2) W(y_2|u_2), \qquad (2.4)
\end{aligned}
$$

which are in the form of (2.1) and (2.2) by taking f as the identity mapping.

It turns out we can write, more generally,

$$
(W_N^{(i)}, W_N^{(i)}) \mapsto (W_{2N}^{(2i-1)}, W_{2N}^{(2i)}). \qquad (2.5)
$$

This follows as a corollary to the following:

Proposition 3 *For any $n \geq 0$, $N = 2^n$, $1 \leq i \leq N$,*

$$
W_{2N}^{(2i-1)}(y_1^{2N}, u_1^{2i-2}|u_{2i-1}) = \\
\sum_{u_{2i}} \frac{1}{2} W_N^{(i)}(y_1^N, u_{1,o}^{2i-2} \oplus u_{1,e}^{2i-2}|u_{2i-1} \oplus u_{2i}) W_N^{(i)}(y_{N+1}^{2N}, u_{1,e}^{2i-2}|u_{2i}) \qquad (2.6)
$$

and

$$
W_{2N}^{(2i)}(y_1^{2N}, u_1^{2i-1}|u_{2i}) = \\
\frac{1}{2} W_N^{(i)}(y_1^N, u_{1,o}^{2i-2} \oplus u_{1,e}^{2i-2}|u_{2i-1} \oplus u_{2i}) W_N^{(i)}(y_{N+1}^{2N}, u_{1,e}^{2i-2}|u_{2i}). \qquad (2.7)
$$

This proposition is proved in the Appendix. The transform relationship (2.5) can now be justified by noting that (2.6) and (2.7) are identical in form to (2.1) and (2.2), respectively, after the following substitutions:

$$
\begin{aligned}
W &\leftarrow W_N^{(i)}, & W' &\leftarrow W_{2N}^{(2i-1)}, \\
W'' &\leftarrow W_{2N}^{(2i)}, & u_1 &\leftarrow u_{2i-1}, \\
u_2 &\leftarrow u_{2i}, & y_1 &\leftarrow (y_1^N, u_{1,o}^{2i-2} \oplus u_{1,e}^{2i-2}), \\
y_2 &\leftarrow (y_{N+1}^{2N}, u_{1,e}^{2i-2}), & f(y_1, y_2) &\leftarrow (y_1^{2N}, u_1^{2i-2}).
\end{aligned}
$$

Thus, we have shown that the blockwise channel transformation from W^N to $(W_N^{(1)}, \ldots, W_N^{(N)})$ breaks at a local level into single-step channel transformations of the form (2.5). The full set of such transformations form a fabric as shown in Fig. 5 for $N = 8$. Reading from right to left, the figure starts with four copies of the transformation $(W, W) \mapsto (W_2^{(1)}, W_2^{(2)})$ and continues in *butterfly* patterns, each

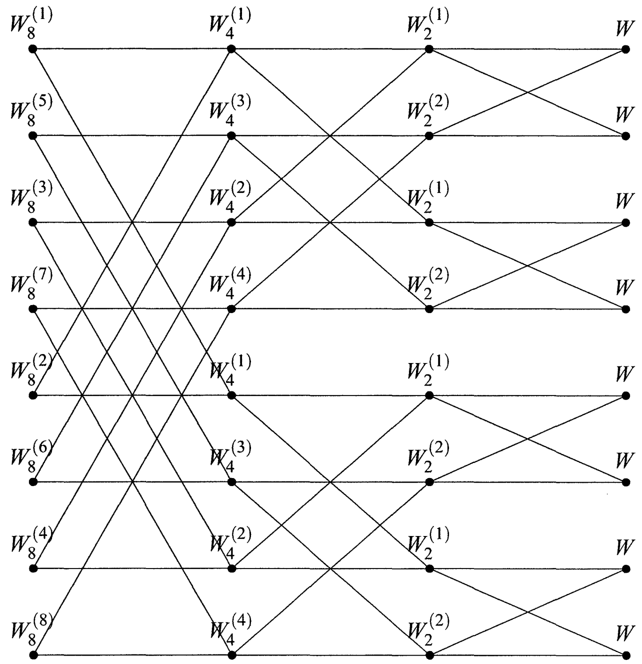


Fig. 2.1 The channel transformation process with $N = 8$ channels.

representing a channel transformation of the form $(W_{2^i}^{(j)}, W_{2^i}^{(j)}) \mapsto (W_{2^{i+1}}^{(2j-1)}, W_{2^{i+1}}^{(2j)})$. The two channels at the right end-points of the butterflies are always identical and independent. At the rightmost level there are 8 independent copies of W; at the next level to the left, there are 4 independent copies of $W_2^{(1)}$ and $W_2^{(2)}$ each; and so on. Each step to the left doubles the number of channel types, but halves the number of independent copies.

2.2 Transformation of rate and reliability

We now investigate how the rate and reliability parameters, $I(W_N^{(i)})$ and $Z(W_N^{(i)})$, change through a local (single-step) transformation (2.5). By understanding the local behavior, we will be able to reach conclusions about the overall transformation from W^N to $(W_N^{(1)}, \dots, W_N^{(N)})$. Proofs of the results in this section are given in the Appendix.

2.2.1 Local transformation of rate and reliability

Proposition 4 *Suppose $(W,W) \mapsto (W',W'')$ for some set of binary-input channels. Then,*

$$I(W') + I(W'') = 2I(W), \tag{2.8}$$
$$I(W') \leq I(W'') \tag{2.9}$$

with equality iff $I(W)$ equals 0 or 1.

The equality (2.8) indicates that the single-step channel transform preserves the symmetric capacity. The inequality (2.9) together with (2.8) implies that the symmetric capacity remains unchanged under a single-step transform, $I(W') = I(W'') = I(W)$, iff W is either a perfect channel or a completely noisy one. If W is neither perfect nor completely noisy, the single-step transform moves the symmetric capacity away from the center in the sense that $I(W') < I(W) < I(W'')$, thus helping polarization.

Proposition 5 *Suppose $(W,W) \mapsto (W',W'')$ for some set of binary-input channels. Then,*

$$Z(W'') = Z(W)^2, \tag{2.10}$$
$$Z(W') \leq 2Z(W) - Z(W)^2, \tag{2.11}$$
$$Z(W') \geq Z(W) \geq Z(W''). \tag{2.12}$$

Equality holds in (2.11) *iff W is a BEC. We have $Z(W') = Z(W'')$ iff $Z(W)$ equals 0 or 1, or equivalently, iff $I(W)$ equals 1 or 0.*

This result shows that reliability can only improve under a single-step channel transform in the sense that

$$Z(W') + Z(W'') \leq 2Z(W) \tag{2.13}$$

with equality iff W is a BEC.

Since the BEC plays a special role w.r.t. extremal behavior of reliability, it deserves special attention.

Proposition 6 *Consider the channel transformation $(W,W) \mapsto (W',W'')$. If W is a BEC with some erasure probability ε, then the channels W' and W'' are BECs with erasure probabilities $2\varepsilon - \varepsilon^2$ and ε^2, respectively. Conversely, if W' or W'' is a BEC, then W is BEC.*

2.2.2 Rate and reliability for $W_N^{(i)}$

We now return to the context at the end of Sect. 2.1.

Proposition 7 *For any B-DMC W, $N = 2^n$, $n \geq 0$, $1 \leq i \leq N$, the transformation $(W_N^{(i)}, W_N^{(i)}) \mapsto (W_{2N}^{(2i-1)}, W_{2N}^{(2i)})$ is rate-preserving and reliability-improving in the sense that*

$$I(W_{2N}^{(2i-1)}) + I(W_{2N}^{(2i)}) = 2I(W_N^{(i)}), \tag{2.14}$$
$$Z(W_{2N}^{(2i-1)}) + Z(W_{2N}^{(2i)}) \leq 2Z(W_N^{(i)}), \tag{2.15}$$

with equality in (2.15) *iff W is a BEC. Channel splitting moves the rate and reliability away from the center in the sense that*

$$I(W_{2N}^{(2i-1)}) \leq I(W_N^{(i)}) \leq I(W_{2N}^{(2i)}), \tag{2.16}$$
$$Z(W_{2N}^{(2i-1)}) \geq Z(W_N^{(i)}) \geq Z(W_{2N}^{(2i)}), \tag{2.17}$$

with equality in (2.16) *and* (2.17) *iff $I(W)$ equals 0 or 1. The reliability terms further satisfy*

$$Z(W_{2N}^{(2i-1)}) \leq 2Z(W_N^{(i)}) - Z(W_N^{(i)})^2, \tag{2.18}$$
$$Z(W_{2N}^{(2i)}) = Z(W_N^{(i)})^2, \tag{2.19}$$
$$Z(W_{2N}^{(2i)}) \leq Z(W_N^{(i)}) \leq Z(W_{2N}^{(2i-1)}), \tag{2.20}$$

with equality in (2.18) *iff W is a BEC and with equality on either side of* (2.20) *iff $I(W)$ is either 0 or 1. The cumulative rate and reliability satisfy*

$$\sum_{i=1}^{N} I(W_N^{(i)}) = NI(W), \tag{2.21}$$
$$\sum_{i=1}^{N} Z(W_N^{(i)}) \leq NZ(W), \tag{2.22}$$

with equality in (2.22) *iff W is a BEC.*

This result follows from Prop. 4 and Prop. 5 as a special case and no separate proof is needed. The cumulative relations (2.21) and (2.22) follow by repeated application of (2.14) and (2.15), respectively. The conditions for equality in Prop. 4 are stated in terms of W rather than $W_N^{(i)}$; this is possible because: (i) by Prop. 4, $I(W) \in \{0,1\}$ iff $I(W_N^{(i)}) \in \{0,1\}$; and (ii) W is a BEC iff $W_N^{(i)}$ is a BEC, which follows from Prop. 6 by induction.

For the special case that W is a BEC with an erasure probability ε, it follows from Prop. 4 and Prop. 6 that the parameters $\{Z(W_N^{(i)})\}$ can be computed through the recursion

$$\begin{aligned} Z(W_N^{(2j-1)}) &= 2Z(W_{N/2}^{(j)}) - Z(W_{N/2}^{(j)})^2, \\ Z(W_N^{(2j)}) &= Z(W_{N/2}^{(j)})^2, \end{aligned} \tag{2.23}$$

with $Z(W_1^{(1)}) = \varepsilon$. The parameter $Z(W_N^{(i)})$ equals the erasure probability of the channel $W_N^{(i)}$. The recursive relations (1.4) follow from (2.23) by the fact that $I(W_N^{(i)}) = 1 - Z(W_N^{(i)})$ for W a BEC.

Appendix

2.3 Proof of Proposition 3

To prove (2.6), we write

$$
\begin{aligned}
W_{2N}^{(2i-1)}(y_1^{2N}, u_1^{2i-2}|u_{2i-1}) &= \sum_{u_{2i}^{2N}} \frac{1}{2^{2N-1}} W_{2N}(y_1^{2N}|u_1^{2N}) \\
&= \sum_{u_{2i,o}^{2N}, u_{2i,e}^{2N}} \frac{1}{2^{2N-1}} W_N(y_1^N|u_{1,o}^{2N} \oplus u_{1,e}^{2N}) W_N(y_{N+1}^{2N}|u_{1,e}^{2N}) \\
&= \sum_{u_{2i}} \frac{1}{2} \sum_{u_{2i+1,e}^{2N}} \frac{1}{2^{N-1}} W_N(y_{N+1}^{2N}|u_{1,e}^{2N}) \sum_{u_{2i+1,o}^{2N}} \frac{1}{2^{N-1}} W_N(y_1^N|u_{1,o}^{2N} \oplus u_{1,e}^{2N}).
\end{aligned} \tag{2.24}
$$

By definition (1.3), the sum over $u_{2i+1,o}^{2N}$ for any fixed $u_{1,e}^{2N}$ equals

$$
W_N^{(i)}(y_1^N, u_{1,o}^{2i-2} \oplus u_{1,e}^{2i-2}|u_{2i-1} \oplus u_{2i}),
$$

because, as $u_{2i+1,o}^{2N}$ ranges over $\mathscr{X}^{N-i}$, $u_{2i+1,o}^{2N} \oplus u_{2i+1,e}^{2N}$ ranges also over $\mathscr{X}^{N-i}$. We now factor this term out of the middle sum in (2.24) and use (1.3) again to obtain (2.6). For the proof of (2.7), we write

$$
\begin{aligned}
W_{2N}^{(2i)}(y_1^{2N}, u_1^{2i-1}|u_{2i}) &= \sum_{u_{2i+1}^{2N}} \frac{1}{2^{2N-1}} W_{2N}(y_1^{2N}|u_1^{2N}) \\
&= \frac{1}{2} \sum_{u_{2i+1,e}^{2N}} \frac{1}{2^{N-1}} W_N(y_{N+1}^{2N}|u_{1,e}^{2N}) \sum_{u_{2i+1,o}^{2N}} \frac{1}{2^{N-1}} W_N(y_1^N|u_{1,o}^{2N} \oplus u_{1,e}^{2N}).
\end{aligned}
$$

By carrying out the inner and outer sums in the same manner as in the proof of (2.6), we obtain (2.7).

2.4 Proof of Proposition 4

Let us specify the channels as follows: $W : \mathscr{X} \to \mathscr{Y}$, $W' : \mathscr{X} \to \tilde{Y}$, and $W'' : \mathscr{X} \to \tilde{Y} \times \mathscr{X}$. By hypothesis there is a one-to-one function $f : \mathscr{Y} \to \tilde{\mathscr{Y}}$ such that (2.1) and (2.2) are satisfied. For the proof it is helpful to define an ensemble of RVs $(U_1,U_2,X_1,X_2,Y_1,Y_2,\tilde{Y})$ so that the pair (U_1,U_2) is uniformly distributed over $\mathscr{X}^2$, $(X_1,X_2) = (U_1 \oplus U_2, U_2)$, $P_{Y_1,Y_2|X_1,X_2}(y_1,y_2|x_1,x_2) = W(y_1|x_1)W(y_2|x_2)$, and $\tilde{Y} = f(Y_1,Y_2)$. We now have

$$\begin{aligned} W'(\tilde{y}|u_1) &= P_{\tilde{Y}|U_1}(\tilde{y}|u_1), \\ W''(\tilde{y},u_1|u_2) &= P_{\tilde{Y}U_1|U_2}(\tilde{y},u_1|u_2). \end{aligned}$$

From these and the fact that $(Y_1,Y_2) \mapsto \tilde{Y}$ is invertible, we get

$$\begin{aligned} I(W') &= I(U_1;\tilde{Y}) = I(U_1;Y_1Y_2), \\ I(W'') &= I(U_2;\tilde{Y}U_1) = I(U_2;Y_1Y_2U_1). \end{aligned}$$

Since U_1 and U_2 are independent, $I(U_2;Y_1Y_2U_1)$ equals $I(U_2;Y_1Y_2|U_1)$. So, by the chain rule, we have

$$I(W') + I(W'') = I(U_1U_2;Y_1Y_2) = I(X_1X_2;Y_1Y_2)$$

where the second equality is due to the one-to-one relationship between (X_1,X_2) and (U_1,U_2). The proof of (2.8) is completed by noting that $I(X_1X_2;Y_1Y_2)$ equals $I(X_1;Y_1) + I(X_2;Y_2)$ which in turn equals $2I(W)$.

To prove (2.9), we begin by noting that

$$\begin{aligned} I(W'') &= I(U_2;Y_1Y_2U_1) \\ &= I(U_2;Y_2) + I(U_2;Y_1U_1|Y_2) \\ &= I(W) + I(U_2;Y_1U_1|Y_2). \end{aligned}$$

This shows that $I(W'') \geq I(W)$. This and (2.8) give (2.9). The above proof shows that equality holds in (2.9) iff $I(U_2;Y_1U_1|Y_2) = 0$, which is equivalent to having

$$P_{U_1,U_2,Y_1|Y_2}(u_1,u_2,y_1|y_2) = P_{U_1,Y_1|Y_2}(u_1,y_1|y_2)P_{U_2|Y_2}(u_2|y_2)$$

for all (u_1,u_2,y_1,y_2) such that $P_{Y_2}(y_2) > 0$, or equivalently,

$$P_{Y_1,Y_2|U_1,U_2}(y_1,y_2|u_1,u_2)P_{Y_2}(y_2) = P_{Y_1,Y_2|U_1}(y_1,y_2|u_1)P_{Y_2|U_2}(y_2|u_2) \tag{2.25}$$

for all (u_1,u_2,y_1,y_2). Since $P_{Y_1,Y_2|U_1,U_2}(y_1,y_2|u_1,u_2) = W(y_1|u_1 \oplus u_2)W(y_2|u_2)$, eq. (2.25) can be written as

$$W(y_2|u_2)\left[W(y_1|u_1 \oplus u_2)P_{Y_2}(y_2) - P_{Y_1,Y_2}(y_1,y_2|u_1)\right] = 0. \tag{2.26}$$

Substituting $P_{Y_2}(y_2) = \frac{1}{2}W(y_2|u_2) + \frac{1}{2}W(y_2|u_2 \oplus 1)$ and

$$P_{Y_1,Y_2|U_1}(y_1,y_2|u_1) = \frac{1}{2}W(y_1|u_1 \oplus u_2)W(y_2|u_2) + \frac{1}{2}W(y_1|u_1 \oplus u_2 \oplus 1)W(y_2|u_2 \oplus 1)$$

into (2.26) and simplifying, we obtain

$$W(y_2|u_2)W(y_2|u_2 \oplus 1)\,[W(y_1|u_1 \oplus u_2) - W(y_1|u_1 \oplus u_2 \oplus 1)] = 0,$$

which for all four possible values of (u_1, u_2) is equivalent to

$$W(y_2|0)W(y_2|1)\,[W(y_1|0) - W(y_1|1)] = 0.$$

Thus, either there exists no y_2 such that $W(y_2|0)W(y_2|1) > 0$, in which case $I(W) = 1$, or for all y_1 we have $W(y_1|0) = W(y_1|1)$, which implies $I(W) = 0$.

2.5 Proof of Proposition 5

Proof of (2.10) is straightforward.

$$\begin{aligned}
Z(W'') &= \sum_{y_1^2, u_1} \sqrt{W''(f(y_1,y_2),u_1|0)}\,\sqrt{W''(f(y_1,y_2),u_1|1)} \\
&= \sum_{y_1^2, u_1} \frac{1}{2}\sqrt{W(y_1 \mid u_1)W(y_2 \mid 0)}\,\sqrt{W(y_1 \mid u_1 \oplus 1)W(y_2 \mid 1)} \\
&= \sum_{y_2}\sqrt{W(y_2 \mid 0)W(y_2 \mid 1)}\,\sum_{u_1}\frac{1}{2}\sum_{y_1}\sqrt{W(y_1 \mid u_1)W(y_1 \mid u_1 \oplus 1)} \\
&= Z(W)^2.
\end{aligned}$$

To prove (2.11), we put for shorthand $\alpha(y_1) = W(y_1|0)$, $\delta(y_1) = W(y_1|1)$, $\beta(y_2) = W(y_2|0)$, and $\gamma(y_2) = W(y_2|1)$, and write

$$\begin{aligned}
Z(W') &= \sum_{y_1^2}\sqrt{W'(f(y_1,y_2)|0)\,W'(f(y_1,y_2)|1)} \\
&= \sum_{y_1^2}\frac{1}{2}\sqrt{\alpha(y_1)\beta(y_2) + \delta(y_1)\gamma(y_2)}\,\sqrt{\alpha(y_1)\gamma(y_2) + \delta(y_1)\beta(y_2)} \\
&\le \sum_{y_1^2}\frac{1}{2}\left[\sqrt{\alpha(y_1)\beta(y_2)} + \sqrt{\delta(y_1)\gamma(y_2)}\right]\left[\sqrt{\alpha(y_1)\gamma(y_2)} + \sqrt{\delta(y_1)\beta(y_2)}\right] \\
&\quad - \sum_{y_1^2}\sqrt{\alpha(y_1)\beta(y_2)\delta(y_1)\gamma(y_2)}
\end{aligned}$$

where the inequality follows from the identity

$$\begin{aligned}
&\left[\sqrt{(\alpha\beta + \delta\gamma)(\alpha\gamma + \delta\beta)}\right]^2 + 2\sqrt{\alpha\beta\delta\gamma}\,(\sqrt{\alpha} - \sqrt{\delta})^2(\sqrt{\beta} - \sqrt{\gamma})^2 \\
&\qquad = \left[(\sqrt{\alpha\beta} + \sqrt{\delta\gamma})(\sqrt{\alpha\gamma} + \sqrt{\delta\beta}) - 2\sqrt{\alpha\beta\delta\gamma}\right]^2.
\end{aligned}$$

Next, we note that

$$\sum_{y_1^2} \alpha(y_1)\sqrt{\beta(y_2)\gamma(y_2)} = Z(W).$$

Likewise, each term obtained by expanding

$$(\sqrt{\alpha(y_1)\beta(y_2)} + \sqrt{\delta(y_1)\gamma(y_2)})(\sqrt{\alpha(y_1)\gamma(y_2)} + \sqrt{\delta(y_1)\beta(y_2)})$$

gives $Z(W)$ when summed over y_1^2. Also, $\sqrt{\alpha(y_1)\beta(y_2)\delta(y_1)\gamma(y_2)}$ summed over y_1^2 equals $Z(W)^2$. Combining these, we obtain the claim (2.11). Equality holds in (2.11) iff, for any choice of y_1^2, one of the following is true: $\alpha(y_1)\beta(y_2)\gamma(y_2)\delta(y_1) = 0$ or $\alpha(y_1) = \delta(y_1)$ or $\beta(y_2) = \gamma(y_2)$. This is satisfied if W is a BEC. Conversely, if we take $y_1 = y_2$, we see that for equality in (2.11), we must have, for any choice of y_1, either $\alpha(y_1)\delta(y_1) = 0$ or $\alpha(y_1) = \delta(y_1)$; this is equivalent to saying that W is a BEC.

To prove (2.12), we need the following result which states that the parameter $Z(W)$ is a convex function of the channel transition probabilities.

Lemma 3 *Given any collection of B-DMCs $W_j : \mathscr{X} \to \mathscr{Y}$, $j \in \mathscr{J}$, and a probability distribution Q on $\mathscr{J}$, define $W : \mathscr{X} \to \mathscr{Y}$ as the channel $W(y|x) = \sum_{j\in\mathscr{J}} Q(j)W_j(y|x)$. Then,*

$$\sum_{j\in\mathscr{J}} Q(j)Z(W_j) \le Z(W). \tag{2.27}$$

Proof. This follows by first rewriting $Z(W)$ in a different form and then applying Minkowsky's inequality [6, p. 524, ineq. (h)].

$$\begin{aligned} Z(W) &= \sum_y \sqrt{W(y|0)W(y|1)} \\ &= -1 + \frac{1}{2}\sum_y \left[\sum_x \sqrt{W(y|x)}\right]^2 \\ &\ge -1 + \frac{1}{2}\sum_y \sum_{j\in\mathscr{J}} Q(j) \left[\sum_x \sqrt{W_j(y|x)}\right]^2 \\ &= \sum_{j\in\mathscr{J}} Q(j)Z(W_j). \end{aligned}$$

We now write W' as the mixture

$$W'(f(y_1,y_2)|u_1) = \frac{1}{2}\left[W_0(y_1^2 \mid u_1) + W_1(y_1^2|u_1)\right]$$

where

$$W_0(y_1^2|u_1) = W(y_1|u_1)W(y_2|0),$$

$$W_1(y_1^2|u_1) = W(y_1|u_1 \oplus 1)W(y_2|1),$$

and apply Lemma 3 to obtain the claimed inequality

$$Z(W') \geq \frac{1}{2}[Z(W_0) + Z(W_1)] = Z(W).$$

Since $0 \leq Z(W) \leq 1$ and $Z(W'') = Z(W)^2$, we have $Z(W) \geq Z(W'')$, with equality iff $Z(W)$ equals 0 or 1. Since $Z(W') \geq Z(W)$, this also shows that $Z(W') = Z(W'')$ iff $Z(W)$ equals 0 or 1. So, by Prop. 1, $Z(W') = Z(W'')$ iff $I(W)$ equal to 1 or 0.

2.6 Proof of Proposition 6

From (2.1), we have the identities

$$\begin{aligned} W'(f(y_1,y_2)|0)W'(f(y_1,y_2)|1) = \\ &\frac{1}{4}\left[W(y_1|0)^2 + W(y_1|1)^2\right] W(y_2|0)W(y_2|1) + \\ &\frac{1}{4}\left[W(y_2|0)^2 + W(y_2|1)^2\right] W(y_1|0)W(y_1|1) \end{aligned} \tag{2.28}$$

and

$$\begin{aligned} W'(f(y_1,y_2)|0) - W'(f(y_1,y_2)|1) = \\ &\frac{1}{2}[W(y_1|0) - W(y_1|1)][W(y_2|0) - W(y_2|1)]. \end{aligned} \tag{2.29}$$

Suppose W is a BEC, but W' is not. Then, there exists (y_1, y_2) such that the left sides of (2.28) and (2.29) are both different from zero. From (2.29), we infer that neither y_1 nor y_2 is an erasure symbol for W. But then the RHS of (2.28) must be zero, which is a contradiction. Thus, W' must be a BEC. From (2.29), we conclude that $f(y_1, y_2)$ is an erasure symbol for W' iff either y_1 or y_2 is an erasure symbol for W. This shows that the erasure probability for W' is $2\varepsilon - \varepsilon^2$, where ε is the erasure probability of W.

Conversely, suppose W' is a BEC but W is not. Then, there exists y_1 such that $W(y_1|0)W(y_1|1) > 0$ and $W(y_1|0) - W(y_1|1) \neq 0$. By taking $y_2 = y_1$, we see that the RHSs of (2.28) and (2.29) can both be made non-zero, which contradicts the assumption that W' is a BEC.

The other claims follow from the identities

$$\begin{aligned} W''(f(y_1,y_2),u_1|0)\, W''(f(y_1,y_2),u_1|1) \\ &= \frac{1}{4}W(y_1|u_1)W(y_1|u_1 \oplus 1)W(y_2|0)W(y_2|1) \end{aligned}$$

and

$$W''(f(y_1,y_2),u_1|0) - W''(f(y_1,y_2),u_1|1)$$
$$= \frac{1}{2}[W(y_1|u_1)W(y_2|0) - W(y_1|u_1 \oplus 1)W(y_2|1)].$$

The arguments are similar to the ones already given and we omit the details, other than noting that $(f(y,y_2),u_1)$ is an erasure symbol for W'' iff both y_1 and y_2 are erasure symbols for W.

Chapter 3
Channel Polarization

Abstract This chapter proves the main polarization theorems.

3.1 Polarization Theorems

The goal of this chapter is to prove the main polarization theorems, restated below.

Theorem 1 *For any B-DMC W, the channels $\{W_N^{(i)}\}$* polarize *in the sense that, for any fixed $\delta \in (0,1)$, as N goes to infinity through powers of two, the fraction of indices $i \in \{1,\ldots,N\}$ for which $I(W_N^{(i)}) \in (1-\delta,1]$ goes to $I(W)$ and the fraction for which $I(W_N^{(i)}) \in [0,\delta)$ goes to $1-I(W)$.*

Theorem 2 *Let W be a B-DMC. For any fixed rate $R < I(W)$ and constant $\beta < \frac{1}{2}$, there exists a sequence of sets $\{\mathscr{A}_N\}$ such that $\mathscr{A}_N \subset \{1,\ldots,N\}$, $|\mathscr{A}_N| \geq NR$, and*

$$\sum_{i \in \mathscr{A}_N} Z(W_N^{(i)}) = o(2^{-N^\beta}). \tag{3.1}$$

Conversely, if $R > 0$ and $\beta > \frac{1}{2}$, then for any sequence of sets $\{\mathscr{A}_N\}$ with $\mathscr{A}_N \subset \{1,\ldots,N\}$, $|\mathscr{A}_N| \geq NR$, we have

$$\max\{Z(W_N^{(i)}) : i \in \mathscr{A}_N\} = \omega(2^{-N^\beta}). \tag{3.2}$$

3.2 A stochastic process framework for analysis

The analysis is based on the recursive relationships depicted in Fig. 5; however, it will be more convenient to re-sketch Fig. 5 as a binary tree as shown in Fig. 6. The root node of the tree is associated with the channel W. The root W gives birth to an upper channel $W_2^{(1)}$ and a lower channel $W_2^{(2)}$, which are associated with the two nodes at level 1. The channel $W_2^{(1)}$ in turn gives birth to the channels $W_4^{(1)}$ and $W_4^{(2)}$, and so on. The channel $W_{2^n}^{(i)}$ is located at level n of the tree at node number i counting from the top.

There is a natural indexing of nodes of the tree in Fig. 6 by bit sequences. The root node is indexed with the null sequence. The upper node at level 1 is indexed with 0 and the lower node with 1. Given a node at level n with index $b_1b_2\cdots b_n$, the upper node emanating from it has the label $b_1b_2\cdots b_n0$ and the lower node $b_1b_2\cdots b_n1$. According to this labeling, the channel $W_{2^n}^{(i)}$ is situated at the node $b_1b_2\cdots b_n$ with $i = 1+\sum_{j=1}^{n} b_j 2^{n-j}$. We denote the channel $W_{2^n}^{(i)}$ located at node $b_1b_2\cdots b_n$ alternatively as $W_{b_1\ldots b_n}$.

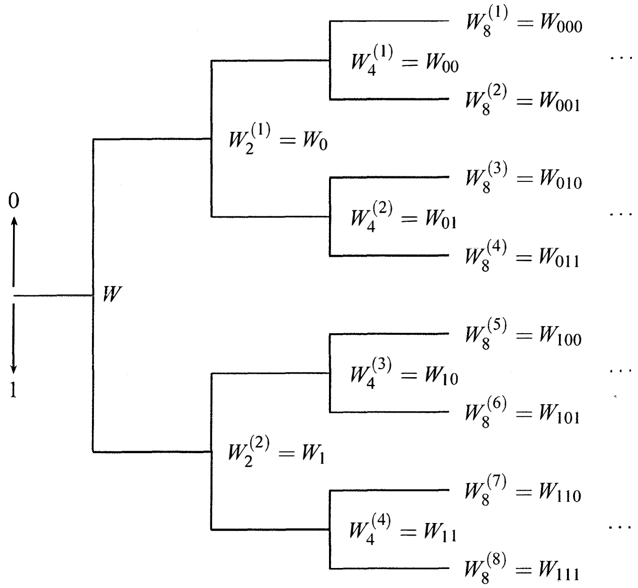


Fig. 3.1 The tree process for the recursive channel construction.

We define a random tree process, denoted $\{K_n; n \geq 0\}$, in connection with Fig. 6. The process begins at the root of the tree with $K_0 = W$. For any $n \geq 0$, given that $K_n = W_{b_1\cdots b_n}$, K_{n+1} equals $W_{b_1\cdots b_n 0}$ or $W_{b_1\cdots b_n 1}$ with probability 1/2 each. Thus, the path taken by $\{K_n\}$ through the channel tree may be thought of as being driven by a sequence of i.i.d. Bernoulli RVs $\{B_n; n = 1,2,\ldots\}$ where B_n equals 0 or 1 with equal probability. Given that $B_1,\ldots,B_n$ has taken on a sample value $b_1,\ldots,b_n$, the random channel process takes the value $K_n = W_{b_1\cdots b_n}$. In order to keep track of the rate and reliability parameters of the random sequence of channels K_n, we define the random processes $I_n = I(K_n)$ and $Z_n = Z(K_n)$.

For a more precise formulation of the problem, we consider the probability space $(\Omega, \mathscr{F}, P)$ where Ω is the space of all binary sequences $(b_1, b_2, \ldots) \in \{0,1\}^\infty$, $\mathscr{F}$ is the Borel field (BF) generated by the *cylinder sets* $S(b_1, \ldots, b_n) \triangleq \{\omega \in \Omega : \omega_1 = b_1, \ldots, \omega_n = b_n\}$, $n \geq 1$, $b_1, \ldots, b_n \in \{0,1\}$, and P is the probability measure defined on $\mathscr{F}$ such that $P(S(b_1, \ldots, b_n)) = 1/2^n$. For each $n \geq 1$, we define $\mathscr{F}_n$ as the BF generated by the cylinder sets $S(b_1, \ldots, b_i)$, $1 \leq i \leq n$, $b_1, \ldots, b_i \in \{0,1\}$. We define $\mathscr{F}_0$ as the trivial BF consisting of the null set and Ω only. Clearly, $\mathscr{F}_0 \subset \mathscr{F}_1 \subset \cdots \subset \mathscr{F}$.

