信州大学工学部

学士論文

# 制約付き乱数に基づく符号の Z 通信路に対する基礎的検証

指導教員　西新 幹彦 准教授

学科　電子情報システム工学科
学籍番号　19T2145B
氏名　南澤 航

2023 年 3 月 14 日

# 目次

# 1　はじめに

通信を行う際，送信元から受信先に正しくメッセージ（情報）を伝えられない場合がある．これは，途中経路である通信路において雑音が発生し，本来伝えようとしていた情報に誤りが生じてしまうためである．このため，発生した誤りを検出し正しく復号する，誤り訂正技術は雑音のある環境下において必要不可欠である．

Shannon は文献 [1] において，雑音のある通信路が与えられた場合，その通信路において，メッセージを正しく送ることができる効率の上限を示した．この上限は通信路容量と呼ばれ，通信路容量を達成するような符号は文献 [1] ですでに提案されている．しかし，この符号では計算時間，メモリ量が指数的に増大するため，現実的な実行は困難である．このため，通信路容量を達成するような実行可能な符号を考えることは，情報理論における課題とされてきた．

従来の符号技術では，実装の容易さから線形符号と呼ばれるものが用いられ，これらの符号の一部は通信路容量に近い性能を示すことが確認されている．しかし，線形符号が通信路容量を達成するのは，線形符号の性質上，特定の通信路に限られており一般には通信路容量を達成するとは限らない．そこで，一般の通信路に対して通信路容量を達成する符号技術として文献 [2] において CoCoNuTS(Code based on Constrained Numbers Theoretically-achieving the Shannon limit，拘束条件を満たす系列に基づくシャノン限界を達成する符号）が提案された．この符号は，特殊な乱数生成器を符号器，復号器で使用することによって，通信路容量を達成することが理論的に証明されている．

本研究では文献 [2] で提案された手法に従って，CoCoNuTS の符号器，復号器を作成し，送受信シミュレーションを行うことによって，その基本的な性質を確認する．また，従来の線形符号では通信路容量を達成できないような通信路での復号誤り確率を計測し，線形符号であるハミング（Hamming）符号，リードソロモン (Reed-Solomon) 符号と比較することにより，復号性能を検証する．

# 2　通信路容量

この章では本研究で扱う通信モデルと，符号の性能を測る尺度を説明する．また，誤りのない通信を行う効率の限界である，通信路容量を定義し，通信路容量を達成するような最適な入力分布がどのようなものになるかを，具体的な通信路を例にしながら述べる．


図 1　通信路モデル

## 2.1　符号の性能

図 1 は本研究で使用される通信路モデルである．ここで，通信路とは現実の通信システムを数学的にモデル化したものであり，通信路入出力 $X$ と $Y$ との間の条件付き確率分布 $P_{Y|X}$ で表される．また，通信路入力 $X$ と通信路出力 $Y$ の分布はそれぞれ $P_X$，$P_Y$ と表される．これらの確率は，通信システムを考える上で重要であり，符号器や復号器の設計と深く関係する．以下では，このような確率モデルとして与えられた通信路に対して，符号の性能がどのように測られているのかを述べていく．

図 1 において，符号器では送信者から受け取ったメッセージ $M$ を符号化して通信路入力 $X$ へ変換する．変換された $X$ は通信路へと入力され，復号器は通信路出力 $Y$ を得る．ここで，通信路において発生した雑音の影響により，通信路入出力 $X$ と $Y$ において $X \neq Y$ となるような場合が考えられる．このため，復号器では観測された $Y$ をもとに送信メッセージ $M$ の推定を行い，推定値を $M'$ として出力する．ここで，このようにして出力された $M'$ は，送信者が本来送りたかったメッセージ $M$ とは異なる場合が考えられる．このように $M \neq M'$ となるような確率を復号誤り確率と定義し，符号の性能を測る尺度の一つと考える．

次に通信を行う上での効率を測る尺度である，符号化レートについて定義する．符号語長を $n$，メッセージシンボル数を $M_n$ とした場合，符号化レート $R$ は

$$R = \frac{\log_2 M_n}{n} \tag{1}$$

と表される．符号化レートが大きいほど通信が高速であることを意味するが，大きくしすぎると復号誤り確率を 0 に近づけることができなくなる．

最後に，図 1 で示した一連の送受信操作において，その実行にかかる計算量を性能評価の尺度の一つとする．ここで計算量とは，メッセージを送信し受信するまでに必要な計算手順の数と，それらの計算を行う上で必要な記憶領域のことである．通信を行う際，計算能力の低い端末でも高速な通信を行うことを可能にするため，計算手順は少なく，必要な記憶領域は小さい

方が好ましい.

## 2.2 通信路容量と最適な入力分布

通信を行う上で性能の良い符号とは，符号化レートが可能な限り大きく，同時に復号誤り確率が限りなく 0 に近い符号である．ここで，復号誤り確率を 0 に近づけながら，符号化レートを大きくするとき，大きく出来る符号化レートの上限は，通信路ごとに決まる．このような符号化レートの限界を通信路容量とよび，文献 [2] で次のように定義されている．

**定義１**　与えられた通信路に対して定まる値 $C$ が以下の二つの条件を満たすとき，$C$ をこの通信路の通信路容量と呼ぶ.

- 復号誤り確率が 0 に限りなく近く，かつ符号化レート $R$ が $C$ に限りなく近い符号が存在する.
- 復号誤り確率が 0 に限りなく近く，かつ符号化レート $R$ が $R > C$ を満たす符号は存在しない.

定義 1 のもとで次の定理が成り立つ.

**定理１**　通信路 $P_{Y|X}$ の通信路容量 $C(P_{Y|X})$ は

$$C(P_{Y|X}) = \max_{P_X}(H(X) - H(X|Y)) \tag{2}$$

で表される．ここで，$H(X)$ は $X$ のエントロピー，$H(X|Y)$ は $Y$ が与えられた元での $X$ の条件付きエントロピーと呼ばれ，それぞれ

$$H(X) \triangleq -\sum_x P_X(x) \log P_X(x) \tag{3}$$

$$H(X|Y) \triangleq -\sum_x \sum_y P_{XY}(x,y) \log P_{X|Y}(x|y) \tag{4}$$

と定義される．また，式 (2) の右辺に現われる，$H(X) - H(X|Y)$ は相互情報量 $I(X;Y)$ を用いて

$$C(P_{Y|X}) = \max_{P_X}(I(X;Y)) \tag{5}$$

と置き換えることが出来る.

式 (5) から通信路容量とは相互情報量 $I(X;Y)$ の最大値であるということがわかる．このとき，式 (5) の最大値を与えるような通信路の入力分布 $P_X$ は，最適な入力分布と呼ばれる．またこのとき，入力分布 $P_X$ は通信路容量を達成するという．最適な入力分布は通信路ごとに異なり，性能の良い符号を設計する上で重要となる．次節ではいくつかの通信路を例に，どのような入力分布によって通信路容量が達成されるのかを表していく.

