Polar 符号の紹介

— C プログラミングによる Polar 符号の体験 —

An Introduction to Polar Codes
— Experience of Polar Codes in C Programming —

岩田賢一 Ken-ichi IWATA

アブストラクト

情報理論は、与えられたシステムにおける符号化問題の理論的境界を数理的な情報量を用いて明確にすると共に、その理論境界を達成する符号化法を明確にすることを目的としている。近年、Arıkan は、確率過程の分極操作に基づいた polar 符号と呼ばれる新しいクラスの符号を提案している。polar 符号は、定常無記憶情報源及び対称通信路の理論的境界を低計算複雑度と低空間複雑度で達成できることを明らかにしている。polar 符号は、多端子符号化問題を含め様々な符号化問題に対して、その理論的境界を低計算複雑度と低空間複雑度で達成する符号の構成法に関して一つの基本的な原理をもたらすことが期待される。本稿では、polar 符号を情報源符号化と通信路符号化について紹介するとともに、polar 符号の C プログラミングによる例を紹介する。

キーワード 情報理論, 符号理論, polar 符号, 情報源分極, 情報源符号, 通信路分極, 通信路符号

Abstract Information theory clarifies the theoretical limit of a coding problem on a given communication system by using the mathematical theory of information, and also establishes attempts to establish coding techniques that will achieve the theoretical limit. In recent years, Arıkan has proposed a new class of codes, called polar codes, based on the polarization of a random process. The polar codes provably achieve the theoretical limit for communication systems with both stationary memoryless sources and binary input memoryless channels with low computational and space complexities. It is expected that polar codes will provide fundamental principles for encoding and decoding techniques with low computational and space complexities for various coding problems including multi terminal coding problems. In this paper, introduce polar codes based on polarization used for two typical communication systems with source coding or channel coding, and give an implementation of the polar codes in the C programming language.

Key words information theory, coding theory, polar codes, source polarization, source codes, channel polarization, channel codes

1. はじめに

情報理論は、与えられたシステムにおける符号化問題の理論的境界を数理的な情報量により明確にするとともに、その理論境界を達成する操作的な符号化法を明確にする理論である⁽¹⁾。今回の記事は、情報理論分野における論文賞である 2010 IEEE Information Theory Society Paper Award を受賞した下記の論文⁽²⁾において、E. Arıkan が提案した polar 符号の紹介である。

E. Arıkan, “Channel polarization: a method for constructing capacity-achieving codes for symmetric binary-input memoryless channels,” IEEE Trans. Inf. Theory, vol.55, no.7, pp.3051–3073, 2009.

岩田賢一 正員 福井大学大学院工学研究科情報・メディア工学専攻
E-mail k-iwata@u-fukui.ac.jp
Ken-ichi IWATA, Member (Graduate School of Engineering, University of Fukui, Fukui-shi, 910-8507 Japan).

電子情報通信学会 基礎・境界ソサイエティ
Fundamentals Review Vol.6 No.3 pp.175–198 2013 年 1 月
©電子情報通信学会 2013

Arıkan が同論文で提案した polar 符号は、線形符号のクラスに属する通信路符号であり、次の顕著な特徴を有している。

- 符号長 n に対して符号化と復号が $O(n \log n)$ の計算複雑度で実装可能であり、この復号は逐次除去復号と呼ばれる。
- 符号化レート (伝送速度) が対称通信路容量よりも小さければ、任意に固定した $0 < \delta < \frac{1}{2}$ に対して、符号長 n の増加に対して逐次除去復号法による平均ブロック誤り確率は $o(2^{-n^{\frac{1}{2}-\delta}})$ であり、その極限值は 0 へ収束する厳密な証明が与えられている。

ここで、上記の対称通信路容量は次のように定義される。通信路への入力記号における確率分布を一様分布と仮定して、通信路での入出力間の同時分布から求められる相互情報量を対称通信路容量とする。この対称通信路容量は、定常無記憶対称通信路に対して達成できる符号化レートの漸近的理論境界を表している。上記の二つの特徴は、polar 符号が定常無記憶対称通信路の通信路容量を $O(n \log n)$ の符号化と復号の計算複雑度で達成できることを示している。言い換えれば、「実用的な複雑度で符号化と復号が実行でき、かつ符号化レートの良い符号をどのように設計するか」という符号理論⁽³⁾の主要な課題に対し

て、一つの明確な解を与えているといえる。

polar 符号では、通信路符号を構成するときの一つの大きな特徴として、通信路分極 (channel polarization) という概念が Arikan により提案されている。この通信路分極に対比して、無ひずみ情報源符号の構成においても情報源分極 (source polarization) という概念がやはり Arikan より提案されている⁽⁴⁾。polar 符号の符号構成における基礎的な原理の一つとして情報量の分極操作があり、情報源若しくは通信路の再帰的な可逆変換による次数の拡大構成とチェインルールによる条件付情報量への分解を行い、符号長を大きくした極限において以降の章で紹介する情報量の分極操作が可能であることを用いている。3. 以降で説明する分極操作とは確率過程における確率的な関係をほぼ決定的な場合とほぼ一様にランダムな場合の二つの極端な組合せに変換する操作である。情報量の分極操作は、多端子符号化問題を含め多くの符号化問題の理論的限界を達成する符号化法を低計算複雑度と低空間複雑度で与える基本的な一つの原理をもたらすことが期待される。

本稿では、2. に情報理論における固定長情報源符号化問題と固定長通信路符号化問題の設定について述べ、理論限界を示す情報源符号化定理と通信路符号化定理を確認する⁽⁵⁾⁽⁶⁾。情報源符号化定理と通信路符号化定理の証明として、典型集合⁽⁷⁾⁽⁸⁾を用いる方法がある。典型集合の系列を直接的に操作する符号化と復号の方法は取り扱う集合の要素数が大きくなり計算複雑度や空間複雑度において困難を伴う。これに対して、Arikan により提案された固定長情報源符号は符号長 n に対して符号化と復号の計算複雑度が $O(n \log n)$ であり、符号化の空間複雑度は $O(n)$ 、復号の空間複雑度は $O(n \log n)$ である。情報源分極は再帰的な可逆変換による次数の拡大変換と情報量のチェインルールによる分解で構成されていることを 3. で確認する。3. で紹介した情報源分極に基づく固定長情報源符号について 4. に述べる。固定長通信路符号については、通信路分極を 5. で確認し、確認した通信路分極に基づく固定長通信路符号について 6. で紹介する^(注1)。C 言語による polar 符号のプログラミング例を 7. に紹介する。情報源分極に基づく情報源符号の例が 7.1 にあり、通信路分極に基づく通信路符号の例が 7.2 と 7.3 にある。最後にむすびを 8. に述べる。

2. 固定長符号化の問題設定

本稿で紹介する polar 符号は、一定の長さの入力系列を一定の長さの符号語に変換する固定長符号のクラスに属する符号である。本章では韓の教科書⁽⁵⁾を参考にして、固定長符号のクラスにおける固定長情報源符号化問題と固定長通信路符号化問題について定式化するとともに、その理論的限界を明確にする符号化定理を述べる。

図1 情報源符号化における情報通信のモデル

2.1 固定長情報源符号化問題

まず、図1にシステムの概略を示す最も簡単な情報源符号化の一つである固定長符号化について考える。定常無記憶情報源を集合 $S = \{0, 1\}$ 上の同一の確率分布 $(P_S(0), P_S(1)) = (1-p, p)$ に従う互いに独立な確率変数の列によりモデル化する。情報源からの長さ n の出力を表す確率変数列を $S_0^{n-1} = (S_0, S_1, \dots, S_{n-1}) \in S^n$ と表記し、確率変数列 S_0^{n-1} が値として $s_0^{n-1} = (s_0, s_1, \dots, s_{n-1}) \in \{0, 1\}^n$ を取る確率を $P_{S_0^{n-1}}(s_0^{n-1})$ と表記する^(注2)。本稿ではCプログラミングを考慮に入れて添え字は0から始める。

定常無記憶情報源からの出力系列 $s_0^{n-1} \in S^n$ は $P_{S_0^{n-1}}(s_0^{n-1}) = \prod_{i=0}^{n-1} P_S(s_i)$ を満たす。情報源符号として固定長符号を用い、固定長符号を符号化写像 (符号器) $f_n: \{0, 1\}^n \rightarrow \{0, 1\}^k$ と復号写像 (復号器) $g_n: \{0, 1\}^k \rightarrow \{0, 1\}^n$ によって定める。情報源からの長さ n の出力 s_0^{n-1} に対する $f_n(s_0^{n-1}) \in \{0, 1\}^k$ を s_0^{n-1} の符号語と呼び、 k を符号 (f_n, g_n) の符号長と呼ぶ。本章では、通信路には雑音がなく符号語が正しく伝送されることを仮定する。受信側では、符号語 $f(s_0^{n-1})$ から情報源からの出力である二元系列 s_0^{n-1} をひずみなく復元することを試みる。

符号 (f_n, g_n) では復号により必ずしも情報源出力系列 s_0^{n-1} を正しく復元できず、符号 (f_n, g_n) の誤り率 $\varepsilon_n^{(s)}(f_n, g_n)$ と符号化レート $r_n^{(s)}(f_n, g_n)$ を

$$\varepsilon_n^{(s)}(f_n, g_n) = \sum_{\substack{s_0^{n-1} \in \{0,1\}^n \\ g_n(f_n(s_0^{n-1})) \neq s_0^{n-1}}} P_{S_0^{n-1}}(s_0^{n-1}),$$

$$r_n^{(s)}(f_n, g_n) = \frac{k}{n}$$

と定め、固定長符号化の性能の評価を行う。情報源符号化における固定長符号化では、符号の誤り確率 $\varepsilon_n^{(s)}(f_n, g_n)$ を0に近づけ、符号長 k を短くし符号化レート $r_n^{(s)}(f_n, g_n)$ を可能な限り小さくしたい。この問題は確率分布 P_S でモデル化される定常無記憶情報源に対して定義1で定める最小許容固定長符号化レート $R_s(P_S)$ を求める問題として定式化される。

[定義1]

$$R_s(P_S) = \inf\{R: \text{符号化レート } R \text{ が達成可能}\}.$$

ただし、符号化レート R が達成可能とは $\lim_{n \rightarrow \infty} \varepsilon_n^{(s)}(f_n, g_n) = 0$ と $\limsup_{n \rightarrow \infty} r_n^{(s)}(f_n, g_n) \leq R$ を満たす符号の列 $\{(f_n, g_n)\}$ が存在することである。□

(注1): 無ひずみ情報源符号と対称通信路における通信路符号の関係から 3. と 4. での情報源符号が 5. と 6. での通信路符号のどちらか片方の論理展開をもう片方の論理展開に帰着できる。

(注2): 本稿では一般に確率変数 S が取る値の集合を S と表し、確率変数 S が従う確率分布を P_S で表し、 $P_S(s) = \Pr\{S = s\}, s \in S$ とする。

図2 通信路符号化における情報通信のモデル

情報源符号化における固定長符号化問題について定理1が成り立つ。

[定理1](Shannon⁽¹⁾)

$$R_s(P_S) = H(S).$$

ただし、 $H(S)$ は確率変数 S のエントロピーを表し、 S が従う確率分布 P_S を用いて $H(S) = -\sum_{s \in S} P_S(s) \log P_S(s)$ と定める。なお、 $0 \log_2 0$ は0と定義する。本稿を通して特に明記しない場合、 $\log$ の底は2とする。

定理1の証明は韓の教科書⁽⁵⁾の定理1.1や植松による教科書⁽⁸⁾の定理1.5と定理1.6に述べられている。これらの証明では典型集合を用いて情報源符号を構成している。

4.で紹介するArikanによる情報源分極に基づく固定長符号 $(f_n^{(A)}, g_n^{(A)})$ は次の性質を満たす符号である。

- 符号長 n に対して符号化と復号が $O(n \log n)$ の計算複雑度で実装可能である。
- 与えられた定常無記憶情報源に対して適切な符号の設計を行い、符号化レート $r_n^{(s)}(f_n^{(A)}, g_n^{(A)})$ が定常無記憶情報源のエントロピー $H(S)$ よりも大きければ、任意に固定した $0 < \delta < \frac{1}{2}$ に対して、誤り確率 $\varepsilon_n^{(s)}(f_n^{(A)}, g_n^{(A)})$ は $o\left(2^{-n^{(\frac{1}{2}-\delta)}}\right)$ であり、符号長 n を大きくした極限值は0に収束する。

2.2 固定長通信路符号化問題

次に、図2にシステムの概略を示す最も簡単な通信路符号化の一つである固定長符号化について考える。

通信路への入力記号の集合を $\mathcal{X} = \{0, 1\}$ とし、通信路からの出力記号の集合を $\mathcal{Y} = \{0, 1, \dots\}$ とする。 $\mathcal{X}$ の要素数は2であり、 $\mathcal{Y}$ の要素数は2以上の整数とし、 $|\mathcal{Y}|$ で表す。以下、集合 $\mathcal{E}$ に対して $|\mathcal{E}|$ によりその要素数を表す。記号 $x \in \mathcal{X}$ が通信路に入力されたとき、記号 $y \in \mathcal{Y}$ が通信路から出力される条件付確率 $P_{Y|X}(y|x)$ を $W(y|x)$ と表し、確率遷移行列 $W = \{W(y|x)\}_{x \in \mathcal{X}, y \in \mathcal{Y}}$ により定常無記憶通信路をモデル化する。通信路への入力系列 $x_0^{n-1} = (x_0, x_1, \dots, x_{n-1}) \in \mathcal{X}^n$ が与えられたとき、通信路から出力系列 $y_0^{n-1} = (y_0, y_1, \dots, y_{n-1}) \in \mathcal{Y}^n$ が出現する条件付確率 $P_{Y_0^{n-1}|X_0^{n-1}}(y_0^{n-1}|x_0^{n-1})$ を $W^n(y_0^{n-1}|x_0^{n-1})$ と表し、 $W^n(y_0^{n-1}|x_0^{n-1})$ が $\prod_{i=0}^{n-1} W(y_i|x_i)$ で定まる通信路を定常無記憶通信路と呼び、定常無記憶通信路 W 若しくは単に通信路 W と表す。

通信路符号として固定長符号を用い、固定長符号を符号化写像 (符号器) $\varphi_n : \{0, 1\}^k \rightarrow \{0, 1\}^n$ と復号写像 (復号器) $\psi_n : \mathcal{Y}^n \rightarrow \{0, 1\}^k$ によって定める。伝送する k ビットのメッ

セージ $m_0^{k-1} \in \{0, 1\}^k$ に対して符号語 $\varphi_n(m_0^{k-1}) \in \{0, 1\}^n$ を通信路に入力する。受信側では、入力した符号語 $\varphi_n(m_0^{k-1}) \in \{0, 1\}^n$ に対応して通信路から出力された $y_0^{n-1} \in \mathcal{Y}^n$ を受け取り、復号器により $\psi_n(y_0^{n-1}) \in \{0, 1\}^k$ を復号の結果とする。このとき、 n を符号 (φ_n, ψ_n) の符号長と呼ぶ。符号 (φ_n, ψ_n) では復号により必ずしも伝送した k ビットのメッセージ m_0^{k-1} を正しく復元できず、符号 (φ_n, ψ_n) の誤り率 $\varepsilon_n^{(c)}(\varphi_n, \psi_n)$ と符号化レート $r_n^{(c)}(\varphi_n, \psi_n)$ を

$$\begin{aligned} \varepsilon_n^{(c)}(\varphi_n, \psi_n) &= \frac{1}{2^k} \sum_{m_0^{k-1} \in \{0, 1\}^k} \sum_{\substack{y_0^{n-1} \in \mathcal{Y}^n, \\ \psi_n(y_0^{n-1}) \neq m_0^{k-1}}} W^n(y_0^{n-1} | \varphi_n(m_0^{k-1})), \\ r_n^{(c)}(\varphi_n, \psi_n) &= \frac{k}{n} \end{aligned}$$

と定め、固定長符号化の性能の評価を行う。ここで、 $\varepsilon_n^{(c)}(\varphi_n, \psi_n)$ は、伝送する 2^k 個のメッセージが等確率で生起することを仮定した平均復号誤り率である。通信路符号化における固定長符号化では、符号の誤り確率 $\varepsilon_n^{(c)}(\varphi_n, \psi_n)$ を0に近づけ、メッセージ長 k を長くして符号化レート $r_n^{(c)}(\varphi_n, \psi_n)$ を可能な限り大きくしたい。この問題は確率遷移行列 W でモデル化される定常無記憶通信路に対して定義2で定まる通信路容量 $R_c(W)$ を求める問題として定式化される。