The random processes described above can now be formally defined as follows. For $\omega = (\omega_1, \omega_2, \ldots) \in \Omega$ and $n \geq 1$, define $B_n(\omega) = \omega_n$, $K_n(\omega) = W_{\omega_1 \cdots \omega_n}$, $I_n(\omega) = I(K_n(\omega))$, and $Z_n(\omega) = Z(K_n(\omega))$. For $n = 0$, define $K_0 = W$, $I_0 = I(W)$, $Z_0 = Z(W)$. It is clear that, for any fixed $n \geq 0$, the RVs B_n, K_n, I_n, and Z_n are measurable with respect to the BF $\mathscr{F}_n$.

3.3 Proof of Theorem 1

We will prove Theorem 1 by considering the stochastic convergence properties of the random sequences $\{I_n\}$ and $\{Z_n\}$.

Proposition 8 *The sequence of random variables and Borel fields $\{I_n, \mathscr{F}_n; n \geq 0\}$ is a martingale, i.e.,*

$$\mathscr{F}_n \subset \mathscr{F}_{n+1} \text{ and } I_n \text{ is } \mathscr{F}_n\text{-measurable}, \tag{3.3}$$

$$E[|I_n|] < \infty, \tag{3.4}$$

$$I_n = E[I_{n+1} | \mathscr{F}_n]. \tag{3.5}$$

Furthermore, the sequence $\{I_n; n \geq 0\}$ converges a. e. to a random variable I_∞ such that $E[I_\infty] = I_0$.

Proof. Condition (3.3) is true by construction and (3.4) by the fact that $0 \leq I_n \leq 1$. To prove (3.5), consider a cylinder set $S(b_1, \ldots, b_n) \in \mathscr{F}_n$ and use Prop. 7 to write

$$\begin{aligned} E[I_{n+1} | S(b_1, \cdots, b_n)] &= \frac{1}{2} I(W_{b_1 \cdots b_n 0}) + \frac{1}{2} I(W_{b_1 \cdots b_n 1}) \\ &= I(W_{b_1 \cdots b_n}). \end{aligned}$$

Since $I(W_{b_1 \cdots b_n})$ is the value of I_n on $S(b_1, \ldots, b_n)$, (3.5) follows. This completes the proof that $\{I_n, \mathscr{F}_n\}$ is a martingale. Since $\{I_n, \mathscr{F}_n\}$ is a uniformly integrable martingale, by general convergence results about such martingales (see, e.g., [3, Theorem 9.4.6]), the claim about I_∞ follows.

It should not be surprising that the limit RV I_∞ takes values a. e. in $\{0,1\}$, which is the set of fixed points of $I(W)$ under the transformation $(W, W) \mapsto (W_2^{(1)}, W_2^{(2)})$, as determined by the condition for equality in (2.9). For a rigorous proof of this statement, we take an indirect approach and bring the process $\{Z_n; n \geq 0\}$ also into the picture.

Proposition 9 *The sequence of random variables and Borel fields $\{Z_n, \mathscr{F}_n; n \geq 0\}$ is a supermartingale, i.e.,*

$$\mathscr{F}_n \subset \mathscr{F}_{n+1} \text{ and } Z_n \text{ is } \mathscr{F}_n\text{-measurable}, \tag{3.6}$$

$$E[|Z_n|] < \infty, \tag{3.7}$$

$$Z_n \geq E[Z_{n+1}|\mathscr{F}_n]. \tag{3.8}$$

Furthermore, the sequence $\{Z_n; n \geq 0\}$ converges a.e. to a random variable Z_∞ which takes values a.e. in $\{0,1\}$.

Proof. Conditions (3.6) and (3.7) are clearly satisfied. To verify (3.8), consider a cylinder set $S(b_1, \ldots, b_n) \in \mathscr{F}_n$ and use Prop. 7 to write

$$\begin{aligned} E[Z_{n+1}|S(b_1, \ldots, b_n)] &= \frac{1}{2} Z(W_{b_1 \cdots b_n 0}) + \frac{1}{2} Z(W_{b_1 \cdots b_n 1}) \\ &\leq Z(W_{b_1 \cdots b_n}). \end{aligned}$$

Since $Z(W_{b_1 \cdots b_n})$ is the value of Z_n on $S(b_1, \ldots, b_n)$, (3.8) follows. This completes the proof that $\{Z_n, \mathscr{F}_n\}$ is a supermartingale. For the second claim, observe that the supermartingale $\{Z_n, \mathscr{F}_n\}$ is uniformly integrable; hence, it converges a.e. and in $\mathscr{L}^1$ to a RV Z_∞ such that $E[|Z_n - Z_\infty|] \to 0$ (see, e.g., [3, Theorem 9.4.5]). It follows that $E[|Z_{n+1} - Z_n|] \to 0$. But, by Prop. 7, $Z_{n+1} = Z_n^2$ with probability 1/2; hence, $E[|Z_{n+1} - Z_n|] \geq (1/2)E[Z_n(1 - Z_n)] \geq 0$. Thus, $E[Z_n(1 - Z_n)] \to 0$, which implies $E[Z_\infty(1 - Z_\infty)] = 0$. This, in turn, means that Z_∞ equals 0 or 1 a.e.

Proposition 10 *The limit RV I_∞ takes values a.e. in the set $\{0,1\}$: $P(I_\infty = 1) = I_0$ and $P(I_\infty = 0) = 1 - I_0$.*

Proof. The fact that Z_∞ equals 0 or 1 a.e., combined with Prop. 1, implies that $I_\infty = 1 - Z_\infty$ a.e. Since $E[I_\infty] = I_0$, the rest of the claim follows.

As a corollary to Prop. 10, we can conclude that, as N tends to infinity, the symmetric capacity terms $\{I(W_N^{(i)} : 1 \leq i \leq N\}$ cluster around 0 and 1, except for a vanishing fraction. This completes the proof of Theorem 1.

3.4 Proof of the converse part of Theorem 2

We first prove the converse part of Theorem 2 which we restate as follows.

Proposition 11 *For any $\beta > 1/2$ and with $P(Z_0 > 0) > 0$,*

$$\lim_{n \to \infty} P\left(Z_n < 2^{-2^{n\beta}}\right) = 0. \tag{3.9}$$

Proof. Observe that the random process Z_n is lower-bounded by the process $\{L_n : n \in \mathbb{N}\}$ defined by $L_0 := Z_0$ and for $n \geq 1$

$$L_n = L_{n-1}^2 \qquad \text{when } B_n = 1,$$
$$L_n = L_{n-1} \qquad \text{when } B_n = 0.$$

Thus, $L_n = L_0^{2^{S_n}}$ where $S_n := \sum_{i=1}^n B_i$. So, we have

$$P\left(Z_n \leq 2^{-2^{\beta n}}\right) \leq P\left(L_n \leq 2^{-2^{\beta n}}\right)$$
$$= P\left(S_n \geq n\beta - \log_2(-\log_2(Z_0))\right).$$

For $\beta > \frac{1}{2}$, this last probability goes to zero as n increases by the law of large numbers.

3.5 Proof of Theorem 2: The direct part

In this part, we will establish the direct part of Theorem 2 which may be stated as follows.

Proposition 12 *For any given $\beta < \frac{1}{2}$ and $\varepsilon > 0$, there exists n such that*

$$P\left(Z_n < 2^{-2^{n\beta}}\right) \geq I_0 - \varepsilon. \tag{3.10}$$

The proof of this result is quite lengthy and will be split into several parts. It will be convenient to introduce some notation and state an elementary fact before beginning the proof.

For $n > m \geq 0$ and $0 \leq \beta \leq 1$, define $S_{m,n} = \sum_{i=m+1}^n B_i$ and

$$\mathscr{S}_{m,n}(\beta) = \{\omega \in \Omega : S_{m,n}(\omega) > (n-m)\beta\}.$$

By Chernoff's bound (see, e.g., [6, p. 531]), for $0 \leq \beta \leq \frac{1}{2}$, the probability of this set is bounded as

$$P[\mathscr{S}_{m,n}(\beta)] \geq 1 - 2^{-(n-m)[1-\mathscr{H}(\beta)]} \tag{3.11}$$

where $\mathscr{H}(\beta) = -\beta\log_2(\beta) - (1-\beta)\log_2(1-\beta)$ is the binary entropy function. Clearly, for $0 \leq \beta < 1/2$, the probability of $\mathscr{S}_{m,n}$ goes to 1 as $(n-m)$ increases. Define $n_0(\beta, \varepsilon)$ as the smallest value of $(n-m)$ such that the RHS of (3.11) is greater than or equal to $1 - \varepsilon$.

3.5.1 A bootstrapping method

We first give a bound to majorize the process $\{Z_n\}$ on a sample function basis. For this it is more convenient to consider the logarithmic process $V_n := \log_2(Z_n)$. This process evolves as

$$\begin{aligned} V_{i+1} &= 2V_i && \text{when } B_{i+1} = 1, \\ V_{i+1} &\leq V_i + 1 && \text{when } B_{i+1} = 0. \end{aligned}$$

Thus, at each step either the value is doubled or incremented by an amount not exceeding one. In terms of this process, we wish to show that with probability close to I_0 we have $V_n \approx -2^{\frac{n}{2}}$.

The following lemma is key to analyzing the behavior of the process $\{V_n\}$.

Lemma 4 *Let $A : \mathbb{R} \to \mathbb{R}$, $A(x) = x + 1$ denote adding one, and $D : \mathbb{R} \to \mathbb{R}$, $D(x) = 2x$ denote doubling. Suppose a sequence of numbers $a_0, a_1, \ldots, a_n$ is defined by specifying a_0 and the recursion*

$$a_{i+1} = f_i(a_i)$$

with $f_i \in \{A, D\}$. Suppose $\left|\{0 \leq i \leq n-1 : f_i = D\}\right| = k$ and $\left|\{0 \leq i \leq n-1 : f_i = A\}\right| = n - k$, i.e., during the first n iterations of the recursion we encounter doubling k times and adding-one $n - k$ times. Then

$$a_n \leq D^{(k)}\left(A^{(n-k)}(a_0)\right) = 2^k(a_0 + n - k).$$

Proof. Observe that the upper bound on a_n corresponds to choosing

$$f_0 = \cdots f_{n-k-1} = A \quad \text{and} \quad f_{n-k} = \cdots = f_{n-1} = D.$$

We will show that any other choice of $\{f_i\}$ can be modified to yield a higher value of a_n. To that end suppose $\{f_i\}$ is not chosen as above. Then there exists $j \in \{1, \ldots, n-1\}$ for which $f_{j-1} = D$ and $f_j = A$. Define $\{f_i'\}$ by swapping f_j and f_{j-1}, i.e.,

$$f_i' = \begin{cases} A & i = j-1 \\ D & i = j \\ f_i & \text{else} \end{cases}$$

and let $\{a_i'\}$ denote the sequence that results from $\{f_i'\}$. Then

$$\begin{aligned} a_i' &= a_i \quad \text{for } i < j \\ a_j' &= a_{j-1} + 1 \\ a_{j+1}' &= 2a_j' = 2a_{j-1} + 2 \\ &> 2a_{j-1} + 1 = a_{j+1}. \end{aligned}$$

Since the recursion from $j+1$ onwards is identical for the $\{f_i\}$ and $\{f_i'\}$ sequences, and since both A and D are order preserving, $a'_{j+1} > a_{j+1}$ implies that $a'_n > a_n$.

By Lemma 4, we can write for any $n > m$

$$\begin{aligned} V_n &\leq \left[V_m + (n-m) - S_{m,n}\right]2^{S_{m,n}} \\ &\leq \left[V_m + (n-m)\right]2^{S_{m,n}} \end{aligned}$$

The process $\{V_n\}$ takes values in $(-\infty, 0]$ and the above bound is effective only when $V_m + (n-m)$ is less than 0. This means that for fixed m, there is a limit to how large n can be taken before rendering the bound useless. On the other hand, in order to obtain the desired rate of exponential convergence one wishes to take n much larger than m so that the exponent can be approximated with high probability as

$$S_{m,n} \approx n/2.$$

Fortunately, by applying the same bound repeatedly these two conflicting constraints on the choice of n can be alleviated. For example, applying the bound first over $[m,k]$ and then over $[k,n]$ we obtain

$$V_n \leq \left[(V_m + (k-m))2^{S_{m,k}} + (n-k)\right]2^{S_{n,k}} \tag{3.12}$$

Now, a value of k modestly larger than m can ensure that V_k takes on a sufficiently large negative value to ensure that we can choose $n \gg k$. This will be shown below. However, still one needs to be able to begin with a large enough negative value for V_m to initiate the bootstrapping operation. The following result states that this can be done.

Proposition 13 *For any given $\varepsilon > 0$ and there exists $m_0(\varepsilon)$ such that for all $m \geq m_0(\varepsilon)$*

$$P\left(V_m \leq -2m\right) \geq I_0 - \varepsilon \tag{3.13}$$

Accepting the validity of Proposition 13 momentarily, we will show how to complete the proof of Proposition 12. We will prove Proposition 13 in the following two subsections.

Let $m \geq m_0(\varepsilon/3)$ be arbitrary. Set $k = 2m$ and $n = m^2$. Then, with probability at least $I_0 - \varepsilon/3$, we have by (3.12) that

$$V_{m^2} \leq \left(-m2^{S_{m,2m}} + (m^2 - 2m)\right)2^{S_{2m,m^2}}$$

For any given $\beta < 1/2$, we can choose $\beta' \in (\beta, 1/2)$ such that for m sufficiently large we have

$$P\left(S_{m,2m} > \beta' m\right) \geq 1 - \varepsilon/3$$

and

$$P\left(S_{2m,m^2} > \beta'(m^2 - m)\right) \geq 1 - \varepsilon/3$$

So, for such m we have with probability at least $I_0 - \varepsilon$

$$V_{m^2} \leq \left[-m2^{m\beta'} + (m^2 - 2m)\right]2^{(m^2-2m)\beta'}.$$

For a non-trivial bound we need to ensure that the term in square brackets is bounded away from zero on the negative side. So, we impose the following additional constraint on m:

$$\left[-m2^{m\beta'} + (m^2 - 2m)\right] < -1$$

which clearly can be met by choosing m large enough. Then, for all m satisfying all the constraints above we have

$$V_{m^2} \leq -2^{(m^2-2m)\beta'}$$

with probability at least $I_0 - \varepsilon$. This, written in terms of $n = m^2$ reads as

$$V_n \leq -2^{(n-o(n))\beta'} \leq -2^{n\beta}$$

where the second inequality holds for n large enough since $\beta' > \beta$.

3.5.2 Sealing the process in $[0, \zeta]$

The proof of Proposition 13 also contains a bootstrapping argument, but of a different type. We first establish a result that "seals" as much of the sample paths of $\{Z_n\}$ as possible in a small interval around zero. For $\zeta \geq 0$ and $\ell \geq 0$, define

$$\mathscr{T}_\ell(\zeta) \stackrel{\Delta}{=} \{\omega \in \Omega : Z_i(\omega) \leq \zeta \text{ for all } i \geq \ell\}.$$

Lemma 5 *For any $\zeta > 0$ and $\varepsilon > 0$, there exists $\ell_0(\zeta, \varepsilon)$ such that for all $\ell \geq \ell_0$*

$$P[\mathscr{T}_\ell(\zeta)] \geq I_0 - \varepsilon.$$

Proof. Fix $\zeta > 0$. Let $\Omega_0 \stackrel{\Delta}{=} \{\omega \in \Omega : \lim_{n\to\infty} Z_n(\omega) = 0\}$. By Prop. 10, $P(\Omega_0) = I_0$. Fix $\omega \in \Omega_0$. $Z_n(\omega) \to 0$ implies that there exists $n_0(\omega, \zeta)$ such that $n \geq n_0(\omega, \zeta) \Rightarrow Z_n(\omega) \leq \zeta$. Thus, $\omega \in \mathscr{T}_\ell(\zeta)$ for some m. So, $\Omega_0 \subset \bigcup_{\ell=1}^\infty \mathscr{T}_\ell(\zeta)$. Therefore, $P(\bigcup_{\ell=1}^\infty \mathscr{T}_\ell(\zeta)) \geq P(\Omega_0)$. Since $\mathscr{T}_\ell(\zeta) \uparrow \bigcup_{\ell=1}^\infty \mathscr{T}_\ell(\zeta)$, by the monotone convergence property of a measure, $\lim_{\ell\to\infty} P[\mathscr{T}_\ell(\zeta)] = P[\bigcup_{\ell=1}^\infty \mathscr{T}_\ell(\zeta)]$. So, $\lim_{\ell\to\infty} P[\mathscr{T}_\ell(\zeta)] \geq I_0$. It follows that, for any $\zeta > 0$, $\varepsilon > 0$, there exists a finite $\ell_0 = \ell_0(\zeta, \varepsilon)$ such that, for all $\ell \geq \ell_0$, $P[\mathscr{T}_\ell(\zeta)] \geq I_0 - \varepsilon$. This completes the proof.

3.5.3 Proof of Proposition 13

For $\omega \in \mathscr{T}_\ell(\zeta)$ and $i \geq \ell$, we have

$$\frac{Z_{i+1}(\omega)}{Z_i(\omega)} \leq \begin{cases} 2, & \text{if } B_{i+1}(\omega) = 0 \\ \zeta, & \text{if } B_{i+1}(\omega) = 1 \end{cases}$$

which implies

$$Z_m(\omega) \leq Z_\ell(\omega)\, 2^{m-\ell-S_{\ell,m}(\omega)}\, \zeta^{S_{\ell,m}(\omega)}, \quad \omega \in \mathscr{T}_\ell(\zeta),\ m > \ell.$$

This gives

$$Z_m(\omega) \leq Z_\ell(\omega) \left(2^{1-\beta}\, \zeta^\beta\right)^{m-\ell}, \quad \omega \in \mathscr{T}_\ell(\zeta) \cap \mathscr{S}_{\ell,m}(\beta).$$

Now, we set $\zeta = \zeta_0 := 2^{-9}$, $\beta = \beta_0 := 9/20$, $m = (7\ell/3)$, and note that $Z_\ell \leq 1$, to obtain

$$Z_m(\omega) \leq 2^{-2m}, \quad \omega \in \mathscr{T}_{(3m/7)}(\zeta_0) \cap \mathscr{S}_{(3m/7),m}(\beta_0). \tag{3.14}$$

The bound (3.11) and Lemma 5 ensure that there exists $m_0(\varepsilon)$ such that, for all $m \geq m_0(\varepsilon)$, (3.14) holds with probability greater than $I_0 - \varepsilon$. Specifically, it suffices to take m greater than both $(7/4)n_0(\beta_0, \varepsilon/2)$ and $(7/3)\ell_0(\zeta_0, \varepsilon/2)$.

3.5.4 Complementary remarks

Theorem 2 was first proved in [2] and the proof of the theorem proved above followed that paper closely. The channel polarization result as expressed by Theorem 2 does not show an explicit dependence on the rate parameter R except for the condition that $R < I_0$. Rate-dependent refinements of this theorem have appeared in [18], [8], [17] soon after the publication of [2]. For a more recent work on the same subject, see [7]. To state this refined polarization theorem, let $Q : \mathbb{R} \to [0,1]$ denote the complementary cumulative distribution function for the standard normal distribution:

$$Q(t) = \frac{1}{\sqrt{2\pi}} \int_t^\infty e^{-u^2/2} du.$$

Let Q^{-1} denote the inverse of Q. Then, the refined result can be stated in the present notation as follows.

Theorem 6 *For any $0 \leq R < I(W)$, the Bhattacharyya random process in polarization has asymptotic probabilities given by*

$$P\left(Z_n \leq 2^{-2^{[n+Q^{-1}(R/I_0)\sqrt{n}]/2+o(\sqrt{n})}}\right) \to R.$$

3.6 A side result

It is interesting that Propositon 9 gives a new interpretation to the symmetric capacity $I(W)$ as the probability that the random process $\{Z_n; n \geq 0\}$ converges to zero. Here, we use this to strengthen the lower bound in (0.1).

Proposition 14 *For any B-DMC W, we have $I(W) + Z(W) \geq 1$ with equality iff W is a BEC.*

This result can be interpreted as saying that, among all B-DMCs W, the BEC presents the most favorable rate-reliability trade-off: it minimizes $Z(W)$ (maximizes reliability) among all channels with a given symmetric capacity $I(W)$; equivalently, it minimizes $I(W)$ required to achieve a given level of reliability $Z(W)$.

Proof. Consider two channels W and W' with $Z(W) = Z(W') \stackrel{\Delta}{=} z_0$. Suppose that W' is a BEC. Then, W' has erasure probability z_0 and $I(W') = 1 - z_0$. Consider the random processes $\{Z_n\}$ and $\{Z'_n\}$. By the condition for equality in (2.18), the process $\{Z_n\}$ is stochastically dominated by $\{Z'_n\}$ in the sense that $P(Z_n \leq z) \geq P(Z'_n \leq z)$ for all $n \geq 1$, $0 \leq z \leq 1$. Thus, the probability of $\{Z_n\}$ converging to zero is lower-bounded by the probability that $\{Z'_n\}$ converges to zero, i.e., $I(W) \geq I(W')$. This implies $I(W) + Z(W) \geq 1$.

Chapter 4
Polar Coding

Abstract We show in this section that polar coding can achieve the symmetric capacity $I(W)$ of any B-DMC W.

4.1 Plan of chapter

The main technical task in this chapter will be to prove Prop. 2. We will carry out the analysis over the class of G_N-coset codes before specializing the discussion to polar codes. Recall that individual G_N-coset codes are identified by a parameter vector $(N, K, \mathscr{A}, u_{\mathscr{A}^c})$. In the analysis, we will fix the parameters $(N, K, \mathscr{A})$ while keeping $u_{\mathscr{A}^c}$ free to take any value over $\mathscr{X}^{N-K}$. In other words, the analysis will be over the ensemble of 2^{N-K} G_N-coset codes with a fixed $(N, K, \mathscr{A})$. The decoder in the system will be the SC decoder described in Sect. 1.2.2.

4.2 A probabilistic setting for the analysis

Let $(\mathscr{X}^N \times \mathscr{Y}^N, P)$ be a probability space with the probability assignment

$$P(\{(u_1^N, y_1^N)\}) \stackrel{\Delta}{=} 2^{-N} W_N(y_1^N | u_1^N) \tag{4.1}$$

for all $(u_1^N, y_1^N) \in \mathscr{X}^N \times \mathscr{Y}^N$. On this probability space, we define an ensemble of random vectors $(U_1^N, X_1^N, Y_1^N, \hat{U}_1^N)$ that represent, respectively, the input to the synthetic channel W_N, the input to the product-form channel W^N, the output of W^N (and also of W_N), and the decisions by the decoder. For each sample point $(u_1^N, y_1^N) \in \mathscr{X}^N \times \mathscr{Y}^N$, the first three vectors take on the values $U_1^N(u_1^N, y_1^N) = u_1^N$, $X_1^N(u_1^N, y_1^N) = u_1^N G_N$, and $Y_1^N(u_1^N, y_1^N) = y_1^N$, while the decoder output takes on the value $\hat{U}_1^N(u_1^N, y_1^N)$ whose coordinates are defined recursively as

$$\hat{U}_i(u_1^N, y_1^N) = \begin{cases} u_i, & i \in \mathscr{A}^c \\ h_i(y_1^N, \hat{U}_1^{i-1}(u_1^N, y_1^N)), & i \in \mathscr{A} \end{cases} \tag{4.2}$$

for $i = 1, \ldots, N$.

A realization $u_1^N \in \mathscr{X}^N$ for the input random vector U_1^N corresponds to sending the data vector $u_{\mathscr{A}}$ together with the frozen vector $u_{\mathscr{A}^c}$. As random vectors, the data part $U_{\mathscr{A}}$ and the frozen part $U_{\mathscr{A}^c}$ are uniformly distributed over their respective ranges and statistically independent. By treating $U_{\mathscr{A}^c}$ as a random vector over $\mathscr{X}^{N-K}$, we obtain a convenient method for analyzing code performance averaged over all codes in the ensemble $(N, K, \mathscr{A})$.

The main event of interest in the following analysis is the block error event under SC decoding, defined as

$$\mathscr{E} \stackrel{\Delta}{=} \{(u_1^N, y_1^N) \in \mathscr{X}^N \times \mathscr{Y}^N : \hat{U}_{\mathscr{A}}(u_1^N, y_1^N) \neq u_{\mathscr{A}}\}. \tag{4.3}$$

Since the decoder never makes an error on the frozen part of U_1^N, i.e., $\hat{U}_{\mathscr{A}^c}$ equals $U_{\mathscr{A}^c}$ with probability one, that part has been excluded from the definition of the block error event.

The probability of error terms $P_e(N, K, \mathscr{A})$ and $P_e(N, K, \mathscr{A}, u_{\mathscr{A}^c})$ that were defined in Sect. 1.2.3 can be expressed in this probability space as

$$\begin{aligned} P_e(N, K, \mathscr{A}) &= P(\mathscr{E}), \\ P_e(N, K, \mathscr{A}, u_{\mathscr{A}^c}) &= P(\mathscr{E} \mid \{U_{\mathscr{A}^c} = u_{\mathscr{A}^c}\}), \end{aligned} \tag{4.4}$$

where $\{U_{\mathscr{A}^c} = u_{\mathscr{A}^c}\}$ denotes the event $\{(\tilde{u}_1^N, y_1^N) \in \mathscr{X}^N \times \mathscr{Y}^N : \tilde{u}_{\mathscr{A}^c} = u_{\mathscr{A}^c}\}$.

4.3 Proof of Proposition 2

We may express the block error event as $\mathscr{E} = \cup_{i \in \mathscr{A}} \mathscr{B}_i$ where

$$\mathscr{B}_i \stackrel{\Delta}{=} \{(u_1^N, y_1^N) \in \mathscr{X}^N \times \mathscr{Y}^N : u_1^{i-1} = \hat{U}_1^{i-1}(u_1^N, y_1^N),\ u_i \neq \hat{U}_i(u_1^N, y_1^N)\} \tag{4.5}$$

is the event that the first decision error in SC decoding occurs at stage i. We notice that

$$\begin{aligned} \mathscr{B}_i &= \{(u_1^N, y_1^N) \in \mathscr{X}^N \times \mathscr{Y}^N : u_1^{i-1} = \hat{U}_1^{i-1}(u_1^N, y_1^N), u_i \neq h_i(y_1^N, \hat{U}_1^{i-1}(u_1^N, y_1^N))\} \\ &= \{(u_1^N, y_1^N) \in \mathscr{X}^N \times \mathscr{Y}^N : u_1^{i-1} = \hat{U}_1^{i-1}(u_1^N, y_1^N), u_i \neq h_i(y_1^N, u_1^{i-1})\} \\ &\subset \{(u_1^N, y_1^N) \in \mathscr{X}^N \times \mathscr{Y}^N : u_i \neq h_i(y_1^N, u_1^{i-1})\} \\ &\subset \mathscr{E}_i \end{aligned}$$

where

$$\mathscr{E}_i \triangleq \{(u_1^N, y_1^N) \in \mathscr{X}^N \times \mathscr{Y}^N : W_N^{(i-1)}(y_1^N, u_1^{i-1} \mid u_i) \leq W_N^{(i-1)}(y_1^N, u_1^{i-1} \mid u_i \oplus 1)\}. \tag{4.6}$$

Thus, we have

$$\mathscr{E} \subset \bigcup_{i \in \mathscr{A}} \mathscr{E}_i, \qquad P(\mathscr{E}) \leq \sum_{i \in \mathscr{A}} P(\mathscr{E}_i).$$

For an upper bound on $P(\mathscr{E}_i)$, note that

$$\begin{aligned} P(\mathscr{E}_i) &= \sum_{u_1^N, y_1^N} \frac{1}{2^N} W_N(y_1^N \mid u_1^N) 1_{\mathscr{E}_i}(u_1^N, y_1^N) \\ &\leq \sum_{u_1^N, y_1^N} \frac{1}{2^N} W_N(y_1^N \mid u_1^N) \sqrt{\frac{W_N^{(i)}(y_1^N, u_1^{i-1} | u_i \oplus 1)}{W_N^{(i)}(y_1^N, u_1^{i-1} | u_i)}} \\ &= Z(W_N^{(i)}). \end{aligned} \tag{4.7}$$

We conclude that

$$P(\mathscr{E}) \leq \sum_{i \in \mathscr{A}} Z(W_N^{(i)}),$$

which is equivalent to (1.13). This completes the proof of Prop. 2. The main coding theorem of the paper now follows readily.

4.4 Proof of Theorem 3

By Theorem 2, for any fixed rate $R < I(W)$ and constant $\beta < \frac{1}{2}$, there exists a sequence of sets $\{\mathscr{A}_N\}$ such that $\mathscr{A}_N \subset \{1, \ldots, N\}$, $|\mathscr{A}_N| \geq NR$, and

$$\sum_{i \in \mathscr{A}_N} Z(W_N^{(i)}) = o(2^{-N^\beta}). \tag{4.8}$$

In particular, the bound (4.8) holds if $\mathscr{A}_N$ is chosen in accordance with the polar coding rule because by definition this rule minimizes the sum in (4.8). Combining this fact about the polar coding rule with Prop. 2, Theorem 3 follows.

4.5 Symmetry under channel combining and splitting

Let $W : \mathscr{X} \to \mathscr{Y}$ be a symmetric B-DMC with $\mathscr{X} = \{0, 1\}$ and $\mathscr{Y}$ arbitrary. By definition, there exists a a permutation π_1 on $\mathscr{Y}$ such that (i) $\pi_1^{-1} = \pi_1$ and (ii) $W(y|1) = W(\pi_1(y)|0)$ for all $y \in \mathscr{Y}$. Let π_0 be the identity permutation on $\mathscr{Y}$.

Clearly, the permutations (π_0, π_1) form an abelian group under function composition. For a compact notation, we will write $x \cdot y$ to denote $\pi_x(y)$, for $x \in \mathscr{X}, y \in \mathscr{Y}$.

Observe that $W(y|x \oplus a) = W(a \cdot y|x)$ for all $a, x \in \mathscr{X}, y \in \mathscr{Y}$. This can be verified by exhaustive study of possible cases or by noting that $W(y|x \oplus a) = W((x \oplus a) \cdot y|0) = W(x \cdot (a \cdot y)|0) = W(a \cdot y|x)$. Also observe that $W(y|x \oplus a) = W(x \cdot y|a)$ as $\oplus$ is a commutative operation on $\mathscr{X}$.

For $x_1^N \in \mathscr{X}^N$, $y_1^N \in \mathscr{Y}^N$, let

$$x_1^N \cdot y_1^N \triangleq (x_1 \cdot y_1, \ldots, x_N \cdot y_N). \tag{4.9}$$

This associates to each element of $\mathscr{X}^N$ a permutation on $\mathscr{Y}^N$.

Proposition 15 *If a B-DMC W is symmetric, then W^N is also symmetric in the sense that*

$$W^N(y_1^N|x_1^N \oplus a_1^N) = W^N(x_1^N \cdot y_1^N|a_1^N) \tag{4.10}$$

for all $x_1^N, a_1^N \in \mathscr{X}^N$, $y_1^N \in \mathscr{Y}^N$.

The proof is immediate and omitted.

Proposition 16 *If a B-DMC W is symmetric, then the channels W_N and $W_N^{(i)}$ are also symmetric in the sense that*

$$W_N(y_1^N \mid u_1^N) = W_N(a_1^N G_N \cdot y_1^N \mid u_1^N \oplus a_1^N), \tag{4.11}$$

$$W_N^{(i)}(y_1^N, u_1^{i-1} \mid u_i) = W_N^{(i)}(a_1^N G_N \cdot y_1^N, u_1^{i-1} \oplus a_1^{i-1} \mid u_i \oplus a_i) \tag{4.12}$$

for all $u_1^N, a_1^N \in \mathscr{X}^N$, $y_1^N \in \mathscr{Y}^N$, $N = 2^n$, $n \geq 0$, $1 \leq i \leq N$.