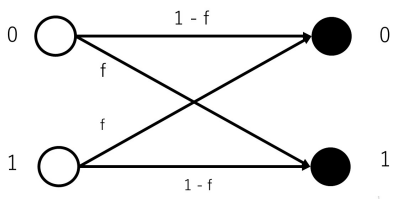


図 2　二元対称通信路

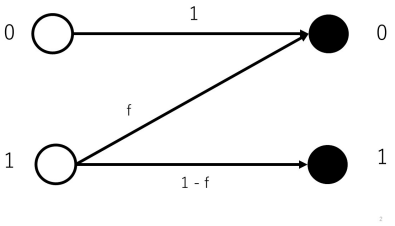


図 3　Z 通信路

## 2.3　通信路の例

代表的な通信路の例として二元対称通信路があげられる．この通信路は図 2 に示すような通信路であり，入力シンボル 0 と 1 がどちらも確率 $f$ で反転するような通信路である．この通信路について，相互情報量 $I(X;Y)$ の上界は

$$I(X;Y) \leq 1 - h(f) \tag{6}$$

で表される [3]．ここで式 (6) の右辺に現われる $h(f)$ は二値エントロピー関数と呼ばれ

$$h(f) \triangleq -f \log_2 f - (1-f) \log_2 (1-f) \tag{7}$$

と定義される．また，式 (6) において等号成立条件は入力分布が一様分布であることが知られている [3]．式 (5) から通信路容量は相互情報量の最大値であるため，二元対称通信路は入力分布が一様分布であるとき通信路容量を達成する．すなわち，一様分布は二元対称通信路の最適な入力分布であることがわかる．

一方，最適な入力分布が一様分布でないような通信路の具体例としては，図 3 で示すような Z 通信路があげられる．Z 通信路は 2 つの入力シンボルのうち一方の反転確率が 0，もう一方の反転確率が $f$ となるような通信路である．

通信路へ 1 が入力される確率を $p$ とおくとき，Z 通信路が通信路容量を達成するような，最適な入力分布は

$$p = \frac{1}{(1-f)(1+2^{\frac{h(f)}{1-f}})} \tag{8}$$

となる．式 (8) より Z 通信路の最適な入力分布は二元対称通信路とは違い，一様分布では与えられず，反転確率 $f$ で決まることがわかる．式 (8) の導出については付録 A で示す．

以上のように通信路容量を達成する最適な入力分布は，一様分布であるとは限らない．このことは，線形符号で通信路容量を達成する通信路が限られることと深く関係している．次章で

は線形符号と最適な入力分布との関係を述べることによって，線形符号で通信路容量を達成する通信路が限られている理由を明らかにする．

# 3 線形符号

本研究で実装，検証を行う CoCoNuTS は，線形符号と密接に関係している．このため本章ではまず，線形符号の構成方法を具体例を用いて述べる．また本研究で，従来の線形符号として CoCoNuTS の比較対象に用いられる，ハミング符号，リードソロモン符号について，その構成や符号化，復号化の仕組みを文献 [4] を元に説明する．さらに，線形符号と通信路容量がどのような関係を持つか，最適な入力分布を踏まえて述べる．

## 3.1 検査行列と生成行列

$p$ 元の要素を持つ体を $\mathbb{F}_p$ と表記する．本節では $\mathbb{F}_2$ 上で定義され，符号語長が 7 となるような線形符号の構成方法の例を示す．

符号語長が 7 の符号語を送信することは，通信路を独立に 7 回使用すると考えることが出来る．このため，符号シンボルは $x_1, x_2, \cdots, x_7$ と表され，符号語 $\boldsymbol{x}$ は $\boldsymbol{x} = (x_1, x_2, \cdots, x_7)$ となる．

線形符号で符号化を行う際，通信路に入力される符号語 $\boldsymbol{x}$ は制約条件として，任意の連立一次方程式の解となるものの中から選ばれる．制約条件は使用する符号ごとに異なるが，ここでは例として

$$\begin{cases} x_1 + x_2 + x_3 = 0 \\ x_1 + x_4 = 0 \\ x_4 + x_5 = 0 \\ x_4 + x_6 = 0 \\ x_4 + x_7 = 0 \end{cases} \tag{9}$$

を制約条件とし，解となるような $\boldsymbol{x}$ を符号語とする．ここで $\boldsymbol{x}$ の成分は $\mathbb{F}_2$ の要素であり，加減乗除も $\mathbb{F}_2$ 上の演算となる．この式 (9) で示した連立一次方程式は $5 \times 7$ 行列を用いて，

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{10}$$

と表すことが出来る．このとき左辺の $5 \times 7$ 行列を検査行列と呼ぶ．検査行列を $H$ と表すと，

符号語が満たす制約条件 (10) は，零列ベクトル $\mathbf{0}$ を用いて

$$H \times \boldsymbol{x} = \mathbf{0} \tag{11}$$

と表現される．

ここで，上記のような符号語と，送信するメッセージとの間には次のような関係がある．検査行列 $H$ と零列ベクトル $\mathbf{0}$ を合体させた拡大係数行列 $H'$ を

$$\left(\begin{array}{ccccccc|c} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{array}\right) \tag{12}$$

としたとき，$H'$ は行列の基本変形を行うと

$$\left(\begin{array}{ccccccc|c} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{array}\right) \tag{13}$$

のように変形できる．ここで，式 (13) で表される拡大係数行列 $H'$ を連立一次方程式の形に戻すと

$$\begin{cases} x_1 + x_2 + x_3 = 0 \\ x_1 + x_4 = 0 \\ x_1 + x_5 = 0 \\ x_1 + x_6 = 0 \\ x_1 + x_7 = 0 \end{cases} \tag{14}$$

となり，恒等式 $x_1 = x_1$, $x_2 = x_2$ を追加して式 (14) を整理すると

$$\begin{cases} x_1 = x_1 \\ x_2 = x_2 \\ x_3 = x_1 + x_2 \\ x_4 = x_1 \\ x_5 = x_1 \\ x_6 = x_1 \\ x_7 = x_1 \end{cases} \tag{15}$$

となる．式 (15) の連立一次方程式を，再び行列の積で表現すると

$$\boldsymbol{x} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \tag{16}$$

のように表され，式 (16) から $x_1$, $x_2$ を定めたときに符号語 $\boldsymbol{x}$ も定まることがわかる．ここで，式 (16) の $7 \times 2$ 行列を $G$ とし，$x_1$, $x_2$ をメッセージベクトル $\boldsymbol{m} = (m_1, m_2)$ とすると式 (16) は

$$\boldsymbol{x} = G \times \boldsymbol{m} \tag{17}$$

となる．式 (17) より，行列 $G$ とメッセージベクトル $\boldsymbol{m}$ の積から，符号語 $\boldsymbol{x}$ を生成することができ，$\boldsymbol{x} = G \times \boldsymbol{m}$ は制約条件 $H\boldsymbol{x} = \boldsymbol{0}$ を変形して得られたものであるため，生成された $\boldsymbol{x}$ は制約条件 $H\boldsymbol{x} = \boldsymbol{0}$ を満たす．以上のように制約条件 $H\boldsymbol{x} = \boldsymbol{0}$ を変形して得られる行列 $G$ を生成行列と呼ぶ．

式 (17) よりメッセージを一つ決めたとき，制約を満たすような符号語 $\boldsymbol{x}$ がただ一つに定まることがわかる．このことは，線形符号の原理上，メッセージと符号語が一対一に対応していることを表している．

上記のようにして生成された符号語 $\boldsymbol{x}$ は通信路を通り，復号器へと入力される．ここで符号語 $\boldsymbol{x}$ は，通信路で生じる雑音の影響を受け，誤りが発生し，復号器へは受信語 $\boldsymbol{y}$ として入力される．復号器ではこのようにして得られた，受信語 $\boldsymbol{y}$ から正しいメッセージを復元する必要がある．このため，復号器はこの受信語 $\boldsymbol{y}$ と，検査行列 $H$ を用いて，次のような計算を行うことにより，発生した誤りについての情報を得る．