[定義2]

$$R_c(W) = \sup\{R : \text{符号化レート } R \text{ が達成可能}\}.$$

ただし、符号化レート R が達成可能とは $\lim_{n \rightarrow \infty} \varepsilon_n^{(c)}(\varphi_n, \psi_n) = 0$ と $\liminf_{n \rightarrow \infty} r_n^{(c)}(\varphi_n, \psi_n) \geq R$ を満たす符号の列 $\{(\varphi_n, \psi_n)\}$ が存在することである。□

定常無記憶通信路での通信路符号化における固定長符号化問題について定理2が成り立つ。

[定理2](Shannon⁽¹⁾)

$$R_c(W) = \max_{P_X} I(X; Y).$$

ただし、 $I(X; Y)$ は、確率変数 X と Y の相互情報量を表し、 X が従う通信路への入力確率分布 P_X と通信路をモデル化する W を用いて

$$I(X; Y) = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} P_X(x) W(y|x) \log \frac{W(y|x)}{P_Y(y)},$$

$P_Y(y) = \sum_{x \in \mathcal{X}} P_X(x) W(y|x)$ と定める。

定理2の証明は韓の教科書⁽⁵⁾の定理3.1、CoverとThomasによる教科書⁽⁶⁾の定理7.7.1、植松による教科書⁽⁸⁾の定理1.8と定理4.8に述べられている。これらの証明では、同時典型集合を用いて通信路容量を達成する通信路符号の存在を示している。

6.で紹介するArikanによる通信路分極に基づく固定長符号 $(\varphi_n^{(A)}, \psi_n^{(A)})$ は次の性質を満たす符号である。

- 符号長 n に対して符号化と復号が $O(n \log n)$ の計算複雑度で実装可能である。
- 与えられた定常無記憶通信路に対して適切な符号の設計を行い、符号化レート $r_n^{(c)}(\varphi_n^{(A)}, \psi_n^{(A)})$ が定常無記憶通信路

W の対称通信路容量 $I(W)$ よりも小さければ、任意に固定した $0 < \delta < \frac{1}{2}$ に対して、平均ブロック誤り確率 $\varepsilon_n^{(c)}(\varphi_n^{(A)}, \psi_n^{(A)})$ は $o\left(2^{-n^{(\frac{1}{2}-\delta)}}\right)$ であり、符号長 n を大きくした極限値は 0 に収束する。

ここで、対称通信路容量 $I(W)$ は

$$\begin{aligned} I(W) &= \frac{1}{2} \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} W(y|x) \log \frac{W(y|x)}{\frac{1}{2} \sum_{x' \in \mathcal{X}} W(y|x')} \\ &= 1 + \frac{1}{2} \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} W(y|x) \log \frac{W(y|x)}{\sum_{x' \in \mathcal{X}} W(y|x')} \end{aligned} \quad (1)$$

で定義される通信路 W への入力確率分布 P_X を一様分布とした相互情報量 $I(X; Y)$ である。

3. 情報源の再帰的な拡大変換と情報量のチェーンルールによる情報源分極

本稿では、情報源の出力記号が 0 か 1 の二元情報源で説明を行う。情報源として確率変数 $S \in \{0, 1\}$ でモデル化される定常無記憶情報源を考え、確率変数 S が従う確率分布 P_S を $0 \leq p \leq 1$ なる実数 p を用いて $P_S(0) = 1-p$ と $P_S(1) = p$ とする。このとき、確率変数 S のエントロピー $H(S) = -p \log p - (1-p) \log(1-p)$ であり、以下では、 $h(p) = -p \log p - (1-p) \log(1-p)$ と表記する。

情報源分極の操作は、情報源からの出力系列である確率変数列 $S_0^{n-1} = (S_0, S_1, \dots, S_{n-1}) \in S^n$ に対して、再帰的な可逆変換により別の確率変数列 $U_0^{n-1} = (U_0, U_1, \dots, U_{n-1}) \in U^n$ に変換することを考える。まず、二次拡大した情報源 S^2 からの出力 $S_0^1 = (S_0, S_1) \in S^2$ に対しては線形変換 T を

$$T = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \quad (2)$$

と定めて、

$$(U_0, U_1) = (S_0, S_1)T = (S_0 \oplus S_1, S_1)$$

を考える。加法 $\oplus$ は mod 2 の加法である。このとき、 (S_0, S_1) から (U_0, U_1) への行列 T による線形変換は全単射であり、 S_0, S_1 のエントロピー $H(S_0, S_1) = -\sum_{s_0^1 \in \{0,1\}^2} P_{S_0^1}(s_0^1) \log P_{S_0^1}(s_0^1)$ と U_0, U_1 のエントロピー $H(U_0, U_1)$ は $H(S_0, S_1) = H(U_0, U_1)$ を満たす^(注3)。また、定常無記憶情報源であることから $H(S_0, S_1) = 2H(S)$ である。いま、 $2H(S) = H(U_0, U_1) = H(U_0) + H(U_1|U_0) \leq H(U_0) + H(U_1) = H(U_0) + H(S_1) = H(U_0) + H(S)$ より、

$$H(U_0) \geq H(S) \geq H(U_1|U_0) \quad (3)$$

が導かれる。ただし、条件付きエントロピー $H(U_1|U_0)$ は $H(U_1|U_0) = H(U_0, U_1) - H(U_0)$ で定められる。図 3 に式 (3) の $H(S)$ と $H(U_0), H(U_1|U_0)$ の関係を示す。式 (3) の等号成立条件は $p = 0, \frac{1}{2}, 1$ である。ここまでの操作を次の (操作

図 3 $n = 2$ での $H(S)$ と $H(U_0), H(U_1|U_0)$ の関係

a)~(操作 c) にまとめる。

(操作 a) 同じ確率分布に従う確率変数 S を 2 個用意し (S_0, S_1) と二次拡大を行う。

(操作 b) 二次拡大した (S_0, S_1) に可逆な線形変換 T を行い、 $(S_0 \oplus S_1, S_1)$ を得る。

(操作 c) $H(S_0 \oplus S_1, S_1)$ をチェーンルールにより $H(S_0 \oplus S_1) + H(S_1|S_0 \oplus S_1)$ と展開する。

この操作が情報量の分極操作の基本的操作であり式 (3) の関係が示すように元の情報量 $H(S)$ に対して情報量の大きくなる部分 $H(U_0) = H(S_0 \oplus S_1)$ と情報量の小さくなる部分 $H(U_1|U_0) = H(S_1|S_0 \oplus S_1)$ に分解できる。更に、分解した情報量 $H(S_0 \oplus S_1)$ と $H(S_1|S_0 \oplus S_1)$ の和は元の情報量 $H(S)$ の 2 倍に等しい。すなわち、

$$H(S_0 \oplus S_1) \geq H(S_0) = H(S_1) \geq H(S_1|S_0 \oplus S_1) \quad (4)$$

$$H(S) = H(S_0) = \frac{1}{2}H(S_0 \oplus S_1) + \frac{1}{2}H(S_1|S_0 \oplus S_1) \quad (5)$$

が成り立つ。

(操作 a) から (操作 c) を再帰的に行うことを考えるため、 (S_0, S_1) と (U_0, U_1) の関係を回路図を用いて図 4 の左図に示す。四次拡大の $S_0^3 = (S_0, S_1, S_2, S_3)$ を考えるには、同じ確率分布に従う確率変数の組 $(S_0 \oplus S_1, S_2 \oplus S_3)$ 、 $H(S_0 \oplus S_1) = H(S_2 \oplus S_3)$ に対して (操作 b) と (操作 c) を行う。同様に $H(S_1|S_0 \oplus S_1) = H(S_3|S_2 \oplus S_3)$ を満たす同じ確率分布に従う確率変数の組 (S_1, S_3) に対して (操作 b) と (操作 c) を行う。これらの関係を回路図で示すと図 4 の右図が得られる。 $n = 2^2$ で $U_0^3 = (U_0, U_1, U_2, U_3)$ を

$$U_0 = S_0 \oplus S_1 \oplus S_2 \oplus S_3,$$

$$U_1 = S_2 \oplus S_3,$$

$$U_2 = S_1 \oplus S_3,$$

$$U_3 = S_3$$

(注3): 例えば、Cover と Thomas による教科書⁽⁶⁾の演習問題 2.5 を参照。

図 4 回路表現, 左: $n = 2^1$, 右: $n = 2^2$

図 5 $n = 4$ での $H(U_i|U_0^{i-1}), i \in \mathcal{I}_{2^2}$ の関係

と定める．このとき，式 (4) と同様に

$$H(U_0) \geq H(S_0 \oplus S_1) = H(S_2 \oplus S_3) \geq H(U_1|U_0), \quad (6)$$

$$\begin{aligned} H(U_2|U_0^1) &\geq H(S_1|S_0 \oplus S_1) \\ &= H(S_3|S_2 \oplus S_3) \geq H(U_3|U_0^2) \end{aligned} \quad (7)$$

が得られる．分解した情報量の和は元の情報量 $H(S_0 \oplus S_1), H(S_1|S_0 \oplus S_1)$ の 2 倍に等しく，式 (5) と同様に

$$H(S_0 \oplus S_1) = \frac{1}{2}H(U_0) + \frac{1}{2}H(U_1|U_0) \quad (8)$$

$$H(S_1|S_0 \oplus S_1) = \frac{1}{2}H(U_2|U_0^1) + \frac{1}{2}H(U_3|U_0^2) \quad (9)$$

が成り立つ．

$H(U_0), H(U_i|U_0^{i-1}), i = 1, 2, 3$ のグラフを図 5 に示す．以降では，表記の簡単化のため $H(U_0|U_0^{-1})$ により $H(U_0)$ を表し，集合 $\{0, 1, \dots, 2^m - 1\}$ を $\mathcal{I}_{2^m}$ と表記する．更に，再帰的に同じ分布に従う 2 組に対して (操作 a) から (操作 c) を繰返して，八次拡大の $S_0^7 = (S_0, S_1, \dots, S_7)$ を考えると，図 6 の回路図と $H(U_i|U_0^{i-1}), i \in \mathcal{I}_{2^3}$ について図 7 のグラフが得られる．図 7 のグラフにおいて $p = 0.1$ のとき， $H(U_4|U_0^3) > H(U_3|U_0^2)$ であり，インデックス i と情報量 $H(U_i|U_0^{i-1})$ の大小関係の順序において順番の入れ替わりが生じることが分かる．後の図 9, 10 で確認できるように，この大小関係の順序は一般には p の値に応じて入れ替わる可能性がある． $n = 2^4$ の場合は $p \approx 0.18488, 0.81512$ でこの大小関係の順序が入れ替わる．このことを 7.1 に示す C 言語によるプログラミング例でも用いて

図 6 $n = 2^3$ の場合の回路表現

図 7 $n = 8$ での $H(U_i|U_0^{i-1}), i \in \mathcal{I}_{2^3}$ の関係

図 8 $n = 2^m, m = 2, 3$ の場合の回路の別表現

いる．

図 4 の右図と図 6 の回路図を $U_i, i = 0, 1, \dots, 7$ の配置を換え書き直すと図 8 が得られる．図 8 の回路の変換を行列 T と行列のクロネッカー積 $\otimes$ を用いて表現すると， $n = 2^2$ の場合は $T^{\otimes 2}$ であり， $n = 2^3$ の場合は $T^{\otimes 3}$ と表現できる．ここで， $T^{\otimes 1} = T$ とし， $T^{\otimes m}$ を再帰的に $T^{\otimes m} = T \otimes T^{\otimes m-1}$ ， $m \geq 2$ と定義する．拡大次数を 2 のべき乗とし，その拡

大次数を n で表し, 1 以上の自然数 m を用いて $n = 2^m$ とする. また, $U_0^{n-1} = (U_0, U_1, \dots, U_{n-1})$ の U_i に対して i の m ビットの 2 進展開を $(b_{m-1}b_{m-2}\dots b_0)_2$ とし, $U_i = U_{(b_{m-1}b_{m-2}\dots b_0)_2}$ と表記すると, $\hat{U}_0^{n-1} = U_0^{n-1}B_n$ により $\hat{U}_{(b_{m-1}b_{m-2}\dots b_0)_2} = U_{(b_0b_1\dots b_{m-1})_2}$ を満たす n 行 n 列の置換行列を B_n , $n = 2^m$ で表現する. つまり, B_n , $n = 2^m$ はインデックスの 2 進展開におけるビット反転による置換を行う行列とする. このとき, B_{2^m} と $T^{\otimes m}$ を用いて, S_0^{n-1} と U_0^{n-1} の関係は

$$U_0^{n-1} = S_0^{n-1} T^{\otimes m} B_{2^m} \quad (10)$$

と表記できる. 以下では, G_n を

$$G_n = T^{\otimes m} B_{2^m} \quad (11)$$

と定める^(注4). なお, $G_n G_n$ は単位行列 E_n であり, G_n の逆行列は G_n である.

定常無記憶情報源 S の拡大次数を n とし, 式 (10) を考えると, G_n は可逆変換による変換であり, $H(S_0^{n-1}) = H(U_0^{n-1})$ を満たす. 更に $H(U_0^{n-1})$ のチェーンルールによる情報量の分解

$$H(U_0^{n-1}) = \sum_{i=0}^{n-1} H(U_i|U_0^{i-1})$$

を行った結果における情報量 $H(U_i|U_0^{i-1})$, $i = 0, \dots, n-1$ の様子を図 9, 10 と図 11 に示す. 図 9 では $P_S(1) = p = 0.11$ であり, 図 10 では $P_S(1) = p = 0.01$ である. 図 9 と 10 は拡大次数は $n = 2^7$ であり, 各 $i \in \mathcal{I}_{2^7}$ における $H(U_i|U_0^{i-1})$ の値をプロットした. p の値により $H(U_i|U_0^{i-1})$ の値をプロットしたグラフの様相が異なることが見て取れる. 図 11 は $p = 0.11$, $n = 2^m$, $m = 7, 10, 13, 16, 21$ において, 横軸を $0 \leq a \leq 1$ とし, $H(U_i|U_0^{i-1})$ の情報量が a 以下となるインデックス i の割合 $|\{i \in \mathcal{I}_{2^m} : H(U_i|U_0^{i-1}) \leq a\}|/2^m$ を近似したグラフである^(注5). 拡大次数 n を大きくするにつれて情報量 $H(U_i|U_0^{i-1})$ がより分極していることが見て取れ, $H(U_i|U_0^{i-1})$ が 0 と 1 に漸近する割合がそれぞれ $1 - H(S)$, $H(S) = h(0.11) \approx 0.4999$ に近づいているように見える.

$H(U_i|U_0^{i-1})$, $i \in \mathcal{I}_{2^m}$ は, $m \rightarrow \infty$ の極限として, 情報量の分極操作が可能であり, 次の定理 3 が成り立つ.

[定理 3] (文献(4) Theorem 1 で補助情報がない場合) 確率変数 S でモデル化される定常無記憶情報源 $\{S_i\}_{i=0}^\infty$ に対して, $U_0^{n-1} = S_0^{n-1} G_n$, $n = 2^m$, $m = 1, 2, \dots$ と定める. このとき, 任意の $0 < \epsilon < 1$ に対して,

$$\lim_{m \rightarrow \infty} \frac{|\{i \in \mathcal{I}_{2^m} : H(U_i|U_0^{i-1}) \in (1 - \epsilon, 1]\}|}{2^m} = H(S),$$

$$\lim_{m \rightarrow \infty} \frac{|\{i \in \mathcal{I}_{2^m} : H(U_i|U_0^{i-1}) \in [0, \epsilon]\}|}{2^m} = 1 - H(S)$$

を満たす.

(注4): 本稿では, G_n を式 (11) で定義したが, G_n を $G_n = T^{\otimes m}$ と定義することも可能である. $G_n = T^{\otimes m}$ と定義した場合は, S_0^{n-1} におけるインデックスの付け直しを考える. 例えば, $m = 3$ の例では, $S_0^7 = (S_0, S_1, \dots, S_7)$ に対して $(S_0, S_4, S_2, S_6, S_1, S_5, S_3, S_7)$ とインデックスの付け直しを考える.

(注5): 十分大きな m において $H(U_i|U_0^{i-1})$ を正確に計算することは計算複雑度や空間複雑度からの困難を伴い, 図 11 での $m = 10, 13, 16, 21$ における $H(U_i|U_0^{i-1})$ は近似値を用いた.