Proof. Let $x_1^N = u_1^N G_N$ and observe that $W_N(y_1^N \mid u_1^N) = \prod_{i=1}^N W(y_i \mid x_i) = \prod_{i=1}^N W(x_i \cdot y_i \mid 0) = W_N(x_1^N \cdot y_1^N \mid 0_1^N)$. Now, let $b_1^N = a_1^N G_N$, and use the same reasoning to see that $W_N(b_1^N \cdot y_1^N \mid u_1^N \oplus a_1^N) = W_N((x_1^N \oplus b_1^N) \cdot (b_1^N \cdot y_1^N) \mid 0_1^N) = W_N(x_1^N \cdot y_1^N \mid 0_1^N)$. This proves the first claim. To prove the second claim, we use the first result.

$$\begin{aligned}
W_N^{(i)}(y_1^N, u_1^{i-1} \mid u_i) &= \sum_{u_{i+1}^N} \frac{1}{2^{N-1}} W_N(y_1^N \mid u_1^N) \\
&= \sum_{u_{i+1}^N} \frac{1}{2^{N-1}} W_N(a_1^N G_N \cdot y_1^N \mid u_1^N \oplus a_1^N) \\
&= W_N(a_1^N G_N \cdot y_1^N, u_1^{i-1} \oplus a_1^{i-1} \mid u_i \oplus a_i)
\end{aligned}$$

where we used the fact that the sum over $u_{i+1}^N \in \mathscr{X}^{N-i}$ can be replaced with a sum over $u_{i+1}^N \oplus a_{i+1}^N$ for any fixed a_1^N since $\{u_{i+1}^N \oplus a_{i+1}^N : u_{i+1}^N \in \mathscr{X}^{N-i}\} = X^{N-i}$.

4.6 Proof of Theorem 4

We return to the analysis in Sect. 4.3 and consider a code ensemble $(N, K, \mathscr{A})$ under SC decoding, only this time assuming that W is a symmetric channel. We first show that the error events $\{\mathscr{E}_i\}$ defined by (4.6) have a symmetry property.

Proposition 17 *For a symmetric B-DMC W, the event $\mathscr{E}_i$ has the property that*

$$(u_1^N, y_1^N) \in \mathscr{E}_i \quad \textit{iff} \quad (a_1^N \oplus u_1^N, a_1^N G_N \cdot y_1^N) \in \mathscr{E}_i \tag{4.13}$$

for each $1 \le i \le N$, $(u_1^N, y_1^N) \in \mathscr{X}^N \times \mathscr{Y}^N$, $a_1^N \in \mathscr{X}^N$.

Proof. This follows directly from the definition of $\mathscr{E}_i$ by using the symmetry property (4.12) of the channel $W_N^{(i)}$.

Now, consider the transmission of a particular source vector $u_{\mathscr{A}}$ and frozen vector $u_{\mathscr{A}^c}$, jointly forming an input vector u_1^N for the channel W_N. This event is denoted below as $\{U_1^N = u_1^N\}$ instead of the more formal $\{u_1^N\} \times \mathscr{Y}^N$.

Corollary 1 *For a symmetric B-DMC W, for each $1 \le i \le N$ and $u_1^N \in \mathscr{X}^N$, the events $\mathscr{E}_i$ and $\{U_1^N = u_1^N\}$ are independent; hence, $P(\mathscr{E}_i) = P(\mathscr{E}_i \mid \{U_1^N = u_1^N\})$.*

Proof. For $(u_1^N, y_1^N) \in \mathscr{X}^N \times \mathscr{Y}^N$ and $x_1^N = u_1^N G_N$, we have

$$\begin{aligned} P(\mathscr{E}_i \mid \{U_1^N = u_1^N\}) &= \sum_{y_1^N} W_N(y_1^N \mid u_1^N)\, 1_{\mathscr{E}_i}(u_1^N, y_1^N) \\ &= \sum_{y_1^N} W_N(x_1^N \cdot y_1^N \mid 0_1^N)\, 1_{\mathscr{E}_i}(0_1^N, x_1^N \cdot y_1^N) \qquad (4.14) \\ &= P(\mathscr{E}_i \mid \{U_1^N = 0_1^N\}). \qquad (4.15) \end{aligned}$$

Equality follows in (4.14) from (4.11) and (4.13) by taking $a_1^N = u_1^N$, and in (4.15) from the fact that $\{x_1^N \cdot y_1^N : y_1^N \in \mathscr{Y}^N\} = \mathscr{Y}^N$ for any fixed $x_1^N \in \mathscr{X}^N$. The rest of the proof is immediate.

Now, by (4.7), we have, for all $u_1^N \in \mathscr{X}^N$,

$$P(\mathscr{E}_i \mid \{U_1^N = u_1^N\}) \le Z(W_N^{(i)}) \tag{4.16}$$

and, since $\mathscr{E} \subset \cup_{i \in \mathscr{A}} \mathscr{E}_i$, we obtain

$$P(\mathscr{E} \mid \{U_1^N = u_1^N\}) \le \sum_{i \in \mathscr{A}} Z(W_N^{(i)}). \tag{4.17}$$

This implies that, for every symmetric B-DMC W and every $(N, K, \mathscr{A}, u_{\mathscr{A}^c})$ code,

$$P_e(N,K,\mathscr{A},u_{\mathscr{A}^c}) = \sum_{u_{\mathscr{A}} \in \mathscr{X}^K} \frac{1}{2^K} P(\mathscr{E} \mid \{U_1^N = u_1^N\})$$
$$\le \sum_{i \in \mathscr{A}} Z(W_N^{(i)}). \tag{4.18}$$

This bound on $P_e(N,K,\mathscr{A},u_{\mathscr{A}^c})$ is independent of the frozen vector $u_{\mathscr{A}^c}$. Theorem 4 is now obtained by combining Theorem 2 with Prop. 2, as in the proof of Theorem 3.

Note that although we have given a bound on $P(\mathscr{E}|\{U_1^N = u_1^N\})$ that is independent of u_1^N, we stopped short of claiming that the error event $\mathscr{E}$ is independent of U_1^N because our decision functions $\{h_i\}$ break ties always in favor of $\hat{u}_i = 0$. If this bias were removed by randomization, then $\mathscr{E}$ would become independent of U_1^N.

4.7 Further symmetries of the channel $W_N^{(i)}$

We may use the degrees of freedom in the choice of a_1^N in (4.12) to explore the symmetries inherent in the channel $W_N^{(i)}$. For a given (y_1^N, u_1^i), we may select a_1^N with $a_1^i = u_1^i$ to obtain

$$W_N^{(i)}(y_1^N, u_1^{i-1} \mid u_i) = W_N^{(i)}(a_1^N G_N \cdot y_1^N, 0_1^{i-1} \mid 0). \tag{4.19}$$

So, if we were to prepare a look-up table for the transition probabilities $\{W_N^{(i)}(y_1^N, u_1^{i-1} \mid u_i) : y_1^N \in \mathscr{Y}^N, u_1^i \in \mathscr{X}^i\}$, it would suffice to store only the subset of probabilities $\{W_N^{(i)}(y_1^N, 0_1^{i-1} \mid 0) : y_1^N \in \mathscr{Y}^N\}$.

The size of the look-up table can be reduced further by using the remaining degrees of freedom in the choice of a_{i+1}^N. Let $\mathscr{X}_{i+1}^N \stackrel{\Delta}{=} \{a_1^N \in \mathscr{X}^N : a_1^i = 0_1^i\}$, $1 \le i \le N$. Then, for any $1 \le i \le N$, $a_1^N \in \mathscr{X}_{i+1}^N$, and $y_1^N \in \mathscr{Y}^N$, we have

$$W_N^{(i)}(y_1^N, 0^{i-1}|0) = W_N^{(i)}(a_1^N G_N \cdot y_1^N, 0_1^{i-1}|0) \tag{4.20}$$

which follows from (4.19) by taking $u_1^i = 0_1^i$ on the left hand side.

To explore this symmetry further, let $\mathscr{X}_{i+1}^N \cdot y_1^N \stackrel{\Delta}{=} \{a_1^N G_N \cdot y_1^N : a_1^N \in \mathscr{X}_{i+1}^N\}$. The set $\mathscr{X}_{i+1}^N \cdot y_1^N$ is the *orbit* of y_1^N under the *action group* $\mathscr{X}_{i+1}^N$. The orbits $\mathscr{X}_{i+1}^N \cdot y_1^N$ over variation of y_1^N partition the space $\mathscr{Y}^N$ into equivalence classes. Let $\mathscr{Y}_{i+1}^N$ be a set formed by taking one representative from each equivalence class. The output alphabet of the channel $W_N^{(i)}$ can be represented effectively by the set $\mathscr{Y}_{i+1}^N$.

For example, suppose W is a BSC with $\mathscr{Y} = \{0,1\}$. Each orbit $\mathscr{X}_{i+1}^N \cdot y_1^N$ has 2^{N-i} elements and there are 2^i orbits. In particular, the channel $W_N^{(1)}$ has effectively two outputs, and being symmetric, it has to be a BSC. This is a great simplification since $W_N^{(1)}$ has an apparent output alphabet size of 2^N. Likewise, while $W_N^{(i)}$ has an apparent output alphabet size of 2^{N+i-1}, due to symmetry, the size shrinks to 2^i.

Further output alphabet size reductions may be possible by exploiting other properties specific to certain B-DMCs. For example, if W is a BEC, the channels $\{W_N^{(i)}\}$ are known to be BECs, each with an effective output alphabet size of three.

The symmetry properties of $\{W_N^{(i)}\}$ help simplify the computation of the channel parameters.

Proposition 18 *For any symmetric B-DMC W, the parameters $\{Z(W_N^{(i)})\}$ given by (1.5) can be calculated by the simplified formula*

$$Z(W_N^{(i)}) = 2^{i-1} \sum_{y_1^N \in \mathscr{Y}_{i+1}^N} |\mathscr{X}_{i+1}^N \cdot y_1^N| \sqrt{W_N^{(i)}(y_1^N, 0_1^{i-1}|0) W_N^{(i)}(y_1^N, 0_1^{i-1}|1)}.$$

We omit the proof of this result.

For the important example of a BSC, this formula becomes

$$Z(W_N^{(i)}) = 2^{N-1} \sum_{y_1^N \in \mathscr{Y}_{i+1}^N} \sqrt{W_N^{(i)}(y_1^N, 0_1^{i-1}|0)\, W_N^{(i)}(y_1^N, 0_1^{i-1}|1)}.$$

This sum for $Z(W_N^{(i)})$ has 2^i terms, as compared to 2^{N+i-1} terms in (1.5).

Chapter 5
Encoding, Decoding and Construction of Polar Codes

Abstract This chapter considers the encoding, decoding, and construction problems for polar coding.

5.1 Encoding

In this section, we will consider the encoding of polar codes and prove the part of Theorem 5 about encoding complexity. We begin by giving explicit algebraic expressions for G_N, the generator matrix for polar coding, which so far has been defined only in a schematic form by Fig. 3. The algebraic forms of G_N naturally point at efficient implementations of the encoding operation $x_1^N = u_1^N G_N$. In analyzing the encoding operation G_N, we exploit its relation to fast transform methods in signal processing; in particular, we use the bit-indexing idea of [4] to interpret the various permutation operations that are part of G_N.

5.1.1 Formulas for G_N

In the following, assume $N = 2^n$ for some $n \geq 0$. Let I_k denote the k-dimensional identity matrix for any $k \geq 1$. We begin by translating the recursive definition of G_N as given by Fig. 3 into an algebraic form:

$$G_N = (I_{N/2} \otimes F) R_N (I_2 \otimes G_{N/2}), \quad \text{for } N \geq 2,$$

with $G_1 = I_1$.

Either by verifying algebraically that $(I_{N/2} \otimes F)R_N = R_N(F \otimes I_{N/2})$ or by observing that channel combining operation in Fig. 3 can be redrawn equivalently as in Fig. 8, we obtain a second recursive formula

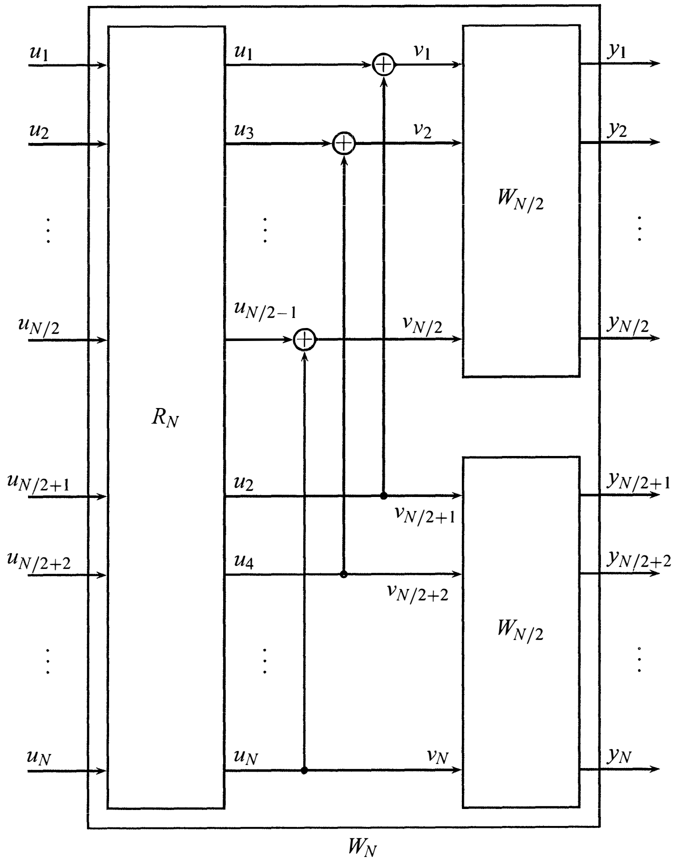


Fig. 5.1 An alternative realization of the recursive construction for W_N.

$$\begin{aligned} G_N &= R_N(F \otimes I_{N/2})(I_2 \otimes G_{N/2}) \\ &= R_N(F \otimes G_{N/2}), \end{aligned} \tag{5.1}$$

valid for $N \geq 2$. This form appears more suitable to derive a recursive relationship. We substitute $G_{N/2} = R_{N/2}(F \otimes G_{N/4})$ back into (5.1) to obtain

$$\begin{aligned} G_N &= R_N \left(F \otimes \left(R_{N/2} \left(F \otimes G_{N/4}\right)\right)\right) \\ &= R_N \left(I_2 \otimes R_{N/2}\right) \left(F^{\otimes 2} \otimes G_{N/4}\right) \end{aligned} \tag{5.2}$$

where (5.2) is obtained by using the identity $(AC) \otimes (BD) = (A \otimes B)(C \otimes D)$ with $A = I_2, B = R_{N/2}, C = F, D = F \otimes G_{N/4}$. Repeating this, we obtain

$$G_N = B_N F^{\otimes n} \tag{5.3}$$

where $B_N \triangleq R_N(I_2 \otimes R_{N/2})(I_4 \otimes R_{N/4}) \cdots (I_{N/2} \otimes R_2)$. It can seen by simple manipulations that

$$B_N = R_N(I_2 \otimes B_{N/2}). \tag{5.4}$$

We can see that B_N is a permutation matrix by the following induction argument. Assume that $B_{N/2}$ is a permutation matrix for some $N \geq 4$; this is true for $N = 4$ since $B_2 = I_2$. Then, B_N is a permutation matrix because it is the product of two permutation matrices, R_N and $I_2 \otimes B_{N/2}$.

In the following, we will say more about the nature of B_N as a permutation.

5.1.2 Analysis by bit-indexing

To analyze the encoding operation further, it will be convenient to index vectors and matrices with bit sequences. Given a vector a_1^N with length $N = 2^n$ for some $n \geq 0$, we denote its *i*th element, a_i, $1 \leq i \leq N$, alternatively as $a_{b_1 \cdots b_n}$ where $b_1 \cdots b_n$ is the binary expansion of the integer $i-1$ in the sense that $i = 1 + \sum_{j=1}^{n} b_j 2^{n-j}$. Likewise, the element A_{ij} of an N-by-N matrix A is denoted alternatively as $A_{b_1 \cdots b_n, b_1' \cdots b_n'}$ where $b_1 \cdots b_n$ and $b_1' \cdots b_n'$ are the binary representations of $i-1$ and $j-1$, respectively. Using this convention, it can be readily verified that the product $C = A \otimes B$ of a 2^n-by-2^n matrix A and a 2^m-by-2^m matrix B has elements $C_{b_1 \cdots b_{n+m}, b_1' \cdots b_{n+m}'} = A_{b_1 \cdots b_n, b_1' \cdots b_n'} B_{b_{n+1} \cdots b_{n+m}, b_{n+1}' \cdots b_{n+m}'}$.

We now consider the encoding operation under bit-indexing. First, we observe that the elements of F in bit-indexed form are given by $F_{b,b'} = 1 \oplus b' \oplus bb'$ for all $b, b' \in \{0,1\}$. Thus, $F^{\otimes n}$ has elements

$$F^{\otimes n}_{b_1 \cdots b_n, b_1' \cdots b_n'} = \prod_{i=1}^{n} F_{b_i, b_i'} = \prod_{i=1}^{n} (1 \oplus b_i' \oplus b_i b_i'). \tag{5.5}$$

Second, the reverse shuffle operator R_N acts on a row vector u_1^N to replace the element in bit-indexed position $b_1 \cdots b_n$ with the element in position $b_2 \cdots b_n b_1$; that is, if $v_1^N = u_1^N R_N$, then $v_{b_1 \cdots b_n} = u_{b_2 \cdots b_n b_1}$ for all $b_1, \ldots, b_n \in \{0,1\}$. In other words, R_N cyclically rotates the bit-indexes of the elements of a left operand u_1^N to the right by one place.

Third, the matrix B_N in (5.3) can be interpreted as the *bit-reversal* operator: if $v_1^N = u_1^N B_N$, then $v_{b_1 \cdots b_n} = u_{b_n \cdots b_1}$ for all $b_1, \ldots, b_n \in \{0,1\}$. This statement can be proved by induction using the recursive formula (5.4). We give the idea of such a proof by an example. Let us assume that B_4 is a bit-reversal operator and show that the same is true for B_8. Let u_1^8 be any vector over $GF(2)$. Using bit-indexing, it can be written as $(u_{000}, u_{001}, u_{010}, u_{011}, u_{100}, u_{101}, u_{110}, u_{111})$. Since $u_1^8 B_8 = u_1^8 R_8 (I_2 \otimes B_4)$, let us first consider the action of R_8 on u_1^8. The reverse shuffle R_8 rearranges the elements of u_1^8 with respect to odd-even parity of their indices, so $u_1^8 R_8$ equals $(u_{000}, u_{010}, u_{100}, u_{110}, u_{001}, u_{011}, u_{101}, u_{111})$. This has two halves, $c_1^4 \stackrel{\Delta}{=} (u_{000}, u_{010}, u_{100}, u_{110})$ and $d_1^4 \stackrel{\Delta}{=} (u_{001}, u_{011}, u_{101}, u_{111})$, corresponding to odd-even index classes. Notice that $c_{b_1 b_2} = u_{b_1 b_2 0}$ and $d_{b_1 b_2} = u_{b_1 b_2 1}$ for all $b_1, b_2 \in \{0,1\}$. This is to be expected since the reverse shuffle rearranges the indices in increasing order within each odd-even index class. Next, consider the action of $I_2 \otimes B_4$ on (c_1^4, d_1^4). The result is $(c_1^4 B_4, d_1^4 B_4)$. By assumption, B_4 is a bit-reversal operation, so $c_1^4 B_4 = (c_{00}, c_{10}, c_{01}, c_{11})$, which in turn equals $(u_{000}, u_{100}, u_{010}, u_{110})$. Likewise, the result of $d_1^4 B_4$ equals $(u_{001}, u_{101}, u_{011}, u_{111})$. Hence, the overall operation B_8 is a bit-reversal operation.

Given the bit-reversal interpretation of B_N, it is clear that B_N is a symmetric matrix, so $B_N^T = B_N$. Since B_N is a permutation, it follows from symmetry that $B_N^{-1} = B_N$.

It is now easy to see that, for any N-by-N matrix A, the product $C = B_N^T A B_N$ has elements $C_{b_1 \cdots b_n, b_1' \cdots b_n'} = A_{b_n \cdots b_1, b_n' \cdots b_1'}$. It follows that if A is invariant under bit-reversal, i.e., if $A_{b_1 \cdots b_n, b_1' \cdots b_n'} = A_{b_n \cdots b_1, b_n' \cdots b_1'}$ for every $b_1, \ldots, b_n, b_1', \ldots, b_n' \in \{0,1\}$, then $A = B_N^T A B_N$. Since $B_N^T = B_N^{-1}$, this is equivalent to $B_N A = A B_T$. Thus, bit-reversal-invariant matrices commute with the bit-reversal operator.

Proposition 19 *For any $N = 2^n$, $n \geq 1$, the generator matrix G_N is given by $G_N = B_N F^{\otimes n}$ and $G_N = F^{\otimes n} B_N$ where B_N is the bit-reversal permutation. G_N is a bit-reversal invariant matrix with*

$$(G_N)_{b_1 \cdots b_n, b_1' \cdots b_n'} = \prod_{i=1}^{n} (1 \oplus b_i' \oplus b_{n-i} b_i'). \tag{5.6}$$

Proof. $F^{\otimes n}$ commutes with B_N because it is invariant under bit-reversal, which is immediate from (5.5). The statement $G_N = B_N F^{\otimes n}$ was established before; by proving that $F^{\otimes n}$ commutes with B_N, we have established the other statement: $G_N = F^{\otimes n} B_N$. The bit-indexed form (5.6) follows by applying bit-reversal to (5.5).

A fact useful for estimation of minimum Hamming distances of polar codes is the following.

Proposition 20 *For any $N = 2^n$, $n \geq 0$, $b_1, \ldots, b_n \in \{0,1\}$, the rows of G_N and $F^{\otimes n}$ with index $b_1 \cdots b_n$ have the same Hamming weight given by $2^{w_H(b_1, \ldots, b_n)}$.*

Proof. For fixed $b_1, \ldots, b_n$, the sum of the terms $(G_N)_{b_1 \cdots b_n, b_1' \cdots b_n'}$ (as integers) over all $b_1', \ldots, b_n' \in \{0,1\}$ gives the Hamming weight of the row of G_N with index $b_1 \cdots b_n$. This sum is easily seen to be $2^{w_H(b_1, \ldots, b_n)}$ where

$$w_H(b_1, \ldots, b_n) \stackrel{\Delta}{=} \sum_{i=1}^{n} b_i \tag{5.7}$$

is the Hamming weight of $(b_1, \ldots, b_n)$. The proof for $F^{\otimes n}$ is obtained by using the same argument on (5.5).

5.1.3 Encoding complexity

For complexity estimation, our computational model will be a single processor machine with a random access memory. The complexities expressed will be time complexities. The discussion will be given for an arbitrary G_N-coset code with parameters $(N, K, \mathscr{A}, u_{\mathscr{A}^c})$.

Let $\chi_E(N)$ denote the worst-case encoding complexity over all $(N, K, \mathscr{A}, u_{\mathscr{A}^c})$ codes with a given block-length N. If we take the complexity of a scalar mod-2 addition as 1 unit and the complexity of the reverse shuffle operation R_N as N units, we see from Fig. 3 that $\chi_E(N) \leq N/2 + N + 2\chi_E(N/2)$. Starting with an initial value $\chi_E(2) = 3$ (a generous figure), we obtain by induction that $\chi_E(N) \leq \frac{3}{2} N \log N$ for all $N = 2^n$, $n \geq 1$. Thus, the encoding complexity is $O(N \log N)$.

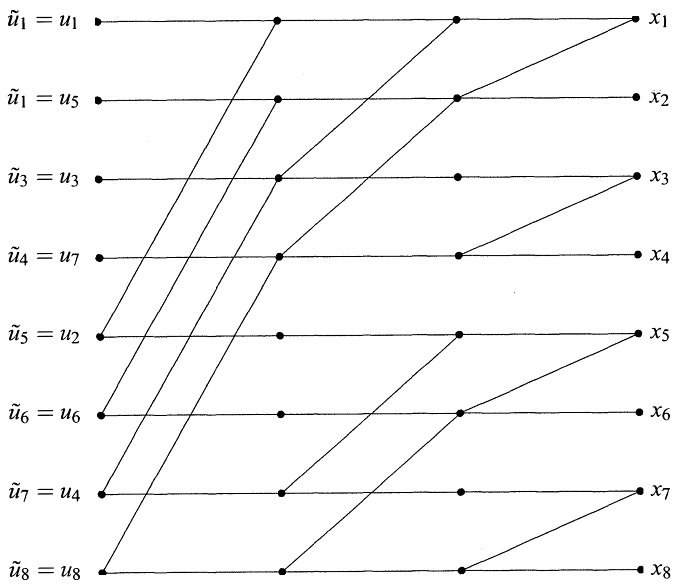


Fig. 5.2 A circuit for implementing the transformation $F^{\otimes 3}$. Signals flow from left to right. Each edge carries a signal 0 or 1. Each node adds (mod-2) the signals on all incoming edges from the left and sends the result out on all edges to the right. (Edges carrying the signals u_i and x_i are not shown.)

A specific implementation of the encoder using the form $G_N = B_N F^{\otimes n}$ is shown in Fig. 9 for $N = 8$. The input to the circuit is the bit-reversed version of u_1^8, i.e., $\tilde{u}_1^8 = u_1^8 B_8$. The output is given by $x_1^8 = \tilde{u}_1^8 F^{\otimes 3} = u_1^8 G_8$. In general, the complexity of this implementation is $O(N \log N)$ with $O(N)$ for B_N and $O(N \log N)$ for $F^{\otimes n}$.

An alternative implementation of the encoder would be to apply u_1^8 in natural index order at the input of the circuit in Fig. 9. Then, we would obtain $\tilde{x}_1^8 = u_1^8 F^{\otimes 3}$

at the output. Encoding could be completed by a post bit-reversal operation: $x_1^8 = \tilde{x}_1^8 B_8 = u_1^8 G_8$.

The encoding circuit of Fig. 9 suggests many parallel implementation alternatives for $F^{\otimes n}$: for example, with N processors, one may do a "column by column" implementation, and reduce the total latency to $\log N$. Various other trade-offs are possible between latency and hardware complexity.

In an actual implementation of polar codes, it may be preferable to use $F^{\otimes n}$ in place of $B_N F^{\otimes n}$ as the encoder mapping in order to simplify the implementation. In that case, the SC decoder should compensate for this by decoding the elements of the source vector u_1^N in bit-reversed index order. We have included B_N as part of the encoder in this paper in order to have a SC decoder that decodes u_1^N in the natural index order, which simplified the notation.

5.2 Decoding

In this section, we consider the computational complexity of the SC decoding algorithm. As in the previous section, our computational model will be a single processor machine with a random access memory and the complexities expressed will be time complexities. Let $\chi_D(N)$ denote the worst-case complexity of SC decoding over all G_N-coset codes with a given block-length N. We will show that $\chi_D(N) = O(N \log N)$.

5.2.1 A first decoding algorithm

Consider SC decoding for an arbitrary G_N-coset code with parameter $(N, K, \mathscr{A}, u_{\mathscr{A}^c})$. Recall that the source vector u_1^N consists of a random part $u_{\mathscr{A}}$ and a frozen part $u_{\mathscr{A}^c}$. This vector is transmitted across W_N and a channel output y_1^N is obtained with probability $W_N(y_1^N | u_1^N)$. The SC decoder observes $(y_1^N, u_{\mathscr{A}^c})$ and generates an estimate $\hat{u}_1^N$ of u_1^N. We may visualize the decoder as consisting of N decision elements (DEs), one for each source element u_i; the DEs are activated in the order 1 to N. If $i \in \mathscr{A}^c$, the element u_i is known; so, the ith DE, when its turn comes, simply sets $\hat{u}_i = u_i$ and sends this result to all succeeding DEs. If $i \in \mathscr{A}$, the ith DE waits until it has received the previous decisions $\hat{u}_1^{i-1}$, and upon receiving them, computes the likelihood ratio (LR)

$$L_N^{(i)}(y_1^N, \hat{u}_1^{i-1}) \triangleq \frac{W_N^{(i)}(y_1^N, \hat{u}_1^{i-1} | 0)}{W_N^{(i)}(y_1^N, \hat{u}_1^{i-1} | 1)}$$

and generates its decision as

$$\hat{u}_i = \begin{cases} 0, & \text{if } L_N^{(i)}(y_1^N, \hat{u}_1^{i-1}) \geq 1 \\ 1, & \text{otherwise} \end{cases}$$

which is then sent to all succeeding DEs. This is a single-pass algorithm, with no revision of estimates. The complexity of this algorithm is determined essentially by the complexity of computing the LRs.

A straightforward calculation using the recursive formulas (2.6) and (2.7) gives

$$L_N^{(2i-1)}(y_1^N, \hat{u}_1^{2i-2}) = \frac{L_{N/2}^{(i)}(y_1^{N/2}, \hat{u}_{1,o}^{2i-2} \oplus \hat{u}_{1,e}^{2i-2})\, L_{N/2}^{(i)}(y_{N/2+1}^N, \hat{u}_{1,e}^{2i-2}) + 1}{L_{N/2}^{(i)}(y_1^{N/2}, \hat{u}_{1,o}^{2i-2} \oplus \hat{u}_{1,e}^{2i-2}) + L_{N/2}^{(i)}(y_{N/2+1}^N, \hat{u}_{1,e}^{2i-2})} \tag{5.8}$$

and

$$L_N^{(2i)}(y_1^N, \hat{u}_1^{2i-1}) = \left[L_{N/2}^{(i)}(y_1^{N/2}, \hat{u}_{1,o}^{2i-2} \oplus \hat{u}_{1,e}^{2i-2})\right]^{1-2\hat{u}_{2i-1}} \cdot L_{N/2}^{(i)}(y_{N/2+1}^N, \hat{u}_{1,e}^{2i-2}). \tag{5.9}$$

Thus, the calculation of an LR at length N is reduced to the calculation of two LRs at length $N/2$. This recursion can be continued down to block-length 1, at which point the LRs have the form $L_1^{(1)}(y_i) = W(y_i|0)/W(y_i|1)$ and can be computed directly.

To estimate the complexity of LR calculations, let $\chi_L(k)$, $k \in \{N, N/2, N/4, \ldots, 1\}$, denote the worst-case complexity of computing $L_k^{(i)}(y_1^k, v_1^{i-1})$ over $i \in [1,k]$ and $(y_1^k, v_1^{i-1}) \in \mathscr{Y}^k \times \mathscr{X}^{i-1}$. From the recursive LR formulas, we have the complexity bound

$$\chi_L(k) \leq 2\chi_L(k/2) + \alpha \tag{5.10}$$

where α is the worst-case complexity of assembling two LRs at length $k/2$ into an LR at length k. Taking $\chi_L^{(1)}(y_i)$ as 1 unit, we obtain the bound

$$\chi_L(N) \leq (1+\alpha)N = O(N). \tag{5.11}$$

The overall decoder complexity can now be bounded as $\chi_D(N) \leq K\chi_L(N) \leq N\chi_L(N) = O(N^2)$. This complexity corresponds to a decoder whose DEs do their LR calculations privately, without sharing any partial results with each other. It turns out, if the DEs pool their scratch-pad results, a more efficient decoder implementation is possible with overall complexity $O(N \log N)$, as we will show next.