受信語 $\boldsymbol{y}$ は，符号語 $\boldsymbol{x}$ と通信路で混入した誤りとの和によって表される．すなわち，通信路で生じた誤りを列ベクトル $\boldsymbol{e} = (e_1, e_2, \cdots, e_7)$ とすると，受信語 $\boldsymbol{y} = (y_1, y_2, \cdots, y_7)$ は

$$\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{e} \tag{18}$$

となる．ここで符号語 $\boldsymbol{x}$ は制約条件 $H\boldsymbol{x} = \boldsymbol{0}$ を満たすことを考慮すると，受信語 $\boldsymbol{y}$ と検査行列 $H$ の積 $\boldsymbol{s}$ は

$$\boldsymbol{s} = H\boldsymbol{y} = H(\boldsymbol{x} + \boldsymbol{e}) = H\boldsymbol{e} \tag{19}$$

と計算される．ここで $\boldsymbol{s}$ は受信語 $\boldsymbol{y}$ のシンドロームと呼ばれる．このようにして計算されるシンドロームは，発生した誤りを特定するための手がかりとなる．

線形符号では，式 (19) の計算によって得られる，シンドローム $\boldsymbol{s} = H\boldsymbol{y}$ を元に送信メッセージを推定する．このため，線形符号の性能は，検査行列 $H$ をどのような構成にするかによって，大きく左右される．次節以降で解説するハミング符号とリードソロモン符号は，検査行列を一定の規則に基づいて定義することにより，通信路で発生した誤りを検出，訂正可能な符号となっている．

## 3.2 ハミング符号

ハミング符号は $\mathbb{F}_2$ 上で定義される線形符号であり，通信路で符号語に生じた任意の 1 シンボル誤りを訂正することが可能である．ここではその符号化，復号化の手順を簡単に説明する．

まず符号化について符号語長が 7 の場合，ハミング符号の検査行列 $H$ は

$$H \triangleq \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \tag{20}$$

のように定義される．式 (20) より，ハミング符号の検査行列の特徴として，検査行列は非ゼロかつ，互いに異なる列ベクトルで構成されていることがあげられる．

ハミング符号は線形符号であるため，ハミング符号の検査行列 $H$ を用いた制約 $H\boldsymbol{x} = \boldsymbol{0}$ を満たすような $\boldsymbol{x}$ が符号語となる．符号器では，この検査行列 $H$ に対して式 (17) で示したような生成行列 $G$ を作成し，生成行列 $G$ とメッセージベクトル $\boldsymbol{m}$ で $\boldsymbol{x} = G \times \boldsymbol{m}$ を計算することによって，符号語 $\boldsymbol{x}$ を生成する．

受信語からメッセージを復号するには，まず受信語 $\boldsymbol{y}$ のシンドローム

$$\boldsymbol{s} = H\boldsymbol{y} = H(\boldsymbol{x} + \boldsymbol{e}) = H\boldsymbol{e} \tag{21}$$

を計算する．ただし，$\boldsymbol{e}$ は通信路での誤りである．もし，シンボル誤りが一か所のみである場合，$\boldsymbol{e}$ は第 $i$ 番目の要素のみが 1 でその他が 0 であるようなベクトルとなるため，シンドローム $\boldsymbol{s}$ はハミング検査行列 $H$ の第 $i$ 列目と等しくなる．すなわち，復号器は受信語からシンドロームを計算し，検査行列の列ベクトルと一致する位置を探すことによって，誤り訂正を行うことが出来る．一方，二か所以上のシンボル誤りが発生した場合シンドローム $\boldsymbol{s}$ から正しい誤り位置を特定できないため，誤り訂正は不可能となる．

## 3.3 リードソロモン符号

ハミング符号が $\mathbb{F}_2$ 上で定義される線形符号であるのに対して，リードソロモン符号は $p$ 元の要素を持つ素体 $\mathbb{F}_p$ 上で定義される線形符号である．本研究では，$\mathbb{F}_2$ の拡大体 $\mathbb{F}_{2^3}$ 上のリードソロモン符号を考える．検査行列は複数あるが，本研究では検査行列 $H$ を

$$H \triangleq \begin{pmatrix} 1 & \alpha^1 & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \\ 1 & \alpha^2 & \alpha^4 & \alpha^6 & \alpha^8 & \alpha^{10} & \alpha^{12} \\ 1 & \alpha^3 & \alpha^6 & \alpha^9 & \alpha^{12} & \alpha^{15} & \alpha^{18} \\ 1 & \alpha^4 & \alpha^8 & \alpha^{12} & \alpha^{16} & \alpha^{20} & \alpha^{24} \end{pmatrix} \tag{22}$$

のように定義する．ここで $\alpha$ は，$\mathbb{F}_2$ 上の既約多項式 $f(z) = z^3 + z + 1$ の根であり，$f(\alpha) = 0$ を満たす．また，検査行列 $H$ の成分は拡大体 $\mathbb{F}_{2^3}$ 上の要素であり，加減乗除も拡大体上の計算となる．さらに，検査行列 $H$ は $\alpha^7 = 1$ となることを利用すると

$$H = \begin{pmatrix} 1 & \alpha^1 & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \\ 1 & \alpha^2 & \alpha^4 & \alpha^6 & \alpha^1 & \alpha^3 & \alpha^5 \\ 1 & \alpha^3 & \alpha^6 & \alpha^2 & \alpha^5 & \alpha^1 & \alpha^4 \\ 1 & \alpha^4 & \alpha^1 & \alpha^5 & \alpha^2 & \alpha^6 & \alpha^3 \end{pmatrix} \tag{23}$$

と表すことが出来る．リードソロモン符号は線形符号であるため，その符号化はハミング符号と同様になされる．

復号するためにはハミング符号と同様に，受け取った受信語 $\boldsymbol{y}$ を元に，シンドローム $\boldsymbol{s}$ を計算する必要がある．誤り訂正の目標は，シンドローム $\boldsymbol{s}$ から誤りベクトル $\boldsymbol{e}$ を正しく求めることである．ここで，$\boldsymbol{e}$ を推定し誤り訂正を行う具体的な方法として，ピーターソン復号法 [4] と呼ばれる方法が知られている．ピーターソン復号法では，リードソロモン符号の検査行列の大きさによって復号可能なシンボル数が決まっており，式 (23) の検査行列 $H$ では 2 個までの誤りを訂正できる．

### 3.4 線形符号と通信路入力

通信路符号化の問題では通常，送信されるメッセージは一様分布に従うことが仮定される．ここで線形符号において，メッセージが一様分布に従うことは，式 (17) 右辺の $\boldsymbol{m}$ が一様分布に従うことと等しい．このとき，通信路入力である $\boldsymbol{x}$ の各シンボルも一様分布に従う．よって，線形符号により通信路容量が達成されるのは，最適な入力分布が一様分布となるような通信路だけである [5][6].

通信路入力が一様分布で通信路容量を達成する通信路の例としては，第 2 章で述べた二元対称通信路のような対称通信路があげられる．一方，Z 通信路のような入力分布 $P_X$ が一様分布のときに通信路容量を達成できない通信路の場合，メッセージと符号語が一対一対応する線形符号の原理上，線形符号では通信路容量を達成できない [7].