図 9 $p = 0.11$ での $H(U_i|U_0^{i-1})$, $i \in \mathcal{I}_{2^7}$

図 10 $p = 0.01$ での $H(U_i|U_0^{i-1})$, $i \in \mathcal{I}_{2^7}$

図 11 $n = 2^m$, $m = 7, 10, 13, 16, 21$ で $H(U_i|U_0^{i-1})$ の情報量が a 以下となるインデックス i の割合 $|\{i \in \mathcal{I}_{2^m} : H(U_i|U_0^{i-1}) \leq a\}|/2^m$ を近似したグラフ

定理 3 は, $H(U_i|U_0^{i-1})$, $i \in \mathcal{I}_{2^m}$ を $m \rightarrow \infty$ とともに情報量が 1 に漸近する $H(U_i|U_0^{i-1})$ と情報量が 0 に漸近する $H(U_i|U_0^{i-1})$ に二分化できることを示しており, 情報量が 1 に漸近する割合が $H(S)$ であり, 情報量が 0 に漸近する割合が $1 - H(S)$ であることを述べている. この現象を Arıkan は「情報源分極」と呼んでいる. 定理 3 の証明は文献(9)の定理 3.1 に補助情報がある場合で述べられており, 5. で紹介する定理

図 12 式 (4) から式 (9) に続く繰返しによる確率過程
 $\{H_j, j = 0, 1, 2, \dots, m, \dots\}$

5 の証明と同じ論理展開により証明される．定理 5 の証明の概要については和書として文献 (10)~(12) がある．Arıkan が最初に示した基本的な証明の流れではマルチンゲールの収束定理を用いた．図 12 に示すように，式 (4) と式 (5) に対応して左端の節点 $H(S)$ から $H(S) > H(S_0 \oplus S_1)$ である右の節点 $H(S_0 \oplus S_1)$ に確率 $\frac{1}{2}$ で移動し， $H(S) < H(S_0 \oplus S_1)$ である右の節点 $H(S_1 | S_0 \oplus S_1)$ に確率 $\frac{1}{2}$ で移動することを考える．同様に式 (6) から式 (9) に対応して節点 $H(S_0 \oplus S_1 \oplus S_2 \oplus S_3)$ から節点 $H(S_3 | S_0 \oplus S_1 \oplus S_2 \oplus S_3, S_2 \oplus S_3, S_1 \oplus S_3)$ のいずれかの節点に等確率で移動すると考える．この等確率による $j = 1, 2, \dots$ 回の移動において， j 番目の情報量を値にとる確率変数を H_j で表す．ただし， $H_0 = H(S)$ とする．このとき，確率過程 $\{H_j, j = 0, 1, \dots, m-1, \dots\}$ は有界な離散時間マルチンゲールの性質を満たす．

もし， $H(U_i | U_0^{i-1}) = 0$ であれば， U_0^{i-1} の値 u_0^{i-1} から U_i の値 u_i を正しく求められる．それゆえ，符号器では $H(U_i | U_0^{i-1}) = 0$ である U_i の値を送信せず $H(U_i | U_0^{i-1}) \neq 0$ である U_i のみを送信し，復号器では $H(U_i | U_0^{i-1}) = 0$ である U_i の値を推定することで u_0^{n-1} を正しく推定できる． u_0^{n-1} が既知であれば G_n の逆変換を行うことで s_0^{n-1} を求められる．この原理により符号化レートを $R_s(P_S) = H(S)$ に上から近づけても誤り率が 0 に漸近する情報源符号化が可能である．情報源分極に基づき Arıkan⁽⁴⁾ が提案した情報源符号化を次章で紹介する．

4. 情報源分極に基づく情報源符号

確率変数 S によりモデル化される定常無記憶情報源 $\{S_i\}_{i=0}^{\infty}$ に対して， $n = 2^m, m = 1, 2, \dots$ として，情報源分極に基づく情報源符号の符号化写像 $f_n^{(A)} : \{0, 1\}^n \rightarrow \{0, 1\}^k$ と復号写像

$g_n^{(A)} : \{0, 1\}^k \rightarrow \{0, 1\}^n$ を考える．まず， n と k は符号器と復号器で既知と仮定する．定常無記憶情報源 $\{S_i\}_{i=0}^{\infty}$ の S_0^{n-1} に対して， $U_0^{n-1} = S_0^{n-1} G_n$ を考え， $H(U_i | U_0^{i-1}), i \in \mathcal{I}_n$ の値の大きい k 個のインデックス i からなる集合 $\mathcal{E} \subset \mathcal{I}_n$ を定める．このとき， $\mathcal{E} \subset \mathcal{I}_n$ は次の条件 1, 2 を満たす．

条件 1 $\forall i \in \mathcal{E}, \forall j \in \mathcal{E}^c$ に対して $H(U_i | U_0^{i-1}) \geq H(U_j | U_0^{j-1})$ ． $\mathcal{E}^c$ は $\mathcal{I}_n$ に対する $\mathcal{E}$ の補集合である．

条件 2 $|\mathcal{E}| = k$ ．

このインデックス集合 $\mathcal{E} \subset \mathcal{I}_n$ を符号器と復号器は既知と仮定する^(注6)．

情報源からの出力系列 $s_0^{n-1} \in \{0, 1\}^n$ に対して，符号器では， $u_0^{n-1} = s_0^{n-1} G_n$ を求め，符号語 $f_n^{(A)}(s_0^{n-1}) = u_{\mathcal{E}}$ とする．ただし $u_{\mathcal{E}}$ は u_0^{n-1} のうち u_i が $i \in \mathcal{E}$ を満たす要素のみを i の小さい順番に並べ直した系列である．例えば， $n = 8, k = 4, p = 0.1$ の場合，図 7 より $\mathcal{E} = \{0, 1, 2, 4\}$ であり，情報源からの出力系列 s_0^7 に対して $u_0^7 = s_0^7 G_8$ を求め， s_0^7 に対する符号語を $f_8^{(A)}(s_0^7) = (u_0, u_1, u_2, u_4)$ とする．

polar 符号の長所である符号長 n に対して符号化が $O(n \log n)$ の計算複雑度であることを確認する．ここでの計算複雑度は XOR 演算の回数に起因する．符号長 n の符号器 $f_n^{(A)}$ の計算複雑度を $\Lambda_E(n)$ で表す．符号長 2 の符号器 $f_2^{(A)}$ の計算複雑度は，図 4 の左図に示すように $s_0^1 G_2$ では 1 回の XOR 演算で可能であり， $\Lambda_E(2) = 1$ である． $s_0^{n-1} G_n, n = 2^m$ は m 回の再帰的な結合の繰返しで構成でき，計算複雑度 $\Lambda_E(n)$ について次の関係式が導かれる．

$$\begin{aligned} \Lambda_E(n) &= \frac{n}{2} + 2\Lambda_E\left(\frac{n}{2}\right) \\ &= \frac{n}{2} + 2\left(\frac{n}{4} + 2\Lambda_E\left(\frac{n}{4}\right)\right) \\ &= \frac{n}{2} + \frac{n}{2} + 4\left(\frac{n}{8} + 2\Lambda_E\left(\frac{n}{8}\right)\right) \\ &= \frac{nm}{2} \end{aligned} \quad (12)$$

よって，計算複雑度 $\Lambda_E(n) = O(n \log n)$ である．例えば， $n = 2^3$ ならば図 8 の右図を縦に $m = 3$ 分割し， $\frac{n}{2} = 4$ 回の XOR 演算の層が 3 層あるとみなすことができる．また，各層ごとに XOR 演算を行えば空間複雑度は $O(n)$ で実装可能である．7. に符号長 $n = 2^4$ の符号器 $f_{16}^{(A)}$ の C 言語によるプログラミング例を示す．

符号語 $f_n^{(A)}(s_0^{n-1}) \in \{0, 1\}^k$ を受け取った復号器では U_0^{n-1} のうち $U_i, i \in \mathcal{E}$ の値を既知であり，未知である $U_i, i \in \mathcal{E}^c$ に対して i の小さい順に逐次的に推定を行う．具体的には， U_0^{i-1} が $U_0^{i-1} = \hat{u}_0^{i-1}$ であると仮定したもとで U_i のゆわ度比に基づき逐次的に行う．すなわち， $\hat{u}_0^{n-1}$ を $i = 0, 1, \dots, n-1$ の順に

$$\hat{u}_i = \begin{cases} u_i & \text{if } i \in \mathcal{E}, \\ 0 & \text{if } i \in \mathcal{E}^c \text{ かつ } D_{2^m}^{(i)}(\hat{u}_0^{i-1}) \geq 1, \\ 1 & \text{if } i \in \mathcal{E}^c \text{ かつ } D_{2^m}^{(i)}(\hat{u}_0^{i-1}) < 1 \end{cases} \quad (13)$$

と定める．ここで $D_{2^m}^{(i)}(\hat{u}_0^{i-1})$ は， $P_{U_i | U_0^{i-1}}(0 | \hat{u}_0^{i-1})$ と

(注6)：なお，符号器と復号器の実装において大きな n で $\mathcal{E}$ を正確に求めることに計算複雑度や空間複雑度からの困難を伴う場合は近似計算を用いる．

$P_{U_i|U_0^{i-1}}(1|\hat{u}_0^{i-1})$ の比

$$D_{2^m}^{(i)}(\hat{u}_0^{i-1}) = \frac{P_{U_i|U_0^{i-1}}(0|\hat{u}_0^{i-1})}{P_{U_i|U_0^{i-1}}(1|\hat{u}_0^{i-1})}$$

で定義される． $D_{2^m}^{(i)}(\hat{u}_0^{i-1})$ は i の偶奇で場合分けを行い，文献 (4) で記述されている次の再帰式で計算できる．

$$D_{2^m}^{(2i)}(\hat{u}_0^{2i-1}) = \frac{1 + D_{2^{m-1}}^{(i)}(\hat{u}_{0,\text{偶}}^{2i-1} \oplus \hat{u}_{0,\text{奇}}^{2i-1}) D_{2^{m-1}}^{(i)}(\hat{u}_{0,\text{奇}}^{2i-1})}{D_{2^{m-1}}^{(i)}(\hat{u}_{0,\text{偶}}^{2i-1} \oplus \hat{u}_{0,\text{奇}}^{2i-1}) + D_{2^{m-1}}^{(i)}(\hat{u}_{0,\text{偶}}^{2i-1})}, \quad (14)$$

$$D_{2^{m+1}}^{(2i+1)}(\hat{u}_0^{2i}) = \begin{cases} D_{2^{m-1}}^{(i)}(\hat{u}_{0,\text{偶}}^{2i} \oplus \hat{u}_{0,\text{奇}}^{2i}) D_{2^{m-1}}^{(i)}(\hat{u}_{0,\text{奇}}^{2i}) & \text{if } \hat{u}_{2i} = 0, \\ \frac{D_{2^{m-1}}^{(i)}(\hat{u}_{0,\text{偶}}^{2i})}{D_{2^{m-1}}^{(i)}(\hat{u}_{0,\text{偶}}^{2i} \oplus \hat{u}_{0,\text{奇}}^{2i})} & \text{if } \hat{u}_{2i} = 1. \end{cases} \quad (15)$$

ただし， $\hat{u}_0^{-1}$ は空列 λ を表し， $D_{2^0}^{(0)}(\lambda) = \frac{P_S(0)}{P_S(1)}$ とする． $u_{0,\text{偶}}^{2i-1}$ は u_0^{2i-1} における偶数インデックスの要素で構成される $u_{0,\text{偶}}^{2i-1} = (u_0, u_2, \dots, u_{2i-2})$ とし， $u_{0,\text{奇}}^{2i-1}$ は u_0^{2i-1} における奇数インデックスの要素で構成される $u_{0,\text{奇}}^{2i-1} = (u_1, u_3, \dots, u_{2i-1})$ とする．式 (14) と式 (15) の再帰式は通信路符号における式 (28) と式 (29) の再帰式と対応しており，式 (28) と式 (29) の再帰式は補題 1 の式 (22) と式 (23) の再帰式から導かれる．

式 (13) で定まる $\hat{u}_0^{n-1}$ に G_n の逆変換 $G_n^{-1} = G_n$ を行うことで $\hat{s}_0^{n-1} = \hat{u}_0^{n-1} G_n$ を求めることができ，復号語を $g_n^{(A)}(u_E) = \hat{s}_0^{n-1}$ とする．以上が復号器 $g_n^{(A)}$ での操作であり，式 (13) により $\hat{u}_0^{n-1}$ の $\hat{u}_i$ を i の小さい順に定めていくので逐次除去復号法と呼ばれる．

polar 符号の長所である符号長 n に対して復号が $O(n \log n)$ の計算複雑度であることを確認する．無限精度の演算が単位コストで計算できると仮定し，符号長 n の復号器 $g_n^{(A)}$ での計算複雑度を $\Lambda_D(n)$ で表す． $n = 2$ の復号器 $g_2^{(A)}$ では $D_{2^0}^{(0)}(\lambda) = \frac{P_S(0)}{P_S(1)}$ と式 (14) と式 (15) より $D_2^{(0)}(\lambda)$ と $D_2^{(1)}(\hat{u}_0)$ が求まる．更に， $\{D_{2^m}^{(i)}(\hat{u}_0^{i-1})\}_{i \in \mathcal{I}_{2^m}}$ は式 (14) と式 (15) より $\{D_{2^{m-1}}^{(i)}(\hat{u}_{0,\text{偶}}^{2i-1} \oplus \hat{u}_{0,\text{奇}}^{2i-1})\}_{i \in \mathcal{I}_{2^{m-1}}}$ と $\{D_{2^{m-1}}^{(i)}(\hat{u}_{0,\text{偶}}^{2i-1})\}_{i \in \mathcal{I}_{2^{m-1}}}$ が既知であれば $O(n)$ の計算複雑度で求められる．それゆえ，再帰的に長さ n における問題は長さが半分の $\frac{n}{2}$ である問題二つに分解できる． $\Lambda_D(2)$ の計算複雑度を定数としたとき，

$$\Lambda_D(n) = O(n) + 2\Lambda_D\left(\frac{n}{2}\right)$$

の再帰式が導かれる．よって，式 (12) で導いた符号器の場合と同様に復号器の計算複雑度は $O(n \log n)$ である．

符号 $(f_n^{(A)}, g_n^{(A)})$ の誤り率 $\varepsilon_n^{(s)}(f_n^{(A)}, g_n^{(A)})$ について次の定理 4 が成り立つ⁽⁴⁾．

[定理 4] 情報源分極に基づく情報源符号 $(f_n^{(A)}, g_n^{(A)})$ の符号化レート $\frac{k}{n}$ が $\frac{k}{n} > H(S)$ ならば，誤り率 $\varepsilon_n^{(s)}(f_n^{(A)}, g_n^{(A)})$ は，任意に固定した $0 < \delta < \frac{1}{2}$ に対して，

$$\varepsilon_n^{(s)}(f_n^{(A)}, g_n^{(A)}) = o\left(2^{-n^{\left(\frac{1}{2}-\delta\right)}}\right)$$

を満たす．

図 13 情報源分極に基づく情報源符号 $(f_n^{(A)}, g_n^{(A)})$ による符号化レート $\frac{k}{n}$ と誤り率 $\varepsilon_n^{(s)}(f_n^{(A)}, g_n^{(A)})$ の計算機実験結果 $p = 0.11$ ， $n = 2^m$ ， $m = 7, 9, 13, 17$ ， $R_s(P_S) \approx 0.4999$ ．

確率変数 S でモデル化される定常無記憶情報源 $\{S_i\}_{i=0}^{\infty}$ からの出力系列を情報源符号 $(f_n^{(A)}, g_n^{(A)})$ を用いて伝送するときの polar 符号 $(f_n^{(A)}, g_n^{(A)})$ の誤り率 $\varepsilon_n^{(s)}(f_n^{(A)}, g_n^{(A)})$ の様子を計算機実験で試行した例を図 13 に示す．図 13 のグラフにおける横軸は符号化レート $\frac{k}{n}$ であり，縦軸は符号 $(f_n^{(A)}, g_n^{(A)})$ の誤り率である． $P_S(1)$ の確率が $p = 0.11$ ， $R_s(P_S) = h(0.11) \approx 0.4999$ ，符号長 $n = 2^m$ ， $m = 7, 9, 13, 17$ における $\varepsilon_n^{(s)}(f_n^{(A)}, g_n^{(A)})$ の実験結果である．定理 4 は $n \rightarrow \infty$ とともに符号化レート $\frac{k}{n}$ を $R_s(P_S) = H(S)$ に近づけても符号 $(f_n^{(A)}, g_n^{(A)})$ の誤り率 $\varepsilon_n^{(s)}(f_n^{(A)}, g_n^{(A)})$ を 0 に漸近できることを示している．

5. 通信路の再帰的な拡大変換と情報量のチェーンルールによる通信路分極

前章までで情報源分極による情報源符号 $(f_n^{(A)}, g_n^{(A)})$ を確認した．ここからは通信路分極について述べる．本稿では，通信路への入力記号が 0 か 1 の二元入力通信路で説明を行う．遷移確率行列 $W = \{P_{Y|X}(y|x)\}_{x \in \mathcal{X}, y \in \mathcal{Y}}$ でモデル化される定常無記憶通信路 W を考える．このとき，定常無記憶通信路の対称通信路容量 $I(W)$ は式 (1) で定義され， $\mathcal{X} = \{0, 1\}$ である．