5.2.2 Refinement of the decoding algorithm

We now consider a decoder that computes the full set of LRs, $\{L_N^{(i)}(y_1^N, \hat{u}_1^{i-1}) : 1 \leq i \leq N\}$. The previous decoder could skip the calculation of $L_N^{(i)}(y_1^N, \hat{u}_1^{i-1})$ for $i \in \mathscr{A}^c$; but now we do not allow this. The decisions $\{\hat{u}_i : 1 \leq i \leq N\}$ are made in exactly the same manner as before; in particular, if $i \in \mathscr{A}^c$, the decision $\hat{u}_i$ is set to the known frozen value u_i, regardless of $L_N^{(i)}(y_1^N, \hat{u}_1^{i-1})$.

To see where the computational savings will come from, we inspect (5.8) and (5.9) and note that each LR value in the pair

$$(L_N^{(2i-1)}(y_1^N, \hat{u}_1^{2i-2}), L_N^{(2i)}(y_1^N, \hat{u}_1^{2i-1}))$$

is assembled from the same pair of LRs:

$$(L_{N/2}^{(i)}(y_1^{N/2}, \hat{u}_{1,o}^{2i-2} \oplus \hat{u}_{1,e}^{2i-2}), L_{N/2}^{(i)}(y_{N/2+1}^N, \hat{u}_{1,e}^{2i-2})).$$

Thus, the calculation of all N LRs at length N requires exactly N LR calculations at length $N/2$.[1] Let us split the N LRs at length $N/2$ into two classes, namely,

$$\begin{aligned} &\{L_{N/2}^{(i)}(y_1^{N/2}, \hat{u}_{1,o}^{2i-2} \oplus \hat{u}_{1,e}^{2i-2}) : 1 \leq i \leq N/2\}, \\ &\{L_{N/2}^{(i)}(y_{N/2+1}^N, \hat{u}_{1,e}^{2i-2}) : 1 \leq i \leq N/2\}. \end{aligned} \tag{5.12}$$

Let us suppose that we carry out the calculations in each class independently, without trying to exploit any further savings that may come from the sharing of LR values between the two classes. Then, we have two problems of the same type as the original but at half the size. Each class in (5.12) generates a set of $N/2$ LR calculation requests at length $N/4$, for a total of N requests. For example, if we let $\hat{v}_1^{N/2} \triangleq \hat{u}_{1,o}^{N/2} \oplus \hat{u}_{1,e}^{N/2}$, the requests arising from the first class are

$$\begin{aligned} &\{L_{N/4}^{(i)}(y_1^{N/4}, \hat{v}_{1,o}^{2i-2} \oplus \hat{v}_{1,e}^{2i-2}) : 1 \leq i \leq N/4\}, \\ &\{L_{N/4}^{(i)}(y_{N/4+1}^{N/2}, \hat{v}_{1,e}^{2i-2}) : 1 \leq i \leq N/4\}. \end{aligned}$$

Using this reasoning inductively across the set of all lengths $\{N, N/2, \ldots, 1\}$, we conclude that the total number of LRs that need to be calculated is $N(1 + \log N)$.

So far, we have not paid attention to the exact order in which the LR calculations at various block-lengths are carried out. Although this gave us an accurate count of the total number of LR calculations, for a full description of the algorithm, we need to specify an order. There are many possibilities for such an order, but to be specific we will use a depth-first algorithm, which is easily described by a small example.

---

[1] Actually, some LR calculations at length $N/2$ may be avoided if, by chance, some duplications occur, but we will disregard this.

We consider a decoder for a code with parameter $(N, K, \mathscr{A}, u_{\mathscr{A}^c})$ chosen as $(8, 5, \{3, 5, 6, 7, 8\}, (0, 0, 0)\}$. The computation for the decoder is laid out in a graph as shown in Fig. 10. There are $N(1 + \log N) = 32$ nodes in the graph, each responsible for computing an LR request that arises during the course of the algorithm. Starting from the left-side, the first column of nodes correspond to LR requests at length 8 (decision level), the second column of nodes to requests at length 4, the third at length 2, and the fourth at length 1 (channel level).

Each node in the graph carries two labels. For example, the third node from the bottom in the third column has the labels $(y_5^6, \hat{u}_2 \oplus \hat{u}_4)$ and 26; the first label indicates that the LR value to be calculated at this node is $L_8^{(2)}(y_5^6, \hat{u}_2 \oplus \hat{u}_4)$ while the second label indicates that this node will be the 26th node to be activated. The numeric labels, 1 through 32, will be used as quick identifiers in referring to nodes in the graph.

The decoder is visualized as consisting of N DEs situated at the left-most side of the decoder graph. The node with label $(y_1^8, \hat{u}_1^{i-1})$ is associated with the ith DE, $1 \leq i \leq 8$. The positioning of the DEs in the left-most column follows the bit-reversed index order, as in Fig. 9.

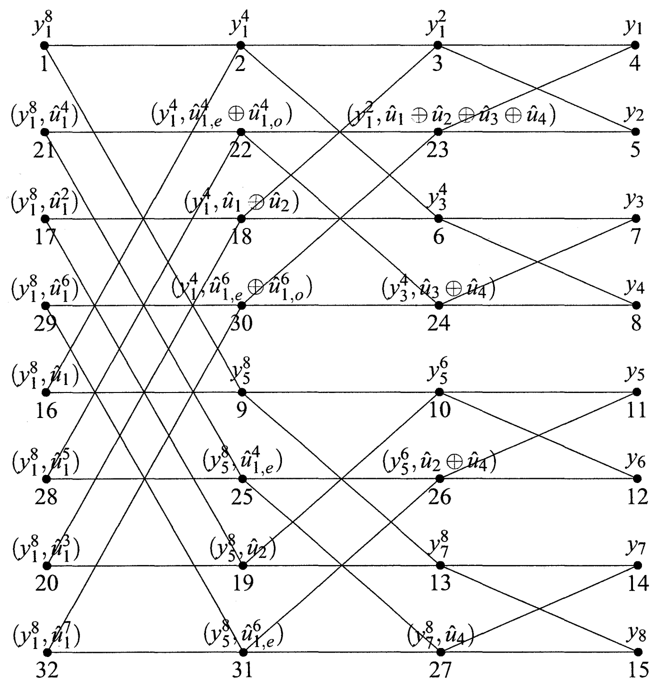


Fig. 5.3 An implementation of the successive cancellation decoder for polar coding at block-length $N = 8$.

Decoding begins with DE 1 activating node 1 for the calculation of $L_8^{(1)}(y_1^8)$. Node 1 in turn activates node 2 for $L_4^{(1)}(y_1^4)$. At this point, program control passes to node 2, and node 1 will wait until node 2 delivers the requested LR. The process continues. Node 2 activates node 3, which activates node 4. Node 4 is a node at the channel level; so it computes $L_1^{(1)}(y_1)$ and passes it to nodes 3 and 23, its left-side neighbors. In general a node will send its computational result to all its left-side neighbors (although this will not be stated explicitly below). Program control will be passed back to the left neighbor from which it was received.

Node 3 still needs data from the right side and activates node 5, which delivers $L_1^{(1)}(y_2)$. Node 3 assembles $L_2^{(1)}(y_1^2)$ from the messages it has received from nodes 4 and 5 and sends it to node 2. Next, node 2 activates node 6, which activates nodes 7 and 8, and returns its result to node 2. Node 2 compiles its response $L_4^{(1)}(y_1^4)$ and sends it to node 1. Node 1 activates node 9 which calculates $L_4^{(1)}(y_5^8)$ in the same manner as node 2 calculated $L_4^{(1)}(y_1^4)$, and returns the result to node 1. Node 1 now assembles $L_8^{(1)}(y_1^8)$ and sends it to DE 1. Since u_1 is a frozen node, DE 1 ignores the received LR, declares $\hat{u}_1 = 0$, and passes control to DE 2, located next to node 16.

DE 2 activates node 16 for $L_8^{(2)}(y_1^8, \hat{u}_1)$. Node 16 assembles $L_8^{(2)}(y_1^8, \hat{u}_1)$ from the already-received LRs $L_4^{(1)}(y_1^4)$ and $L_4^{(1)}(y_5^8)$, and returns its response without activating any node. DE 2 ignores the returned LR since u_2 is frozen, announces $\hat{u}_2 = 0$, and passes control to DE 3.

DE 3 activates node 17 for $L_8^{(3)}(y_1^8, \hat{u}_1^2)$. This triggers LR requests at nodes 18 and 19, but no further. The bit u_3 is not frozen; so, the decision $\hat{u}_3$ is made in accordance with $L_8^{(3)}(y_1^8, \hat{u}_1^2)$, and control is passed to DE 4. DE 4 activates node 20 for $L_8^{(4)}(y_1^8, \hat{u}_1^3)$, which is readily assembled and returned. The algorithm continues in this manner until finally DE 8 receives $L_8^{(7)}(y_1^8, \hat{u}_1^7)$ and decides $\hat{u}_8$.

There are a number of observations that can be made by looking at this example that should provide further insight into the general decoding algorithm. First, notice that the computation of $L_8^{(1)}(y_1^8)$ is carried out in a subtree rooted at node 1, consisting of paths going from left to right, and spanning all nodes at the channel level. This subtree splits into two disjoint subtrees, namely, the subtree rooted at node 2 for the calculation of $L_4^{(1)}(y_1^4)$ and the subtree rooted at node 9 for the calculation of $L_4^{(1)}(y_5^8)$. Since the two subtrees are disjoint, the corresponding calculations can be carried out independently (even in parallel if there are multiple processors). This splitting of computational subtrees into disjoint subtrees holds for all nodes in the graph (except those at the channel level), making it possible to implement the decoder with a high degree of parallelism.

Second, we notice that the decoder graph consists of *butterflies* (2-by-2 complete bipartite graphs) that tie together adjacent levels of the graph. For example, nodes 9, 19, 10, and 13 form a butterfly. The computational subtrees rooted at nodes 9 and 19 split into a single pair of computational subtrees, one rooted at node 10, the other at node 13. Also note that among the four nodes of a butterfly, the upper-left node is always the first node to be activated by the above depth-first algorithm and the lower-left node always the last one. The upper-right and lower-right nodes are activated by the upper-left node and they may be activated in any order or even in parallel. The algorithm we specified always activated the upper-right node first, but this choice was arbitrary. When the lower-left node is activated, it finds the LRs from its right neighbors ready for assembly. The upper-left node assembles the LRs it receives from the right side as in formula (5.8), the lower-left node as in (5.9). These formulas show that the butterfly patterns impose a constraint on the completion time of LR calculations: in any given butterfly, the lower-left node needs to wait for the result of the upper-left node which in turn needs to wait for the results of the right-side nodes.

Variants of the decoder are possible in which the nodal computations are scheduled differently. In the "left-to-right" implementation given above, nodes waited to be activated. However, it is possible to have a "right-to-left" implementation in which each node starts its computation autonomously as soon as its right-side neighbors finish their calculations; this allows exploiting parallelism in computations to the maximum possible extent.

For example, in such a fully-parallel implementation for the case in Fig. 10, all eight nodes at the channel-level start calculating their respective LRs in the first time slot following the availability of the channel output vector y_1^8. In the second time slot, nodes 3, 6, 10, and 13 do their LR calculations in parallel. Note that this is the maximum degree of parallelism possible in the second time slot. Node 23, for example, cannot calculate $L_N^{(2)}(y_1^2, \hat{u}_1 \oplus \hat{u}_2 \oplus \hat{u}_3 \oplus \hat{u}_4)$ in this slot, because $\hat{u}_1 \oplus \hat{u}_2 \oplus \hat{u}_3 \oplus \hat{u}_4$ is not yet available; it has to wait until decisions $\hat{u}_1$, $\hat{u}_2$, $\hat{u}_3$, $\hat{u}_4$ are announced by the corresponding DEs. In the third time slot, nodes 2 and 9 do their calculations. In time slot 4, the first decision $\hat{u}_1$ is made at node 1 and broadcast to all nodes across the graph (or at least to those that need it). In slot 5, node 16 calculates $\hat{u}_2$ and broadcasts it. In slot 6, nodes 18 and 19 do their calculations. This process continues until time slot 15 when node 32 decides $\hat{u}_8$. It can be shown that, in general, this fully-parallel decoder implementation has a latency of $2N - 1$ time slots for a code of block-length N.

5.3 Code construction

The original polar coding paper [1] left the polar coding construction problem unsolved. Only for the BEC, a solution was given. For the general case, a Monte Carlo simulation method was suggested. Although the problem looked very formidable, rapid progress has been made in this area starting with Mori and Tanaka [10] who proposed a density evolution approach but did not address the numerical problems in computing the densities with sufficient precision. A major advance was made by Tal and Vardy [16] who exploited the notions of channel degradation and "upgradation" to provide not just approximations but also upper and lower bounds on the channel parameters, such as $I(W_N^{(i)})$ and $Z(W_N^{(i)})$, that are involved in code construction. This line of work has been extended in Pedarsani *et al.* [12] where specific bounds on the approximation error were derived. The presentation below follows largely [12] and Şaşoğlu [5].

For polar code construction, we seek an algorithm that accepts as input a triple (W,N,K) where W is the B-DMC on which the code will be used, N is the code block-length, and K is the dimensionality of the code and produces as output an information set $\mathscr{A} \subset \{1,\ldots,N\}$ of size K such that $\sum_{i\in\mathscr{A}} Z(W_N^{(i)})$ is as small as possible. Finding a good frozen vector $u_{\mathscr{A}^c}$ should also be included as part of the desired output of a code construction algorithm in general. However, if W is a symmetric channel then the code performance is not affected by the choice of $u_{\mathscr{A}^c}$ and this second issue disappears. The following discussion is restricted to symmetric channels and we will exclude finding a good frozen vector from the code construction problem. We use the abbreviation BMS to refer to binary-input memoryless symmetric channels. The output alphabet for a BMS will be assumed finite but the methods here applicable to BMS channels with a continuous output alphabet such as binary-input additive Gaussian noise channels.

In principle, the code construction problem can be solved by computing the transition probabilities of all the channels $\{W_{2^{n-k}}^{(i)} : 0 \le k \le n, 1 \le i \le 2^{n-k}\}$ created through the course of the polarization construction, as depicted in Fig. 3.1. Such a computation would use the recursive relations given in Proposition 3 starting with $W_1^{(1)} = W$. Altogether there are $2N-1$ channels in this collection and it may appear that this calculation should have complexity $O(N)$ where $N = 2^n$ is the code block length. Unfortunately, this computation is complicated by the exponentially growing size of the output spaces of the channels involved. For example, the output of the channel $W_N^{(i)}$ is the vector $y^N u^{i-1}$ which can take on $M^N 2^{i-1}$ possible values if W is a channel with M outputs.

There is an exceptional case where the above recursive calculation is feasible. If W is a BEC, each channel in the collection $\{W_{2^{n-k}}^{(i)}\}$ is a BEC and the erasure probabilities can be calculated using the recursive formulas (2.23) with overall complexity $O(N)$. Although the channels created from a BEC W also appear to have an exponentially growing size for their output spaces, after merging equivalent output letters, only three letters remain: 0,1, and erasure. The BEC example suggests that merging similar output letters may lead to a low-complexity approximate code construction algorithm for general channels. This is indeed the key idea of the methods that will be presented in the rest of this section.

Before we present the specific methods for polar code construction we need to develop some general results about BMS channels.

5.3.1 A general representation of BMS channels

Definition 1 *A channel* $W : \mathscr{X} \to \mathscr{Y}$ *is said to be the sum of channels* $\{W_i : 1 \le i \le M\}$ *with weights* $\{p_i : 1 \le i \le M\}$ *if the following hold:*

- $\{p_i : 1 \le i \le M\}$ *is a probability distribution*

- *The channels entering into the sum have the form*

$$W_i : \mathscr{X} \to \mathscr{Y}_i$$

with the output alphabets $\mathscr{Y}_i$, $1 \le i \le M$, forming a partition of the output alphabet $\mathscr{Y}$ of the original channel:

$$\mathscr{Y} = \bigcup_{i=1}^{M} \mathscr{Y}_i, \qquad \mathscr{Y}_i \cap \mathscr{Y}_j = \emptyset,\ i \neq j.$$

- *The transition probabilities are related by*

$$W(y|x) = p_i W_i(y|x), \qquad \text{whenever } y \in \mathscr{Y}_i,\ 1 \le i \le M.$$

We write $W = \sum_{i=1}^{M} p_i W_i$ to denote that W is a sum of channels in this sense.

Proposition 21 *Any BMS channel $W : \{0,1\} \to \mathscr{Y}$ with a finite output alphabet can be written as the sum of BSCs:*

$$W = \sum_{i=1}^{M} p_i BSC(\varepsilon_i),$$

where the crossover probabilities ε_i are between 0 and 1/2.

Proof. Since W is symmetric, for each output letter y there exists a conjugate letter $\bar{y}$ so that $W(y|0) = W(\bar{y}|1)$ and $W(y|1) = W(\bar{y}|0)$. Thus, each output letter, together with its conjugate $\bar{y}$ defines a BSC with input alphabet $\{0,1\}$ and output alphabet $\{y, \bar{y}\}$. Some of these BSCs may have identical crossover probabilities; in that case, we merge the BSCs with identical crossover probabilities into a single BSC. Output symbols y for which $W(y|0) = W(y|1)$ (which are effectively erasures) may be split into two symbols if necessary to represent them as a BSC with crossover probability 1/2.

Example 1 *A binary erasure channel W with erasure probability ε can be written as $W = (1-\varepsilon)BSC(0) + \varepsilon BSC(1/2)$.*

It will be convenient to generalize the above definitions to the case where the channel output alphabet can be continuous. In this more general case, we may represent any BMS channel W in the form

$$W = \int_0^{1/2} f(\varepsilon) \mathrm{BSC}(\varepsilon)\, d\varepsilon$$

where f is a pdf on $[0, 1/2]$. This representation covers the previous one by taking $f(\varepsilon) = \sum_{i=1}^{M} p_i \delta(\varepsilon - \varepsilon_i)$.

Given the characterization of a BMS channel W as a sum of BSCs, it is easy to see that the symmetric capacity $I(W)$ and the Bhattacharyya parameter $Z(W)$ can be calculated as

$$I(W) = \int_0^{1/2} f(\varepsilon)[1 - \mathscr{H}(\varepsilon)]\, d\varepsilon$$

and

$$Z(W) = \int_0^{1/2} f(\varepsilon)\sqrt{4\varepsilon(1-\varepsilon)}\, d\varepsilon.$$

These parameters may alternatively be denoted as $I(f)$ and $Z(f)$.

5.3.2 Channel approximation

A given BMS channel W may be approximated for a given purpose by suitably approximating its characterizing pdf f. In polar coding, typically, we wish to replace a given f with a simpler f' while keeping the approximation error, as measured by $|I(f) - I(f')|$ or $|Z(f) - Z(f')|$, small. Since both $I(f)$ and $Z(f)$ are continuous functions of f taking values in a closed compact interval (namely, $[0,1]$), this approximation problem can be solved without much difficulty. For our purposes it will be sufficient to use the following simple "quantizer" for approximating BMS channels.

Proposition 22 *Let $L \geq 1$ be a fixed integer. For $i = 0, 1, \ldots, L$, let $\delta_i \in [0, 1/2]$ be (the unique real number) such that a BSC with crossover probability δ_i has symmetric capacity $1 - (i/L)$, i.e., $\mathscr{H}(\delta_i) = i/L$. Let W be a symmetric binary-input memoryless channel characterized by a PDF f. Let $\tilde{W}$ be the channel*

$$\tilde{W} = \sum_{i=0}^{L} \tilde{p}_i BSC(\delta_i)$$

where

$$\tilde{p}_i = \int_{\delta_{i-1}}^{\delta_i} f(\delta) d\delta, \qquad i = 1, \ldots, L.$$

(The integrals are over $[\delta_{i-1}, \delta_i)$ except for the last one which is over $[\delta_{L-1}, \delta_L]$.) Then, $I(\tilde{W}) \leq I(W) \leq I(\tilde{W}) + 1/L$.

Proof. Since $\mathscr{H}(\delta)$ is an increasing function of δ in the interval $[0, 1/2]$, we have $0 = \delta_0 < \delta_1 < \cdots < \delta_L = 1/2$. Thus, these points partition $[0, 1/2]$ into disjoint quantization intervals. The first half of the desired inequality is obtained as

$$\begin{aligned}
I(W) &= \int_0^{1/2} f(\delta)[1 - \mathscr{H}(\delta)]d\delta \\
&= \sum_{i=1}^{L} \int_{\delta_{i-1}}^{\delta_i} f(\delta)[1 - \mathscr{H}(\delta)]d\delta \\
&\geq \sum_{i=1}^{L} \int_{\delta_{i-1}}^{\delta_i} f(\delta)[1 - \mathscr{H}(\delta_i)]d\delta
\end{aligned}$$

$$= I(\tilde{W})$$

where the inequality uses the monotone increasing property of $\mathcal{H}(\delta)$ for $\delta \in [0,1/2]$. To obtain the second half, we use the monotone property again but in the reverse direction.

$$\begin{aligned} I(W) &\leq \sum_{i=1}^{L} \int_{\delta_{i-1}}^{\delta_i} f(\delta)[1-\mathcal{H}(\delta_i{}_{-1}]d\delta \\ &= \sum_{i=1}^{L} p_i[1-(i-1)/L] \\ &= I(\tilde{W}) + 1/L. \end{aligned}$$

We will show that the above type of quantization creates a degraded channel in the following sense.

Definition 2 *Let $W : \mathcal{X} \to \mathcal{Y}$ and $W' : \mathcal{X} \to \mathcal{Y}'$ be two channels. We say that W' is degraded wrt W if there exists a third channel $P : \mathcal{Y} \to \mathcal{Y}'$ such that*

$$W'(y'|x) = \sum_y P(y'|y)W(y|x).$$

We write $W' \preceq W$ to indicate that W' is degraded wrt W.

Proposition 23 *Let W be a BMS channel and $\tilde{W}$ be its quantized version as above. Then, $\tilde{W} \preceq W$.*

Proof. We may represent the quantizer as a channel (a deterministic one).

Proposition 24 *Let W and W' be two B-DMCs with $W \preceq W'$. Then, $I(W) \leq I(W')$ and $Z(W) \geq Z(W')$. Furthermore, channel degradedness relationship propagates through the polarization construction in the sense that*

$$W_N^{(i)} \preceq (W')_N^{(i)}, \qquad \text{for all } N = 2^n,\ 1 \leq i \leq N.$$

Corollary 2 *Let $W_2^{(1)}$ and $W_2^{(2)}$ be the channels obtained from W by one-step polarization. Similarly let $\tilde{W}_2^{(1)}$ and $\tilde{W}_2^{(2)}$ be obtained from the quantized channel $\tilde{W}$. Then,*

$$I(\tilde{W}_2^{(1)}) \leq I(W_2^{(1)}) \quad \text{and} \quad I(\tilde{W}_2^{(2)}) \leq I(W_2^{(2)}).$$

5.3.3 A code construction algorithm

We have completed intoducing the basic notions that underly the code construction algorithm that follows. Let W be a given BMS and let $\tilde{W}$ be a downward quantization of W with resolution L as defined above. From the identities

$$I(W_2^{(1)}) + I(W_2^{(2)}) = 2I(W)$$

and

$$I(\tilde{W}_2^{(1)}) + I(\tilde{W}_2^{(2)}) = 2I(\tilde{W})$$

we obtain

$$[I(W_2^{(1)}) - I(\tilde{W}_2^{(1)})] + [I(W_2^{(2)}) - I(\tilde{W}_2^{(2)})] = 2[I(W) - I(\tilde{W})]$$

This shows that the average approximation error after one-step polarization is the same as the error before the polarization step. Since the two difference terms on the left are non-negative (channel degradedness) and the difference term on the right is bounded by $1/L$, we have

$$|I(W_2^{(1)}) - I(\tilde{W}_2^{(1)})| + |I(W_2^{(2)}) - I(\tilde{W}_2^{(2)}) \leq 2/L.$$

Thus, the average *absolute* error is also bounded by $2/L$. The fact that we have a bound on the absolute error is essential for the final result.,

While the quantized channel $\tilde{W}$ has at most $2(L+1)$ output letters, the channels $\tilde{W}_2^{(1)}$ and $\tilde{W}_2^{(2)}$ have many more output letters. The idea of low-complexity polar code construction is to quantize the channels $\tilde{W}_2^{(i)}$ again before continuing with the next step of polarization. The method can be described more precisely by referring to Fig. 3.1 again. The quantization procedure replaces the root node by $\tilde{W}$ before applying the first polarization step. The two channels created at level 1 are now $\tilde{W}_2^{(1)}$ and $\tilde{W}_2^{(2)}$. Before continuing further, these channels are quantized to resolution L and polarization is applied to obtain the four channels at level 2. We shall abuse the notation to denote by $\{\tilde{W}_{2^{n-k}}^{(i)} : 0 \leq k \leq n, 1 \leq i \leq 2^{n-k}\}$ the channels obtained in the course of this quantize-polarize procedure. Each branching point in Fig. 3.1 causes an incremental quantization error. The average quantization error at each node is bounded by $1/L$. An inductive argument shows that the overall average absolute quantization error at level k of this procedure is bounded as

$$\frac{1}{2^{n-k}} \sum_{i=1}^{2^{n-k}} |I(W_{2^{n-k}}^{(i)} - I(\tilde{W}_{2^{n-k}}^{(i)})| \leq k/L, \qquad k = 1, \ldots, n. \tag{5.13}$$

In particular, the average absolute quantization error at the last level is bounded by n/L. We conclude by Markov's inequality that at least a fraction $1 - \sqrt{n/L}$ of the quantities $\{I(W_N^{(i)}) : 1 \leq i \leq N\}$ are computed with an error not exceeding $\sqrt{n/L}$. (It is here that having a bound on average absolute error is crucial.) By taking $L = n^2$, one can ensure that, with the exception of at most a fraction $1/\sqrt{n}$, the terms $\{I(W_N^{(i)})\}$ are computed with an error not exceeding $1/\sqrt{n}$. This means that with a negligible loss in rate we can identify the good coordinates. The overall complexity of this calculation is roughly $O(L^2 N)$ or $O(Nn^2)$ if L is chosen as n^2.

References

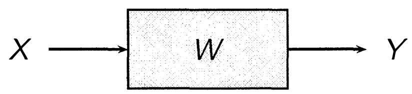
1. Arıkan, E.: Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels. IEEE Trans. Inform. Theory **IT-55**(7), 3051–3073 (2009)
2. Arıkan, E., Telatar, E.: On the rate of channel polarization. In: Proc. 2009 IEEE Int. Symp. Inform. Theory, pp. 1493–1495. Seoul, S. Korea (2009)
3. Chung, K.L.: A Course in Probability Theory, 2nd ed. Academic: New York (1974)
4. Cooley, J.W., Tukey, J.W.: An algorithm for the machine calculation of complex Fourier series. *Math. Comput.* **19**(90), 297–301 (1965)
5. Şaşoğlu, E.: Polarization and polar coding (Spring 2012). Unpublished notes (to appear in the "Foundation Series."
6. Gallager, R.G.: Information Theory and Reliable Communication. Wiley: New York (1968)
7. Hassani, S.H., Mori, R., Tanaka, T., Urbanke, R.: Rate-dependent analysis of the asymptotic behavior of channel polarization (Oct. 2011). ArXiv.org:1110.0194
8. Hassani, S.H., Urbanke, R.: On the scaling of polar codes: I. the behavior of polarized channels. In: Proc. 2010 IEEE Int. Symp. Inform. Theory, pp. 874–878. Austin, TX (2010)
9. Lin, S., Costello, Jr., D.J.: Error Control Coding, (2nd ed). Pearson: N.J. (2004)
10. Mori, R., Tanaka, T.: Performance of polar codes with the construction using density evolution. IEEE Communications Letters **13**(7), 519–521 (2009)
11. Muller, D.E.: Application of boolean algebra to switching circuit design and to error correction. IRE Trans. Electronic Computers **EC-3**, 6–12 (1954)
12. Pedarsani, R., Hassani, S.H., Tal, I., Telatar, E.: On the construction of polar codes. In: Proc. 2011 IEEE Int. Symp. Inform. Theory, pp. 11–15. St. Petersburg, Russia (2011)
13. Plotkin, M.: Binary codes with specified minimum distance. IRE Trans. Inform. Theory **6**(4), 445–450 (1960)
14. Reed, I.: A class of multiple-error-correcting codes and the decoding scheme. IRE Trans. Inform. Theory **4**(4), 39–44 (1954)
15. Shannon, C.E.: A mathematical theory of communication. Bell System Tech. J. **27**(2), 379–423, 623–656 (1948)
16. Tal, I., Vardy, A.: How to construct polar codes (May 2011). ArXiv.org:1105.6164
17. Tanaka, T.: On speed of channel polarization. In: Proc. 2010 IEEE Information Theory Workshop, pp. 1–5. Dublin, Ireland (2010)
18. Tanaka, T., Mori, R.: Refined rate of channel polarization. In: Proc. 2010 IEEE Int. Symp. Inform. Theory, pp. 889–893. Austin, TX (2010)

Polar Coding Tutorial

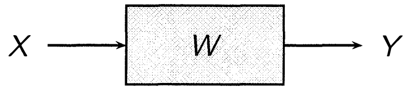
Erdal Arıkan and Emre Telatar

1 July 2011
2012 IEEE International Symposium on Information Theory
ISIT'2012
Cambridge, MA

Binary-Input Memoryless Channels

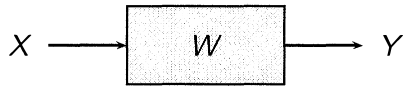


Binary-Input Memoryless Channels



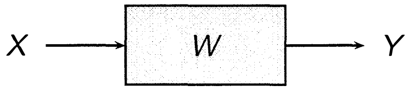
- input alphabet: $\mathcal{X} = \{0, 1\}$

Binary-Input Memoryless Channels



- input alphabet: $\mathcal{X} = \{0, 1\}$
- output alphabet: $\mathcal{Y}$

Binary-Input Memoryless Channels

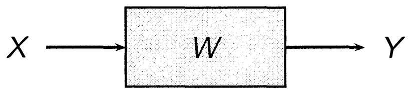


- input alphabet: $\mathcal{X} = \{0, 1\}$
- output alphabet: $\mathcal{Y}$
- transition probabilities:

$$W(y|x), \quad x \in \mathcal{X}, y \in \mathcal{Y}$$

Setup Polarization Encoding Decoding Construction

Binary-Input Memoryless Channels



- input alphabet: $\mathcal{X} = \{0, 1\}$
- output alphabet: $\mathcal{Y}$
- transition probabilities:

$$W(y|x), \quad x \in \mathcal{X}, y \in \mathcal{Y}$$

- memoryless:

$$W^N(y^N|x^N) = \prod_{i=1}^{N} W(y_i|x_i)$$

Assumption on channel inputs

Throughout we assume that channel input random variable X is unifom on $\{0, 1\}$.

Capacity with uniform inputs

The capacity of a binary-input channel W subject to using the inputs equiprobably is given by

$$I(W) \triangleq I(X; Y)$$

where the channel input random variable X is uniform on $\{0, 1\}$.

Capacity with uniform inputs

The capacity of a binary-input channel W subject to using the inputs equiprobably is given by

$$I(W) \triangleq I(X;Y)$$

where the channel input random variable X is uniform on $\{0,1\}$.

More explicitly,

$$I(W) = \sum_{x,y} \frac{1}{2} W(y|x) \log \frac{W(y|x)}{\frac{1}{2}W(y|0) + \frac{1}{2}W(y|1))}.$$

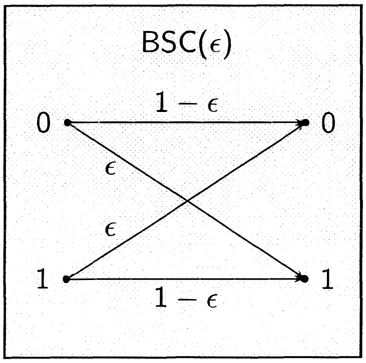
Symmetric channels

If the channel has sufficient symmetries, $I(W)$ equals the true channel capacity.

Symmetric channels

If the channel has sufficient symmetries, $I(W)$ equals the true channel capacity.