## 4 CoCoNuTS による通信路符号

線形符号を用いる場合，その原理上，通信路容量を達成できない通信路が存在する．本章ではこの問題を解決するための符号である CoCoNuTS について，その符号器，復号器の設計方法を文献 [2] に基づいて説明する．

## 4.1 拘束条件を満たす乱数生成器

CoCoNuTS の符号器，復号器ではともに拘束条件を満たす乱数生成器が使用される．この乱数生成器は指定された分布で乱数を発生させるものであり，その分布は次のように表される．

系列 $\boldsymbol{x} = (x_1, \cdots, x_n)$ を $n$ 次元列ベクトルとして，$l \times n$ 行列 $U$ と $l$ 次元列ベクトル $\boldsymbol{v}$ を用いて，連立一次方程式で与えられる拘束条件 $U\boldsymbol{x} = \boldsymbol{v}$ を表現する．また，拘束条件 $U\boldsymbol{x} = \boldsymbol{v}$ を満たすような解 $\boldsymbol{x}$ が複数あることを保証するため $l < n$ を仮定する．

拘束条件を満たす乱数生成器とは，拘束条件 $U\boldsymbol{x} = \boldsymbol{v}$ を満たす系列 $\boldsymbol{x}$ を，与えられた確率 $P(\boldsymbol{x})$ に比例して生成させるものである．すなわち，拘束条件を満たす乱数生成器は

$$Q(\boldsymbol{x}|\boldsymbol{v}) = \frac{P(\boldsymbol{x})\mathbb{1}\{U\boldsymbol{x} = \boldsymbol{v}\}}{\sum_{\boldsymbol{x}} P(\boldsymbol{x})\mathbb{1}\{U\boldsymbol{x} = \boldsymbol{v}\}} \tag{24}$$

のように定義される条件付き確率分布 $Q$ にしたがって乱数系列 $\boldsymbol{x}$ を出力する．ここで，$\mathbb{1}\{U\boldsymbol{x} = \boldsymbol{v}\}$ は定義関数であり

$$\mathbb{1}\{U\boldsymbol{x} = \boldsymbol{v}\} \triangleq \begin{cases} 1 & U\boldsymbol{x} = \boldsymbol{v} \text{ を満たしているとき} \\ 0 & U\boldsymbol{x} = \boldsymbol{v} \text{ を満たしていないとき} \end{cases} \tag{25}$$

を表している．$\sum_{\boldsymbol{x}}$ はすべての $n$ 次元列ベクトル $\boldsymbol{x}$ にわたる和である．このような乱数生成器を使用することにより，拘束条件 $U\boldsymbol{x} = \boldsymbol{v}$ を満たす複数の解の中から一つを決定している．

## 4.2 符号器

CoCoNuTS の符号器は前節で述べた，拘束条件を満たす乱数生成器を用いて構成される．符号器に現れる乱数生成器の分布 $F$ は

$$F(\boldsymbol{x}|\boldsymbol{c}, \boldsymbol{m}) = \frac{P(\boldsymbol{x})\mathbb{1}\{A\boldsymbol{x} = \boldsymbol{c}, B\boldsymbol{x} = \boldsymbol{m}\}}{\sum_{\boldsymbol{x}} P(\boldsymbol{x})\mathbb{1}\{A\boldsymbol{x} = \boldsymbol{c}, B\boldsymbol{x} = \boldsymbol{m}\}} \tag{26}$$

で定義される．ここで，$\boldsymbol{m}$ は $k$ 次元のメッセージ列ベクトルであり，$A$ は $l \times n$ 行列，$B$ は $k \times n$ 行列，$\boldsymbol{c}$ は $l$ 次元列ベクトルとし，拘束条件 $A\boldsymbol{x} = \boldsymbol{c}$, $B\boldsymbol{x} = \boldsymbol{m}$ を満たすような $\boldsymbol{x}$ の存在を保証するために $l + k \leq n$ を仮定する．$A$ と $B$ を縦に連結した行列を $U$，$\boldsymbol{c}$ と $\boldsymbol{m}$ を縦に連結した列ベクトルを $\boldsymbol{v}$，$P(\boldsymbol{x})$ を通信路に対する最適な入力分布とする．すると，式（26）は式 (24) で定義した，乱数生成器として考えることができる．

ここで，式 (10) の左辺に現われる検査行列 $H$ を例として CoCoNuTS の符号化方法を示す．検査行列 $H$ の，1 行目から 3 行目までを $A$，4 行目から 5 行目を $B$ となるように分割し，さらに $\boldsymbol{c}$ をすべての要素が 0 である列ベクトルとすると，符号器に現われる拘束条件 $A\boldsymbol{x} = \boldsymbol{c}$,

$B\boldsymbol{x} = \boldsymbol{m}$ は,

$$A\boldsymbol{x} = \boldsymbol{c} \Leftrightarrow \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \tag{27}$$

$$B\boldsymbol{x} = \boldsymbol{m} \Leftrightarrow \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix} = \begin{pmatrix} m_1 \\ m_2 \end{pmatrix} \tag{28}$$

のように表される．CoCoNuTS 符号器ではこの拘束条件 $A\boldsymbol{x} = \boldsymbol{c}, B\boldsymbol{x} = \boldsymbol{m}$ を満たすような，符号語 $\boldsymbol{x}$ を生成することでメッセージの符号化を行っている．

また，元となる行列をどのような大きさの $A$，$B$ 分割するかについては制約がない．よって式 (20) のハミング検査行列 $H$ を例に CoCoNuTS 符号化を考えると，行列 $A$，$B$ の分割方法は

$$A = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \end{pmatrix} \qquad B = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \tag{29}$$

$$A = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} \qquad B = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \tag{30}$$

$$A = \quad \text{なし} \qquad B = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \tag{31}$$

の 3 種類となる．行列 $B$ にメッセージが割り当てられるため，式 (29) から式 (31) で定義される CoCoNuTS のメッセージ数はそれぞれ 2, $2^2$, $2^3$ となり，その符号化レート $R$ はすべて

の分割方法で符号語長が 7 であることを考慮すると，式 (1) より

$$\text{式 (29) の符号化レート} \qquad R = \frac{\log_2 2}{7} = \frac{1}{7} \tag{32}$$

$$\text{式 (30) の符号化レート} \qquad R = \frac{\log_2 2^2}{7} = \frac{2}{7} \tag{33}$$

$$\text{式 (31) の符号化レート} \qquad R = \frac{\log_2 2^3}{7} = \frac{3}{7} \tag{34}$$

となる．このように，CoCoNuTS による符号化を考えた際，元となる検査行列の分割方法は，複数考えることが出来る．さらに，メッセージ数は分割方法ごとに変化するため，CoCoNuTS の符号化レートは，元となる検査行列をどのように分割するかによって定まる．

## 4.3 復号器

復号器に現われる拘束条件を満たす乱数生成器の分布 $G$ は

$$G(\boldsymbol{x}|\boldsymbol{y}, \boldsymbol{c}) = \frac{P_{X|Y}(\boldsymbol{x}|\boldsymbol{y})\mathbb{1}\{A\boldsymbol{x} = \boldsymbol{c}\}}{\sum_{\boldsymbol{x}} P_{X|Y}(\boldsymbol{x}|\boldsymbol{y})\mathbb{1}\{A\boldsymbol{x} = \boldsymbol{c}\}} \tag{35}$$

で定義される．ここで $\boldsymbol{y}$ は受信語である．文献 [2] では，この分布 $G$ を用いて乱数を発生させ復号を行っているが，本研究では使用する符号のサイズが小さいため，復号にはブロック MAP 復号法を用いる．

ブロック MAP 復号法は，観測された受信語 $\boldsymbol{y}$ に基づいて得られる事後確率 $P_{X|Y}(\boldsymbol{x}|\boldsymbol{y})$ を計算し，その値を最大化するような $\boldsymbol{x}$ を推定符号語 $\hat{\boldsymbol{x}}$ として出力するような復号方法である．ここで，事後確率 $P_{X|Y}(\boldsymbol{x}|\boldsymbol{y})$ はベイズ則を利用して

$$P_{X|Y}(\boldsymbol{x}|\boldsymbol{y}) = \frac{P_{Y|X}(\boldsymbol{y}|\boldsymbol{x})F(\boldsymbol{x})}{P_Y(\boldsymbol{y})} \tag{36}$$