通信路分極の操作を情報源分極の操作と同様に線形変換 T として式 (2) を用いて紹介する．まず，集合 $\mathcal{Z} = \{0, 1\}$ 上に一様分布 $P_Z(0) = P_Z(1) = \frac{1}{2}$ に従う互いに独立な確率変数の列 $\{Z_i\}_{i=0}^{\infty}$ を用意する． $n = 2^m$ ， $m = 1, 2, \dots$ とし， n 次拡大した通信路 $W^n : \mathcal{X}^n \rightarrow \mathcal{Y}^n$ への入力 $X_0^{n-1} = (X_0, X_1, \dots, X_{n-1}) \in \mathcal{X}^n$ の構成を

$$X_0^{n-1} = Z_0^{n-1} T^{\otimes m}$$

と定める． $m = 1$ ， $n = 2$ のときを確認すると，

$$(X_0, X_1) = (Z_0, Z_1)T = (Z_0 \oplus Z_1, Z_1)$$

であり， Z_0^1 と X_0^1 と Y_0^1 の関係を図 14 の左図に示す．ここで， $W_2 : \mathcal{Z}^2 \rightarrow \mathcal{Y}^2$ を

図 14 $n = 2, 4$ の場合の概念図 (右: $n = 2$, 左: $n = 4$)

$$W_2(y_0^1|z_0^1) = \prod_{i=0}^1 W(y_i|x_i) = \prod_{i=0}^1 W(y_i|(z_0^1 T)_i) \quad (16)$$

と定める．ただし， $(z_0^1 T)_i$ は系列 $z_0^1 T$ の 0 から数えて i 番目の要素を表す． Z_0^1 から X_0^1 への線形変換は全単射であり，通信路が無記憶であることとあわせて，

$$I(Z_0^1; Y_0^1) = I(X_0^1; Y_0^1) = 2I(W)$$

を満たす．相互情報量のチェインルールより，

$$I(Z_0^1; Y_0^1) = I(Z_0; Y_0^1) + I(Z_1; Y_0^1|Z_0)$$

であり， Z_0 と Z_1 は互いに独立な確率変数であり $I(Z_0; Z_1) = 0$ を満たす．それゆえ，

$$\begin{aligned} I(Z_0^1; Y_0^1) &= I(Z_0; Y_0^1) + I(Z_1; Y_0^1|Z_0) + I(Z_0; Z_1) \\ &= I(Z_0; Y_0^1) + I(Z_1; Y_0^1, Z_0) \end{aligned}$$

と変形できる．ここで， $I(Z_0; Y_0^1)$ の対称通信路容量を有する入力 Z_0 ，出力 Y_0^1 の通信路を考えることができ，この通信路を $W_2^{(0)}: \mathcal{Z} \rightarrow \mathcal{Y}^2$ で表すと，

$$\begin{aligned} W_2^{(0)}(y_0^1|z_0) &= \frac{1}{2} \sum_{z_1} W_2(y_0^1|z_0^1) \\ &= \frac{1}{2} \sum_{z_1} W(y_0|z_0 \oplus z_1) W(y_1|z_1) \end{aligned}$$

を満たす．同様に， $I(Z_1; Y_0^1, Z_0)$ の対称通信路容量を有する入力 Z_1 ，出力 Y_0^1, Z_0 の通信路を考えることができ，この通信路を $W_2^{(1)}: \mathcal{Z} \rightarrow \mathcal{Y}^2 \times \mathcal{Z}$ で表すと，

$$W_2^{(1)}(y_0^1, z_0|z_1) = \frac{1}{2} W_2(y_0^1|z_0^1) = \frac{1}{2} W(y_0|z_0 \oplus z_1) W(y_1|z_1)$$

を満たす．このとき，式 (1) の表記を用いて，

$$2I(W) = I(Z_0; Y_0^1) + I(Z_1; Y_0^1, Z_0) = I(W_2^{(0)}) + I(W_2^{(1)})$$

が成り立つ．更に， $I(W_2^{(1)}) = H(Z_1) - H(Z_1|Y_0^1, Z_0) \geq H(Z_1) - H(Z_1|Y_1) = I(W)$ より，

$$I(W_2^{(0)}) \leq I(W) \leq I(W_2^{(1)}) \quad (17)$$

が導かれる．ここまでの操作は二つの同一な通信路の組 (W, W) から二つの通信路 $(W_2^{(0)}, W_2^{(1)})$ に変換した．この変換により通信路 W より対称通信路容量が小さい通信路 $W_2^{(0)}$ と通信路 W より対称通信路容量が大きい通信路 $W_2^{(1)}$ が導かれた．通信路分極における式 (17) の関係式は情報源分極における式 (3) の関係式に対応している．

図 15 2 個の $W_{2^{m-1}}: \mathcal{Z}^{2^{m-1}} \rightarrow \mathcal{Y}^{2^{m-1}}$ から式 (18) による再帰的な拡大通信路 $W_{2^m}: \mathcal{Z}^{2^m} \rightarrow \mathcal{Y}^{2^m}$ への通信路変換

次に， $n = 2^m$ 個の通信路 W に対して再帰的な方法により通信路 $W_n: \mathcal{Z}^n \rightarrow \mathcal{Y}^n$ を

$$\begin{aligned} W_n(y_0^{n-1}|z_0^{n-1}) &= W_{\frac{n}{2}}(y_0^{\frac{n}{2}-1}|z_0^{\frac{n}{2}-1}) \oplus W_{\frac{n}{2}}(y_{\frac{n}{2}}^{\frac{n}{2}-1}|z_{\frac{n}{2}}^{\frac{n}{2}-1}) \quad (18) \end{aligned}$$

と構成する．ただし， $W_1 = W$ とする．このとき，式 (16) は式 (18) から得られる W_2 と一致する．式 (18) の関係を図示すると図 15 が得られる．図中の A_n は入力系列 $V_0^{2^m-1} = (V_0, V_1, \dots, V_{2^m-1})$ に対して $V_{0, \text{偶}}^{n-1}$ を前に $V_{0, \text{奇}}^{n-1}$ を後ろに並べた出力系列 $(V_0, V_2, \dots, V_{2^m-2}, V_1, V_3, \dots, V_{2^m-1})$ に変換する置換行列 A_n を表す．2 個の $W_{2^{m-1}}: \mathcal{Z}^{2^{m-1}} \rightarrow \mathcal{Y}^{2^{m-1}}$ から式 (18) による再帰的な拡大通信路 $W_{2^m}: \mathcal{Z}^{2^m} \rightarrow \mathcal{Y}^{2^m}$ への通信路変換の操作を次の (操作 1)~(操作 3) にまとめる．

(操作 1) それぞれの (Z_{2i}, Z_{2i+1}) ， $i \in \mathcal{I}_{2^{m-1}}$ に変換行列 T を用いる．

(操作 2) (操作 1) の出力系列 V_0^{n-1} に置換行列 A を用い， V_0^{n-1} を $V_{0, \text{偶}}^{n-1}$ と $V_{0, \text{奇}}^{n-1}$ に分割する．

(操作 3) 2 個の同一な通信路 $W_{\frac{n}{2}}$ を用意し，片方の通信路入力を $V_{0, \text{偶}}^{n-1}$ とし，もう一方の通信路入力を $V_{0, \text{奇}}^{n-1}$ にする．

$n = 4$ として，4 個の通信路 W を再帰的に結合した例を図 14 の右図に示す．なお，8 個の通信路 W を再帰的に結合した W_8 の例は図 6 において， $i \in \mathcal{I}_8$ の U_i を Z_i とし， S_i を W と Y_i で書き換え左右反転した図を考えればよい．

W_n を再帰的に式 (18) で定めたとき， Z_0^{n-1} と Y_0^{n-1} 間の通信路 W_n は通信路 W を用いて，各 y_0^{n-1} と z_0^{n-1} に対して

$$W_n(y_0^{n-1}|z_0^{n-1}) = \prod_{i=0}^{n-1} W(y_i|(z_0^{n-1} G_n)_i)$$

を満たす．式 (11) の G_n は $G_1 = (1)$ ， $G_2 = T$ であり， $m = 1, 2, \dots$ において，式 (18) 及び図 15 より

$$G_{2^m} = (E_{2^{m-1}} \otimes G_2) A_{2^m} (E_2 \otimes G_{2^{m-1}}) \quad (19)$$

を満たす．ただし， $E_{2^{m-1}}$ は単位行列である．

相互情報量のチェインルールと $\{Z_i\}_{i=0}^\infty$ が一様分布に従う互いに独立な確率変数の列であることから

$$I(Z_0^{n-1}; Y_0^{n-1}) = \sum_{i=0}^{n-1} I(Z_i; Y_0^{n-1}, Z_0^{i-1}) \quad (20)$$

と展開できる．展開した $I(Z_i; Y_0^{n-1}, Z_0^{i-1})$, $i \in \mathcal{I}_{2^m}$ の対称通信路容量を有する入力 Z_i , 出力 Y_0^{n-1}, Z_0^{i-1} の通信路を考えることができ, この通信路を $W_{2^m}^{(i)}: \mathcal{Z} \rightarrow \mathcal{Y}^{2^m} \times \mathcal{Z}^{i-1}$ で表すと, $W_1^{(0)} = W_1 = W$ であり,

$$\begin{aligned} & W_n^{(i)}(y_0^{n-1}, z_0^{i-1} | z_i) \\ &= \sum_{z_{i+1}^{n-1}} \frac{P_{Y_0^{n-1} Z_0^{n-1}}(y_0^{n-1}, z_0^{n-1})}{P_{Z_i}(z_i)} \\ &= \frac{1}{2^{n-1}} \sum_{z_{i+1}^{n-1}} W_n(y_0^{n-1} | z_0^{n-1}) \end{aligned} \quad (21)$$

を満たす．

Arikan は式 (18) により再帰的に (操作 1)~(操作 3) で 2^m 個の通信路 $W: \mathcal{X} \rightarrow \mathcal{Y}$ から拡大通信路 $W_{2^m}: \mathcal{Z}^{2^m} \rightarrow \mathcal{Y}^{2^m}$ に組み合わせることを通信路結合 (channel combining) と呼んでいる．また, 式 (20) の関係を用いて拡大通信路 $W_{2^m}: \mathcal{X}^{2^m} \rightarrow \mathcal{Y}^{2^m}$ を $W_{2^m}^{(i)}: \mathcal{Z} \rightarrow \mathcal{Y}^{2^m} \times \mathcal{Z}^{i-1}$, $i \in \mathcal{I}_{2^m}$ に分解することを通信路分解 (channel splitting) と呼んでいる． $\{W_{2^m}^{(i)}\}_{i \in \mathcal{I}_{2^m}}$ と $\{W_{2^{m-1}}^{(i)}\}_{i \in \mathcal{I}_{2^{m-1}}}$ の再帰の関連について次の補題が成り立つ．
[補題 1] (文献 (2) の Proposition 3, 4) 各 $n = 2^m$ と各 $i \in \mathcal{I}_{2^m}$, $m = 0, 1, 2, \dots$ において次が成り立つ．

$$\begin{aligned} & W_{2^m}^{(2i)}(y_0^{2^m-1}, z_0^{2i-1} | z_{2i}) \\ &= \frac{1}{2} \sum_{z_{2i+1}} W_{2^{m-1}}^{(i)}(y_0^{2^{m-1}-1}, z_0^{2i-1} \oplus z_{0, \text{偶}}^{2i-1} | z_{2i} \oplus z_{2i+1}) \\ & \quad W_{2^{m-1}}^{(i)}(y_0^{2^{m-1}-1}, z_0^{2i-1} | z_{2i+1}), \end{aligned} \quad (22)$$

$$\begin{aligned} & W_{2^m}^{(2i+1)}(y_0^{2^m-1}, z_0^{2i} | z_{2i+1}) \\ &= \frac{1}{2} W_{2^{m-1}}^{(i)}(y_0^{2^{m-1}-1}, z_0^{2i-1} \oplus z_{0, \text{偶}}^{2i-1} | z_{2i} \oplus z_{2i+1}) \\ & \quad W_{2^{m-1}}^{(i)}(y_0^{2^{m-1}-1}, z_0^{2i-1} | z_{2i+1}), \end{aligned} \quad (23)$$

$$I(W_{2^m}^{(2i)}) \leq I(W_{2^{m-1}}^{(i)}) \leq I(W_{2^{m-1}}^{(2i+1)}), \quad (24)$$

$$I(W_{2^m}^{(2i)}) + I(W_{2^m}^{(2i+1)}) = 2I(W_{2^{m-1}}^{(i)}). \quad (25)$$

Arikan は補題 1 の証明を文献 (2) の付録に述べており, 式 (22) と式 (23) は式 (21) と式 (18) を用いて導く．通信路結合と通信路分割により, 2 個の同一な通信路 $W_{2^{m-1}}^{(i)}$ と再帰的な式 (22) と式 (23) をなす通信路 $W_{2^m}^{(2i)}$ と $W_{2^m}^{(2i+1)}$ は式 (24) と式 (25) が示すように元の相互情報量 $I(W_{2^m}^{(i)})$ に対して相互情報量の小さくなる部分 $I(W_{2^m}^{(2i)})$ と相互情報量の大きくなる部分 $I(W_{2^m}^{(2i+1)})$ に分解できることを示している．

$I(W_{2^m}^{(i)})$, $i \in \mathcal{I}_{2^m}$ は, $m \rightarrow \infty$ の極限として, 相互情報量の分極操作が可能であり, 次の定理 5 が成り立つ⁽²⁾．

[定理 5] (文献 (2)) 任意の $0 < \epsilon < 1$ に対して,

$$\begin{aligned} \lim_{m \rightarrow \infty} \frac{|\{i \in \mathcal{I}_{2^m} : I(W_{2^m}^{(i)}) \in (1 - \epsilon, 1]\}|}{2^m} &= I(W), \\ \lim_{m \rightarrow \infty} \frac{|\{i \in \mathcal{I}_{2^m} : I(W_{2^m}^{(i)}) \in [0, \epsilon]\}|}{2^m} &= 1 - I(W) \end{aligned}$$

を満たす．

定理 5 は, $I(W_{2^m}^{(i)})$, $i \in \mathcal{I}_{2^m}$ を $m \rightarrow \infty$ とともに相互情報量が 1 に漸近する $I(W_{2^m}^{(i)})$ と相互情報量が 0 に漸近する $I(W_{2^m}^{(i)})$ に二分化できることを示しており, 情報量が 1 に漸近する割合が $I(W)$ であり, 情報量が 0 に漸近する割合が $1 - I(W)$ であることを述べている．この現象を Arikan は「通信路分極」と呼んでいる．

$I(W_{2^m}^{(i)}) = 1$ であれば, 1bit の情報を通信路 $W_{2^m}^{(i)}: \mathcal{Z} \rightarrow \mathcal{Y}^{2^m} \times \mathcal{Z}^{i-1}$ を介して伝送できる．すなわち, $I(W_{2^m}^{(i)}) = I(Z_i; Y_0^{2^m-1}, Z_0^{i-1}) = 1$ より通信路 W_n への入力系列 $z_0^{2^m-1}$ に対応する通信路 W_n からの出力系列が $y_0^{2^m-1}$ のとき, 復号器が出力系列 $y_0^{2^m-1}$ と入力系列 z_0^{i-1} を既知であれば, Z_i の値 z_i を正しく伝送可能である．それゆえ, 符号器では $I(W_{2^m}^{(i)}) = 1$ である i bit 目を用いて Z_i として送信したい情報を伝送する．また, 送信したい情報を伝送しなかった Z_i は符号器と復号器で事前に値を共有する．これにより, 復号器では送信された Z_i の値を i の小さい順に正しく推定できる．この原理により符号化レートを $I(W)$ に下から近づけても誤り率が 0 に漸近する通信路符号化が可能である．通信路分極に基づき Arikan⁽²⁾ が提案した通信路符号化を次章で紹介する．

6. 通信路分極に基づく通信路符号

$\mathcal{X} = \{0, 1\}$ である遷移確率行列 $W = \{W(y|x)\}_{x \in \mathcal{X}, y \in \mathcal{Y}}$ によりモデル化される定常無記憶通信路 $W: \mathcal{X} \rightarrow \mathcal{Y}$ に対して, $n = 2^m$, $m = 1, 2, \dots$ として, 通信路分極に基づく通信路符号の符号化写像 $\varphi_n^{(A)}: \{0, 1\}^k \rightarrow \{0, 1\}^n$ と復号写像 $\psi_n^{(A)}: \{0, 1\}^n \rightarrow \{0, 1\}^k$ を考える．まず, n と k は符号器と復号器で既知と仮定する．定常無記憶通信路 W に対して, 2^m 個の通信路 $W: \mathcal{X} \rightarrow \mathcal{Y}$ を用いた通信路結合と通信路分割により, $W_{2^m}^{(i)}: \mathcal{X} \rightarrow \mathcal{Y}^{2^m} \times \mathcal{X}^{i-1}$, $i \in \mathcal{I}_{2^m}$ を考え, $I(W_{2^m}^{(i)})$, $i \in \mathcal{I}_n$ の値の大きい k 個のインデックス i からなる集合 $\mathcal{M} \subset \mathcal{I}_n$ を定める．このとき, $\mathcal{M} \subset \mathcal{I}_n$ は次の条件 1, 2 を満たす．