Examples:

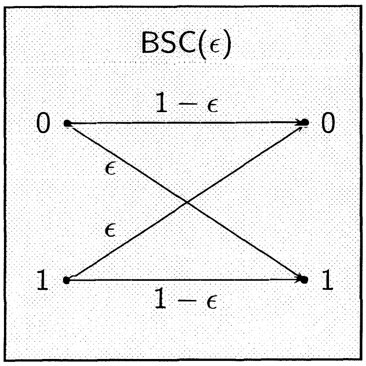


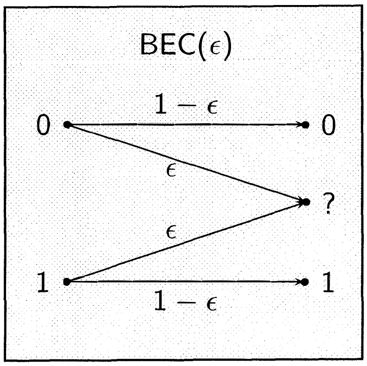
Setup Polarization Encoding Decoding Construction

Symmetric channels

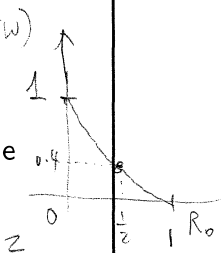
If the channel has sufficient symmetries, $I(W)$ equals the true channel capacity.

Examples:





Cutoff rate, Bhattacharyya parameter



For binary-input channels with uniform distribution on inputs, the cutoff rate equals

$$R_0(W) = \log \frac{2}{1+Z(W)}$$

$$2^{R_0} = \frac{2}{1+Z(W)}$$

$$\Rightarrow 1+Z(W) = 2^{1-R_0}$$

$$\Rightarrow Z(W) = 2^{(1-R_0)} - 1$$

where $Z(W)$ is the Bhattacharyya parameter

$$Z(W) = \sum_y \sqrt{W(y|0)W(y|1)}.$$

$$P_e = \Pr\{\text{error}\} = \Pr\{y \in D_{\text{error}}\} = \sum_{0 \text{ tx'd } y \in D_{\text{error}}} W(y|0) < \sum_y W(y|0)\left(\frac{W(y|1)}{W(y|0)}\right)^{\frac{1}{2}}$$

$$= \sum_y W(y|0)^{\frac{1}{2}} W(y|1)^{\frac{1}{2}}$$

$$= Z(W)$$

What is the significance of $R_0(W)$?

$$P_e \leq Z(W)$$

$$2^{\log_2(Z(W))}$$

- The sequential decoding (Wozencraft, 1957) method achieves $R_0(W)$ with low complexity.
- Came to be regarded as "practical capacity" for a while.
- But $R_0(W)$ is a "flaky" parameter: it can be created or destroyed.
- In a sense polarization is a method of boosting R_0 to channel capacity.

What is the significance of $R_0(W)$?

- The sequential decoding (Wozencraft, 1957) method achieves $R_0(W)$ with low complexity.
- Came to be regarded as "practical capacity" for a while.

Setup Polarization Encoding Decoding Construction

What is the significance of $R_0(W)$?

- The sequential decoding (Wozencraft, 1957) method achieves $R_0(W)$ with low complexity.
- Came to be regarded as "practical capacity" for a while.
- But $R_0(W)$ is a 'flaky" parameter: it can be created or destroyed.

What is the significance of $R_0(W)$?

- The sequential decoding (Wozencraft, 1957) method achieves $R_0(W)$ with low complexity.
- Came to be regarded as "practical capacity" for a while.
- But $R_0(W)$ is a 'flaky" parameter: it can be created or destroyed.
- In a sense polarization is a method of boosting R_0 to channel capacity.

The significance of $Z(W)$

Recall the definition:

$$Z(W) = \sum_{y} \sqrt{W(y|0)W(y|1)}.$$

- $Z(W)$ is an upper-bound on uncoded bit error rate
- $Z(W) = 0$ iff the channel is perfect
- $Z(W) = 1$ iff the channel is useless
- Easier to track than $R_0(W)$
- Gives a bound on the error probabilities in successive cancelation decoding

The significance of $Z(W)$

Recall the definition:

$$Z(W) = \sum_y \sqrt{W(y|0)W(y|1)}.$$

- $Z(W)$ is an upper-bound on uncoded bit error rate
- $Z(W) = 0$ iff the channel is perfect

Setup Polarization Encoding Decoding Construction

The significance of $Z(W)$

Recall the definition:

$$Z(W) = \sum_y \sqrt{W(y|0)W(y|1)}.$$

- $Z(W)$ is an upper-bound on uncoded bit error rate
- $Z(W) = 0$ iff the channel is perfect
- $Z(W) = 1$ iff the channel is useless

The significance of $Z(W)$

Recall the definition:

$$Z(W) = \sum_y \sqrt{W(y|0)W(y|1)}.$$

- $Z(W)$ is an upper-bound on uncoded bit error rate
- $Z(W) = 0$ iff the channel is perfect
- $Z(W) = 1$ iff the channel is useless
- Easier to track than $R_0(W)$

The significance of $Z(W)$

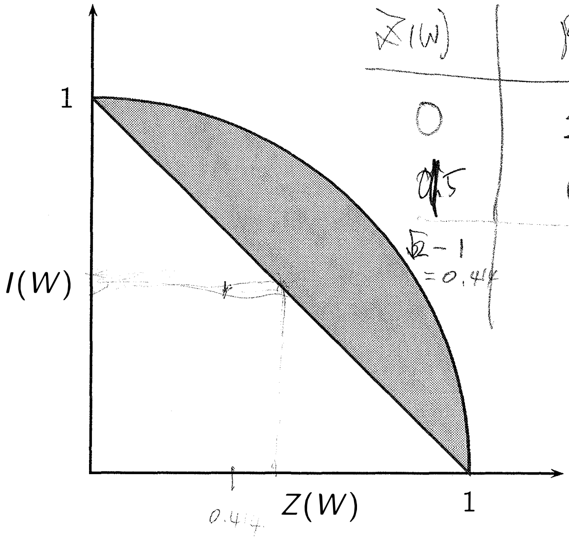
Recall the definition:

$$Z(W) = \sum_y \sqrt{W(y|0)W(y|1)}.$$

- $Z(W)$ is an upper-bound on uncoded bit error rate
- $Z(W) = 0$ iff the channel is perfect
- $Z(W) = 1$ iff the channel is useless
- Easier to track than $R_0(W)$
- Gives a bound on the error probabilities in successive cancelation decoding

Don't know!

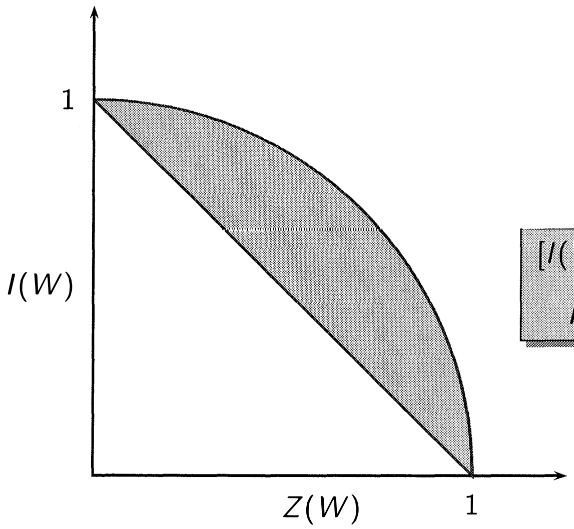
A fact about $I(W)$ vs $Z(W)$ $\cong 2^{(1-R_0)} - 1$. $R_0 = \log \frac{2}{1+Z(W)}$



$Z(W)$	R_0	
0	1	
$\cancel{0}$.5	0	
$\sqrt{2}-1$ $=0.44$	$\frac{1}{2}$	

Setup Polarization Encoding Decoding Construction

A fact about $I(W)$ vs $Z(W)$



$$[I(W)]^2 + [Z(W)]^2 \leq 1$$

$$I(W) + Z(W) \geq 1$$

Extreme value relations

Lemma

For any binary-input channel W,

$$I(W) = 1 \quad \text{iff} \quad Z(W) = 0 \quad \text{iff} \quad R_0(W) = 1$$

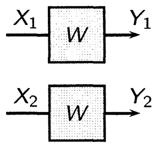
and

$$I(W) = 0 \quad \text{iff} \quad Z(W) = 1 \quad \text{iff} \quad R_0(W) = 0.$$

Proof: Given in the next presentation.

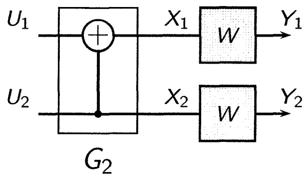
Basic module for a low-complexity scheme

Combine two copies of W



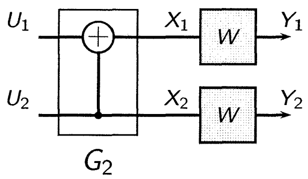
Basic module for a low-complexity scheme

Combine two copies of W



Basic module for a low-complexity scheme

Combine two copies of W



and split to create two bit-channels

$$W^- : U_1 \to (Y_1, Y_2)$$
$$W^+ : U_2 \to (Y_1, Y_2, U_1)$$

The worse channel W^-

$W^-(Y_1, Y_2 | U_1)$

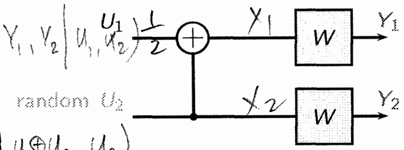
U_1	(Y_1, Y_2)	U_2	$(\cancel{X_1}, X_2)$
0	$(1-\varepsilon)$	0	(0, 0)
		1	(1, 0)
1		0	(1, 0)
		1	(1, 1)

$$W^- : U_1 \to (Y_1, Y_2)$$

$$W^-(Y_1, Y_2 | U_1)$$
$$= \sum_{U_2} W^-(Y_1, Y_2, U_2 | U_1)$$
$$= \sum_{U_2} W^-(Y_1, Y_2 | U_1, U_2) \frac{1}{2}$$
$$= \sum_{U_2} \frac{1}{2} W(Y_1, Y_2 | U_1 \oplus U_2, U_2)$$

→ Transition probabilities

$$= \sum_{U_2} \frac{1}{2} W(Y_1 | U_1 \oplus U_2) W(Y_2 | U_2)$$

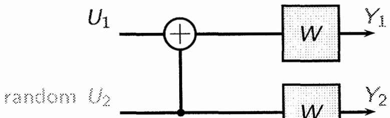


U_1	U_2	(Y_1, Y_2) 0 0	0 1	1 0	1 1
0	0	$\frac{1}{2}$			
	1				
1	0				
	1				

The worse channel W^-

BEC

$$W^- : U_1 \to (Y_1, Y_2)$$



$$I(W^-) = I(U_1; Y_1, Y_2)$$
$$= H(U_1) - H(Y_1, Y_2 | U_1)$$

다른 방향으로 chain Rule 적용

$I(U_2;U_1) + I(U_2;Y_1,Y_2|U_1)$

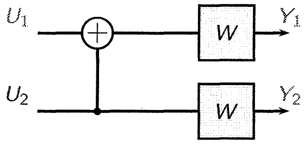
$I(U_2;U_1) = 0$

The better channel W^+

$I(U_2;Y_1,Y_2,U_1) = I(U_2;Y_2) + I(U_2;Y_1,U_1|Y_2)$

Given the output (U_1,Y_1,Y_2) determine the input U_2

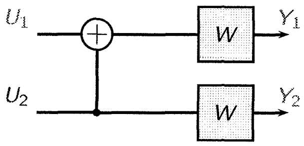
$$W^+ : U_2 \to (Y_1, Y_2, U_1)$$



Setup · Polarization · Encoding · Decoding · Construction

The better channel W^+

$$W^+ : U_2 \to (Y_1, Y_2, U_1)$$



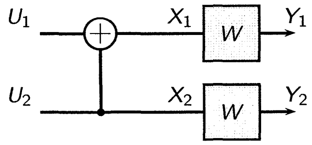
$$I(W^+) = I(U_2; Y_1, Y_2, U_1)$$

$= I(U_2;Y_1,Y_2|U_1)$

$= \frac{1}{2} I(U_2;Y_1,Y_2|U_1=0) + \frac{1}{2} I(U_2;Y_1,Y_2|U_1=1)$

I need ch transition for $p(y_1,y_2|u_1,u_2)$

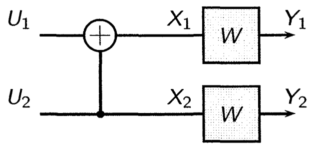
Capacity conserved but redistributed unevenly



- Conservation:

$$I(W^-) + I(W^+) = 2I(W)$$

Capacity conserved but redistributed unevenly



- Conservation:

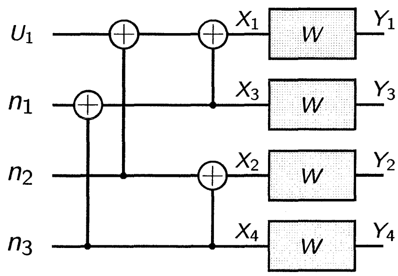
$$I(W^-) + I(W^+) = 2I(W)$$

- Extremization:

$$I(W^-) \leq I(W) \leq I(W^+)$$

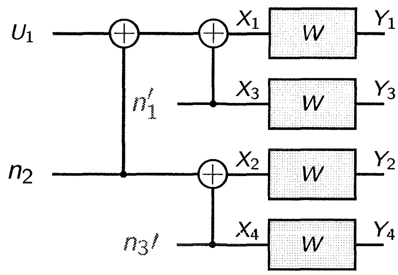
with equality iff $I(W)$ equals 0 or 1.

... decoding U_1



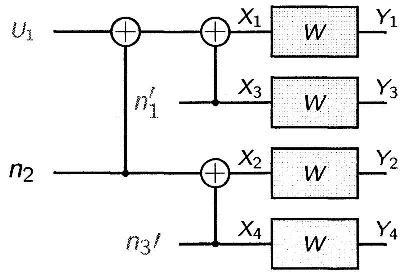
n_1, n_2, n_3 are i.i.d. Bernoulli 1/2 "noise."

... decoding U_1 with equivalent noise model



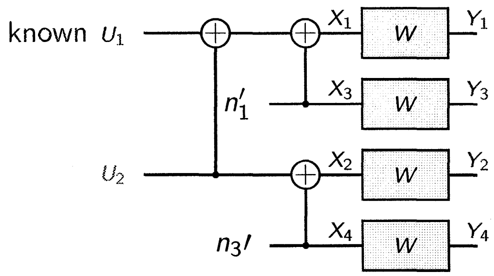
n'_1, n_2, n'_3 are i.i.d. Bernoulli 1/2 "noise."

... identify the channel W^{--}



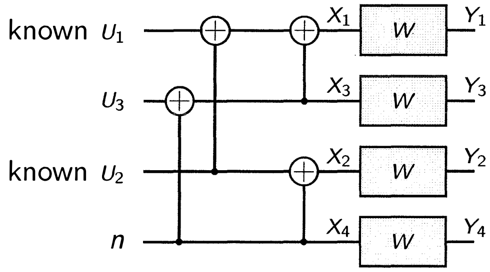
$$W^{--} : U_1 \to Y_1 Y_2 Y_3 Y_4$$

... decoding U_2 given U_1



$$W^{-+} : U_2 \to Y_1 Y_2 Y_3 Y_4 U_1$$

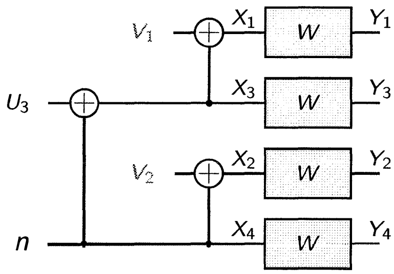
... decoding U_3 given U_1 and U_2



n is Bernoulli 1/2 noise, U_1 and U_2 are known.

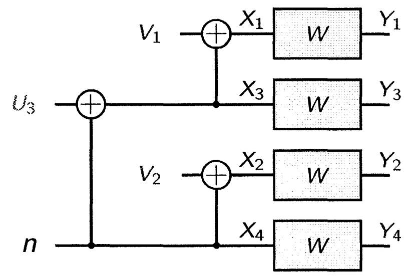
Setup Polarization Encoding Decoding Construction

... decoding U_3 given V_1 and V_2



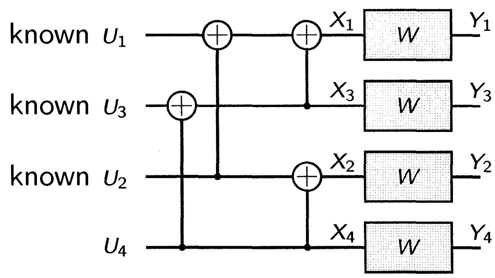
n is Bernoulli 1/2 noise, V_1 and V_2 are known.

... identify the channel W^{+-}



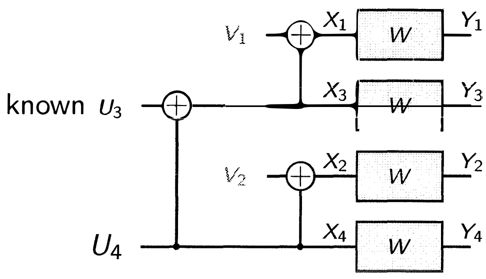
$$W^{+-} : U_3 \to Y_1 Y_2 Y_3 Y_4 V_1 V_2$$

... decoding U_4 given U_1, U_2, and U_3



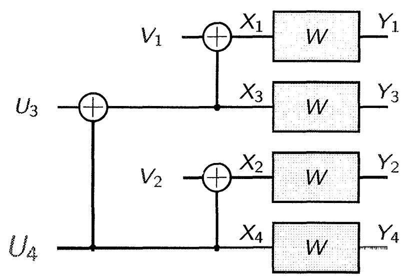
U_1, U_2, and U_3 are known.

... decoding U_4 given V_1, V_2, and U_3



U_3 is known, V_1 and V_2 are known.

... identify the channel W^{++}



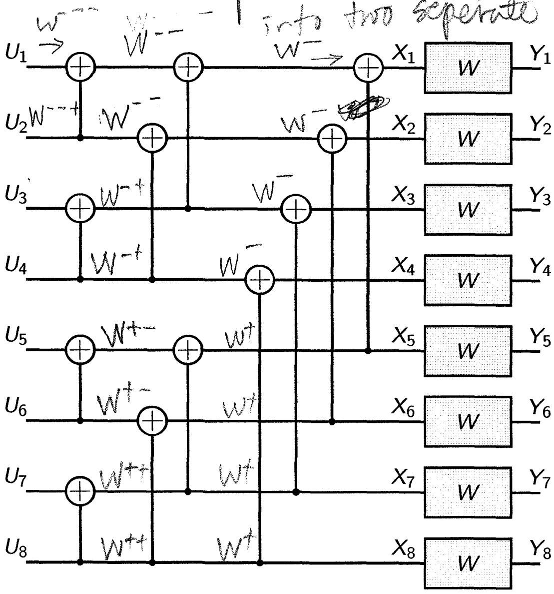
W^{++} is the channel $U_4 to Y_1 Y_2 Y_3 Y_4 V_1 V_2 U_3$

$W^{①,②,③}$

③ ② ①

Size 8 construction

Split each channel W^{--}, W^{-+}, W^{+-}, W^{++} into two seperate one.



Demonstration of polarization

Polarization is easy to analyze when W is a BEC.

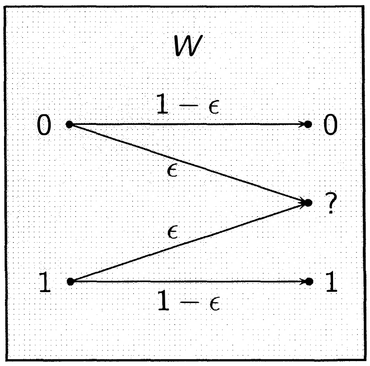
If W is a BEC(ϵ), then so are W^- and W^+, with erasure probabilities

$$\epsilon^- \triangleq 2\epsilon - \epsilon^2$$

and

$$\epsilon^+ \triangleq \epsilon^2$$

respectively.



Demonstration of polarization

Polarization is easy to analyze when W is a BEC.

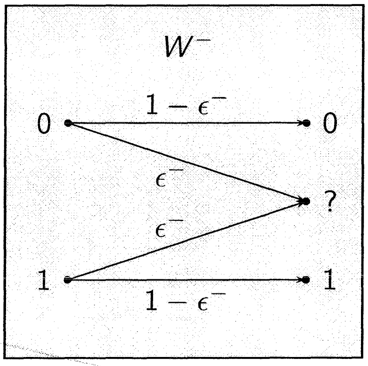
If W is a BEC(ϵ), then so are W^- and W^+, with erasure probabilities

$$\epsilon^- \triangleq 2\epsilon - \epsilon^2$$

and

$$\epsilon^+ \triangleq \epsilon^2$$

respectively.



W^{--} $\quad \varepsilon^{--} \triangleq 2 \cdot \varepsilon^- - \varepsilon^{-2}$

W^{-+} $\quad \varepsilon^{-+} \triangleq \varepsilon^{-2}$

W^{+-} $\quad \varepsilon^{+-} \triangleq 2\varepsilon^+ - \varepsilon^{+2}$

W^{++} $\quad \varepsilon^{++} \triangleq \varepsilon^{+2}$

Setup Polarization Encoding Decoding Construction

Demonstration of polarization

Polarization is easy to analyze when W is a BEC.

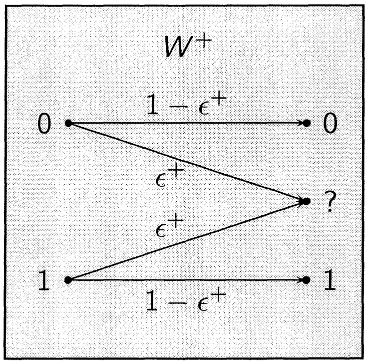
If W is a BEC(ϵ), then so are W^- and W^+, with erasure probabilities

$$\epsilon^- \triangleq 2\epsilon - \epsilon^2$$

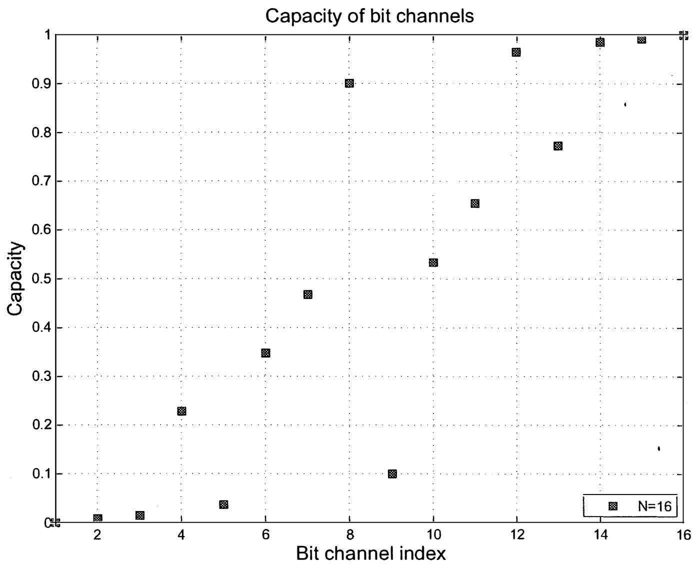
and

$$\epsilon^+ \triangleq \epsilon^2$$

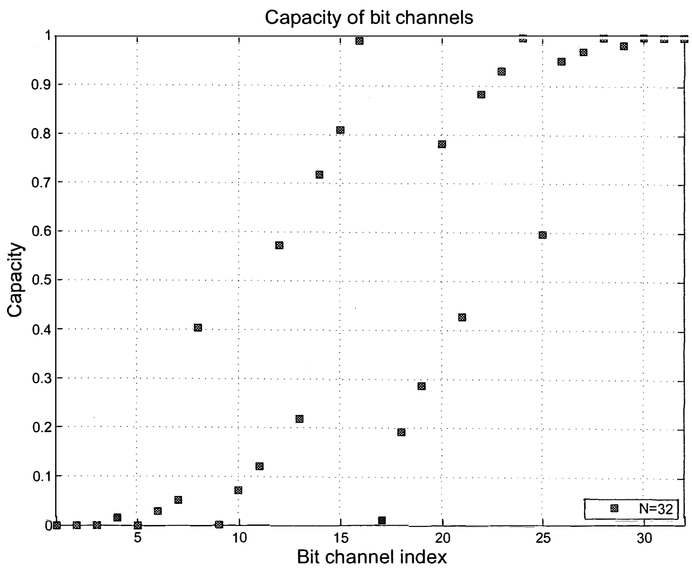
respectively.



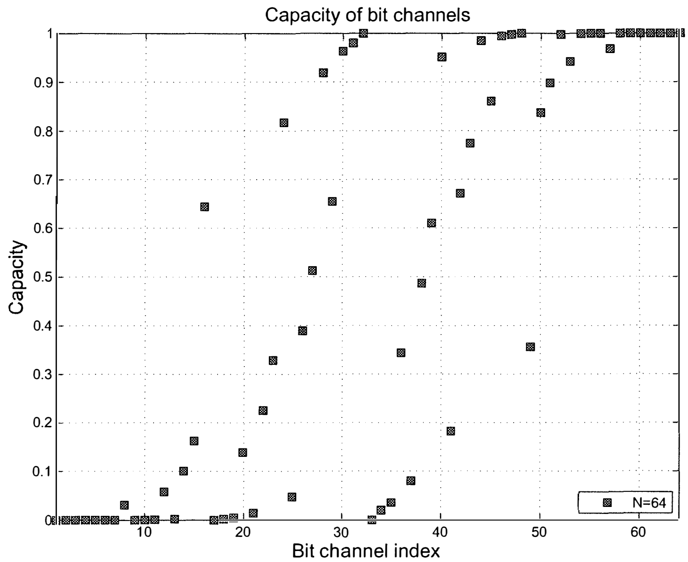
Polarization for BEC($\frac{1}{2}$): $N = 16$



Polarization for BEC($\frac{1}{2}$): $N = 32$

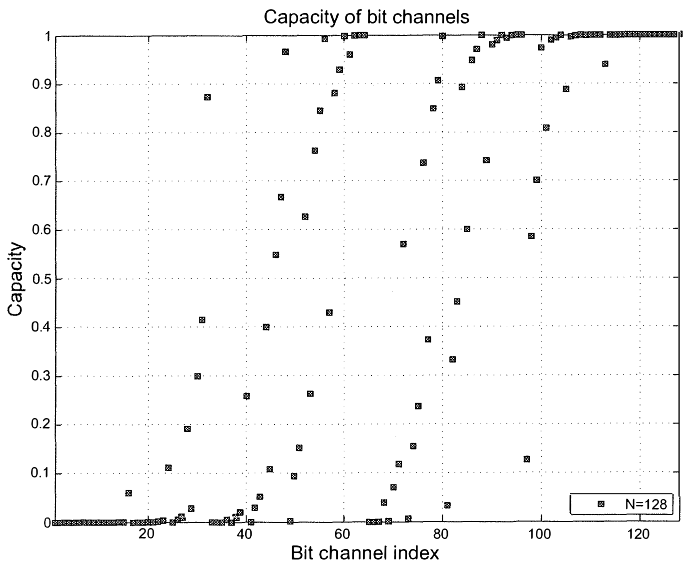


Polarization for BEC($\frac{1}{2}$): $N = 64$

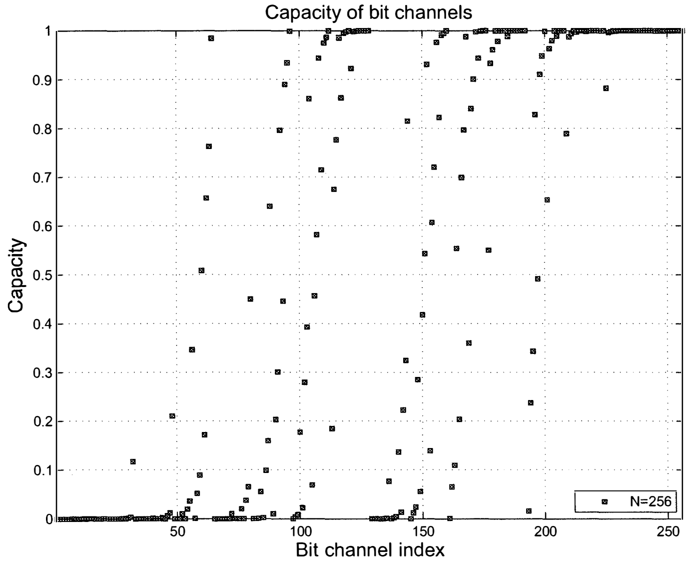


Setup Polarization Encoding Decoding Construction

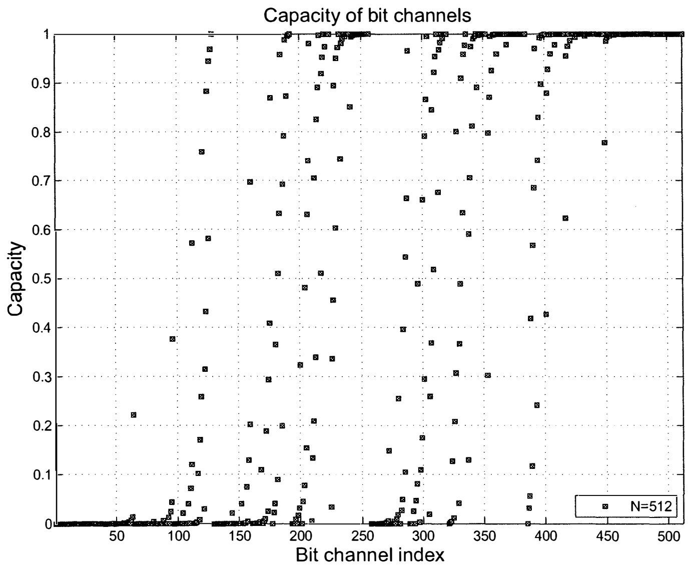
Polarization for BEC($\frac{1}{2}$): $N = 128$



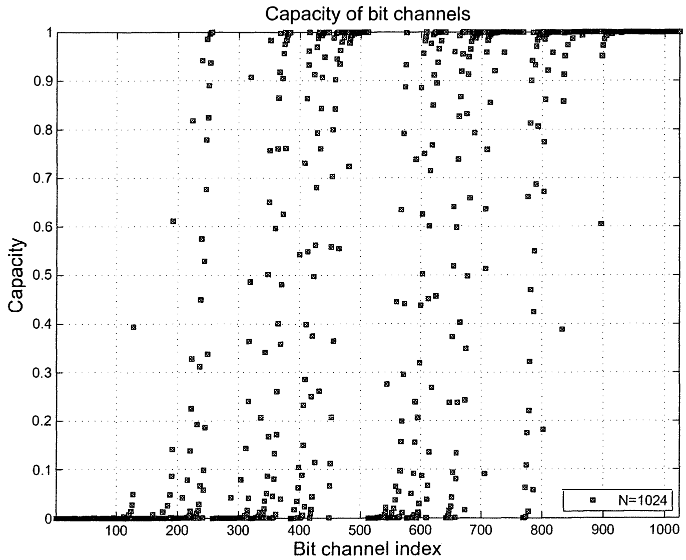
Polarization for BEC($\frac{1}{2}$): $N = 256$



Polarization for BEC($\frac{1}{2}$): $N = 512$



Polarization for BEC($\frac{1}{2}$): $N = 1024$



Polarization and rate of polarization

Polarization

The bit-channel capacities $\{I(W_N^{(i)})\}$ polarize: for any $\delta \in (0,1)$, as the construction size N grows

$$\left[\frac{\text{no. channels with } I(W_N^{(i)}) > 1-\delta}{N}\right] \longrightarrow I(W)$$

and

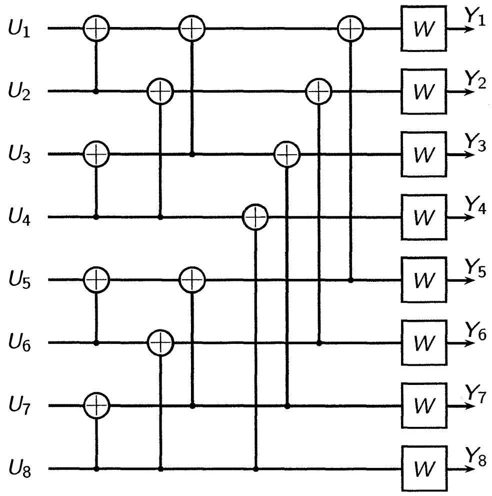
$$\left[\frac{\text{no. channels with } I(W_N^{(i)}) < \delta}{N}\right] \longrightarrow 1 - I(W)$$

This result holds with $\delta = \mathcal{O}(2^{-N^{\beta}})$ for any fixed $\beta < 1/2$.