のように表現される．式 (36) において，分子に登場する $P_{Y|X}(\boldsymbol{y}|\boldsymbol{x})$ は通信路に対する条件付き確率，$F(\boldsymbol{x})$ は符号語の分布であり，ともに $\boldsymbol{x}$ の関数となる．また，分母の $P_Y(\boldsymbol{y})$ は受信語 $\boldsymbol{y}$ にのみ依存するため，$\boldsymbol{x}$ に関して $P_{X|Y}(\boldsymbol{x}|\boldsymbol{y})$ を最大化する際には関係がない．したがって

$$P_{Y|X}(\boldsymbol{y}|\boldsymbol{x})F(\boldsymbol{x}) \tag{37}$$

を最大化するような $\boldsymbol{x}$ を求め，推定符号語 $\hat{\boldsymbol{x}}$ として出力すれば良い．

このようにして求められた $\hat{\boldsymbol{x}}$ が，もし送信された符号語 $\boldsymbol{x}$ であるならば，CoCoNuTS 行列 $B$ を用いた制約条件 $B\boldsymbol{x} = \boldsymbol{m}$ を満たすため，推定符号語 $\hat{\boldsymbol{x}}$ から $B\hat{\boldsymbol{x}} = \boldsymbol{m}$ を計算することによってメッセージ $\boldsymbol{m}$ を正しく復元することができる．

# 5　実装と復号性能の検証

本研究では，CoCoNuTS と従来の線形符号との比較を行う．ここでは従来の線形符号として，ハミング符号，リードソロモン符号を用い，CoCoNuTS と復号誤り確率を比較することによって，復号性能を検証する．検証実験を行うにあたり，CoCoNuTS を以下のように実装した．

## 5.1　実装方法

式 (10) の検査行列 $H$ を式 (27) 式 (28) のような行列 $A$, $B$ で分割した CoCoNuTS を考える．このとき，CoCoNuTS では符号器，復号器の両方で拘束条件を満たす乱数生成器を使用するため，一連の送受信シミュレーションを行う前に式 (26) の分母

$$\sum_{\boldsymbol{x}} P(\boldsymbol{x}) \mathbb{1}\{A\boldsymbol{x} = \boldsymbol{c}, B\boldsymbol{x} = \boldsymbol{m}\} \tag{38}$$

を計算して保存する必要がある．メッセージ $\boldsymbol{m}$ ごとに拘束条件 $B\boldsymbol{x} = \boldsymbol{m}$ が変化するため，保存する分母の数はメッセージ数と等しくなる．本節の例では，$B$ は $2 \times 7$ 行列であるため，メッセージ数が 4 となり，事前に 4 つの分母が保存される．

符号器では送信者から送られたメッセージ $\boldsymbol{m}$ を元に，拘束条件 $A\boldsymbol{x} = \boldsymbol{c}$, $B\boldsymbol{x} = \boldsymbol{m}$ を満たす符号語の候補を生成する．生成された候補の中から式 (26) で表される分布 $F$ に基づいて通信路に入力される符号語が決められる．例えば，$\boldsymbol{m} = (0, 0)$ の時，拘束条件を満たす符号語の候補とその分布は最適な入力分布を $p$ とすると

$$(0,0,0,0,0,0,0) \longrightarrow F = \frac{(1-p)^7}{(1-p)^7 + p^6(1-p)^1 + p^6(1-p)^1 + p^2(1-p)^5} \tag{39}$$

$$(1,0,1,1,1,1,1) \longrightarrow F = \frac{p^6(1-p)^1}{(1-p)^7 + p^6(1-p)^1 + p^6(1-p)^1 + p^2(1-p)^5} \tag{40}$$

$$(1,1,0,1,1,1,1) \longrightarrow F = \frac{p^6(1-p)^1}{(1-p)^7 + p^6(1-p)^1 + p^6(1-p)^1 + p^2(1-p)^5} \tag{41}$$

$$(0,1,1,0,0,0,0) \longrightarrow F = \frac{p^2(1-p)^5}{(1-p)^7 + p^6(1-p)^1 + p^6(1-p)^1 + p^2(1-p)^5} \tag{42}$$

のようになる．ここで，$F$ の計算を行う際，分母は事前に計算されたものを使用するため符号器では分子の計算のみを行う．

復号器ではブロック MAP 復号を用いて，受信語 $\boldsymbol{y}$ から推定符号語 $\hat{\boldsymbol{x}}$ を求める．ここで，CoCoNuTS において符号語は拘束条件 $A\boldsymbol{x} = \boldsymbol{c}$ を満たすため，復号器では拘束条件を満たす，すべての $\boldsymbol{x}$ ごとに式 (37) を計算し，その値を最大化するような $\boldsymbol{x}$ を推定符号語 $\hat{\boldsymbol{x}}$ として出力する．$A$ は $3 \times 7$ 行列であるため，$A\boldsymbol{x} = \boldsymbol{c}$ を満たすような符号語は 16 個存在し，その中から推定符号語 $\boldsymbol{x}$ が一つ選ばれる．

このように，本研究における CoCoNuTS の実装方法では符号化，復号化に際して，式 (26) の分母部分を記憶し，すべての符号語に対して式 (37) を計算している．このため，一連の送受信における計算量は使用する検査行列のサイズに対して指数的に増大する．本研究では使用する検査行列のサイズが非常に小さいものであるため，上記の方法で CoCoNuTS を実装できたが，より大きな行列で CoCoNuTS の実装を考えた場合，指数的に増大する計算時間，メモリ量に対してどのように CoCoNuTS 符号器，復号器を構成するかが今後の課題となる．

## 5.2 CoCoNuTS 行列の任意性

CoCoNuTS の構成において，CoCoNuTS 行列 $A$, $B$ の定義には任意性がある．このため，元となる検査行列が同じ CoCoNuTS でも，その性能に差が生じる．本節では，従来の線形符号との比較を行う前に，その性能差について CoCoNuTS 行列の定義に対する補足点として，考察していく．

式 (10) の $5 \times 7$ 検査行列 $H$ を，$3 \times 7$ 行列 $A$ と，$2 \times 7$ 行列 $B$ に分けて CoCoNuTS を考える．このとき $A$ と $B$ が分割前の検査行列 $H$ のどの行で定義されるかは任意である．このため CoCoNuTS 行列 $B$ は，元となる検査行列 $H$ を構成する 5 つの行の中から，任意に選ばれた 2 つの行によって構成される．このとき，もう片方の CoCoNuTS 行列 $A$ は，元となる検査行列の残された 3 行で定まるため，検査行列 $H$ を $3 \times 7$ 行列と，$2 \times 7$ 行列に分けて CoCoNuTS を構成するとき，構成方法は

$$ {}_5\mathrm{C}_2 = 10 \tag{43} $$

より 10 種類考えることが出来る．

図 4 は式 (10) で定義される 10 種類の CoCoNuTS における復号誤り確率を示したものである．ここで，通信路は Z 通信路であり図 4 では 1 の反転確率 $f$ を 0 から 1 まで変化させて検証を行った．図 4 より，分割する前の検査行列が同じでも，CoCoNuTS 行列 $A$，$B$ を元となる検査行列のどの行で定義するかによって，CoCoNuTS における復号誤り率に変化が生じることがわかる．