条件 1 $\forall i \in \mathcal{M}, \forall j \in \mathcal{M}^c$ に対して $I(W_{2^m}^{(i)}) \geq I(W_{2^m}^{(j)})$ ．

条件 2 $|\mathcal{M}| = k$ ．

このインデックス集合 $\mathcal{M} \subset \mathcal{I}_{2^m}$ を符号器と復号器は既知と仮定し, 値 z_i , $i \in \mathcal{M}^c$ を符号器と復号器で事前に定めておき既知と仮定する^(注 7)．それゆえ, 文献 (2) では z_i , $i \in \mathcal{M}^c$ からなる bit 列を凍結ビット (frozen bits) 若しくは凍結ベクトル (frozen vector) と呼んでいる．

一様分布に従い生起する送信したい k bit の情報を $m_0^{k-1} = (m_0, m_1, \dots, m_{k-1}) \in \{0, 1\}^k$ と表記する． $\{z_i, i \in \mathcal{M}\}$ のインデックス i の小さい順に $z_{i_1}, z_{i_2}, \dots, z_{i_{k-1}}$ とし, 符号器では, 与えられた m_0^{k-1} に対して, z_i , $i \in \mathcal{M}$ を $z_{i_j} = m_j$, $j = 0, 1, \dots, k-1$ と定める． z_i , $i \in \mathcal{M}^c$ は符号器と復号器で既知を仮定しており, 送信したい k bit の情報 m_0^{k-1} に対して z_0^{n-1} が定まる．符号器では, 情報 m_0^{k-1} に対して定めた z_0^{n-1} を用

(注 7): W が対称通信路であるならば z_i , $i \in \mathcal{M}^c$ として $n - k$ bit の系列を任意に定めても符号 $(\varphi_n^{(A)}, \psi_n^{(A)})$ の誤り率の性能に影響を与えないことが知られている⁽²⁾．

いて符号語 $\varphi_n^{(A)}(m_0^{k-1}) = z_0^{n-1} G_n$ と定める。

例えば, $\mathcal{Y} = \{0, 1\}$ とし, $W(0|0) = W(1|1) = 0.9, W(1|0) = W(0|1) = 0.1$ でモデル化される定常無記憶通信路 W と $k = 4, n = 8$ の符号 $(\varphi_8^{(A)}, \psi_8^{(A)})$ 場合, $\mathcal{M} = \{3, 5, 6, 7\}$ であり, 送信したい情報 m_0^3 に対して, $z_0^7 = (z_0, z_1, z_2, m_0, z_4, m_1, m_2, m_3)$ と定め, m_0^3 に対する符号語を $\varphi_8^{(A)}(m_0^3) = z_0^7 G_8$ とする。

polar 符号では使用する通信路 W に応じて $\mathcal{M}$ が異なる可能性があり符号語が異なる場合がある。また, $z_i, i \in \mathcal{M}^c$ の値によっても符号語が異なる。

符号語 $\varphi_n^{(A)}(m_0^{k-1}) \in \{0, 1\}^n$ を通信路 $W^n: \mathcal{X}^n \rightarrow \mathcal{Y}$ への入力系列とし, 通信路 W^n からの出力系列 y_0^{n-1} を受け取った復号器では Z_0^{i-1} のうち, $Z_i, i \in \mathcal{M}^c$ の値を既知であり, 未知である $Z_i, i \in \mathcal{M}$ に対して i の小さい順に逐次的に推定を行う。具体的には, Z_0^{i-1} が $Z_0^{i-1} = \hat{z}_0^{i-1}$ であると仮定したもとして Z_i のゆう度比に基づき逐次的に行う。すなわち, z_0^{n-1} を $i = 0, 1, \dots, n-1$ の順に

$$\hat{z}_i = \begin{cases} z_i & \text{if } i \in \mathcal{M}^c, \\ 0 & \text{if } i \in \mathcal{M} \text{ かつ } L_{2^m}^{(i)}(\hat{z}_0^{i-1}) \geq 1, \\ 1 & \text{if } i \in \mathcal{M} \text{ かつ } L_{2^m}^{(i)}(\hat{z}_0^{i-1}) < 1 \end{cases} \quad (26)$$

と定める。ここで $L_{2^m}^{(i)}(\hat{z}_0^{i-1})$ は, $W_{2^m}^{(i)}(y_0^{2^m-1}, z_0^{i-1}|0)$ と $W_{2^m}^{(i)}(y_0^{2^m-1}, z_0^{i-1}|1)$ の比

$$L_{2^m}^{(i)}(\hat{z}_0^{i-1}) = \frac{W_{2^m}^{(i)}(y_0^{2^m-1}, z_0^{i-1}|0)}{W_{2^m}^{(i)}(y_0^{2^m-1}, z_0^{i-1}|1)} \quad (27)$$

で定義される。 λ により空列を表すと, $L_{2^0}^{(0)}(\lambda) = \frac{W(y|0)}{W(y|1)}$ であり, $L_{2^m}^{(i)}(\hat{z}_0^{i-1})$ は i の偶奇で場合分けを行い, 式 (22) と式 (23) より次の再帰式で計算できる^(注8)。

$$L_{2^m}^{(2i)}(\hat{z}_0^{2i-1}) = \frac{1 + L_{2^{m-1}}^{(i)}(\hat{z}_0^{2i-1} \oplus \hat{z}_0^{2i-1}) L_{2^{m-1}}^{(i)}(\hat{z}_0^{2i-1})}{L_{2^{m-1}}^{(i)}(\hat{z}_0^{2i-1} \oplus \hat{z}_0^{2i-1}) + L_{2^{m-1}}^{(i)}(\hat{z}_0^{2i-1})} \quad (28)$$

$$L_{2^m}^{(2i+1)}(\hat{z}_0^{2i}) = \frac{L_{2^{m-1}}^{(i)}(\hat{z}_0^{2i} \oplus \hat{z}_0^{2i})^{1-2\hat{z}_i} L_{2^{m-1}}^{(i)}(\hat{z}_0^{2i})}{L_{2^{m-1}}^{(i)}(\hat{z}_0^{2i} \oplus \hat{z}_0^{2i})^{1-2\hat{z}_i} L_{2^{m-1}}^{(i)}(\hat{z}_0^{2i})} \quad (29)$$

式 (26) で定まる $\hat{z}_0^{n-1}$ の部分系列 $\hat{z}_{\mathcal{M}}$ を送信された m_0^{k-1} の推定値とし, $\psi_n^{(A)}(y_0^{n-1}) = \hat{z}_{\mathcal{M}}$ とする。以上が復号器 $\psi_n^{(A)}$ での操作であり, 式 (26) により $\hat{z}_0^{n-1}$ の $\hat{z}_i$ を i の小さい順に決めていくので逐次除去復号法と呼ばれる。このとき, 符号長 n の符号器 $\varphi_n^{(A)}$ と復号器 $\psi_n^{(A)}$ の計算複雑度は, 式 (12) と同様に $O(n \log n)$ である。7. に符号長 $n = 2^3$ の C 言語によるプログラミング例を示す。

通信路分極に基づく通信路符号 $(\varphi_n^{(A)}, \psi_n^{(A)})$ の符号化レートは $\frac{k}{n}$ であり, 符号 $(\varphi_n^{(A)}, \psi_n^{(A)})$ の誤り率 $\varepsilon_n^{(c)}(\varphi_n^{(A)}, \psi_n^{(A)})$ について $z_{\mathcal{M}^c}$ を取り替えた期待値を評価して, 次の定理 6 が成り立つ^{(2) (14)}。

[定理 6] 符号化レート $r_n(\varphi_n^{(A)}, \psi_n^{(A)}) = \frac{k}{n}$ が $\frac{k}{n} < I(W)$ であり, 誤り率 $\varepsilon_n^{(c)}(\varphi_n^{(A)}, \psi_n^{(A)})$ が, 任意に固定した $0 < \delta < \frac{1}{2}$ に対して,

$$\varepsilon_n^{(c)}(\varphi_n^{(A)}, \psi_n^{(A)}) = o\left(2^{-n(\frac{1}{2}-\delta)}\right)$$

を満たす通信路分極に基づく通信路符号 $(\varphi_n^{(A)}, \psi_n^{(A)})$ が存在する。

定理 6 では $\varepsilon_n^{(c)}(\varphi_n^{(A)}, \psi_n^{(A)})$ について $Z_{\mathcal{M}^c}$ の期待値を評価しており, 通信路符号 $(\varphi_n^{(A)}, \psi_n^{(A)})$ の存在を述べてる。これに対して W が対称通信路であれば次の定理が成り立つ。

[定理 7] 与えられた二元入力対称通信路 W に対して, 符号化レート $\frac{k}{n}$ が $\frac{k}{n} < I(W) = R_c(W)$ ならば, 任意に $z_{\mathcal{M}^c}$ を定めた通信路符号 $(\varphi_n^{(A)}, \psi_n^{(A)})$ の誤り率 $\varepsilon_n^{(c)}(\varphi_n^{(A)}, \psi_n^{(A)})$ は, 任意に固定した $0 < \delta < \frac{1}{2}$ に対して,

$$\varepsilon_n^{(c)}(\varphi_n^{(A)}, \psi_n^{(A)}) = o\left(2^{-n(\frac{1}{2}-\delta)}\right)$$

を満たす。

定理 7 は $n \rightarrow \infty$ とともに符号化レート $\frac{k}{n}$ を $R_c(W) = I(W)$ に近づけても符号 $(\varphi_n^{(A)}, \psi_n^{(A)})$ の誤り率 $\varepsilon_n^{(c)}(\varphi_n^{(A)}, \psi_n^{(A)})$ を 0 に漸近できることを示している。

ここまでは, 林の解説論文⁽¹⁵⁾にある一次漸近理論であり, 二次漸近理論が更なる興味の対象となる。 $\varepsilon_n^{(c)}(\varphi_n^{(A)}, \psi_n^{(A)})$ に関する符号化レートに依存した詳細化として次の定理^{(11) (16)}が明らかにされている。

[定理 8] 二元入力対称通信路 W が与えられたとき, 通信路分極に基づく通信路符号 $(\varphi_n^{(A)}, \psi_n^{(A)})$ の符号化レート $\frac{k}{n}$ が $\frac{k}{n} < I(W) = R_c(W)$ であり, $|\mathcal{M}| = k$ を満たす $\mathcal{M}$ を適切に選択すれば, 任意に $z_{\mathcal{M}^c}$ を定めた通信路分極に基づく通信路符号 $(\varphi_n^{(A)}, \psi_n^{(A)})$ は誤り率

$$\varepsilon_n^{(c)}(\varphi_n^{(A)}, \psi_n^{(A)}) = o\left(2^{-2^{(m+Q^{-1}(\frac{k}{nI(W)})\sqrt{m})/2}}\right)$$

を満たす。

ここで, Q^{-1} は

$$Q(t) = \frac{1}{\sqrt{2\pi}} \int_t^{-\infty} e^{-u^2/2} du$$

で定義される標準正規分布における相補累積分布関数である Q 関数の逆関数である。

最良の固定長通信路符号 (φ_n, ψ_n) の誤り率 $\varepsilon_n^{(c)}(\varphi_n, \psi_n)$ は, 符号化レート $\frac{k}{n}$ と通信路 W で定まる信頼性関数 $F_c(\frac{k}{n}, W)$ を用いて, 符号長 n の指数関数的に $\varepsilon_n^{(c)}(\varphi_n, \psi_n) = o\left(2^{-nF_c(\frac{k}{n}, W)}\right)$ で減少することが知られている⁽¹⁷⁾。これに対して, 定理 7 が示す符号 $(\varphi_n^{(A)}, \psi_n^{(A)})$ の誤り率 $\varepsilon_n^{(c)}(\varphi_n^{(A)}, \psi_n^{(A)})$ の 0 への漸近は最良の固定長通信路符号の場合より遅い。すなわち, 小さい誤り率を達成するために必要となる符号 $(\varphi_n^{(A)}, \psi_n^{(A)})$ の符号長は最良の固定長通信路符号の符号長より長くなる。

(注8): Z_i の推定値を式 (27) の $L_{2^m}^{(i)}(\hat{z}_0^{i-1})$ を定めたが, $P_{Z_i|Y_0^{2^m-1}, Z_0^{i-1}}(0|y_0^{2^m-1}, z_0^{i-1})$ と $P_{Z_i|Y_0^{2^m-1}, Z_0^{i-1}}(1|y_0^{2^m-1}, z_0^{i-1})$ の比により z_i を評価することも可能である⁽¹³⁾。

7. C 言語によるプログラミング例

情報源分極に基づく情報源符号の C 言語によるプログラミング例を 7.1 に述べ、通信路分極に基づく通信路符号の C 言語によるプログラミング例を 7.2 に述べる。大きな n で符号器と復号器の設計において ε や M を定めるとき、計算複雑度や空間複雑度からの困難が生じる場合は、この困難を回避する方法の一つとして近似を利用できる。 $W_n^{(i)}: \{0, 1\} \rightarrow \mathcal{Y} \times \mathcal{Z}^i$ に対して、 $|\mathcal{Y} \times \mathcal{Z}^i|$ がある一定値以下を満たすように通信路出力を量子化により近似することで通信路 $\hat{W}_n^{(i)}$ を定め、 $I(W_n^{(i)}) \approx I(\hat{W}_n^{(i)})$ なる近似値 $I(\hat{W}_n^{(i)})$ を用いて、 M の代わりに定めるプログラミング例を 7.3 に述べる。

7.1 情報源分極に基づく情報源符号の場合

4. に紹介した情報源分極に基づく情報源符号 $(f_n^{(A)}, g_n^{(A)})$ の C 言語によるプログラミングを試みる。ここでは $m = 4$ と定めた $(f_n^{(A)}, g_n^{(A)})$, $n = 2^m$ を紹介する(注9)。 $P_S(0) = 1 - p$, $P_S(1) = p$ の確率でモデル化される定常無記憶情報源からの出力系列のサンプル $s_0^{15} \in \{0, 1\}^{15}$ を作成し、 s_0^{15} を符号化する source_enc.c を述べる。source_enc.c は common.c を読み込んでおり、common.c は復号とともに共通で用いる関数を記述している。実行は source_enc p d であり、第 1 引数の p は $P_S(1) = p$ の p、第 2 引数の d は符号化しないビット数、 $|\mathcal{E}^c| = d$ である。プログラムの実行結果、疑似符号語として $u_i = 2, i \in \mathcal{E}^c$ とした系列 u_0^{n-1} をファイル名 codeword_p.enc に保存する。

符号長を大きくするには、fb[n] を適切に定め、演算精度を考慮に入れ、main 関数の 6 行目にある int m=4; の m を大きくする。fb[n] を適切に定める 1 手法を 7.3 に通信路の場合について述べる。

```
%//
// common.c
//
// source_{enc,dec}.c
// channel_enc_dec.c
// 同じディレクトリに置く
//

typedef struct ldi{
    double zero; // D もしくは L の分子
    double one;  // D もしくは L の分母
}ldi;

ldi **nd; // D もしくは L の格納用
int n,m;
```

(注9): m を大きくし、符号長を大きくする場合は計算機での演算精度やメモリ量に注意する。これは、7.2 で紹介する通信路符号 $(\varphi_n^{(A)}, \psi_n^{(A)})$ の実装についても同様である。

// 系列 $u[0..n-1]$ を表示する。

```
void print_seq(char *u) {
    int i;
    for(i=0;i<n;i++) printf("%d",u[i]);
    printf("\n");
}
```

// B_n の操作を行う

```
unsigned int bitrev(unsigned int b) {
    b=(b&0x55555555)<<1 |(b&0xaaaaaaaa)>>1;
    b=(b&0x33333333)<<2 |(b&0xcccccccc)>>2;
    b=(b&0xf0f0f0f0)<<4 |(b&0xf0f0f0f0)>>4;
    b=(b&0xff00ff00)<<8 |(b&0xff00ff00)>>8;
    b=(b&0x0000ffff)<<16|(b&0xffff0000)>>16;
    return b>>(32-m);
}
```