Proof: To be given in the next lecture.

Polar code example: $W = \mathrm{BEC}(\frac{1}{2})$, $N = 8$, rate $1/2$

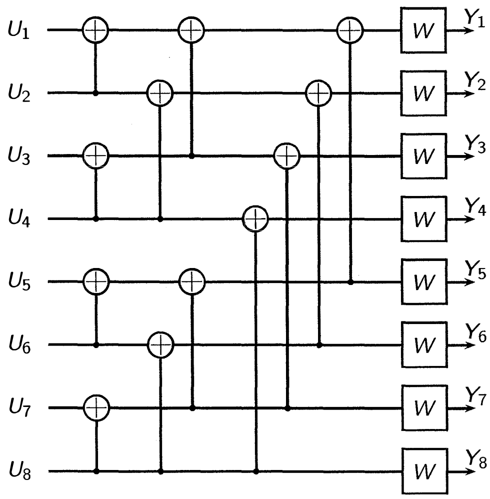
$I(W_i)$
0.0039
0.1211
0.1914
0.6836
0.3164
0.8086
0.8789
0.9961



Setup Polarization Encoding Decoding Construction

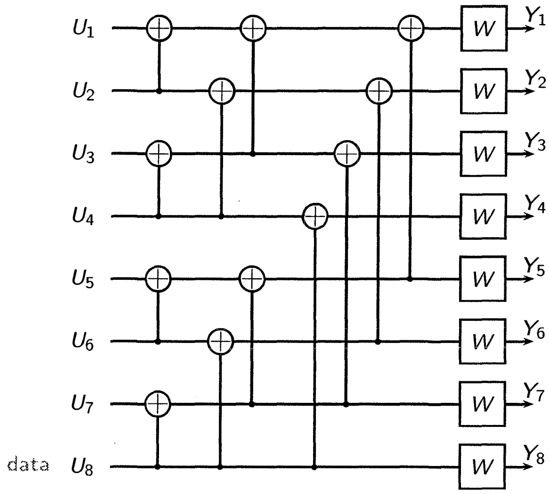
Polar code example: $W = \mathrm{BEC}(\frac{1}{2})$, $N = 8$, rate $1/2$

$I(W_i)$	Rank
0.0039	8
0.1211	7
0.1914	6
0.6836	4
0.3164	5
0.8086	3
0.8789	2
0.9961	1



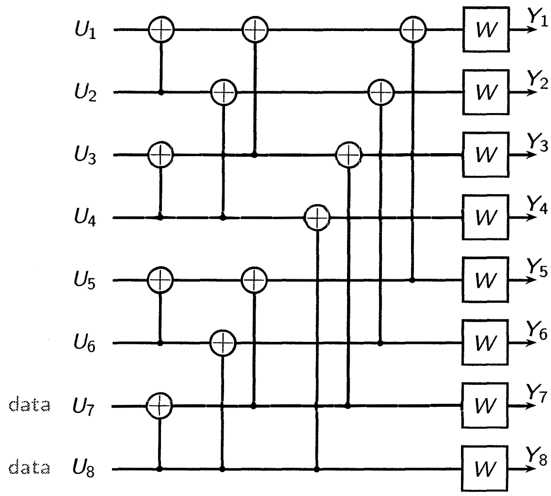
Polar code example: $W = \mathrm{BEC}(\frac{1}{2})$, $N = 8$, rate $1/2$

$I(W_i)$	Rank
0.0039	8
0.1211	7
0.1914	6
0.6836	4
0.3164	5
0.8086	3
0.8789	2
0.9961	1



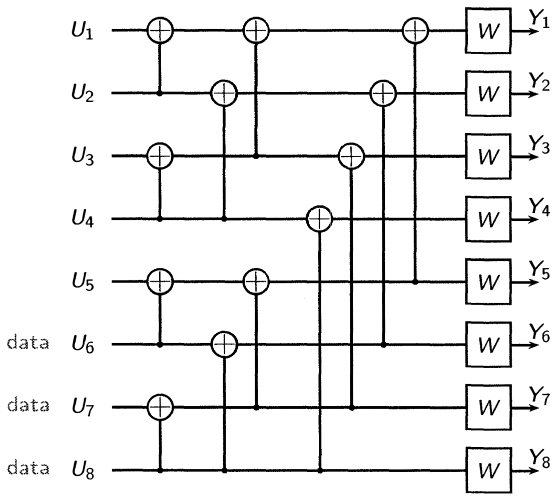
Polar code example: $W = \mathrm{BEC}(\frac{1}{2})$, $N = 8$, rate $1/2$

$I(W_i)$	Rank
0.0039	8
0.1211	7
0.1914	6
0.6836	4
0.3164	5
0.8086	3
0.8789	2
0.9961	1



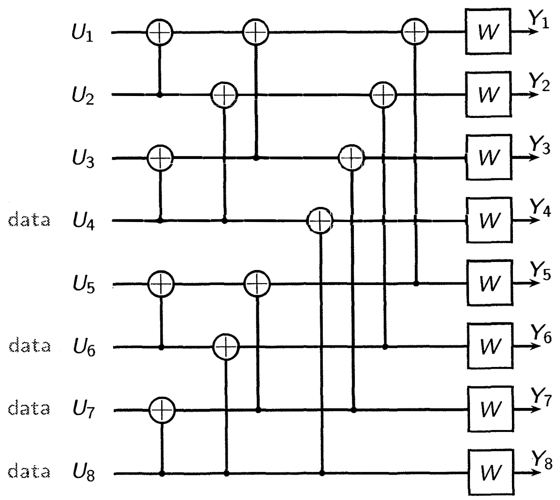
Polar code example: $W = \text{BEC}(\frac{1}{2})$, $N = 8$, rate $1/2$

$I(W_i)$	Rank
0.0039	8
0.1211	7
0.1914	6
0.6836	4
0.3164	5
0.8086	3
0.8789	2
0.9961	1



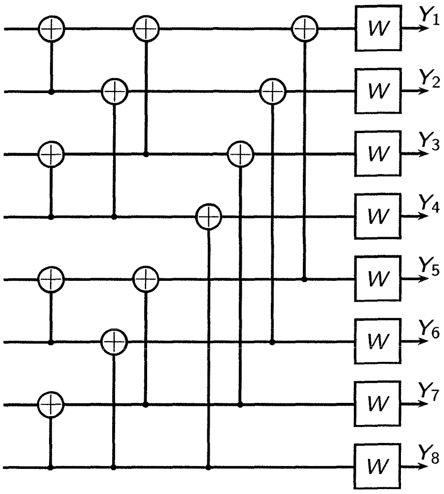
Polar code example: $W = \text{BEC}(\frac{1}{2})$, $N = 8$, rate $1/2$

$I(W_i)$	Rank
0.0039	8
0.1211	7
0.1914	6
0.6836	4
0.3164	5
0.8086	3
0.8789	2
0.9961	1



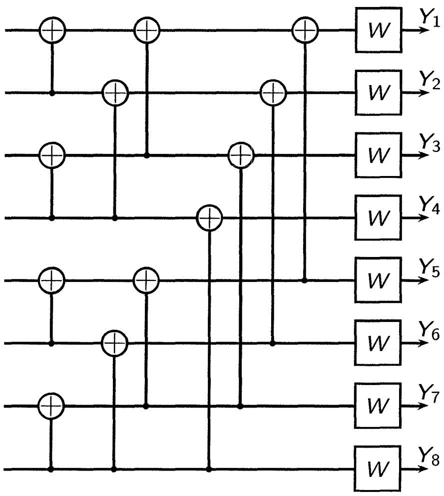
Polar code example: $W = \mathrm{BEC}(\frac{1}{2})$, $N = 8$, rate $1/2$

$I(W_i)$	Rank		
0.0039	8	frozen	U_1
0.1211	7	frozen	U_2
0.1914	6	frozen	U_3
0.6836	4	data	U_4
0.3164	5	frozen	U_5
0.8086	3	data	U_6
0.8789	2	data	U_7
0.9961	1	data	U_8



Polar code example: $W = \mathrm{BEC}(\frac{1}{2})$, $N = 8$, rate $1/2$

$I(W_i)$	Rank		
0.0039	8	frozen	0
0.1211	7	frozen	0
0.1914	6	frozen	0
0.6836	4	data	U_4
0.3164	5	frozen	0
0.8086	3	data	U_6
0.8789	2	data	U_7
0.9961	1	data	U_8



Encoding complexity

Theorem

Encoding complexity for polar coding is $\mathcal{O}(N \log N)$.

Proof:

- Polar coding transform can be represented as a graph with $N[1 + \log(N)]$ variables.

Setup Polarization **Encoding** Decoding Construction

Encoding complexity

Theorem

Encoding complexity for polar coding is $\mathcal{O}(N \log N)$.

Proof:

- Polar coding transform can be represented as a graph with $N[1 + \log(N)]$ variables.
- The graph has $(1 + \log(N))$ levels with N variables at each level.

Encoding complexity

Theorem

Encoding complexity for polar coding is $\mathcal{O}(N \log N)$.

Proof:

- Polar coding transform can be represented as a graph with $N[1 + \log(N)]$ variables.
- The graph has $(1 + \log(N))$ levels with N variables at each level.
- Computation begins at the source level and can be carried out level by level.

Encoding complexity

Theorem

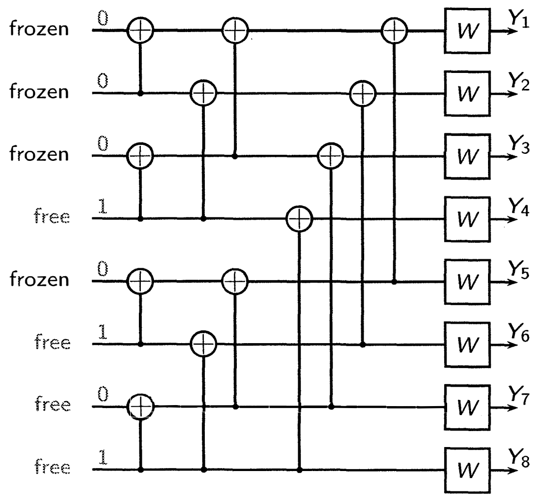
Encoding complexity for polar coding is $\mathcal{O}(N \log N)$.

Proof:

- Polar coding transform can be represented as a graph with $N[1 + \log(N)]$ variables.
- The graph has $(1 + \log(N))$ levels with N variables at each level.
- Computation begins at the source level and can be carried out level by level.

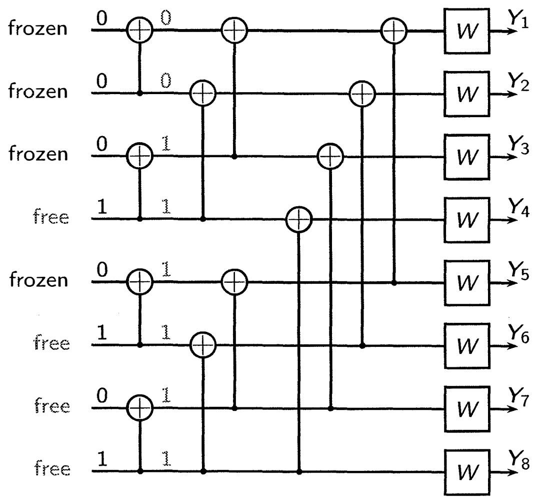
- Space complexity $O(N)$, time complexity $O(N \log N)$.

Encoding: an example

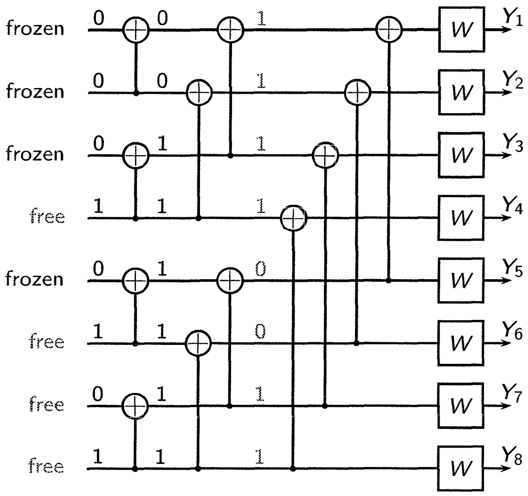


Setup Polarization Encoding Decoding Construction

Encoding: an example

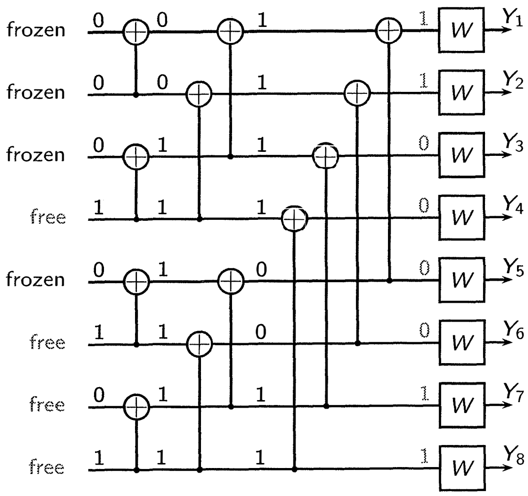


Encoding: an example



Setup Polarization Encoding Decoding Construction

Encoding: an example

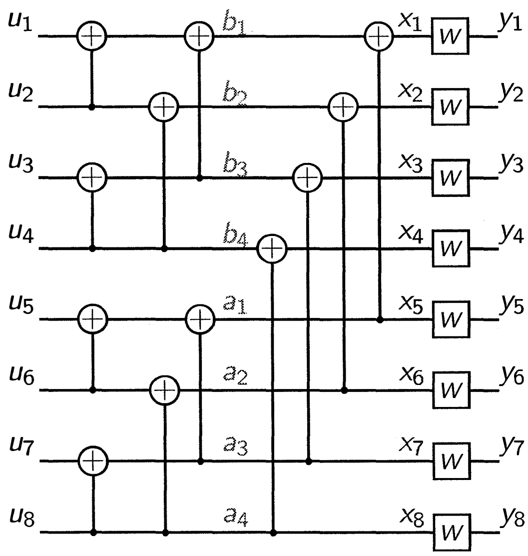


Successive Cancellation Decoding (SCD)

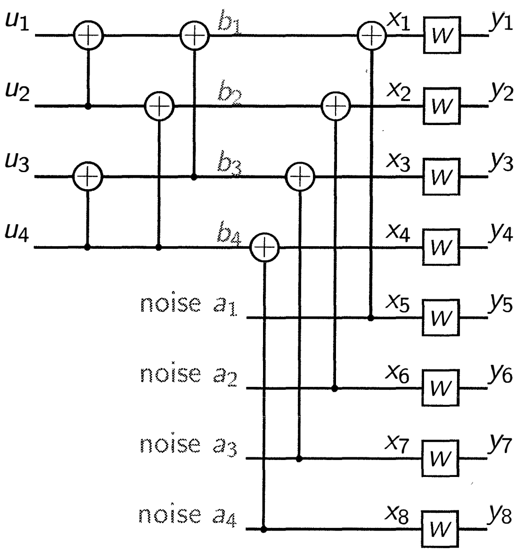
Theorem
The complexity of successive cancellation decoding for polar codes is $\mathcal{O}(N \log N)$.

Proof: Given below.

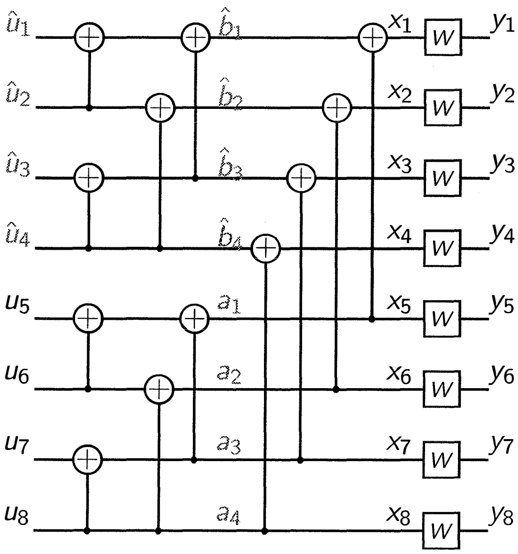
SCD: Exploit the $\mathbf{x} = |\mathbf{a}|\mathbf{a} + \mathbf{b}|$ structure



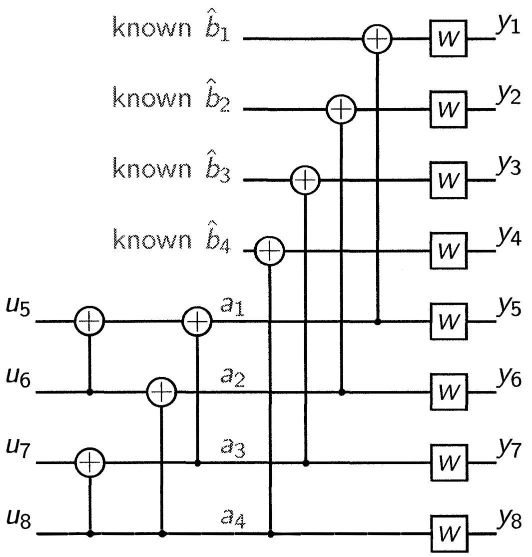
First phase: treat a as noise, decode (u_1, u_2, u_3, u_4)



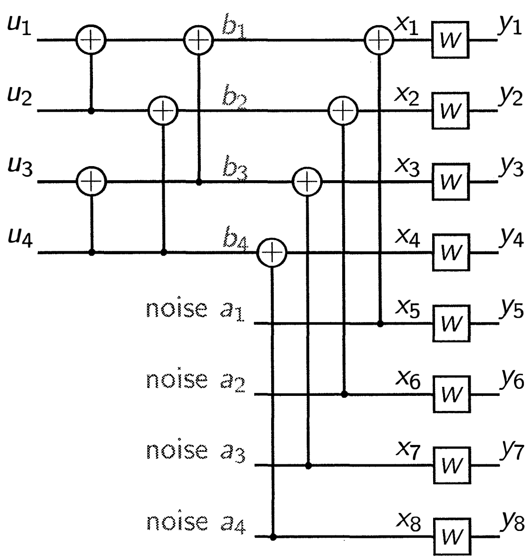
End of first phase



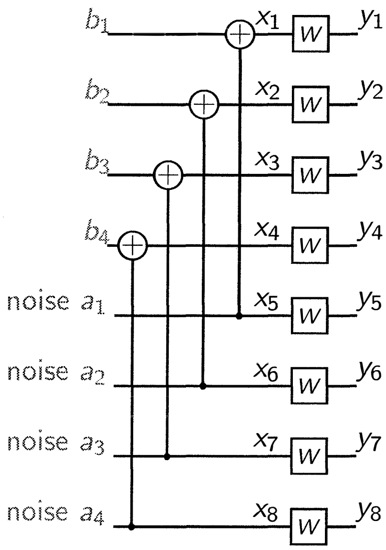
Second phase: Treat $\hat{\mathbf{b}}$ as known, decode (u_5, u_6, u_7, u_8)



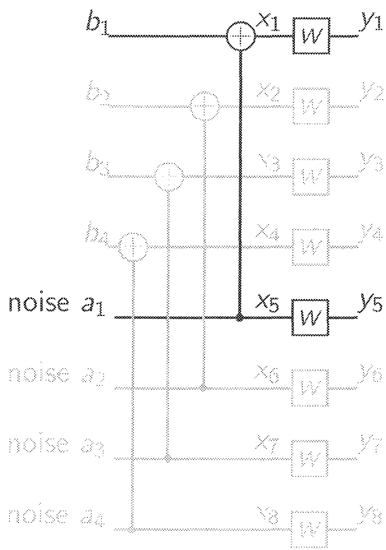
First phase in detail



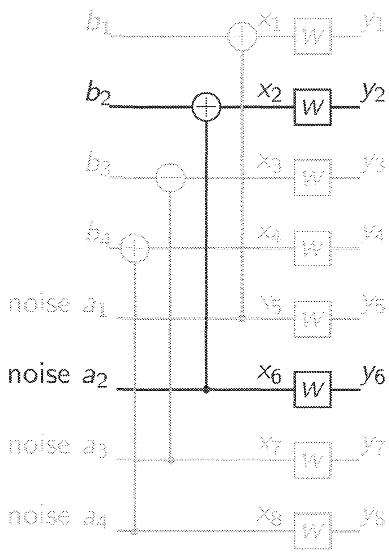
Equivalent channel model



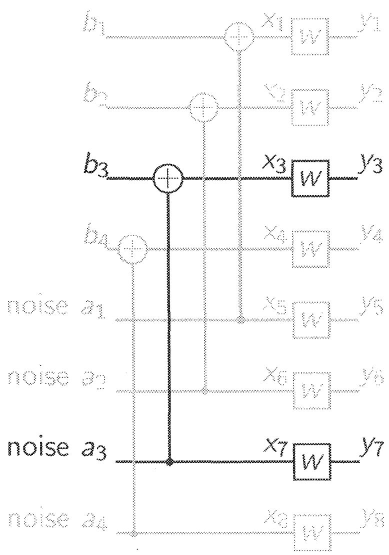
First copy of W^-



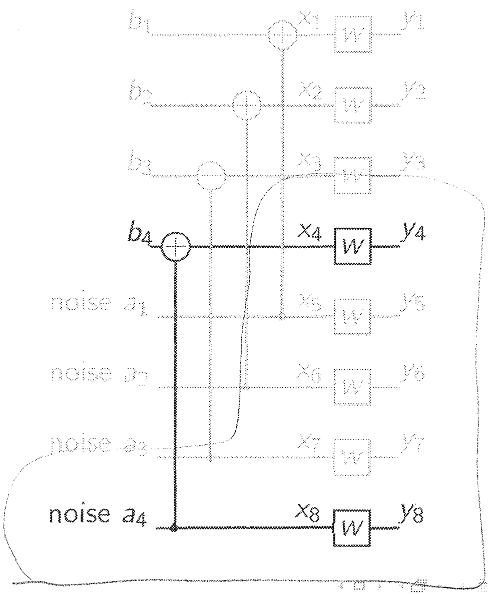
Second copy of W^-



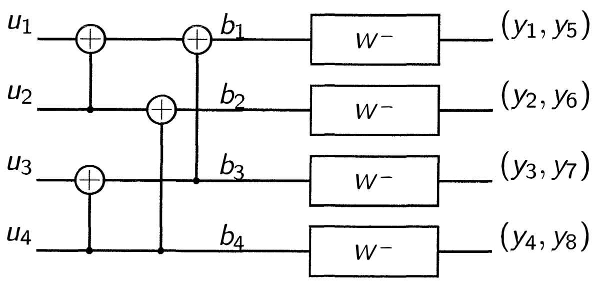
Third copy of W^-



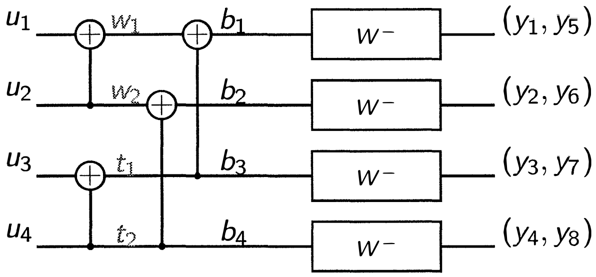
Fourth copy of W^-



Decoding on W^-



$\mathbf{b} = |\mathbf{t}|\mathbf{t} + \mathbf{w}|$

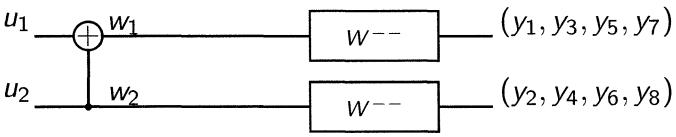


u_3, u_4 是 noise 了

Setup Polarization Encoding Decoding Construction

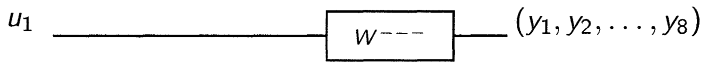
Decoding on W^{--}

b_1, t_2 是 noise 了



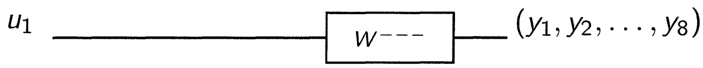
u_2 是 noise 了!

Decoding on W^{---}



Setup Polarization Encoding Decoding Construction

Decoding on W^{---}



Compute

$$L^{---} \triangleq \frac{W^{---}(y_1, \ldots, y_8 \mid u_1 = 0)}{W^{---}(y_1, \ldots, y_8 \mid u_1 = 1)}.$$

Decoding on W^{---}

$$u_1 \longrightarrow \boxed{W^{---}} \longrightarrow (y_1, y_2, \ldots, y_8)$$

Compute

$$L^{---} \triangleq \frac{W^{---}(y_1, \ldots, y_8 \mid u_1 = 0)}{W^{---}(y_1, \ldots, y_8 \mid u_1 = 1)}.$$

Set

$$\hat{u}_1 = \begin{cases} u_1 & \text{if } u_1 \text{ is frozen} \\ 0 & \text{else if } L^{---} > 0 \\ 1 & \text{else} \end{cases}$$

Setup Polarization Encoding Decoding Construction

Decoding on W^{---}

$$u_1 \longrightarrow \boxed{W^{---}} \longrightarrow (y_1, y_2, \ldots, y_8)$$

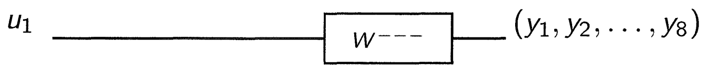
Compute

$$L^{---} \triangleq \frac{W^{---}(y_1, \ldots, y_8 \mid u_1 = 0)}{W^{---}(y_1, \ldots, y_8 \mid u_1 = 1)}.$$

Set

$$\hat{u}_1 = \begin{cases} u_1 & \text{if } u_1 \text{ is frozen} \\ 0 & \text{else if } L^{---} > 0 \\ 1 & \text{else} \end{cases}$$

Decoding on W^{---}



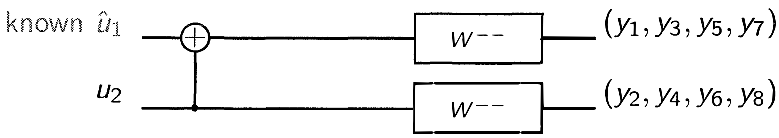
Compute

$$L^{---} \triangleq \frac{W^{---}(y_1, \ldots, y_8 \mid u_1 = 0)}{W^{---}(y_1, \ldots, y_8 \mid u_1 = 1)}.$$

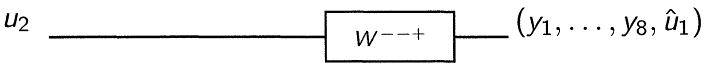
Set

$$\hat{u}_1 = \begin{cases} u_1 & \text{if } u_1 \text{ is frozen} \\ 0 & \text{else if } L^{---} > 0 \\ 1 & \text{else} \end{cases}$$

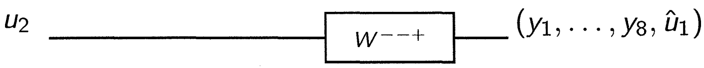
Decoding on W^{--+}



Decoding on W^{--+}



Decoding on W^{--+}



Compute

$$L^{--+} \triangleq \frac{W^{--+}(y_1, \dots, y_8, \hat{u}_1 \mid u_2 = 0)}{W^{--+}(y_1, \dots, y_8, \hat{u}_1 \mid u_2 = 1)}.$$

Decoding on W^{--+}

$$u_2 \;\text{———}\; \boxed{W^{--+}} \;\text{—}\; (y_1, \ldots, y_8, \hat{u}_1)$$

Compute

$$L^{--+} \triangleq \frac{W^{--+}(y_1, \ldots, y_8, \hat{u}_1 \mid u_2 = 0)}{W^{--+}(y_1, \ldots, y_8, \hat{u}_1 \mid u_2 = 1)}.$$

Set

$$\hat{u}_2 = \begin{cases} u_2 & \text{if } u_2 \text{ is frozen} \\ 0 & \text{else if } L^{--+} > 0 \\ 1 & \text{else} \end{cases}$$

Setup Polarization Encoding Decoding Construction

Decoding on W^{--+}

$$u_2 \;\text{———}\; \boxed{W^{--+}} \;\text{—}\; (y_1, \ldots, y_8, \hat{u}_1)$$

Compute

$$L^{--+} \triangleq \frac{W^{--+}(y_1, \ldots, y_8, \hat{u}_1 \mid u_2 = 0)}{W^{--+}(y_1, \ldots, y_8, \hat{u}_1 \mid u_2 = 1)}.$$

Set

$$\hat{u}_2 = \begin{cases} u_2 & \text{if } u_2 \text{ is frozen} \\ 0 & \text{else if } L^{--+} > 0 \\ 1 & \text{else} \end{cases}$$

Decoding on W^{--+}

$$u_2 \longrightarrow \boxed{W^{--+}} \longrightarrow (y_1, \ldots, y_8, \hat{u}_1)$$

Compute

$$L^{--+} \triangleq \frac{W^{--+}(y_1, \ldots, y_8, \hat{u}_1 \mid u_2 = 0)}{W^{--+}(y_1, \ldots, y_8, \hat{u}_1 \mid u_2 = 1)}.$$

Set

$$\hat{u}_2 = \begin{cases} u_2 & \text{if } u_2 \text{ is frozen} \\ 0 & \text{else if } L^{--+} > 0 \\ 1 & \text{else} \end{cases}$$

Complexity for successive cancelation decoding

- Let C_N be the complexity of decoding a code of length N
- Decoding problem of size N for W reduced to two decoding problems of size $N/2$ for W^- and W^+
- So

$$C_N = 2C_{N/2} + kN$$

for some constant k

- This gives $C_N = O(N \log N)$

Complexity for successive cancelation decoding

- Let C_N be the complexity of decoding a code of length N
- Decoding problem of size N for W reduced to two decoding problems of size $N/2$ for W^- and W^+

Complexity for successive cancelation decoding

- Let C_N be the complexity of decoding a code of length N
- Decoding problem of size N for W reduced to two decoding problems of size $N/2$ for W^- and W^+
- So
  $$C_N = 2C_{N/2} + kN$$
  for some constant k
- This gives $C_N = O(N \log N)$

Complexity for successive cancelation decoding

- Let C_N be the complexity of decoding a code of length N
- Decoding problem of size N for W reduced to two decoding problems of size $N/2$ for W^- and W^+
- So
$$C_N = 2C_{N/2} + kN$$
for some constant k
- This gives $C_N = \mathcal{O}(N \log N)$

Performance of polar codes

Theorem

For any rate $R < I(W)$ and block-length N, the probability of frame error for polar codes under successive cancelation decoding is bounded as
$$P_e(N, R) = o\left(2^{-\sqrt{N}+o(\sqrt{N})}\right)$$

Proof: Given in the next presentation.

Construction complexity

Theorem

Given W and a rate $R < I(W)$, a polar code can be constructed in $\mathcal{O}(N\text{poly}(log(N)))$ time that achieves under SCD the performance

$$P_e = o\left(2^{-\sqrt{N}+o(\sqrt{N})}\right)$$

Proof: Given in the next presentation.

Setup Polarization Encoding Decoding Construction

Polar coding summary

Summary

Given W, $N = 2^n$, and $R < I(W)$, a polar code can be constructed such that it has

- construction complexity $\mathcal{O}(N\text{poly}(log(N)))$,
- encoding complexity $\approx N \log N$,
- successive-cancellation decoding complexity $\approx N \log N$,
- frame error probability $P_e(N, R) = o\left(2^{-\sqrt{N}+o(\sqrt{N})}\right)$.