復号誤り確率が変化する理由として，$2 \times 7$ 行列 $B$ について，元となる検査行列 $H$ の 1 行目と 2 行目で定義される場合と，4 行目と 5 行目で定義される場合と例に考えてみると，メッ

セージに対する制約 $B\boldsymbol{x} = \boldsymbol{m}$ がそれぞれ

$$1, 2 \text{ 行目で定義されたとき} \qquad B\boldsymbol{x} = \boldsymbol{m} \Leftrightarrow \begin{cases} x_1 + x_2 + x_3 = m_1 \\ x_1 + x_4 = m_2 \end{cases} \tag{44}$$

$$4, 5 \text{ 行目で定義されたとき} \qquad B\boldsymbol{x} = \boldsymbol{m} \Leftrightarrow \begin{cases} x_4 + x_6 = m_1 \\ x_4 + x_7 = m_2 \end{cases} \tag{45}$$

のように変化する．メッセージシンボル $m_1$ が定まったとき，式 (44) では $m_1$ が 3 つの符号シンボルの制約に関係するのに対して，式 (45) では 2 つの符号シンボルの制約に関係する．このように，$A$, $B$ の定義の仕方によって符号語が満たす条件がそれぞれ異なる場合があるため，元となる検査行列が等しい場合でも復号誤り確率が変化すると考えられる．

一方，元となる検査行列 $H$ の異なる行で $A$, $B$ を定義したにもかかわらず復号誤り確率が変化しなかった場合について，$B$ が分割前の検査行列 $H$ の 3 行目と 4 行目で定義される場合，4 行目と 5 行目で定義される場合を例に考えてみると，メッセージに対する制約 $B\boldsymbol{x} = \boldsymbol{m}$ は

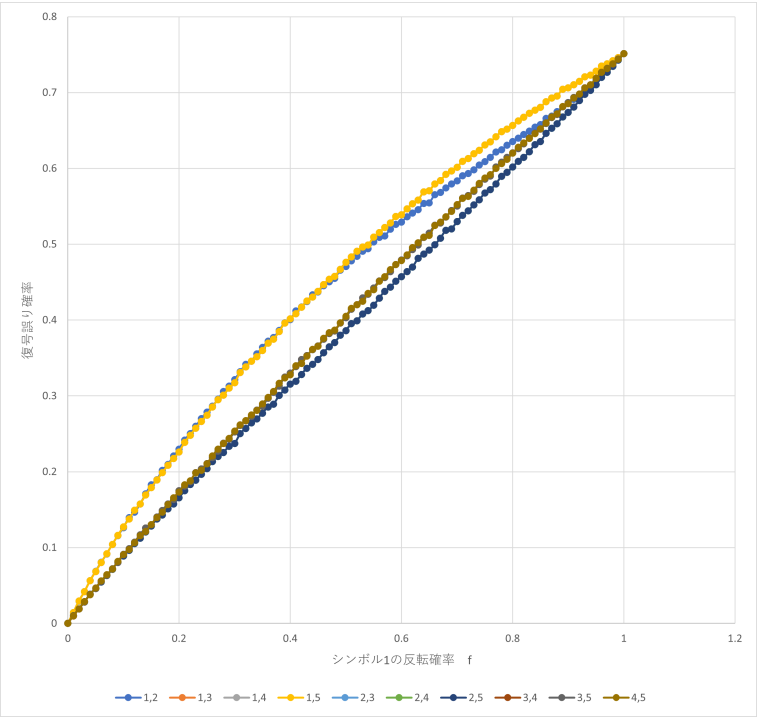


図 4　$5 \times 7$ 検査行列で定義される CoCoNuTS

それぞれ

$$3,4\text{ 行目で定義されたとき} \qquad B\boldsymbol{x}=\boldsymbol{m} \Leftrightarrow \begin{cases} x_4+x_5=m_1 \\ x_4+x_6=m_2 \end{cases} \tag{46}$$

$$3,5\text{ 行目で定義されたとき} \qquad B\boldsymbol{x}=\boldsymbol{m} \Leftrightarrow \begin{cases} x_4+x_5=m_1 \\ x_4+x_7=m_2 \end{cases} \tag{47}$$

となる．このとき式 (46) と式 (47) ではメッセージシンボル $m_1$ に対する符号シンボルの制約が等しく，$m_2$ に対する符号シンボルの制約も $m_2$ が 2 つの符号シンボル制約に関係するという点で等しい．このように式 (46) と式 (47) で定義される CoCoNuTS は，CoCoNuTS を構成する行列 $A$, $B$ がそれぞれ異なるものの，$A$, $B$ が符号語に与える制約は等しいため，復号誤り確率に変化が生じなかったのではないかと考えられる．

以上のように，CoCoNuTS では行列 $A$，$B$ をどの行で定義するかによって復号誤り確率に変化が生じることが分かった．このため，本研究で使用するハミング符号とリードソロモン符号についても同一符号内の異なる行で CoCoNuTS の行列 $A$，$B$ を定義することが可能である．

ここで式 (20) より，ハミング符号の検査行列 $H$ では，検査行列内のどの行をメッセージシンボルと対応させても

$$x_4+x_5+x_6+x_7=m \tag{48}$$
$$x_2+x_3+x_6+x_7=m \tag{49}$$
$$x_1+x_3+x_5+x_7=m \tag{50}$$

のように等しい数の符号シンボルに影響を与える．

また, 式 (23) よりリードソロモン符号の検査行列 $H$ は, すべての行で $(1,\alpha^1,\alpha^2,\alpha^3,\alpha^4,\alpha^5,\alpha^6)$ が一度ずつ現われるように構成されているため，検査行列内のどの行をメッセージシンボルに対応させてもその制約は $1,\alpha^1,\alpha^2,\alpha^3,\alpha^4,\alpha^5,\alpha^6$ が係数として一度ずつ登場する一次方程式

$$x+\alpha^1x+\alpha^2x+\alpha^3x+\alpha^4x+\alpha^5x+\alpha^6x=m \tag{51}$$

となる．よって，ハミング符号，リードソロモン符号ともに，検査行列のどの行を用いて行列 $A$，$B$ を定義しても制約条件に変化が生じない．このため，次節以降で扱うハミング符号とリードソロモン符号については，どの行で $A$，$B$ が定義されるかについては考慮しない．

## 5.3 検証

まず，式 (20) の検査行列 $H$ で定義されるハミング符号と，検査行列 $H$ を分割して得られる CoCoNuTS との復号誤り確率を比較した．検査行列 $H$ から CoCoNuTS を構成する場合，4

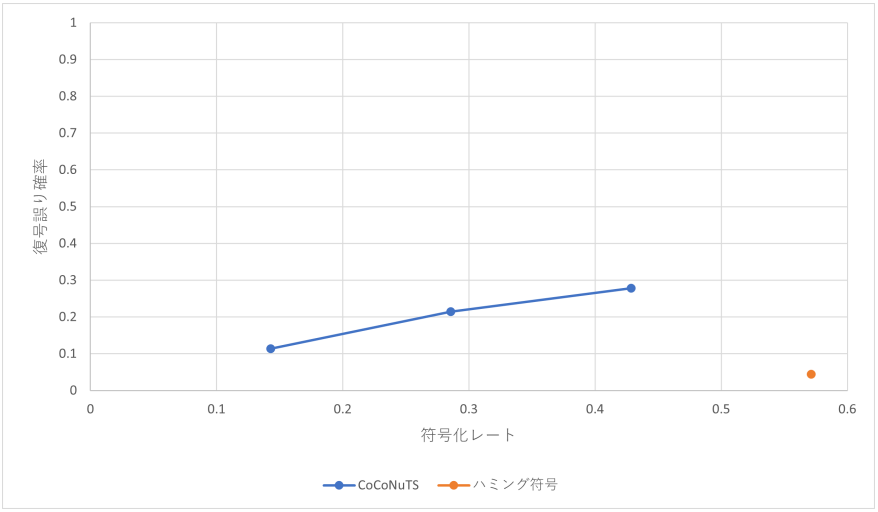


図 5　f = 0.1 の Z 通信路

章 2 節で示した通り符号化レートの異なる 3 種類の CoCoNuTS 符号が構成可能であるため，そのすべての復号誤り確率を求めた．ここで実験は，1 の反転確率 $f$ が $f = 0.1$ と $f = 0.5$ となるような Z 通信路で行い，それぞれの結果を図 5，図 6 に示した．