// s T_n の操作を行う。

```
void matrixt(char *s){
    int i,j;
    for(i=1;i<=m;i++)
        for(j=0;j<n;j++)
            if((j%(1<<i))<(1<<(i-1)))
                s[j]=s[j]^s[(1<<(i-1))+j];
}
```

```
//
// source_enc.c
//
// m=4, n=16, 圧縮する d ビットを削除する。
// 符号語は u[n] に生成され、u[i]=2 は送信されない bit
//
// アルゴリズムの実行例 ./source_enc p d
// s[0..15]= 0000101100010010
// u[0..15]= 1001001101100110
// Codeword = 1001001201120222
// creat codeword_p.enc
//
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <limits.h>
#include <time.h>
#include "common.c"
```

// 情報源出力サンプルを作成する。

```
void make_source_sample(char *s,double p) {
    int i;
```



```

printf("s[0..%d]=\t",n-1);
for(i=0;i<n;i++) { //DMS P_S(1) = p
    s[i]=(rand()/(double)INT_MAX<p)? 1:0;
    printf("%d",(int)s[i]);
}
printf("\n");
}

void fprint_codeword(FILE *fp,char *u) {
    int i;
    for(i=0;i<n;i++) fprintf(fp,"%d",u[i]);
    fclose(fp);
}

void encode(char *s, int d, char *u,
            int *fb, char *fn, FILE *fp) {
    int i;

    // 符号語の元 u = s T_n B_n の生成
    matrixt(s); // s T_n
    for(i=0;i<n;i++) u[bitrev(i)]=s[i]; // B_n
    printf("u[0..%d]=\t",n-1);
    print_seq(u);
    //符号語 u[0..n-1]: d 個を 2 にする
    for(i=0;i<d;i++) u[fb[i]]=2;
    printf("Codeword =\t");
    print_seq(u);
    fprint_codeword(fp,u);
    printf("creat %s \n",fn);
}

int main(int argc,char *argv[]){
    FILE *fp;
    int d;
    double p;

    m = 4; // 入力系列長 n=2^m
    n=1<m;

    if(argc!=3) {
        printf("Usage: %s p d \n",argv[0]);
        exit(1);
    }

    char fn[50]="codeword_";
    char footer[]=".enc";
    strcat(fn,argv[1]);
    strcat(fn,footer);

    if((fp=fopen(fn,"w"))==NULL) {

```

```

        fprintf(stderr,"File error: %s.\n",fn);
        exit(1);
    }

    p=atof(argv[1]); // p =P_S(1)
    if(p<0.0||p>1.0) {
        fprintf(stderr,"Bad parameter: 0 <= p <= 1.0\n");
        exit(1);
    }

    d=atoi(argv[2]); // d は 0 <= d <= 2^m で
    if(d<0||d>1<m) { // 符号化レートは (n-d)/n
        fprintf(stderr,"Bad parameter: 0 <= %s <= 16\n"
            ,argv[2]);
        exit(1);
    }

    char s[n], u[n];

    // H(U_i|U_0^{i-1}) の小さい順に i を並べた fb[n]
    int fb[16] = // p が約 0.185 と約 0.815 の間の場合
        {15, 14, 13, 11, 7, 12, 10,
          9, 6, 5, 3, 8, 4, 2, 1, 0};

    srand(time(NULL));

    if(p < 0.18488 || 0.81512 < p) {
        fb[7]=6; // p によって H(U_i|U_0^{i-1})
        fb[8]=9; // の大小関係における順序が変化する .
    }

    // 情報源出力サンプルを作成 s[0..n-1], P_S(1)=p
    make_source_sample(s,p);
    // s[0..n-1] の符号化
    encode(s,d,u,fb,fn,fp);
    return 0;
}

```

次に, 符号化された codeword_p.enc から復号する source_dec.c を記述する. 実行は source_dec p であり, 第 1 引数の p は $P_S(1) = p$ の p である. 符号化と復号を続けて行う例として,

符号化

```

./source_enc 0.2 3
s[0..15]= 10000010000000000
u[0..15]= 00101010000000000
Codeword = 00101010000000222
creat codeword_0.2.enc

```

復号

```

./source_dec 0.2
u[0..15]= 00101010000000222
hat{u}[0..15]= 00101010000000000

```

```

hat{s}[0..15]= 10000010000000000
ならば, s[0..15]=hat{s}[0..15] で正しく復号できているこ
とを示す. 一方,
符号化
./source_enc 0.2 5
s[0..15]= 1000010010000100
u[0..15]= 0001000001010000
Codeword = 0001000201020222
creat codeword_0.2.enc
復号
./source_dec 0.2
u[0..15]= 0001000201020222
hat{u}[0..15]= 0001000001000000
hat{s}[0..15]= 0100100001001000
ならば,  $\hat{u}_{11}$  の復号を誤っており正しく復号できない例である.
符号長を大きくするには, 演算精度を考慮に入れるとともに,
main 関数の 7 行目にある int m=4; の m を大きくする.

//
// source_dec.c
//
// % source_dec p
// により codeword_p.enc
// の復号を試みる.
//
// アルゴリズムの実行例 ./source_dec p
// Codeword = 1001001201120222
// u[0..15]= 1001001101100110
// s[0..15]= 0000101100010010
//

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "common.c"

// D のための初期化
void init_nd(double p) {
    int i,j;

    nd=(ldi **)malloc(n*sizeof(ldi*));
    for(i=0;i<n;i++) {
        nd[i]=(ldi *)malloc((m+1)*sizeof(ldi));
        nd[i][0].zero=1.0-p; // D の分子の初期化
        nd[i][0].one=p; // D の分母の初期化
        for(j=1;j<=m;j++) // -1.0 は未計算を表す
            nd[i][j].zero=nd[i][j].one=-1.0;
    }
}

```

```

// D の計算 (再帰呼出し)
ldi D(int i,int j,char *u2,int len){
    char *u2_;
    ldi tmpD;
    double sum;
    int k=1<<j,k2=k<<1,t,ix=i^k,lenh=len>>1;

    u2_=(char *)malloc(lenh*sizeof(char));
    if(nd[i][j].zero<0.0){ // 未計算の場合
        if(lenh>0) {
            if((ix%k2)<k) for(t=0;t<lenh;t++)
                u2_[t]=u2[2*t]^u2[2*t+1];
            else for(t=0;t<lenh;t++) u2_[t]=u2[2*t+1];
        }
        nd[i][j]=D(i,j-1,u2_,lenh);
    }
    if(nd[ix][j].zero<0.0){ // 未計算の場合
        if(lenh>0) {
            if((ix%k2)<k) for(t=0;t<lenh;t++)
                u2_[t]=u2[2*t]^u2[2*t+1];
            else for(t=0;t<lenh;t++) u2_[t]=u2[2*t+1];
        }
        nd[ix][j]=D(ix,j-1,u2_,lenh);
    }
    tmpD.zero=(len&1)? // D の分子 (14),(15) 参照
        (u2[len-1]?nd[ix][j].one:nd[ix][j].zero)
        *nd[i][j].zero:
        nd[i][j].zero*nd[ix][j].zero+nd[i][j].one
        *nd[ix][j].one;
    tmpD.one=(len&1)? // D の分母 (14),(15) 参照
        (u2[len-1]?nd[ix][j].zero:nd[ix][j].one)
        *nd[i][j].one:
        nd[i][j].zero*nd[ix][j].one+nd[i][j].one
        *nd[ix][j].zero;
    sum=tmpD.zero+tmpD.one;
    tmpD.zero/=sum;
    tmpD.one/=sum;
    free(u2_);
    return tmpD;
}

// 符号語 u のファイルからの読み出し
void init_codeword(FILE *fp,char *u) {
    int i;
    char c;

    for(i=0;i<n;i++) {
        if(fscanf(fp,"%c",&c)==EOF) {
            fprintf(stderr,"File error\n");
            exit(1);
        }
    }
}

```

```

    }
    u[i]=c-'0';
}
fclose(fp);
}

// 符号語 u の復号
void decode(double p, char *u) {
    int i;
    unsigned int ir;
    char s[n];

    init_nd(p); // D の計算のための初期化

    for(i=0;i<n;i++) // u[i]==2 なら 逐次に u[i] を推定
        if(u[i]==2) {
            ir = bitrev(i);
            nd[ir][m]=D(ir,m-1,u,i);
            u[i]=nd[ir][m].zero<nd[ir][m].one;
        }
    printf("hat{u}[0..%d]=\t",n-1);
    print_seq(u);

    // 以下では u から s を復元
    matrixt(u); // u T_n
    for(i=0;i<n;i++) s[i]=u[bitrev(i)];
    printf("hat{s}[0..%d]=\t",n-1);
    print_seq(s);
}

int main(int argc,char *argv[]){
    FILE *fp;
    char fn[30]="codeword_";
    char footer[]=".enc";
    double p;

    m = 4; // 入力系列長 n=2^m
    n=1<m;

    if(argc!=2) {
        printf("Usage: %s p\n",argv[0]);
        exit(1);
    }

    strcat(fn,argv[1]);
    strcat(fn,footer);

    if((fp=fopen(fn,"r"))==NULL ) {
        fprintf(stderr,"File error: %s.\n",fn);
        exit(1);
    }

```

```

    }

    p=atof(argv[1]); // p = P_S(1)
    if(p<0.0||p>1.0) exit(1);

    char u[n];

    // 圧縮された符号語 u[0..n-1] に格納
    init_codeword(fp,u);
    printf("u[0..%d]=\t",n-1);
    print_seq(u);
    // u[0..n-1] の復号
    decode(p,u);
    return 0;
}

```

7.2 通信路分極に基づく通信路符号の場合

6. に紹介した通信路分極に基づく通信路符号 $(\varphi_n^{(A)}, \psi_n^{(A)})$ の C 言語によるプログラミングを試みる．ここでは $m = 3$ と定めた $(\varphi_8^{(A)}, \psi_8^{(A)})$ を紹介する．まず，遷移確率行列 W によりモデル化される定常無記憶通信路 W に対して， $I(W_{2^m}^{(i)}), i \in \mathcal{I}_n$ を求め， $I(W_{2^m}^{(i)}), i \in \mathcal{I}_n$ の値の小さい順にインデックス i を並べ直すプログラムとして，calIdmc.c を用意する．calIdmc.c は二元対称通信路 $W : \{0,1\} \rightarrow \{0,1\}$ の場合の例であり， $0 \leq p \leq 1$ を用いて $W(1|0) = W(0|1) = p$ とモデル化する．

実行は calIdmc p である．実行結果として， $I(W_{2^m}^{(i)}), i \in \mathcal{I}_{2^m}$ の値の小さい順にインデックス i を並べた結果をファイル名 sort_I_p.dat に保存する．また， $I(W_{2^m}^{(i)}), i \in \mathcal{I}_{2^m}$ の計算に式 (22) と式 (23) を用いた場合の $|\mathcal{Y} \times \mathcal{Z}^i|$ を表示する．このとき， $|\mathcal{Y} \times \mathcal{Z}^i|$ は符号長とともに指数関数的に大きくなる．最後に確認のため， $I(W), nI(W), \sum_{i=0}^{n-1} I(W_n^{(i)})$ を表示する．計算誤差を生じなければ $nI(W) = \sum_{i=0}^{n-1} I(W_n^{(i)})$ が成り立つ．なお，プログラムの記述の簡略化のため同じ情報量の演算を繰返し行っているが，同じ演算の繰返しを省略することは可能である．

二元対称通信路と異なる通信路 W を考えるときは関数 void calI における二元対称通信路 BSC の例の下 7 行分等を変更する．特に，通信路 W が二元消失通信路の場合は文献 (2) の (6) より

$$\begin{aligned}
 I(W_1^{(0)}) &= I(W), \\
 I(W_{2^m}^{(2i)}) &= I(W_{2^{m-1}}^{(i)}), \\
 I(W_{2^m}^{(2i+1)}) &= 2I(W_{2^{m-1}}^{(i)}) - I(W_{2^{m-1}}^{(i)})^2
 \end{aligned}$$

が成り立ち， $I(W_{2^m}^{(i)})$ を符号長 n に対して $O(n)$ で計算可能である．

```

//
// calIdmc.c

```

```

//
// 実行は calIdmc p
//
// p は BSC の cross over probability
// p は 0.0 <= p <= 1.0
//
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>

int m=3;

typedef struct matrix{
    double x0; //入力 0
    double x1; //入力 1
} matrix;

typedef struct channel{
    int size; // 出力記号数 |Y|
    matrix *c; // 条件付確率
} channel;

typedef struct sortidx{ // ソート用
    int idx; // インデックス
    double inf; // 情報量
} sortidx;

int compare_inf(const void *a, const void *b)
{ // qsort における情報量の比較のため
    sortidx ta = *(sortidx *)a;
    sortidx tb = *(sortidx *)b;
    if ( (double)ta.inf == (double)tb.inf )
        return(0);
    else if ((double)ta.inf > (double)tb.inf)
        return(1);
    else return(-1);
}

double inf(double prob) { // 確率 x 自己情報量
    if(prob<=0.0||prob>=1.0) return 0.0;
    return (-prob*log2(prob));
}

double mutualinf(matrix c) { // 相互情報量
    return(0.5*(c.x0+c.x1+inf(c.x0+c.x1)
        -inf(c.x0)-inf(c.x1)));
}

channel do_even(channel Q){ // XOR がある場合

```

```

    int i,j,k;
    channel Wtmp;

    Wtmp.c = (matrix *)malloc(Q.size*Q.size
        *sizeof(matrix));

    for(k=0,i=0;i<Q.size;i++)
        for(j=0;j<Q.size;j++){ // 補題 1 を利用
            Wtmp.c[k].x0 // for u == 0
                =0.5*(Q.c[i].x0*Q.c[j].x0
                    +Q.c[i].x1*Q.c[j].x1);
            Wtmp.c[k++].x1 // for u == 1
                =0.5*(Q.c[i].x1*Q.c[j].x0
                    +Q.c[i].x0*Q.c[j].x1);
        }
    Wtmp.size = k;
    free(Q.c);
    return(Wtmp);
}

channel do_odd(channel Q){ // XOR がない場合
    int i,j,k;
    channel Wtmp;

    Wtmp.c = (matrix *)malloc(2*Q.size*Q.size
        *sizeof(matrix));

    for(k=0,i=0;i<Q.size;i++)
        for(j=0;j<Q.size;j++){ // 補題 1 を利用
            Wtmp.c[k].x0 // (z,x)==(0,0), (z xor x)==0
                =0.5*Q.c[i].x0*Q.c[j].x0;
            Wtmp.c[k++].x1 // (z,x)==(1,0), (z xor x)==1
                =0.5*Q.c[i].x1*Q.c[j].x1;
            Wtmp.c[k].x0 // (z,x)==(0,1), (z xor x)==1
                =0.5*Q.c[i].x1*Q.c[j].x0;
            Wtmp.c[k++].x1 // (z,x)==(1,1), (z xor x)==0
                = 0.5*Q.c[i].x0*Q.c[j].x1;
        }
    Wtmp.size = k;
    free(Q.c);
    return(Wtmp);
}

void calI(double p, int n,char *fn, FILE *fp) {

    int i,j,k;
    channel W;
    sortidx IW[n]; // I(W_n^{(i)}) を IW[i] に保持
    double IW0,sumI = 0.0; // I(W_n^{(i)}) の和=n*I(W)

```



```

for(i=0;i<n;i++){ // i は  $I(W_n^{\{(i)\}})$  の i
    // 2 元対称通信路 BSC の例
    W.size=2; // W の出力記号数
    W.c=(matrix *)malloc(W.size*sizeof(matrix));
    W.c[0].x0 = 1-p; //  $W(0|0)=P_{\{Y|X\}}(0|0)=1-p$ 
    W.c[1].x0 = p; //  $W(1|0)=P_{\{Y|X\}}(1|0)=p$ 
    W.c[0].x1 = p; //  $W(0|1)=P_{\{Y|X\}}(0|1)=p$ 
    W.c[1].x1 = 1-p; //  $W(1|1)=P_{\{Y|X\}}(1|1)=1-p$ 
    IW0=mualinf(W.c[0])+mualinf(W.c[1]); //  $I(W)$ 

    for(j=0;j<m;j++) // 次の?で even と odd の判定
        W= ((i&(1<<(m-j-1)))==0)? do_even(W):do_odd(W);

    // 情報量  $I(W_n^{\{(i)\}}) = IW[i]$  の計算
    for(IW[i].idx=i,IW[i].inf=0.0,k=0;k<W.size;k++)
        IW[i].inf += mualinf(W.c[k]);

    printf("|Y~%dxZ~%d|=%10d, I(W_%d~(%d))=%1.5e\n",
           n,i,W.size,n,i,IW[i].inf);
    sumI += IW[i].inf;
    free(W.c);
}

printf("\n");

qsort(IW,n,sizeof(sortidx),compare_inf);
for(i=0;i<n;i++) fprintf(fp,"%d ",IW[i].idx);
fclose(fp);
printf("I(W) = %3.5e\t",IW0);
printf("%d*I(W) = %3.5e\n",n,n*IW0);
printf("Sum_i I(W_%d~(i)) = %3.5e\n",n,sumI);
printf("Sort I(W_%d~(i)), and creat %s \n",n,fn);
}

int main(int argc, char *argv[]){

    FILE *fp;
    int n;
    double p;
    char fn[50]="sort_I_";
    char footer[5]=".dat";

    if(argc!=2) {
        printf("Usage: %s p \n",argv[0]); exit(1); }

    strcat(fn,argv[1]);
    strcat(fn,footer);

    if((fp=fopen(fn,"w"))==NULL) exit(1);

    p=atof(argv[1]); // p は BSC で  $p=W(1|0)=W(0|1)$ 