Polar coding summary

Summary

Given W, $N = 2^n$, and $R < I(W)$, a polar code can be constructed such that it has

- construction complexity $\mathcal{O}(N\text{poly}(log(N)))$,
- encoding complexity $\approx N \log N$,
- successive-cancellation decoding complexity $\approx N \log N$,
- frame error probability $P_e(N, R) = o\left(2^{-\sqrt{N} - o(\sqrt{N})}\right)$.

Setup Polarization Encoding Decoding Construction

Polar coding summary

Summary

Given W, $N = 2^n$, and $R < I(W)$, a polar code can be constructed such that it has

- construction complexity $\mathcal{O}(N\text{poly}(log(N)))$,
- encoding complexity $\approx N \log N$,
- successive-cancellation decoding complexity $\approx N \log N$,
- frame error probability $P_e(N, R) = o\left(2^{-\sqrt{N} - o(\sqrt{N})}\right)$.

Polar coding summary

Summary

Given W, $N = 2^n$, and $R < I(W)$, a polar code can be constructed such that it has

- construction complexity $\mathcal{O}(N\text{poly}(log(N)))$,
- encoding complexity $\approx N \log N$,
- successive-cancellation decoding complexity $\approx N \log N$,
- frame error probability $P_e(N, R) = o\left(2^{-\sqrt{N}+o(\sqrt{N})}\right)$.

Polar codes: nits and grits

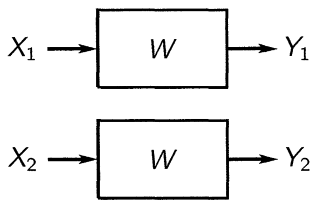
Erdal Arıkan, Emre Telatar

Bilkent U., EPFL

Cambridge — July 1, 2012

Building block

Given two copies of a binary input channel $W: \mathbb{F}_2 \to \mathcal{Y}$



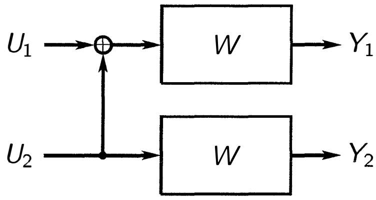
Building block

Given two copies of a binary input channel $W\colon \mathbb{F}_2 \to \mathcal{Y}$

- Set

$$X_1 = U_1 + U_2$$
$$X_2 = U_2$$

with U_1, U_2 i.i.d., uniform on $\mathbb{F}_2$.



Basics Polarization Speed Complexity

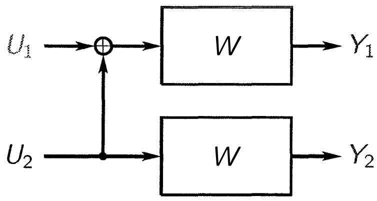
Building block

Given two copies of a binary input channel $W\colon \mathbb{F}_2 \to \mathcal{Y}$

- Set

$$X_1 = U_1 + U_2$$
$$X_2 = U_2$$

with U_1, U_2 i.i.d., uniform on $\mathbb{F}_2$.

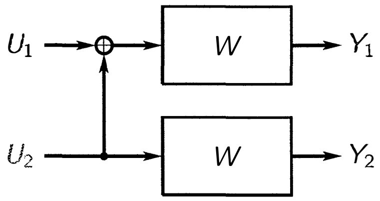


- This induces two synthetic channels $W^{-}\colon \mathbb{F}_2 \to \mathcal{Y}^2$

Building block

Given two copies of a binary input channel $W\colon \mathbb{F}_2 \to \mathcal{Y}$

- Set
$$X_1 = U_1 + U_2$$
$$X_2 = U_2$$
with U_1, U_2 i.i.d., uniform on $\mathbb{F}_2$.



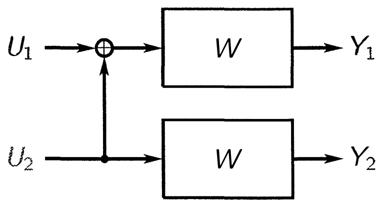
- This induces two synthetic channels $W^-\colon \mathbb{F}_2 \to \mathcal{Y}^2$ and $W^+\colon \mathbb{F}_2 \to \mathcal{Y}^2 \times \mathbb{F}_2$.

Basics Polarization Speed Complexity

Building block

Given two copies of a binary input channel $W\colon \mathbb{F}_2 \to \mathcal{Y}$

- Set
$$X_1 = U_1 + U_2$$
$$X_2 = U_2$$
with U_1, U_2 i.i.d., uniform on $\mathbb{F}_2$.



- This induces two synthetic channels $W^-\colon \mathbb{F}_2 \to \mathcal{Y}^2$ and $W^+\colon \mathbb{F}_2 \to \mathcal{Y}^2 \times \mathbb{F}_2$.
- How come U_1 appears at the output of W^+? Are we being cheated?

Building block: successive decoding

Consider successively decoding $U_1, U_2, \ldots, U_N$ from Y

(a) with a genie-aided decoder:

$$\hat{U}_1 = \phi_1(Y)$$
$$\hat{U}_2 = \phi_2(Y, U_1)$$
$$\hat{U}_3 = \phi_3(Y, U^2)$$
$$\ldots$$
$$\hat{U}_N = \phi_N(Y, U^{N-1})$$

Basics Polarization Speed Complexity

Building block: successive decoding

Consider successively decoding $U_1, U_2, \ldots, U_N$ from Y

(a) with a genie-aided decoder: (b) a Standalone decoder:

$$\begin{aligned} \hat{U}_1 &= \phi_1(Y) & \qquad & \hat{U}_1 = \phi_1(Y) \\ \hat{U}_2 &= \phi_2(Y, U_1) & & \hat{U}_2 = \phi_2(Y, \hat{U}_1) \\ \hat{U}_3 &= \phi_3(Y, U^2) & \text{vs} & \hat{U}_3 = \phi_3(Y, \hat{U}^2) \\ &\ldots & & \ldots \\ \hat{U}_N &= \phi_N(Y, U^{N-1}) & & \hat{U}_N = \phi_N(Y, \hat{U}^{N-1}). \end{aligned}$$

Building block: successive decoding

Consider successively decoding $U_1, U_2, \ldots, U_N$ from Y

(a) with a genie-aided decoder: (b) a Standalone decoder:

$$\begin{aligned} \hat{U}_1 &= \phi_1(Y) \\ \hat{U}_2 &= \phi_2(Y, U_1) \\ \hat{U}_3 &= \phi_3(Y, U^2) \\ &\cdots \\ \hat{U}_N &= \phi_N(Y, U^{N-1}) \end{aligned} \quad \text{vs} \quad \begin{aligned} \hat{U}_1 &= \phi_1(Y) \\ \hat{U}_2 &= \phi_2(Y, \hat{U}_1) \\ \hat{U}_3 &= \phi_3(Y, \hat{U}^2) \\ &\cdots \\ \hat{U}_N &= \phi_N(Y, \hat{U}^{N-1}). \end{aligned}$$

If the genie-aided decoder makes no errors, then, the standalone decoder makes no errors.

Basics Polarization Speed Complexity

Building block: successive decoding

Consider successively decoding $U_1, U_2, \ldots, U_N$ from Y

(a) with a genie-aided decoder: (b) a Standalone decoder:

$$\begin{aligned} \hat{U}_1 &= \phi_1(Y) \\ \hat{U}_2 &= \phi_2(Y, U_1) \\ \hat{U}_3 &= \phi_3(Y, U^2) \\ &\cdots \\ \hat{U}_N &= \phi_N(Y, U^{N-1}) \end{aligned} \quad \text{vs} \quad \begin{aligned} \hat{U}_1 &= \phi_1(Y) \\ \hat{U}_2 &= \phi_2(Y, \hat{U}_1) \\ \hat{U}_3 &= \phi_3(Y, \hat{U}^2) \\ &\cdots \\ \hat{U}_N &= \phi_N(Y, \hat{U}^{N-1}). \end{aligned}$$

If the genie-aided decoder makes no errors, then, the standalone decoder makes no errors. The block error events of the two decoders are the same.

Building block: successive decoding

Consider successively decoding $U_1, U_2, \ldots, U_N$ from Y

(a) with a genie-aided decoder:

$$\hat{U}_1 = \phi_1(Y)$$
$$\hat{U}_2 = \phi_2(Y, U_1)$$
$$\hat{U}_3 = \phi_3(Y, U^2)$$
$$\ldots$$
$$\hat{U}_N = \phi_N(Y, U^{N-1})$$

vs

(b) a Standalone decoder:

$$\hat{U}_1 = \phi_1(Y)$$
$$\hat{U}_2 = \phi_2(Y, \hat{U}_1)$$
$$\hat{U}_3 = \phi_3(Y, \hat{U}^2)$$
$$\ldots$$
$$\hat{U}_N = \phi_N(Y, \hat{U}^{N-1}).$$

If the genie-aided decoder makes no errors, then, the standalone decoder makes no errors. The block error events of the two decoders are the same. As long as the block error probability of the genie-aided decoder is shown to be small, we are not cheated.

Polarization Example: Erasure channel

Suppose W is a BEC(p), i.e., $Y = X$ with probabilty $1 - p$, $Y = ?$ otherwise.

Polarization Example: Erasure channel

Suppose W is a BEC(p), i.e., $Y = X$ with probabilty $1 - p$, $Y = ?$ otherwise.

- W^- has input U_1, output $(Y_1, Y_2) =$

Basics Polarization Speed Complexity

Polarization Example: Erasure channel

Suppose W is a BEC(p), i.e., $Y = X$ with probabilty $1 - p$, $Y = ?$ otherwise.

- W^- has input U_1, output $(Y_1, Y_2) = (U_1 + U_2, U_2)$

Polarization Example: Erasure channel

Suppose W is a BEC(p), i.e., $Y = X$ with probabilty $1-p$, $Y = ?$ otherwise.

- W^- has input U_1, output $(Y_1, Y_2) = (U_1 + U_2,\ ?\ )$

Polarization Example: Erasure channel

Suppose W is a BEC(p), i.e., $Y = X$ with probabilty $1-p$, $Y = ?$ otherwise.

- W^- has input U_1, output $(Y_1, Y_2) = (\quad ? \quad, U_2)$

Polarization Example: Erasure channel

Suppose W is a BEC(p), i.e., $Y = X$ with probabilty $1 - p$, $Y =?$ otherwise.

- W^- has input U_1, output $(Y_1, Y_2) = (\quad ? \quad , ?)$

Basics Polarization Speed Complexity

Polarization Example: Erasure channel

Suppose W is a BEC(p), i.e., $Y = X$ with probabilty $1 - p$, $Y =?$ otherwise.

- W^- is a BEC($2p - p^2$).

Polarization Example: Erasure channel

Suppose W is a BEC(p), i.e., $Y = X$ with probabilty $1-p$, $Y = ?$ otherwise.

- W^- is a BEC($2p - p^2$).
- W^+ has input U_2, output $(Y_1, Y_2, U_1) =$

Basics Polarization Speed Complexity

Polarization Example: Erasure channel

Suppose W is a BEC(p), i.e., $Y = X$ with probabilty $1-p$, $Y = ?$ otherwise.

- W^- is a BEC($2p - p^2$).
- W^+ has input U_2, output $(Y_1, Y_2, U_1) = (U_1 + U_2, U_2, U_1)$

Polarization Example: Erasure channel

Suppose W is a BEC(p), i.e., $Y = X$ with probabilty $1 - p$, $Y = ?$ otherwise.

- W^- is a BEC($2p - p^2$).
- W^+ has input U_2, output $(Y_1, Y_2, U_1) = (\quad ? \quad, U_2, U_1)$

Polarization Example: Erasure channel

Suppose W is a BEC(p), i.e., $Y = X$ with probabilty $1 - p$, $Y = ?$ otherwise.

- W^- is a BEC($2p - p^2$).
- W^+ has input U_2, output $(Y_1, Y_2, U_1) = (U_1 + U_2,\ ?\ , U_1)$

Polarization Example: Erasure channel

Suppose W is a BEC(p), i.e., $Y = X$ with probabilty $1 - p$, $Y =?$ otherwise.

- W^- is a BEC($2p - p^2$).
- W^+ has input U_2, output $(Y_1, Y_2, U_1) = (\quad ? \quad , ? , U_1)$

Polarization Example: Erasure channel

Suppose W is a BEC(p), i.e., $Y = X$ with probabilty $1 - p$, $Y =?$ otherwise.

- W^- is a BEC($2p - p^2$).
- W^+ is a BEC(p^2).

Polarization Example: Erasure channel

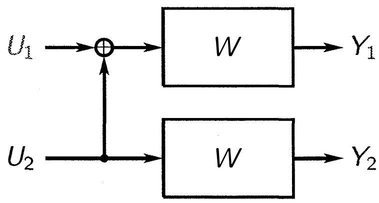
Suppose W is a BEC(p), i.e., $Y = X$ with probabilty $1 - p$, $Y = ?$ otherwise.

- W^- is a BEC($2p - p^2$).
- W^+ is a BEC(p^2).
- We already begin to see some extremalization: W^+ is better than W, while W^- is worse.

Building block: properties

Properties of $W \mapsto (W^-, W^+)$:

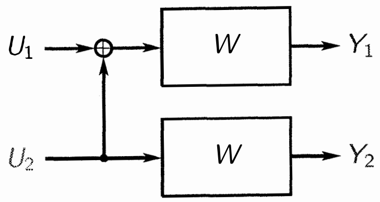
$$I(W^-) = I(U_1; Y_1 Y_2)$$



Building block: properties

Properties of $W \mapsto (W^-, W^+)$:

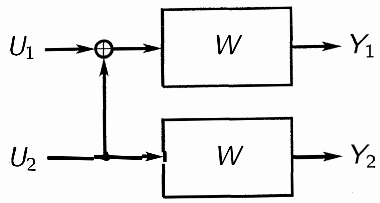
$$I(W^-) = I(U_1; Y_1 Y_2)$$
$$I(W^+) = I(U_2; Y_1 Y_2 U_1)$$



Building block: properties

Properties of $W \mapsto (W^-, W^+)$:

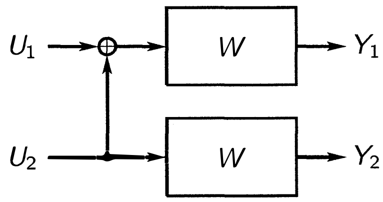
$$I(W^-) = I(U_1; Y_1 Y_2)$$
$$I(W^+) = I(U_2; Y_1 Y_2 U_1)$$
$$I(W^-) + I(W^+) = I(U_1 U_2; Y_1 Y_2)$$



Building block: properties

Properties of $W \mapsto (W^-, W^+)$:

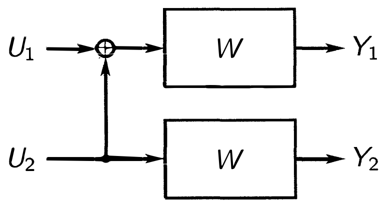
$$\begin{aligned} I(W^-) &= I(U_1; Y_1 Y_2) \\ I(W^+) &= I(U_2; Y_1 Y_2 U_1) \\ I(W^-) + I(W^+) &= I(U_1 U_2; Y_1 Y_2) \\ &= I(X_1 X_2; Y_1 Y_2) \end{aligned}$$



Building block: properties

Properties of $W \mapsto (W^-, W^+)$:

$$\begin{aligned} I(W^-) &= I(U_1; Y_1 Y_2) \\ I(W^+) &= I(U_2; Y_1 Y_2 U_1) \\ I(W^-) + I(W^+) &= I(U_1 U_2; Y_1 Y_2) \\ &= I(X_1 X_2; Y_1 Y_2) \end{aligned}$$

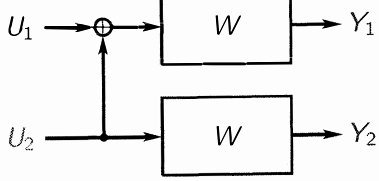


- $\frac{1}{2} I(W^-) + \frac{1}{2} I(W^+) = I(W)$.

Building block: properties

Properties of $W \mapsto (W^-, W^+)$:

$$
\begin{aligned}
I(W^-) &= I(U_1; Y_1 Y_2) \\
I(W^+) &= I(U_2; Y_1 Y_2 U_1) \\
I(W^-) + I(W^+) &= I(U_1 U_2; Y_1 Y_2) \\
&= I(X_1 X_2; Y_1 Y_2)
\end{aligned}
$$

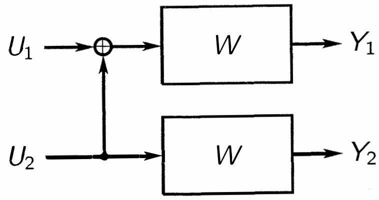


- $\frac{1}{2} I(W^-) + \frac{1}{2} I(W^+) = I(W)$.
- $I(W^+) \geq I(W)$

Building block: properties

Properties of $W \mapsto (W^-, W^+)$:

$$
\begin{aligned}
I(W^-) &= I(U_1; Y_1 Y_2) \\
I(W^+) &= I(U_2; Y_1 Y_2 U_1) \\
I(W^-) + I(W^+) &= I(U_1 U_2; Y_1 Y_2) \\
&= I(X_1 X_2; Y_1 Y_2)
\end{aligned}
$$



- $\frac{1}{2} I(W^-) + \frac{1}{2} I(W^+) = I(W)$.
- $I(W^+) \geq I(W) \geq I(W^-)$.

Building block: properties

Properties of $W \mapsto (W^-, W^+)$:

- $\frac{1}{2}I(W^-) + \frac{1}{2}I(W^+) = I(W)$.
- $I(W^+) \geq I(W) \geq I(W^-)$.

Basics Polarization Speed Complexity

Building block: properties

Properties of $W \mapsto (W^-, W^+)$:

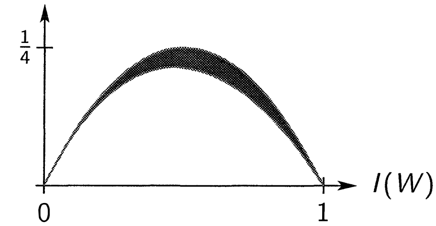
- $\frac{1}{2}I(W^-) + \frac{1}{2}I(W^+) = I(W)$.
- $I(W^+) \geq I(W) \geq I(W^-)$.

Building block: properties

Properties of $W \mapsto (W^-, W^+)$:

- $\frac{1}{2}I(W^-) + \frac{1}{2}I(W^+) = I(W)$.
- $I(W^+) \geq I(W) \geq I(W^-)$.

$I(W^+) - I(W) = I(W) - I(W^-)$

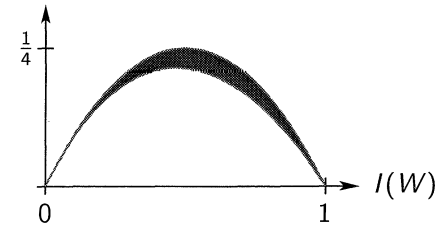


Building block: properties

Properties of $W \mapsto (W^-, W^+)$:

- $\frac{1}{2}I(W^-) + \frac{1}{2}I(W^+) = I(W)$.
- $I(W^+) \geq I(W) \geq I(W^-)$.
- 'Guaranteed progress' unless already extremal.

$I(W^+) - I(W) = I(W) - I(W^-)$



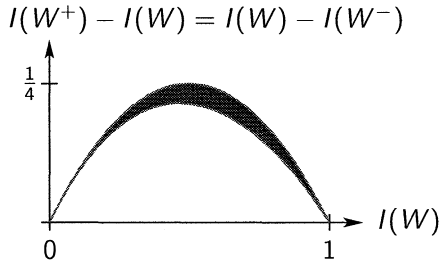
Building block: properties

Properties of $W \mapsto (W^-, W^+)$:

- $\frac{1}{2}I(W^-) + \frac{1}{2}I(W^+) = I(W)$.
- $I(W^+) \geq I(W) \geq I(W^-)$.
- 'Guaranteed progress' unless already extremal.
- $|I(W^\pm) - I(W)| < \delta$ implies

$$I(W) \notin (\epsilon, 1-\epsilon),$$

with $\epsilon(\delta) \to 0$ as $\delta \to 0$.



Guaranteed progress

Notation: $h(p) = -p\log_2 p - (1-p)\log_2(1-p)$, denotes the binary entropy function.
Define $p * q := p(1-q) + (1-p)q$; handy when expressing the distribution of the mod-2 sum of independent binary RVs.

Guaranteed progress

Notation: $h(p) = -p\log_2 p - (1-p)\log_2(1-p)$, denotes the binary entropy function.
Define $p * q := p(1-q) + (1-p)q$; handy when expressing the distribution of the mod-2 sum of independent binary RVs.

Lemma

If (X_1, Y_1) and (X_2, Y_2) are independent, X_1 and X_2 are binary, $H(X_1|Y_1) = h(p_1)$, and $H(X_2|Y_2) = h(p_2)$, then,

$$H(X_1 + X_2|Y_1Y_2) \geq h(p_1 * p_2).$$

Basics Polarization Speed Complexity

Guaranteed progress

Notation: $h(p) = -p\log_2 p - (1-p)\log_2(1-p)$, denotes the binary entropy function.
Define $p * q := p(1-q) + (1-p)q$; handy when expressing the distribution of the mod-2 sum of independent binary RVs.

Lemma

If (X_1, Y_1) and (X_2, Y_2) are independent, X_1 and X_2 are binary, $H(X_1|Y_1) = h(p_1)$, and $H(X_2|Y_2) = h(p_2)$, then,

$$H(X_1 + X_2|Y_1Y_2) \geq h(p_1 * p_2).$$

Proof (Lazy).

This is just Mrs Gerber's Lemma. □

Guaranteed progress

Corollary

If $I(W) = 1 - h(p)$, then $I(W^-) \leq 1 - h(p * p)$, and thus $I(W) - I(W^-) \geq h(p * p) - h(p)$.

Guaranteed progress

Corollary

If $I(W) = 1 - h(p)$, then $I(W^-) \leq 1 - h(p * p)$, and thus $I(W) - I(W^-) \geq h(p * p) - h(p)$.

Proof.

From $I(W) = 1 - h(p)$ we find $H(X_i|Y_i) = h(p)$. Consequently,

$$\begin{aligned} I(W^-) &= I(U_1; Y_1 Y_2) \\ &= 1 - H(U_1|Y_1 Y_2) \\ &= 1 - H(X_1 + X_2|Y_1 Y_2) \\ &\leq 1 - h(p * p) \end{aligned}$$ □

Guaranteed progress

Corollary

For every $\epsilon > 0$, there exists $\delta > 0$ such that

$$\left|I(W) - I(W^{\pm})\right| < \delta$$

implies

$$I(W) \notin (\epsilon, 1-\epsilon).$$

Proof.

See figure. □

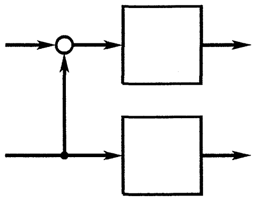
Polarization: why?

Recall the polar construction:

Polarization: why?

Recall the polar construction:

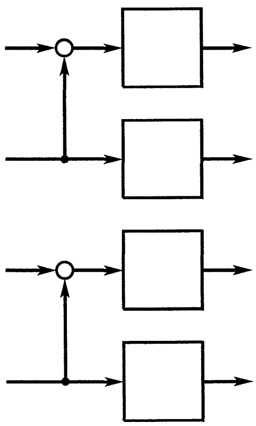
- Duplicate W and obtain W^- and W^+.



Polarization: why?

Recall the polar construction:

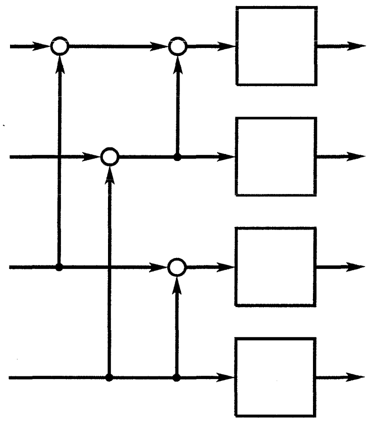
- Duplicate W and obtain W^- and W^+.
- Duplicate W^- (and W^+),



Polarization: why?

Recall the polar construction:

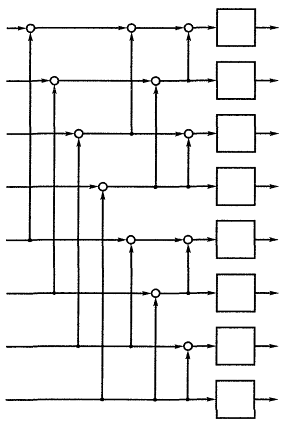
- Duplicate W and obtain W^- and W^+.
- Duplicate W^- (and W^+),
- and obtain W^{--} and W^{-+} (and W^{+-} and W^{++}).



Polarization: why?

Recall the polar construction:

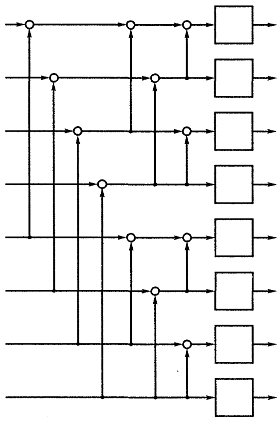
- Duplicate W and obtain W^- and W^+.
- Duplicate W^- (and W^+),
- and obtain W^{--} and W^{-+} (and W^{+-} and W^{++}).
- Duplicate W^{--} (and W^{-+}, W^{+-}, W^{++}) and obtain W^{---} and W^{--+} (and W^{-+-}, W^{-++}, W^{+--}, W^{+-+}, W^{++-}, W^{+++}).



Polarization: why?

Recall the polar construction:

- Duplicate W and obtain W^- and W^+.
- Duplicate W^- (and W^+),
- and obtain W^{--} and W^{-+} (and W^{+-} and W^{++}).
- Duplicate W^{--} (and W^{-+}, W^{+-}, W^{++}) and obtain W^{---} and W^{--+} (and W^{-+-}, W^{-++}, W^{+--}, W^{+-+}, W^{++-}, W^{+++}).
- ...



Polarization: why?

At the nth level into this process we have transformed $N = 2^n$ uses of the channel W to one use each of the 2^n channels

$$W^{b_1 \dots b_n}, \quad b_j \in \{+,-\}.$$

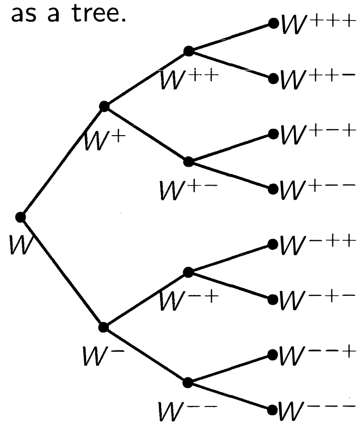
The meaning of polarizatoin is that the 2^n quantities

$$I(W^{-\cdots-}), \dots, I(W^{+\cdots+})$$

are all close to 0 or 1 except for a vanishing fraction (as n grows).

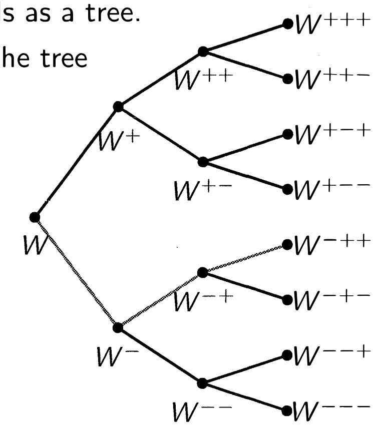
Polarization: why?

- Organize the synthetic channels as a tree.



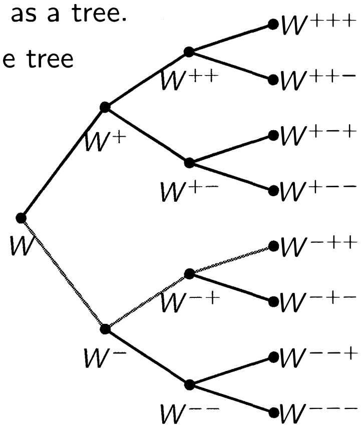
Polarization: why?

- Organize the synthetic channels as a tree.
- Pick a random path climbing the tree according to fair coin flips. This path uniformly samples the nodes at any level n.



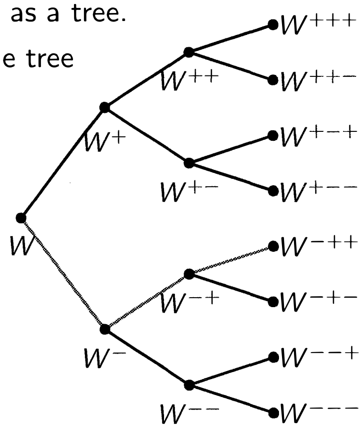
Polarization: why?

- Organize the synthetic channels as a tree.
- Pick a random path climbing the tree according to fair coin flips. This path uniformly samples the nodes at any level n.
- The $I(\cdot)$ sequence we encounter satisfies $E\left[I_{n+1} \mid I_0, \ldots, I_n\right] = I_n$.



Polarization: why?

- Organize the synthetic channels as a tree.
- Pick a random path climbing the tree according to fair coin flips. This path uniformly samples the nodes at any level n.
- The $I(\cdot)$ sequence we encounter satisfies $E\left[I_{n+1} \mid I_0, \ldots, I_n\right] = I_n$.
- Thus, the differences $J_n = I_{n+1} - I_n$ are zero mean, uncorrelated random variables.



Polarization: why?

- $1 \geq (I_n - I_0)^2 = \left(\sum_{k=0}^{n-1} J_k\right)^2 = \sum_{i,k=0}^{n-1} J_i J_k$

Basics Polarization Speed Complexity

Polarization: why?

- $1 \geq (I_n - I_0)^2 = \left(\sum_{k=0}^{n-1} J_k\right)^2 = \sum_{i,k=0}^{n-1} J_i J_k$
- Thus $1 \geq \sum_{k=0}^{n-1} E[J_k^2]$.

Polarization: why?

- $1 \geq (I_n - I_0)^2 = \left(\sum_{k=0}^{n-1} J_k\right)^2 = \sum_{i,k=0}^{n-1} J_i J_k$
- Thus $1 \geq \sum_{k=0}^{n-1} E[J_k^2]$.
- So, $E[J_n^2] \to 0$, thus, for any $\delta > 0$, $\Pr(|J_n| > \delta) \to 0$.

Basics Polarization Speed Complexity

Polarization: why?

- $1 \geq (I_n - I_0)^2 = \left(\sum_{k=0}^{n-1} J_k\right)^2 = \sum_{i,k=0}^{n-1} J_i J_k$
- Thus $1 \geq \sum_{k=0}^{n-1} E[J_k^2]$.
- So, $E[J_n^2] \to 0$, thus, for any $\delta > 0$, $\Pr(|J_n| > \delta) \to 0$.
- By 'guaranteed progress property' the event $\{|J_n| > \delta\}$ includes the event $\{I_n \in (\epsilon, 1-\epsilon)\}$.

Polarization: why?

- $1 \geq (I_n - I_0)^2 = \left(\sum_{k=0}^{n-1} J_k\right)^2 = \sum_{i,k=0}^{n-1} J_i J_k$
- Thus $1 \geq \sum_{k=0}^{n-1} E[J_k^2]$.
- So, $E[J_n^2] \to 0$, thus, for any $\delta > 0$, $\Pr(|J_n| > \delta) \to 0$.
- By 'guaranteed progress property' the event $\{|J_n| > \delta\}$ includes the event $\{I_n \in (\epsilon, 1-\epsilon)\}$.
- Thus the fraction paths for which $I_n \in (\epsilon, 1-\epsilon)$ approaches zero as n gets large. Done! Thanks: H.A. Loeliger

Polarization

- We have shown that $\lim_n \Pr\{I_n \in (\epsilon, 1-\epsilon)\} = 0$.