図 5 では，CoCoNuTS よりハミング符号の方が大きい符号化レートで低い復号誤り率を示し，ハミング符号と比較して CoCoNuTS が悪い性能を示している．また図 6 において，CoCoNuTS の方がハミング符号より小さい復号誤り確率を示す場合があるが，ハミング符号の方が符号化レートが大きいため CoCoNuTS の方が性能が良いと判断することは出来ない．

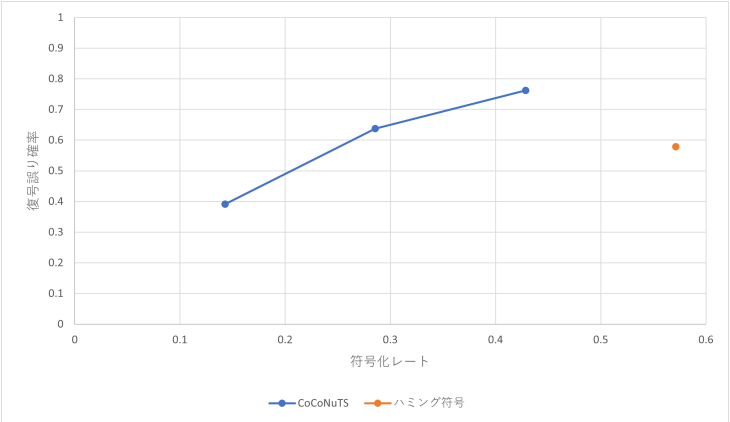


図 6　f = 0.5 の Z 通信路

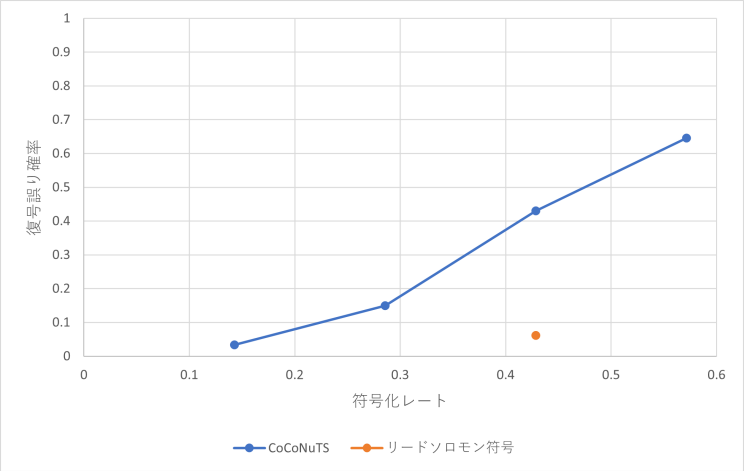


図 7　f = 0.1 の Z 通信路

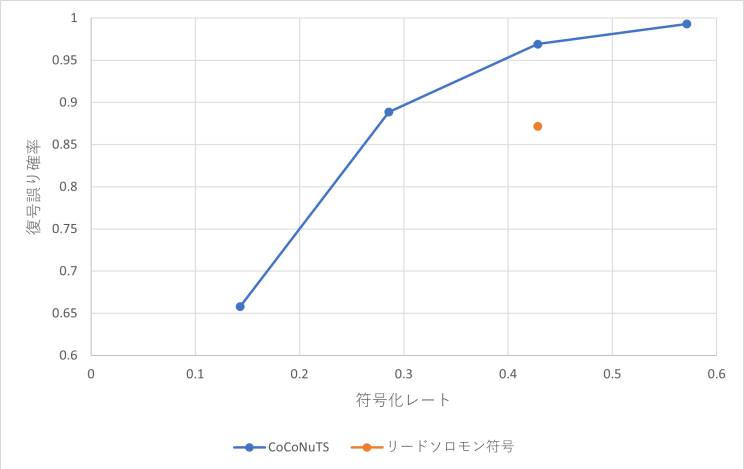


図 8　f = 0.5 の Z 通信路

次に式 (23) の検査行列 $H$ で定義されるリードソロモン符号と，CoCoNuTS との比較を行った．リードソロモン符号は文献 [4] で示されるピーターソン復号法を用いて復号を行い，CoCoNuTS はリードソロモン符号の検査行列 $H$ から符号化レートの異なる 4 種類の符号を構成した．ハミング符号の時と同様に実験は $f = 0.1$，$f = 0.5$ の Z 通信路を使用し，それぞれの結果を図 7, 図 8 に示した．

図 7，図 8 において CoCoNuTS がリードソロモン符号に対して小さい誤り率を示している場合があるものの，符号化レートが等しい場合を比較すると CoCoNuTS の復号誤り確率は

リードソロモン符号のものよりも大きくなることがわかる．

以上の結果から設定した条件ではハミング符号に対して CoCoNuTS は，符号化レートが大きく，復号誤り確率が小さいような性能の良い符号であるとは言えず，リードソロモン符号に対しては符号化レートをそろえた場合，復号誤り確率で劣ることが分かった．CoCoNuTS が良い性能を示さなかった理由として，使用した符号のサイズ自体が非常に小さいものであったことが考えられる．また，ハミング符号，リードソロモン符号の検査行列は誤り訂正が行えるような，最適なものを使用したのに対して，CoCoNuTS を定義した検査行列は，最適なものではなかったことも理由だと考えられる．

## 6　まとめ

本研究では，CoCoNuTS の符号器，復号器を設計し，従来の線形符号では通信路容量を達成できないような通信路で使用することによりその性能を検証した．大きさの小さい符号を用いて検証を行なったため，CoCoNuTS の復号性能は従来の線形符号と比較して悪くなっている結果が得られた．また，ブロック MAP 復号法を使用している以上，復号時の計算量が膨大になってしまうことは避けられない．このため，今後は符号のサイズを大きくするような符号器を設計するとともに，サイズの大きい符号に対応できるような復号器の設計も同時に行う．

## 謝辞

本研究を行うにあたり，丁寧なご指導を賜りました指導教員の西新幹彦准教授に深く感謝申し上げます．

## 参考文献

[1] C.E, Shannon, “A mathematical theory of communication,” Bell System Technical Journal, vol.27, pp.379–423, 623–656, 1948.

[2] 村松純，三宅茂樹，「シャノン限界を達成する符号技術 CoCoNuTS」，電子情報通信学会，電子情報通信学会誌，vol.104, No.8, pp.898–905, Jan. 2021.

[3] T. Cover, J. Thomas, (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，共立出版株式会社， 2012.

[4] 和田山正，誤り訂正技術の基礎，森北出版，2010.

[5] P. Elias, “Coding for noisy channels,” IRE Conv. Rec, vol.3, pp.37–46, 1955.

[6] R.L. Dobrushin, “Asymptotic optimality of group and systematic codes for some channels,” Theory of Probability and Its Applications, vol.8, no.1, pp. 47–60, 1963.