```

```

if(p<0.0||p>1.0) {
    fprintf(stderr,"Bad parameter: 0 <= p <= 1.0\n");
    exit(1);
}

n=1<<m; // 最終的に作成する系列長  $2^m$ 

calI(p,n,fn,fp);
return 0;
}

次に,  $z_0^{n-1}$  のサンプル生成とその符号化と復号の操作を
各符号化レート  $\frac{k}{n}, k = 0, 1, \dots, n$  ごとに 1 万回の試行を
繰返し, 復号による誤りの割合を評価するプログラムとして
channel_enc_dec.c を記述する. 実行は channel_enc_dec p
であり, インデックス集合  $\mathcal{M}$  は sort_I_p.dat に基づき定め,
通信路は  $W(1|0) = W(0|1) = p$  の二元対称通信路を仮定して
いる. 二元対称通信路と異なる通信路  $W$  を考えるには, 関数
init_nd(char *y) と main 関数にある符号器と復号器の間の
通信路の設定を変更する. 符号長を大きくするには, 演算精度
を考慮に入れるとともに, main 関数の 7 行目にある int m=3;
の m を大きくする.

//
// channel_enc_dec.c
//
// % channel_enc_dec p
// により 各符号化レート  $k/n, k = 0, 1, \dots, n$ 
// における復号誤りの割合の評価を
// 試行回数 times=10000 で試みる.
//
// アルゴリズムの実行例 ./channel_enc_dec p
// % ./channel_enc_dec 0.11
// n = 8
// coding rate = 0/8, blk err rate = 0/10000
// coding rate = 1/8, blk err rate = 33/10000
// coding rate = 2/8, blk err rate = 676/10000
// coding rate = 3/8, blk err rate = 1325/10000
// coding rate = 4/8, blk err rate = 1756/10000
// coding rate = 5/8, blk err rate = 3956/10000
// coding rate = 6/8, blk err rate = 4969/10000
// coding rate = 7/8, blk err rate = 5578/10000
// coding rate = 8/8, blk err rate = 6021/10000
//
// 最後の分数は 1 万回の試行回数における
// ブロック誤りの割合である.
//
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <limits.h>
#include <time.h>

```

```

#include "common.c"

#define times 10000 // 試行回数

// L のための初期化 (二元対称通信路の場合)
void init_ndw(double p, char *y) {
    int i, j;
    for(i=0; i<n; i++) {
        nd[i][m].zero=y[i]?p:1-p; //BSC,
        nd[i][m].one=y[i]?1-p:p; //p=W(1|0)=W(0|1)
        for(j=0; j<m; j++)
            nd[i][j].zero=nd[i][j].one=-1.0;
    }
}

// L の計算 (再帰呼出し)
ldi L(int i, int j, char *u2, int len){
    char *u2_;
    ldi tmpL;
    double sum;
    int k=1<<(m-j), k2=k<<1, t, ix=i^k, lenh=len>>1;

    u2_=(char *)malloc(lenh*sizeof(char));
    if(nd[i][j].zero<0.0){ // 未計算の場合
        if(lenh>0) {
            if((i%k2)<k) for(t=0; t<lenh; t++)
                u2_[t]=u2[2*t]^u2[2*t+1];
            else for(t=0; t<lenh; t++) u2_[t]=u2[2*t+1];
        }
        nd[i][j]=L(i, j+1, u2_, lenh);
    }
    if(nd[ix][j].zero<0.0){ // 未計算の場合
        if(lenh>0) {
            if((ix%k2)<k) for(t=0; t<lenh; t++)
                u2_[t]=u2[2*t]^u2[2*t+1];
            else for(t=0; t<lenh; t++) u2_[t]=u2[2*t+1];
        }
        nd[ix][j]=L(ix, j+1, u2_, lenh);
    }
    tmpL.zero=(len&1)? // L の分子 (28), (29) 参照
        (u2[len-1]==1?nd[ix][j].one:nd[ix][j].zero)
        *nd[i][j].zero:
        nd[i][j].zero*nd[ix][j].zero+nd[i][j].one
        *nd[ix][j].one;
    tmpL.one=(len&1)? // L の分母 (28), (29) 参照
        (u2[len-1]==1?nd[ix][j].zero:nd[ix][j].one)
        *nd[i][j].one:
        nd[i][j].zero*nd[ix][j].one+nd[i][j].one
        *nd[ix][j].zero;
    sum=tmpL.zero+tmpL.one;

    tmpL.zero/=sum;
    tmpL.one/=sum;
    free(u2_);
    return tmpL;
}

void init(char *eb, int *fb, char *fn, FILE *fp){
    int i;

    nd=(ldi **)malloc(n*sizeof(ldi*));
    for(i=0; i<n; i++)
        nd[i]=(ldi *)malloc((m+1)*sizeof(ldi));

    for(i=0; i<n; i++) { // eb, fb の初期化
        eb[i]=0;
        if(fscanf(fp, "%d ", &fb[i])==EOF) {
            fprintf(stderr, "File error %s.\n", fn);
            exit(1);
        }
    }
    fclose(fp);

    srand(time(NULL));
}

void eval_blk_err(double p, char *eb, int *fb){
    char x[n], y[n], z[n], hatz[n];
    int i, ir, j, ers=0, tms; // ers はブロック誤り回数

    for(j=0; j<=n; j++){ // j は情報ビット数
        if(ers!=times) ers=0; else break;
        for(tms=0; tms<times; tms++){
            // サンプル生成
            // z を一様分布で作成
            for(i=0; i<n; i++) z[i]=rand()&1;

            // 符号器
            // ここから z から符号語 x を生成
            for(i=0; i<n; i++) x[bitrev(i)]=z[i];
            matrixt(x); // z を保持するため matrixt(x) が後
            // ここまで符号化操作

            // 通信路
            // ここから通信路の設定, 入力 x
            // 符号語 x[0..n-1] を通信路 BSC(p) で伝送
            for(i=0; i<n; i++) //BSC channel
                if(rand()/(double)INT_MAX<p)
                    y[i]=x[i]^1; // ビット誤りが発生
                else y[i]=x[i];
            // ここまで通信路の設定, 出力 y

```

```

// 復号器
// ここから y を受け取り逐次復号操作
init_ndw(p,y); // L のための初期化
for(i=0;i<n;i++)
    if(eb[i]) { // 情報ビット推定の場合
        ir = bitrev(i);
        nd[ir][0]=L(ir,1,hatz,i);
        hatz[i]=nd[ir][0].zero<nd[ir][0].one;
        if(hatz[i]!=z[i]) { // 復号誤りの場合
            ers++;break; // ブロック誤り回数++
        }
    }
}

else hatz[i]=z[i]; // 既知ビットの場合
// ここまで復号操作
}

// 試行結果表示
printf("coding rate=%4d/%4d,\t",j,n);
printf("blk err rate=%5d/%5d\n",ers,tms);
if(j<n) eb[fb[n-j-1]]=1; // 情報ビット数++
}
}

int main(int argc,char *argv[]){
    FILE *fp;
    char fn[50]="sort_I_";
    char footer[]=".dat";
    double p;

    m = 3; // 入力系列長 n=2^m
    n=1<<m;

    if(argc!=2) {
        printf("Usage: %s p \n",argv[0]);
        exit(1); }

    strcat(fn,argv[1]);
    strcat(fn,footer);

    if((fp=fopen(fn,"r"))==NULL){
        fprintf(stderr,"File error: %s.\n",fn);
        exit(1); }

    p=atof(argv[1]); // p は BSC で p=W(1|0)=W(0|1)
    if(p<0||p>1) {
        fprintf(stderr,"Bad parameter: 0 <= p <= 1.0\n");
        exit(1);
    }

    char eb[n];

```

```

int fb[n];

// eb, fb の初期化
init(eb,fb,fn,fp);
// 復号誤り割合の試行評価
eval_blk_err(p,eb,fb);

return 0;
}

```

7.3 $I(W_n^{(i)})$ の近似計算

$I(W_{2^m}^{(i)})$, $i \in \mathcal{I}_{2^m}$ の計算に式 (22) と式 (23) を用いた場合、通信路 $W_{2^m}^{(i)}$ からの出力記号の集合の要素数 $|\mathcal{Y} \times \mathcal{Z}^i|$ が指数関数的に大きくなり $I(W_{2^m}^{(i)})$ の計算に困難が生じる。この困難を回避する方法の一つとして通信路 $W_{2^m}^{(i)}$ の近似を用いる。

森と田中⁽¹⁸⁾は対数ゆ度密度関数を計算する密度発展法 (density evolution)⁽¹⁹⁾を用いて $O(n)$ の計算複雑度で $\mathcal{M}$ を選択する方法を提案した⁽²⁰⁾。文献 (21) (22) は、密度発展法における量子化方法に関して、通信路 $W_n^{(i)}$ の出力において $|\mathcal{Y} \times \mathcal{Z}^i|$ をある一定値以下を満たすように通信路出力を量子化により通信路 $\hat{W}_n^{(i)}$ を定めることを提案している。文献 (23) では、通信路 W が二元入力対称通信路であるとき、Kurkoski と Yagi⁽²⁴⁾ が提案した動的計画法に基づく近似により通信路 $\hat{W}_n^{(i)}$ を求める方法を述べている。ここでは、 W が二元対称通信路として文献 (23) の方法を用いて、 $I(W_n^{(i)}) \approx I(\hat{W}_n^{(i)})$ なる近似値 $I(\hat{W}_n^{(i)})$ を求め、 $\mathcal{M}$ の代用を定める C 言語によるプログラミング例を記述する。与えられた通信路 W が二元入力対称通信路であれば文献 (25) (26) にあるように通信路 W を二元対称通信路の重み付和で等価な通信路が構成できる。それゆえ、ここでの近似法は二元入力対称通信路に用いることができる。文献 (23) の $I(W_n^{(i)})$ を近似する方法は、 $H(U_i|U_0^{i-1})$ を近似する方法にも適応可能である^(注10)。

また、Hassani, Mori, Tanaka, Urbanke⁽²⁷⁾ は $\mathcal{M}$ の選択において、インデックス $i \in \mathcal{M}$ に対して、インデックス i の 2 進数展開における最初の $\log n$ bit 程度を通信路 W に依存して選択し、インデックス i の二進数展開における残りの部分を 2 進展開で 1 の個数が多いインデックスを選択する (Reed-Muller 符号と同様の選択法) の方法を再帰的に適応することにより、漸近的に最適なインデックスの集合を選択できることを示している。

```

//
// calIdmcDp.c
//
// 実行は calIdmc m p alpha
//
// m は 符号長 n=2^m
// p は BSC の cross over probability
// p は 0.0 <= p <= 0.5

```

(注10): 図 11 の $m \geq 8$ では $H(U_i|U_0^{i-1})$ に関してこの近似方法を用いた。

```

// alpha は 近似に用いる BSC の最大個数
//
// アルゴリズムの実行例
// % cal_I_dmc 6 0.11 5
// 出力は calIdmc.c の場合と同じであるが近似をしている
//

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>

typedef struct sub_bsc{
    double w; // weight
    double p; // probability of  $W(1|0) = W(0|1) = p$ 
} sub_bsc;

typedef struct channel{
    int size;
    int hw; // index の Hamming weight
    sub_bsc *c;
} channel;

typedef struct sortidx{
    int idx;
    int hw; // idx の Hamming weight
    double inf;
} sortidx;

typedef struct dpnode{
    int idx;
    double inf;
} dpnode;

channel Wtmp;
static int overflow=0;

int compare_sub(const void *a, const void *b)
{
    sub_bsc ta = *(sub_bsc *)a;
    sub_bsc tb = *(sub_bsc *)b;
    if (ta.p > tb.p) return(1);
    else if (ta.p < tb.p) return(-1);
    else return(0);
}

int compare_inf(const void *a, const void *b)
{
    sortidx ta = *(sortidx *)a;
    sortidx tb = *(sortidx *)b;

```

```

    if (ta.inf > tb.inf) return(1);
    else if (ta.inf < tb.inf) return(-1);
    else if (ta.hw > tb.hw) return(1);
    else if (ta.hw < tb.hw) return(-1);
    else if (ta.idx > tb.idx) return(1);
    else if (ta.idx < tb.idx) return(-1);
    else return(0);
}

double mutualinf(double p) {
    double mi;
    if(p<=0.0||p>=1.0) return 1.0;
    mi = 1.0 + p * log2(p) +(1.0-p)*log2(1.0-p);
    return (mi<=0.0)? 0.0:mi;
}

void dp(int alpha){
    int i,j,k;
    int idxs[alpha];
    dpnode dpn[alpha][Wtmp.size-alpha+1];
    double wsum=0.0,psum,min;

    qsort(Wtmp.c,Wtmp.size,sizeof(sub_bsc),compare_sub);

    for(i=0;i<alpha-1;i++){
        for(j=0;j<Wtmp.size-alpha+1;j++){
            if(i==0){
                psum
                    =(wsum*psum+Wtmp.c[j].w*Wtmp.c[j].p)
                    /(wsum+Wtmp.c[j].w);
                wsum+=Wtmp.c[j].w;
                dpn[i][j].inf = wsum*mutualinf(psum);
                dpn[i][j].idx=0;
            } else{
                for(wsum=0.0,min=-1.0,k=j;k>=0;k--){
                    psum
                        =(wsum*psum+Wtmp.c[i+k].w*Wtmp.c[i+k].p)
                        /(wsum+Wtmp.c[i+k].w);
                    wsum+=Wtmp.c[i+k].w;
                    if(wsum*mutualinf(psum)+dpn[i-1][k].inf>min){
                        dpn[i][j].idx=k;
                        min=wsum*mutualinf(psum)+dpn[i-1][k].inf;
                    }
                }
                dpn[i][j].inf=min;
            }
        }
    }
}

```



```

for(wsum=0.0,min=-1.0,k=Wtmp.size-alpha;k>=0;k--){
    psum
    =(wsum*psum+Wtmp.c[i+k].w*Wtmp.c[i+k].p)
    /(wsum+Wtmp.c[i+k].w);
    wsum+=Wtmp.c[i+k].w;
    if(wsum*mutilinf(psum)+dpn[alpha-2][k].inf>min){
        dpn[alpha-1][Wtmp.size-alpha].idx=k;
        min=wsum*mutilinf(psum)+dpn[alpha-2][k].inf;
    }
}
dpn[alpha-1][Wtmp.size-alpha].inf=min;

idxs[alpha-1]=Wtmp.size;
j=dpn[alpha-1][Wtmp.size-alpha].idx;
for(i=alpha-2;i>=0;i--){
    idxs[i]=i+j+1;
    j=dpn[i][j].idx;
}

for(i=j=0;i<alpha;i++){
    for(wsum=psum=0.0;j<idxs[i];j++) {
        wsum+=Wtmp.c[j].w;
        psum+=Wtmp.c[j].w*Wtmp.c[j].p;
        Wtmp.c[i].w=(wsum<0.0)? 0.0:wsum;
        Wtmp.c[i].p=(psum/wsum<0.0)? 0.0:psum/wsum;
    }
}
Wtmp.size = alpha;
}

void do_even(channel Q, int alpha){ //XOR がある場合
    int i,j,k;
    double tw,tp;

    for(k=0,i=0;i<Q.size;i++){
        for(j=i;j<Q.size;j++){
            tw = Q.c[i].w * Q.c[j].w;
            tp = Q.c[i].p+Q.c[j].p-2.0*Q.c[i].p*Q.c[j].p;
            if ( 0.0<tw && 0.0<=tp && tp<=1.0 ) {
                Wtmp.c[k].w = (j==i)? tw:2.0*tw ;
                Wtmp.c[k++].p
                    = ( 0.0<=tp && tp<= 0.5 ) ? tp:1.0-tp;
            } else overflow++;
        }
    }
    Wtmp.size = k;
    if (Wtmp.size > alpha) dp(alpha);
    Wtmp.hw = Q.hw+1;
}

void do_odd(channel Q, int alpha){ //XOR がない場合
    int i,j,k;
    double tw,tp;

    for(k=0,i=0;i<Q.size;i++){
        for(j=i;j<Q.size;j++){
            tw = Q.c[i].w * Q.c[j].w *
                (Q.c[i].p*(1.0-Q.c[j].p)
                +(1.0-Q.c[i].p)*Q.c[j].p);
            tp = (1.0-Q.c[i].p)*Q.c[j].p
                / (Q.c[i].p*(1.0-Q.c[j].p)
                +(1.0-Q.c[i].p)*Q.c[j].p);
            if ( 0.0<tw && 0.0<=tp && tp<=1.0 ) {
                Wtmp.c[k].w = (j==i)? tw:2.0*tw ;
                Wtmp.c[k++].p
                    = ( 0.0<=tp && tp<= 0.5 ) ? tp:1.0-tp;
            } else overflow++;
        }
    }
    Wtmp.size = k;
    if (Wtmp.size > alpha) dp(alpha);
    Wtmp.hw = Q.hw+1;
}

void calI(double p, int m, int alpha,
          char *fn, FILE *fp) {
    int i,j,k,l,n;
    channel *W,*Q;
    sortidx *IofW; // I(W_n^{(i)}) を IW[i] に保持
    double IW0,sumI = 0.0; // I(W_n^{(i)}) の和=n*I(W)

    n=1<m; // 最終的に作成する系列長 2^m

    W = (channel *)malloc(n*sizeof(channel));
    Q = (channel *)malloc(n*sizeof(channel));
    for(i=0;i<n;i++){
        W[i].c = (sub_bsc *)malloc(alpha*
                                     sizeof(sub_bsc));
        Q[i].c = (sub_bsc *)malloc(alpha*