Polarization

- We have shown that $\lim_n \Pr\{I_n \in (\epsilon, 1-\epsilon)\} = 0$.
- Together with $E[I_n] = I)(W)$ this implies

$$\Pr(I_n \geq 1-\epsilon) \to I(W) \quad \text{and} \quad \Pr(I_n \leq \epsilon) \to 1 - I(W).$$

Basics Polarization Speed Complexity

Polarization

- We have shown that $\lim_n \Pr\{I_n \in (\epsilon, 1-\epsilon)\} = 0$.
- Together with $E[I_n] = I)(W)$ this implies

$$\Pr(I_n \geq 1-\epsilon) \to I(W) \quad \text{and} \quad \Pr(I_n \leq \epsilon) \to 1 - I(W).$$

- Even stronger statements can be made by appealing to the martingale convergence theorem:

$$\Pr\{\lim_n I_n = 1\} = I(W) \quad \text{and} \quad \Pr\{\lim_n I_n = 0\} = 1 - I(W).$$

Polarization speed

- We have seen that polarization takes place.

Polarization speed

- We have seen that polarization takes place.
- But how fast? Fast enough to arrest error propagation?

Polarization speed

- We have seen that polarization takes place.
- But how fast? Fast enough to arrest error propagation?
- Introduce the Bhattacharyya parameter

$$Z(W) = \sum_{y} \sqrt{W(y|0)W(y|1)}$$

as a companion to $I(W)$. Note that this is an upper bound on probability of error for uncoded transmission over W.

A useful representation

$$\begin{aligned} I(W) &= 1 - H(X|Y) \\ &= \sum_{y} W(y)\left[1 - H(X|Y = y)\right] \\ &= \sum_{y} W(y)\left[1 - h\big(W(0|y)\big)\right] \end{aligned}$$

A useful representation

$$\begin{aligned} I(W) &= 1 - H(X|Y) \\ &= \sum_y W(y)[1 - H(X|Y=y)] \\ &= \sum_y W(y)\big[1 - h\big(W(0|y)\big)\big] \end{aligned}$$

Similarly

$$\begin{aligned} Z(W) &= \sum_y \sqrt{W(y|0)W(y|1)} \\ &= \sum_y W(y)\sqrt{4W(0|y)W(1|y)} \\ &= \sum_y W(y)\sqrt{4W(0|y)(1 - W(0|y))} \end{aligned}$$

A useful representation

$$\begin{aligned} I(W) &= 1 - H(X|Y) \\ &= \sum_y W(y)[1 - H(X|Y=y)] \\ &= \sum_y W(y)\big[1 - h\big(W(0|y)\big)\big] \end{aligned}$$

Similarly

$$\begin{aligned} Z(W) &= \sum_y \sqrt{W(y|0)W(y|1)} \\ &= \sum_y W(y)\sqrt{4W(0|y)W(1|y)} \\ &= \sum_y W(y)\sqrt{4W(0|y)(1 - W(0|y))} \end{aligned}$$

So

$$\begin{aligned} I(W) &= E[1 - h(\Delta)] \\ Z(W) &= E[\sqrt{4\Delta(1 - \Delta)}] \end{aligned}$$

A useful representation

$$I(W) = 1 - H(X|Y) = \sum_y W(y)\big[1 - H(X|Y=y)\big] = \sum_y W(y)\big[1 - h\big(W(0|y)\big)\big]$$

Similarly

$$Z(W) = \sum_y \sqrt{W(y|0)W(y|1)} = \sum_y W(y)\sqrt{4W(0|y)W(1|y)} = \sum_y W(y)\sqrt{4W(0|y)(1-W(0|y))}$$

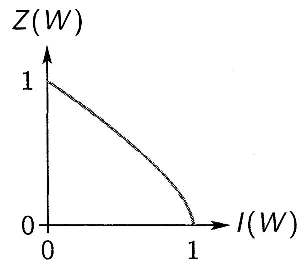
So

$$I(W) = E[1 - h(\Delta)]$$
$$Z(W) = E[\sqrt{4\Delta(1-\Delta)}]$$

Consequently $\big(I(W), Z(W)\big)$ belongs to the Convex hull of the curve

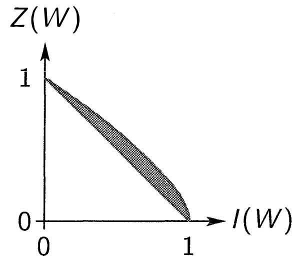
$$\big\{\big(1 - h(\delta), \sqrt{4\delta(1-\delta)}\big) : \delta \in [0,1]\big\}$$

Polarization speed



Polarization speed

Properties of $Z(W)$:

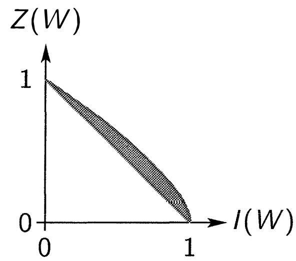


Basics Polarization Speed Complexity

Polarization speed

Properties of $Z(W)$:

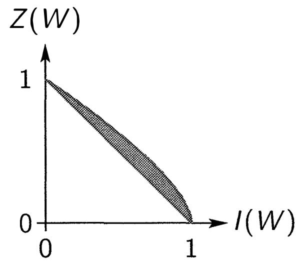
- $Z(W) \approx 0$ iff $I(W) \approx 1$.



Polarization speed

Properties of $Z(W)$:

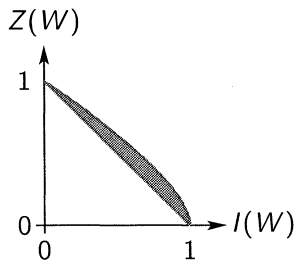
- $Z(W) \approx 0$ iff $I(W) \approx 1$.
- $Z(W) \approx 1$ iff $I(W) \approx 0$.



Polarization speed

Properties of $Z(W)$:

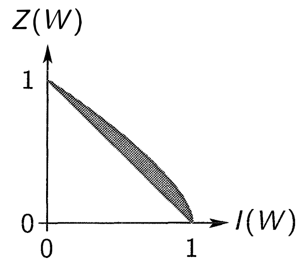
- $Z(W) \approx 0$ iff $I(W) \approx 1$.
- $Z(W) \approx 1$ iff $I(W) \approx 0$.
- $Z(W^+) = Z(W)^2$.



Polarization speed

Properties of $Z(W)$:

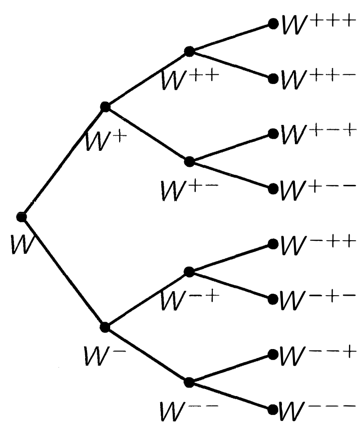
- $Z(W) \approx 0$ iff $I(W) \approx 1$.
- $Z(W) \approx 1$ iff $I(W) \approx 0$.
- $Z(W^+) = Z(W)^2$.
- $Z(W^-) \leq 2Z(W)$.



Since $Z(W)$ upper bounds on probability of error for uncoded transmission over W, we can choose the good indices on the basis of $Z(W)$. The sum of the Z's of the chosen channels will upper bound the block error probability. Good reason to study the polarization speed of Z.

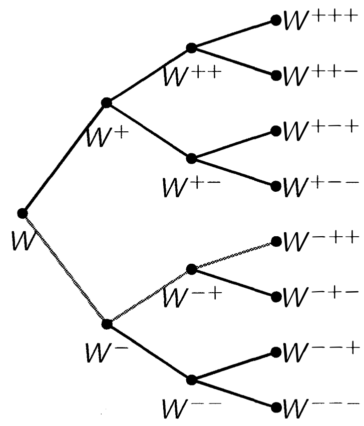
Polarization speed

- Recall the channels organized in a tree.



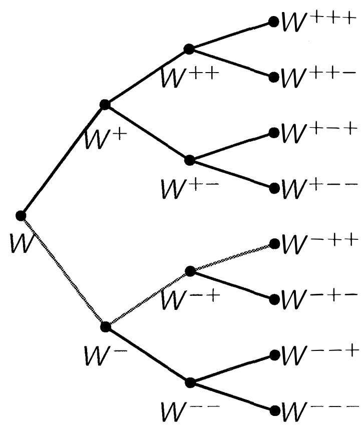
Polarization speed

- Recall the channels organized in a tree.
- Let $Z_0, Z_1, \ldots$ be the $Z(\cdot)$ values we encouter we climb the tree.



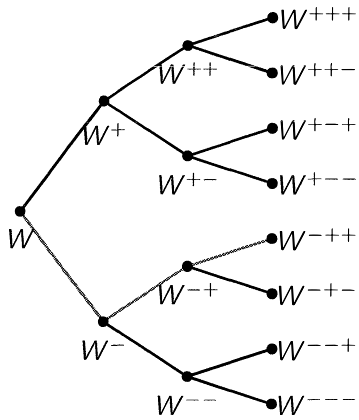
Polarization speed

- Recall the channels organized in a tree.
- Let $Z_0, Z_1, \ldots$ be the $Z(\cdot)$ values we encouter we climb the tree.
- We know that $P(Z_n \to 0) = I(W)$.



Polarization speed

- Recall the channels organized in a tree.
- Let $Z_0, Z_1, \ldots$ be the $Z(\cdot)$ values we encouter we climb the tree.
- We know that $P(Z_n \to 0) = I(W)$.
- We want to show that when $Z_n \to 0$ it does so fast.



Polarization speed

- It is more convenient to work with $V_n = \log_2 Z_n$. This takes values in $(-\infty, 0]$, We already know that $V_n \to -\infty$ with probability $I(W)$, and want to show that it goes to $-\infty$ fast when it does.

Polarization speed

- It is more convenient to work with $V_n = \log_2 Z_n$. This takes values in $(-\infty, 0]$, We already know that $V_n \to -\infty$ with probability $I(W)$, and want to show that it goes to $-\infty$ fast when it does.
- V_n's obey

$$\begin{aligned} V_{n+1} &= 2V_n && \text{for a 'plus' move} \\ V_{n+1} &\le V_n + 1 && \text{for a 'minus' move} \end{aligned}$$

Basics Polarization Speed Complexity

Polarization speed

- It is more convenient to work with $V_n = \log_2 Z_n$. This takes values in $(-\infty, 0]$, We already know that $V_n \to -\infty$ with probability $I(W)$, and want to show that it goes to $-\infty$ fast when it does.
- V_n's obey

$$\begin{aligned} V_{n+1} &= 2V_n && \text{for a 'plus' move} \\ V_{n+1} &\le V_n + 1 && \text{for a 'minus' move} \end{aligned}$$

- E.g., starting with $V_m = -20$, and sequence moves: $-$, $-$, $+$, $-$, $-$, $+$, $+$, $-$, we will see a sequence dominated by -20,

Polarization speed

- It is more convenient to work with $V_n = \log_2 Z_n$. This takes values in $(-\infty, 0]$, We already know that $V_n \to -\infty$ with probability $I(W)$, and want to show that it goes to $-\infty$ fast when it does.
- V_n's obey

$$\begin{aligned} V_{n+1} &= 2V_n && \text{for a 'plus' move} \\ V_{n+1} &\leq V_n + 1 && \text{for a 'minus' move} \end{aligned}$$

- E.g., starting with $V_m = -20$, and sequence moves: $-$, $-$, $+$, $-$, $-$, $+$, $+$, $-$, we will see a sequence dominated by $-20, -19,$

Basics Polarization Speed Complexity

Polarization speed

- It is more convenient to work with $V_n = \log_2 Z_n$. This takes values in $(-\infty, 0]$, We already know that $V_n \to -\infty$ with probability $I(W)$, and want to show that it goes to $-\infty$ fast when it does.
- V_n's obey

$$\begin{aligned} V_{n+1} &= 2V_n && \text{for a 'plus' move} \\ V_{n+1} &\leq V_n + 1 && \text{for a 'minus' move} \end{aligned}$$

- E.g., starting with $V_m = -20$, and sequence moves: $-$, $-$, $+$, $-$, $-$, $+$, $+$, $-$, we will see a sequence dominated by $-20, -19, -18,$

Polarization speed

- It is more convenient to work with $V_n = \log_2 Z_n$. This takes values in $(-\infty, 0]$, We already know that $V_n \to -\infty$ with probability $I(W)$, and want to show that it goes to $-\infty$ fast when it does.
- V_n's obey

$$\begin{aligned} V_{n+1} &= 2V_n && \text{for a 'plus' move} \\ V_{n+1} &\leq V_n + 1 && \text{for a 'minus' move} \end{aligned}$$

- E.g., starting with $V_m = -20$, and sequence moves: $-$, $-$, $+$, $-$, $-$, $+$, $+$, $-$, we will see a sequence dominated by $-20, -19, -18, -36,$

Basics Polarization Speed Complexity

Polarization speed

- It is more convenient to work with $V_n = \log_2 Z_n$. This takes values in $(-\infty, 0]$, We already know that $V_n \to -\infty$ with probability $I(W)$, and want to show that it goes to $-\infty$ fast when it does.
- V_n's obey

$$\begin{aligned} V_{n+1} &= 2V_n && \text{for a 'plus' move} \\ V_{n+1} &\leq V_n + 1 && \text{for a 'minus' move} \end{aligned}$$

- E.g., starting with $V_m = -20$, and sequence moves: $-$, $-$, $+$, $-$, $-$, $+$, $+$, $-$, we will see a sequence dominated by $-20, -19, -18, -36, -35,$

Polarization speed

- It is more convenient to work with $V_n = \log_2 Z_n$. This takes values in $(-\infty, 0]$, We already know that $V_n \to -\infty$ with probability $I(W)$, and want to show that it goes to $-\infty$ fast when it does.
- V_n's obey

$$\begin{aligned} V_{n+1} &= 2V_n && \text{for a 'plus' move} \\ V_{n+1} &\leq V_n + 1 && \text{for a 'minus' move} \end{aligned}$$

- E.g., starting with $V_m = -20$, and sequence moves: $-$, $-$, $+$, $-$, $-$, $+$, $+$, $-$, we will see a sequence dominated by $-20, -19, -18, -36, -35, -34,$

Polarization speed

- It is more convenient to work with $V_n = \log_2 Z_n$. This takes values in $(-\infty, 0]$, We already know that $V_n \to -\infty$ with probability $I(W)$, and want to show that it goes to $-\infty$ fast when it does.
- V_n's obey

$$\begin{aligned} V_{n+1} &= 2V_n && \text{for a 'plus' move} \\ V_{n+1} &\leq V_n + 1 && \text{for a 'minus' move} \end{aligned}$$

- E.g., starting with $V_m = -20$, and sequence moves: $-$, $-$, $+$, $-$, $-$, $+$, $+$, $-$, we will see a sequence dominated by $-20, -19, -18, -36, -35, -34, -68,$

Polarization speed

- It is more convenient to work with $V_n = \log_2 Z_n$. This takes values in $(-\infty, 0]$, We already know that $V_n \to -\infty$ with probability $I(W)$, and want to show that it goes to $-\infty$ fast when it does.
- V_n's obey

$$\begin{aligned} V_{n+1} &= 2V_n && \text{for a 'plus' move} \\ V_{n+1} &\leq V_n + 1 && \text{for a 'minus' move} \end{aligned}$$

- E.g., starting with $V_m = -20$, and sequence moves: $-$, $-$, $+$, $-$, $-$, $+$, $+$, $-$, we will see a sequence dominated by $-20, -19, -18, -36, -35, -34, -68, -136,$

Polarization speed

- It is more convenient to work with $V_n = \log_2 Z_n$. This takes values in $(-\infty, 0]$, We already know that $V_n \to -\infty$ with probability $I(W)$, and want to show that it goes to $-\infty$ fast when it does.
- V_n's obey

$$\begin{aligned} V_{n+1} &= 2V_n && \text{for a 'plus' move} \\ V_{n+1} &\leq V_n + 1 && \text{for a 'minus' move} \end{aligned}$$

- E.g., starting with $V_m = -20$, and sequence moves: $-$, $-$, $+$, $-$, $-$, $+$, $+$, $-$, we will see a sequence dominated by $-20, -19, -18, -36, -35, -34, -68, -136, -135,$

Polarization speed

- It is more convenient to work with $V_n = \log_2 Z_n$. This takes values in $(-\infty, 0]$, We already know that $V_n \to -\infty$ with probability $I(W)$, and want to show that it goes to $-\infty$ fast when it does.
- V_n's obey

$$\begin{aligned} V_{n+1} &= 2V_n && \text{for a 'plus' move} \\ V_{n+1} &\leq V_n + 1 && \text{for a 'minus' move} \end{aligned}$$

- E.g., starting with $V_m = -20$, and sequence moves: $-$, $-$, $+$, $-$, $-$, $+$, $+$, $-$, we will see a sequence dominated by $-20, -19, -18, -36, -35, -34, -68, -136, -135, \ldots$

Basics Polarization Speed Complexity

Polarization speed

- It is more convenient to work with $V_n = \log_2 Z_n$. This takes values in $(-\infty, 0]$, We already know that $V_n \to -\infty$ with probability $I(W)$, and want to show that it goes to $-\infty$ fast when it does.
- V_n's obey

$$\begin{aligned} V_{n+1} &= 2V_n && \text{for a 'plus' move} \\ V_{n+1} &\leq V_n + 1 && \text{for a 'minus' move} \end{aligned}$$

- E.g., starting with $V_m = -20$, and sequence moves: $-$, $-$, $+$, $-$, $-$, $+$, $+$, $-$, we will see a sequence dominated by $-20, -19, -18, -36, -35, -34, -68, -136, -135, \ldots$
- The amounts the 'minus' moves change the V values are negligible compared to the changes made by the 'plus' moves.

Polarizaton speed: heuristics

- To the first approximation, V_n process behaves like

$$\begin{aligned} \tilde{V}_{n+1} &= 2\tilde{V}_n && \text{for a 'plus' move} \\ \tilde{V}_{n+1} &= \tilde{V}_n && \text{for a 'minus' move} \end{aligned}$$

Basics Polarization Speed Complexity

Polarizaton speed: heuristics

- To the first approximation, V_n process behaves like

$$\begin{aligned} \tilde{V}_{n+1} &= 2\tilde{V}_n && \text{for a 'plus' move} \\ \tilde{V}_{n+1} &= \tilde{V}_n && \text{for a 'minus' move} \end{aligned}$$

- In a long sequence of moves we will typlically see an almost equal number of $+$ and $-$'s, thus

$$\tilde{V}_n = O\left(-2^{n/2}\right) = O(-\sqrt{N}).$$

Polarizaton speed: heuristics

- To the first approximation, V_n process behaves like

$$\tilde{V}_{n+1} = 2\tilde{V}_n \qquad \text{for a 'plus' move}$$
$$\tilde{V}_{n+1} = \tilde{V}_n \qquad \text{for a 'minus' move}$$

- In a long sequence of moves we will typically see an almost equal number of $+$ and $-$'s, thus

$$\tilde{V}_n = O(-2^{n/2}) = O(-\sqrt{N}).$$

- So we expect Z_n to behave roughly like $2^{-\sqrt{N}}$.

Polarization speed: more formally

- In going from V_m to V_n we make $n - m$ moves. If $S_{m,n}$ of these are 'plus' moves, then
  $V_n \leq [V_m + (n - m - S_{m,n})]2^{S_{m,n}}$

Polarization speed: more formally

- In going from V_m to V_n we make $n - m$ moves. If $S_{m,n}$ of these are 'plus' moves, then
  $V_n \leq [V_m + (n - m - S_{m,n})]2^{S_{m,n}} \leq [V_m + n - m]2^{S_{m,n}}$

Basics Polarization Speed Complexity

Polarization speed: more formally

- In going from V_m to V_n we make $n - m$ moves. If $S_{m,n}$ of these are 'plus' moves, then
  $V_n \leq [V_m + (n - m - S_{m,n})]2^{S_{m,n}} \leq [V_m + n - m]2^{S_{m,n}}$
- Note that the bound is useful only when $n \leq m - V_m$. So one cannot show too strong a convergence speed based on this alone.

Polarization speed: more formally

- In going from V_m to V_n we make $n-m$ moves. If $S_{m,n}$ of these are 'plus' moves, then
  $V_n \leq [V_m + (n-m-S_{m,n})]2^{S_{m,n}} \leq [V_m + n - m]2^{S_{m,n}}$
- Note that the bound is useful only when $n \leq m - V_m$. So one cannot show too strong a convergence speed based on this alone.
- But using the bound twice by introducing an intermediate destination k:

$$V_n \leq [V_k + n - k]2^{S_{k,n}}$$

Basics Polarization Speed Complexity

Polarization speed: more formally

- In going from V_m to V_n we make $n-m$ moves. If $S_{m,n}$ of these are 'plus' moves, then
  $V_n \leq [V_m + (n-m-S_{m,n})]2^{S_{m,n}} \leq [V_m + n - m]2^{S_{m,n}}$
- Note that the bound is useful only when $n \leq m - V_m$. So one cannot show too strong a convergence speed based on this alone.
- But using the bound twice by introducing an intermediate destination k:

$$\begin{aligned} V_n &\leq [V_k + n - k]2^{S_{k,n}} \\ &\leq \Big[[V_m + k - m]2^{S_{m,k}} + n - k\Big]2^{S_{k,n}} \end{aligned}$$

Polarization speed: more formally

If V_m were less than $-2m$, we could take $k = 2m$, and $n = m^2$ to obtain

$$\begin{aligned} V_{m^2} &\leq [-m2^{S_{m,2m}} + m^2 - 2m]2^{S_{2m,m^2}} \\ &= [-m2^{m(1-\epsilon)} + m^2 - 2m]2^{(m^2-m)(1-\epsilon)/2} \qquad \text{(typically)} \\ &= O\big(-2^{m^2(0.5-\epsilon)}\big) \end{aligned}$$

Equivalently,

$$V_n \leq O(-N^{0.5-\epsilon})$$

Polarization speed: more formally

- Only thing left to show is that $V_m \leq -2m$ is a typical event for the paths where $V_n \to -\infty$.

Polarization speed: more formally

- Only thing left to show is that $V_m \le -2m$ is a typical event for the paths where $V_n \to -\infty$.
- On such paths, there will come a time n_0 so that $V_n \le -11$ for all $n \ge n_0$. The evolution of V_n then satisfies

$$V_{n+1} \le 2V_n \le V_n - 11 \qquad \text{'plus' moves}$$
$$V_{n+1} \le V_n + 1 \qquad \text{'minus' moves}$$

Basics Polarization Speed Complexity

Polarization speed: more formally

- Only thing left to show is that $V_m \le -2m$ is a typical event for the paths where $V_n \to -\infty$.
- On such paths, there will come a time n_0 so that $V_n \le -11$ for all $n \ge n_0$. The evolution of V_n then satisfies

$$V_{n+1} \le 2V_n \le V_n - 11 \qquad \text{'plus' moves}$$
$$V_{n+1} \le V_n + 1 \qquad \text{'minus' moves}$$

- Thus from n_0 onwards, V_n is dominated by a random walk with average drift -5.

Polarization speed: more formally

- Only thing left to show is that $V_m \leq -2m$ is a typical event for the paths where $V_n \to -\infty$.
- On such paths, there will come a time n_0 so that $V_n \leq -11$ for all $n \geq n_0$. The evolution of V_n then satisfies

$$\begin{aligned} V_{n+1} &\leq 2V_n \leq V_n - 11 && \text{'plus' moves} \\ V_{n+1} &\leq V_n + 1 && \text{'minus' moves} \end{aligned}$$

- Thus from n_0 onwards, V_n is dominated by a random walk with average drift -5.
- Thus at time $m = 2n_0$ the typical value of V_m is dominated by $-5n_0 = -2.5m \leq -2m$, which is what we want (with room to spare).

Construction complexity

Let $V \preceq W$ denote that V is stochastically degraded with respect to W.

Lemma

If $V \preceq W$ then $V^{\pm} \preceq W^{\pm}$.

Proof.

Obvious. □

Construction complexity

Lemma

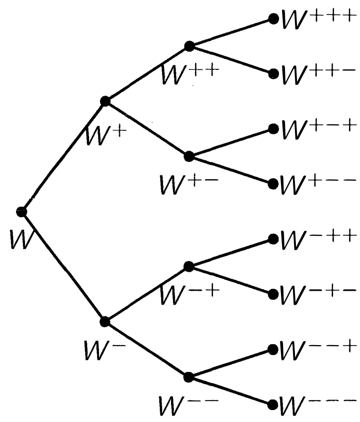
Given any symmetric channel W, and $\delta > 0$ there is a symmetric channel V such that

- $V \preceq W$
- $I(W) - I(V) \leq \delta$
- *V has an output alphabet of cardinality $\leq 2/\delta$.*

Moreover, one can efficiently find such a V.

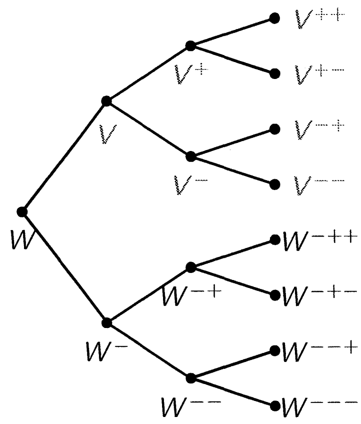
Construction complexity

- If we take the tree of channels,



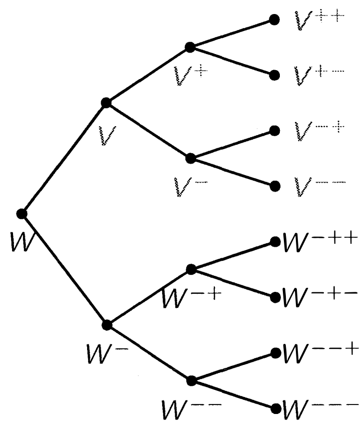
Construction complexity

- If we take the tree of channels,
- Replace a channel on a node by a stochastically degraded version (E.g., replace W^+ by a $V \preceq W^+$) whose mutual information is differs from the original by δ, (E.g, $I(W^+) - I(V) = \delta$)

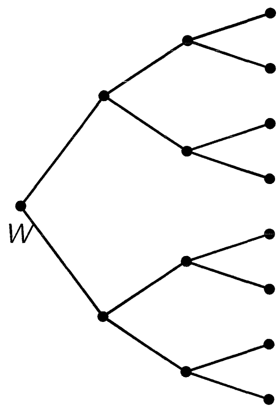


Construction complexity

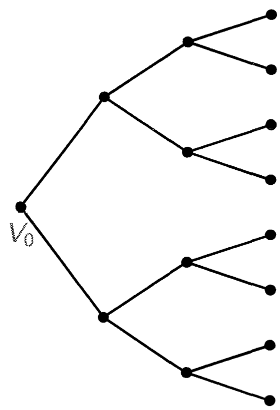
- If we take the tree of channels,
- Replace a channel on a node by a stochastically degraded version (E.g., replace W^+ by a $V \preceq W^+$) whose mutual information is differs from the original by δ, (E.g, $I(W^+) - I(V) = \delta$)
- Then the average loss of mutual information the descendants of this node at any level equals δ.



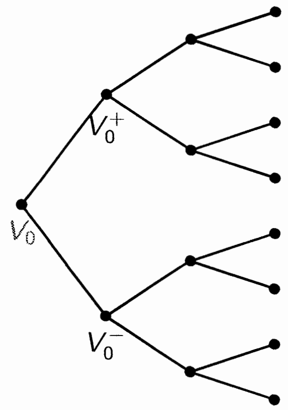
Construction complexity



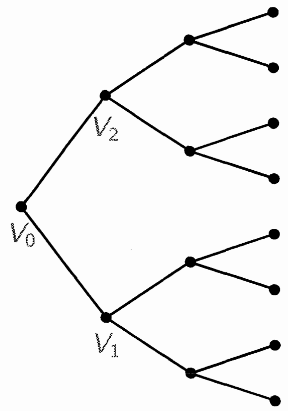
Construction complexity



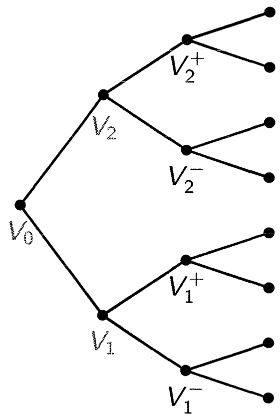
Construction complexity



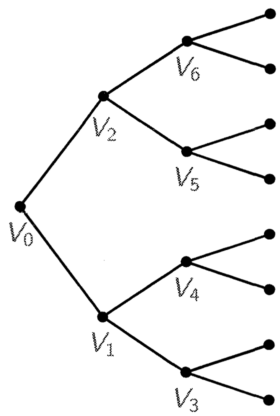
Construction complexity



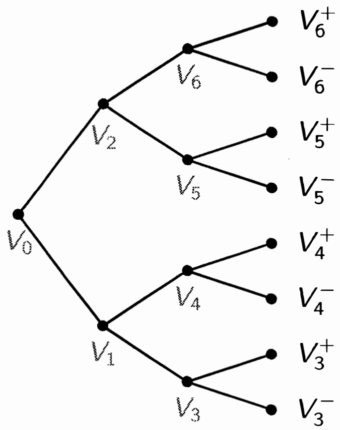
Construction complexity



Construction complexity

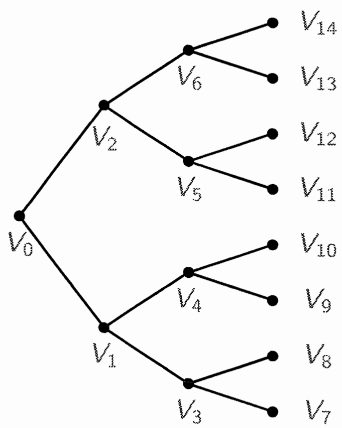


Construction complexity

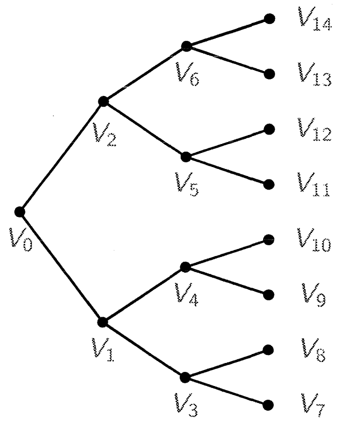


Basics Polarization Speed Complexity

Construction complexity



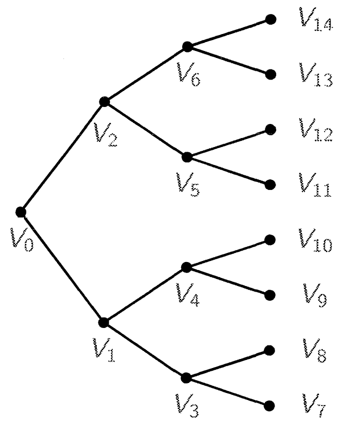
Construction complexity



- If each of the replacements are as in the lemma, their total effect on average loss of mutual information on the nth level of the tree is $(n+1)\delta$

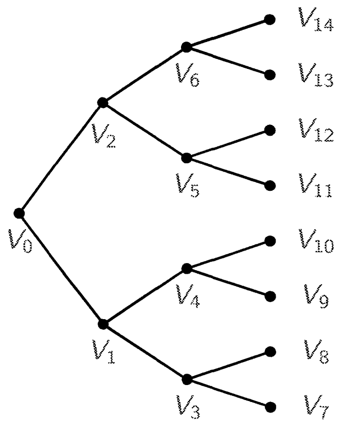
Basics Polarization Speed Complexity

Construction complexity



- If each of the replacements are as in the lemma, their total effect on average loss of mutual information on the nth level of the tree is $(n+1)\delta$
- Choosing $\delta = 1/(n+1)n$ ensures that the average loss is at most $1/n$.

Construction complexity



- If each of the replacements are as in the lemma, their total effect on average loss of mutual information on the nth level of the tree is $(n+1)\delta$
- Choosing $\delta = 1/(n+1)n$ ensures that the average loss is at most $1/n$.
- In particular the fraction of channels that suffer a loss more than $1/\sqrt{n}$ is less than $1/\sqrt{n}$.