```

```

                                     sizeof(sub_bsc));
        Q[i].c = (sub_bsc *)malloc(alpha*

```

```

        sizeof(sub_bsc));
    }

    Wtmp.c = (sub_bsc *)malloc(2*alpha*alpha*
        sizeof(sub_bsc));

    IofW = (sortidx *)malloc(n*sizeof(sortidx));

    W[0].size=1; // W.size は W における BSC 分解の個数
    W[0].hw=0; // W.hw はインデックスのハミングウエイト
    W[0].c[0].w = 1.0; // W の初期化で BSC の w = 1.0
    W[0].c[0].p = p; // W の初期化で BSC の p
    IW0=mutualinf(W[0].c[0].p); // I(W) = I(BSC(p))
    for(k=1,j=0;j<m;j++){
        for(i=0;i<k;i++) {
            do_even(W[i],alpha);
            Q[2*i].size=Wtmp.size;
            Q[2*i].hw=Wtmp.hw;
            for(l=0;l<Wtmp.size;l++) {
                Q[2*i].c[l].w=Wtmp.c[l].w;
                Q[2*i].c[l].p=Wtmp.c[l].p;
            }
            do_odd(W[i],alpha);
            Q[2*i+1].size=Wtmp.size;
            Q[2*i+1].hw=Wtmp.hw;
            for(l=0;l<Wtmp.size;l++) {
                Q[2*i+1].c[l].w=Wtmp.c[l].w;
                Q[2*i+1].c[l].p=Wtmp.c[l].p;
            }
        }
    }
    k<=1;
    for(i=0;i<k;i++) {
        W[i].size=Q[i].size;
        W[i].hw=Q[i].hw;
        for(l=0;l<Q[i].size;l++) {
            W[i].c[l].w =Q[i].c[l].w;
            W[i].c[l].p =Q[i].c[l].p;
        }
    }
}

//情報量 IofW[i] の計算
for(i=0;i<n;i++) {
    IofW[i].idx=i;
    IofW[i].hw=W[i].hw;
    for(IofW[i].inf=0.0,k=0;k<W[i].size;k++)
        if( W[i].c[k].w > 0.0)
            IofW[i].inf +=
                W[i].c[k].w*mutualinf(W[i].c[k].p);
    if( IofW[i].inf > 1.0 ) IofW[i].inf = 1.0;

    sumI += IofW[i].inf;
}

printf("\n");

qsort(IofW,n,sizeof(sortidx),compare_inf);
for(i=0;i<n;i++) {
    fprintf(fp,"%d ",IofW[i].idx);
    printf("%8dth I(W_%d^(%d)) = %1.15e \n"
        ,i,n,IofW[i].idx,IofW[i].inf);
}

fclose(fp);
printf("I(W) = %3.5e \n",IW0);
printf("%d*I(W) = %3.5e \n",n,n*IW0);
printf("Sum_i I(W_%d^(i)) = %3.5e \n",n,sumI);
printf("Sort I(W_%d^(i)), and creat %s \n",n,fn);
}

int main(int argc, char *argv[]){

    FILE *fp;
    int m,alpha;
    double p;
    char fn[50]="sort_I_";
    char footer[]=".dat";

    if(argc!=4) {
        printf("Usage: %s m p alpha \n",argv[0]);
        exit(1);
    }

    strcat(fn,argv[1]);
    strcat(fn,"_");
    strcat(fn,argv[2]);
    strcat(fn,"_");
    strcat(fn,argv[3]);
    strcat(fn,footer);

    if((fp=fopen(fn,"w"))==NULL) exit(1);
    m=atoi(argv[1]); // n=2^m
    if(m<0||m>20){
        fprintf(stderr,"Bad parameter: 0 <= m <= 20");
        exit(1);
    }

    p=atof(argv[2]); // BSC の cross over probability
    if(p<0||p>0.5){ // p は 0 <= p <= 0.5
        fprintf(stderr,"Bad parameter: 0 <= p <= 0.5 \n");
        exit(1);
    }
    alpha=atoi(argv[3]); // alpha は BSC の最大個数

```

```
//出力アルファベットサイズは 2*alpha
if(alpha<3||alpha>10){
    fprintf(stderr,"Bad parameter: 3<=alpha<=10 \ n");
    exit(1);
}

calI(p,m,alpha,fn,fp);
return(0);
}
```

8. む す び

本稿では polar 符号の紹介を行った．Arıkan による polar 符号の発表以来，polar 符号に関する多くの研究がなされている．polar 符号に興味を持たれた方は，田中の文献(10)と森の文献(12)及び発表スライド(28)により，森と田中が貢献した Arıkan の発表以降における polar 符号の一般化やその後の発展について理解が得られる．二元の 2 行 2 列の行列 T に基づく再帰的な変換による polar 符号は，有限体の多元の l 行 l 列の行列に基づく再帰的な変換による polar 符号へ一般化がなされている⁽²⁵⁾．文献(27)では多元 polar 符号により有限長の符号長における顕著な誤り率の改善がなされている．Abbe⁽²⁹⁾は有限体上の多元 polar 符号へ拡張において 1 回の変換行列 T において最も情報量の差が生じるように T を定めること，つまり，式(3)の $H(U_1|U_0)$ を最小にするように有限体上の T を定めることを提案している．これは情報源符号や加法的通信路における通信路符号における polar 符号の設計に関する基本的な一つの指針と考えられる．文献(10)のむすびで述べられているように，符号の構成法(行列 T の定め方と再帰構成の方法及び M の選び方)と復号法を組み合わせる工夫をすることで符号の性能が改善できる可能性があり，短い符号長での性能についても研究がなされている．polar 符号を組織符号化として構成することでビット誤り率を改善する方法も提案されている⁽³⁰⁾．

任意の通信路においてその通信路容量を達成する polar 符号の拡張は，文献(31)～(33)に提案されている．フィードバック通信路を利用した自動再送要求(ARQ)に polar 符号を用いた Hybrid-ARQ 通信方式への拡張⁽¹³⁾もなされている．有ひずみを考慮した情報源符号の拡張は，文献(25)(34)に提案されている．多端子情報理論の問題について通信の理論的境界の解明がなされる⁽³⁵⁾とともに，polar 符号を用いて理論的境界を達成する符号化法について研究が試みられている^{(4)(9)(25)(34)(36)～(39)}．

符号化問題の理論的境界を達成する符号化法を明確にする基本的な原理が polar 符号の発展によりもたらされることが期待される．

謝辞 この解説論文の基になる講演の機会を与えて頂いた情報理論研究専門委員会の先生方に深謝する．本稿の執筆において，東京工業大学の植松友彦先生，筑波大学の古賀弘樹先生，日本電信電話株式会社の村松純先生，湘南工科大学の有村光晴先生から多くの御助言を頂いた．懇切丁寧な御査読下さいました匿名のお二人の先生方と工学院大学の斎藤秀俊先生から貴重な

数多くの御意見を頂いた．本稿の C 言語によるプログラミングと図の作成には鈴木佑輔君と小澤伸也君に御協力を頂いた．本研究の一部は JSPS 科研費 23560447 の助成を受けた．ここに記して深く感謝を申し上げる．

文 献

- (1) C. E. Shannon, "A mathematical theory of communication," Bell Syst. Tech. J., vol.27, pp.379-423, 623-656, July/Oct. 1948.
- (2) E. Arıkan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," IEEE Trans. Inf. Theory, vol.55, no.7, pp.3051-3073, July 2009.
- (3) 今井秀樹, 萩原 学, "日本における符号理論の原点," IEICE Fundamentals Review, vol.2, no.4, pp.9-15, April 2009.
- (4) E. Arıkan, "Source polarization," Proc. of 2010 IEEE Inter. Symp. on Inf. Theory, pp.899-903, Austin, Texas, USA, June 2010.
- (5) 韓 太舜, 情報理論における情報スペクトル的方法, 培風館, 東京, 1998. (T. S. Han, Information-Spectrum Methods in Information Theory, Springer, 2003.)
- (6) T. M. Cover and J. A. Thomas, Elements of Information Theory, Wiley-Interscience, 1991, 2nd ed., Wiley-Interscience, 2006. (山本博資, 古賀弘樹, 有村光晴, 岩本貢(訳), 情報理論 基礎と広がり, 共立出版, 東京, 2012.)
- (7) I. Csiszár and J. Körner, Information Theory: Coding Theorems for Discrete Memoryless Systems, Academic Press, 1981, 2nd ed., Cambridge University Press, 2011.
- (8) 植松友彦, 現代シャノン理論タイプによる情報理論, 培風館, 東京, 1998.
- (9) E. Şaşoğlu, Polar Coding Theorems for Discrete Systems, Ph.D. Thesis, EPFL, 2011. [Online] Available: <http://library.epfl.ch/en/theses/?nr=5219>, 2011.
- (10) 田中利幸, "通信路分極とポーラ符号," 信学技報, IT2009-50, pp.23-30, Dec. 2009.
- (11) 森 立平, 田中利幸, "通信路分極現象のレートに依存した漸近的な評価," 第 32 回情報理論とその応用シンポジウム, pp.279-284, Dec. 2009.
- (12) 森 立平, "Polar 符号について," 信学技報, IT2010-41, pp.43-49, Sept. 2010.
- (13) 永田基樹, 山本博資, "Polar 符号を用いた Hybrid-ARQ 方式," 信学技報, IT2012-25, pp.97-102, July 2012.
- (14) E. Arıkan and E. Telatar, "On the rate of channel polarization," Proc. 2009 IEEE Int. Symp. Inf. Theory, pp.1493-1495, Seoul, S. Korea, June/July 2009. [online] Available: arXiv:0807.3806v3 [cs.IT], 2009.
- (15) 林 正人, "情報スペクトルによる二次オーダーの情報理論—一次漸近理論を超えて—," IEICE Fundamentals Review, vol.6, no.1, pp.12-25, July 2012.
- (16) T. Tanaka, "On speed of channel polarization," Proc. of 2010 IEEE Inf. Theory Workshop, pp.1-5, Dublin, Ireland, Sept. 2010.
- (17) 植松友彦, "シャノン理論—通信路符号化定理の今昔—," IEICE Fundamentals Review, vol.1, no.1, pp.8-18, July 2007.
- (18) R. Mori and T. Tanaka, "Performance of polar codes with the construction using density evolution," IEEE Commun. Lett., vol.13, no.7, pp.519-521, July 2009.
- (19) T. Richardson and R. L. Urbanke, Modern Coding Theory, Cambridge University Press, 2008.
- (20) R. Mori, Properties and Construction of Polar Codes, Master Thesis, Kyoto University, 2010. [Online] Available: arXiv:1002.3521v1 [cs.IT], 2010.
- (21) I. Tal and A. Vardy, "How to construct polar codes," Proc. of 2010 IEEE Inf. Theory Workshop, Dublin, Ireland, Sept. 2010, [Online] Available: arXiv:1105.6164v1 [cs.IT], 2011.
- (22) R. Pedarsani, S. H. Hassani, I. Tal, and E. Telatar, "On the construction of polar codes," Proc. of 2011 IEEE

Int. Symp. on Inf. Theory, pp.11-15, Saint-Petersburg, Russia, July/Aug. 2011.

- (23) 鈴木佑輔, 岩田賢一, “2 入力対称通信路における近似 Polar 符号の動的計画法を用いた符号構成に関する考察,” 信学技報, IT2011-42, pp.19-24, Jan. 2012.
- (24) B. Kurkoski and H. Yagi, “Quantization of binary-input discrete memoryless channels, with applications to LDPC decoding,” [Online] Available: arXiv:1107.5637v1 [cs.IT], 2011.
- (25) S. B. Korada, Polar Codes for Channel and Source Coding, Ph.D. Thesis, EPFL, 2009, [Online] Available: <http://library.epfl.ch/theses/?nr=4461>, 2009.
- (26) I. Land and J. B. Huber, Information Combining, ser. Foundations and Trends in Communications and Information Theory, Now, vol. 3, 2006.
- (27) S. H. Hassani, R. Mori, T. Tanaka, and R. L. Urbanke, “Rate-dependent analysis of the asymptotic behavior of channel polarization,” [Online] Available: arXiv:1110.0194v2 [cs.IT], 2011.
- (28) 森 立平, “Polar 符号とその研究動向,” 第 34 回情報理論とその応用シンポジウム, ワークショップ資料, Nov./Dec. 2011, [Online] Available: http://www-adsys.sys.i.kyoto-u.ac.jp/rmori/slides/Mori_SITA2011_Workshop.pdf
- (29) E. Abbe, “Polar martingales of maximal spread,” Proc. of Int. Zurich Seminar on Communications, pp.44-47, Zurich, Switzerland, Feb.-March. 2012.
- (30) E. Arıkan, “Systematic polar coding,” IEEE communications letters, vol.15, no.8, pp.860-862, Aug. 2011.
- (31) J. Honda and H. Yamamoto, “Polar coding without alphabet extension for asymmetric channels,” Proc. of 2012 IEEE Inter. Symp. on Inf. Theory, pp.2147-2151, Cambridge, MA, USA, July 2012.
- (32) E. Şaşıoğlu, E. Telatar, and E. Arıkan, “Polarization for arbitrary discrete memoryless channels,” Proc. 2009 IEEE Inf. Theory Workshop, pp.144-148, Taormina, Sicily, Italy, Oct. 2009, [online] Available: arXiv:0908.0302v1 [cs.IT], 2009.
- (33) D. Sutter, J. M. Renes, F. Dupuis, R. Renner, “Achieving the capacity of any DMC using only polar codes,” [online] Available: arXiv:1205.3756v2 [cs.IT], 2012.
- (34) S. B. Korada and R. L. Urbanke, “Polar codes are optimal for lossy source coding,” IEEE Trans. Inf. Theory, vol.56, no.4, pp.1751-1768, April 2010.
- (35) 大濱靖匡, “多端子情報源符号化の研究と未解決問題,” Fundamentals Review, vol. 2, no. 2, pp.40-52, Oct. 2008.
- (36) E. Arıkan, “Polar coding for the slepian-wolf problem based on monotone chain rules,” Proc. of 2012 IEEE Inter. Symp. on Inf. Theory, pp.566-570, Cambridge, MA, USA, July 2012.
- (37) E. Abbe and E. Telatar, “Polar codes for the m -user multiple access channel,” IEEE Trans. Inf. Theory, vol.58, no.8, pp.5437-5448, Aug. 2012.
- (38) I. Tal, A. Sharov, and A. Vardy, “Constructing polar codes for non-binary alphabets and MACs,” Proc. of 2012 IEEE Inter. Symp. on Inf. Theory, pp.2132-2136, Cambridge, MA, USA, July 2012.
- (39) H. MahdaviFar and A. Vardy, “Achieving the secrecy capacity of wiretap channels using polar codes,” IEEE Trans. Inf. Theory, vol.57, no.10, pp.6428-6443, Oct. 2011.

(IT 研究会提案, 平成 24 年 9 月 10 日受付)

11 月 9 日最終受付)

岩田賢一 (正員)

1993 和歌山大・教育卒. 1995 北陸先端大大学院情報科学研究科博士前期課程了. 1996 同大学院博士後期課程中退. 徳島大助手, 松山東雲女子大講師, 電通大助手, 山形大助教授, 准教授を経て, 現在, 福井大大学院工学研究科准教授. 博士 (工学). 情報理論の研究に従事.