[7] R. Ahlswede, “Group codes do not achieve Shanon’s channel capacity for general discrete chanenels,” The Annals of Mathematical Statistics, vol.42 , no.1, pp.224–240, 1971.

# 付録 A　Z 通信路の最適な入力分布

Z 通信路の最適な入力分布が式 (8) となることを導出する．

シンボル 1 が入力される確率を $p$，通信路で 1 が反転する確率を $f$ とおく．すなわち，通信路入力 $X$ の分布 $P_X$ を

$$P_X(0) = 1 - p \tag{52}$$
$$P_X(1) = p \tag{53}$$

とし，通信路の条件付き確率分布 $P_{Y|X}$ を

$$P_{Y|X}(0|0) = 1 \tag{54}$$
$$P_{Y|X}(1|0) = 0 \tag{55}$$
$$P_{Y|X}(0|1) = f \tag{56}$$
$$P_{Y|X}(1|1) = 1 - f \tag{57}$$

とする．またシンボル 1 が出力される確率を $q$ とおく．すなわち，通信路出力 $Y$ の分布 $P_Y$ を

$$P_Y(0) = 1 - q \tag{58}$$
$$P_Y(1) = q \tag{59}$$

とする．このとき，$q$ は $p$ と $f$ を用いて

$$q = P_X(0)P_{Y|X}(1|0) + P_X(1)P_{Y|X}(1|1) \tag{60}$$
$$= p(1 - f) \tag{61}$$

と表される．これらの式を利用して以下では相互情報量 $I(X;Y)$ について考えていく．

相互情報量 $I(X;Y)$ は通信路入出力を表す確率変数 $X$, $Y$ を用いて

$$I(X;Y) = H(Y) - H(Y|X) \tag{62}$$

と表される．ここで $X$ の分布 $P_X$ を用いると，式 (62) 右辺の条件付きエントロピー $H(Y|X)$ は

$$H(Y|X) = \sum_x P_X(x)H(Y|X = x) \tag{63}$$
$$= P_X(0)H(Y|X = 0) + P_X(1)H(Y|X = 1) \tag{64}$$

となる．ここで $H(Y|X=0)$ は

$$H(Y|X=0) = \sum_y P_{Y|X}(y|0) \log_2 \frac{1}{P_{Y|X}(y|0)} \tag{65}$$

$$= P_{Y|X}(0|0) \log_2 \frac{1}{P_{Y|X}(0|0)} + P_{Y|X}(1|0) \log_2 \frac{1}{P_{Y|X}(1|0)} \tag{66}$$

となり，式 (54), 式 (56) を用いると

$$H(Y|X=0) = 0 \tag{67}$$

となる．同様に $H(Y|X=1)$ は，式 (55), 式 (57) を用いると

$$H(Y|X=1) = \sum_y P_{Y|X}(y|1) \log_2 \frac{1}{P_{Y|X}(y|1)} \tag{68}$$

$$= P_{Y|X}(0|1) \log_2 \frac{1}{P_{Y|X}(0|1)} + P_{Y|X}(1|1) \log_2 \frac{1}{P_{Y|X}(1|1)} \tag{69}$$

$$= f \log_2 \frac{1}{f} + (1-f) \log_2 \frac{1}{1-f} \tag{70}$$

となる．ここで式 (70) は式 (7) で定義される二値エントロピー関数である．式 (67) と式 (70) より条件付きエントロピー $H(Y|X)$ は

$$H(Y|X) = P_X(1) H(Y|X=1) \tag{71}$$

$$= ph(f) \tag{72}$$

のようになる．

また式 (62) の $H(Y)$ について，$H(Y)$ はエントロピーの定義式 (3) より

$$H(Y) = -\sum_q P_Y(q) \log P_Y(q) \tag{73}$$

$$= P_Y(0) \log_2 \frac{1}{P_Y(0)} + P_Y(1) \log_2 \frac{1}{P_Y(1)} \tag{74}$$

となる．ここで式 (58)，式 (59) より

$$H(Y) = (1-q) \log_2 \frac{1}{1-q} + q \log_2 \frac{1}{q} \tag{75}$$

となる．ここで式 (75) は式 (7) で定義される二値エントロピー関数であるため，$H(Y)$ は

$$H(Y) = h(q) \tag{76}$$

となる．

よって，式 (72) と式 (76) より Z 通信路における相互情報量 $I(X;Y)$ は

$$I(X;Y) = h(q) - ph(f) \tag{77}$$

となる．

通信路容量は相互情報量の最大値であるため，最適な入力分布とは式 (77) の右辺を最大化するような $p$ である．ここで，相互情報量は通信路入力の分布 $P_X$ に対して上に凸な関数である [3]．このため，相互情報量 $I(X;Y)$ を $p$ で微分した

$$\frac{d}{dp}I(X;Y) = (1-f)\log_2\frac{1-q}{q} - h(f) \tag{78}$$

について，$\frac{d}{dp}I(X;Y) = 0$ を満たすような $p$ が与えられたとき，相互情報量 $I(X;Y)$ は極大かつ最大となり，$p$ で定まる $P_X$ は通信路容量を達成する最適な入力分布となる．

まず，反転確率 $f = 1$ の場合を考える．式 (78) について

$$\frac{d}{dp}I(X;Y) = (1-1)\log_2\frac{1-q}{q} - h(1) \tag{79}$$

$$= -h(1) \tag{80}$$

となる．このとき

$$h(1) = (1-1)\log_2\frac{1}{1-1} + 1\log_2\frac{1}{1} \tag{81}$$

$$= 0 \tag{82}$$

である．よって $\frac{d}{dp}I(X;Y) = 0$ は常に成り立つため，最適な入力分布 $p$ は任意の値となる．

次に，反転確率 $f \neq 1$ のときについて考える．式 (78) について

$$(1-f)\log_2\frac{1-q}{q} - h(f) = 0 \tag{83}$$

$$\log_2\frac{1-q}{q} = \frac{h(f)}{1-f} \tag{84}$$

$$\frac{1-q}{q} = 2^{\frac{h(f)}{1-f}} \tag{85}$$

となる．ここで，式 (60) より $q = p(1-f)$ であるため

$$\frac{1-p(1-f)}{p(1-f)} = 2^{\frac{h(f)}{1-f}} \tag{86}$$

$$1-p(1-f) = p(1-f)2^{\frac{h(f)}{1-f}} \tag{87}$$

$$(2^{\frac{h(f)}{1-f}} + 1)p = \frac{1}{1-f} \tag{88}$$

$$p = \frac{1}{(1-f)(2^{\frac{h(f)}{1-f}} + 1)} \tag{89}$$

となる．以上より Z 通信路の最適な入力分布 $p$ は式 (89) で与えられる．

# 付録 B　ハミング符号の検査行列で定義される CoCoNuTS の送受信シミュレーションプログラム

```
#include<stdio.h>
#include<math.h>
#include "MT.h"

/*二値エントロピー関数*/
double h(double a)
{
    double b;
    if(a == 0 || a == 1){
      b = 0;
    } else {
      b = a * log(1 / a)+(1 - a) * log(1 / (1 - a));
    }
    return(b);
}

/*最適な入力分布*/
double p(double f, double g)
{
    double a, b;
    double p1;

    a = 1 - (g + f);

    if (a != 0.0){
      b = exp((1 / a) * (h(f) - h(g)));
      p1 = (b * (1 - f) - f)/(a * (1 + b));
    } else {
      p1 = 0.5;
    }
    return(p1);

}


double c_p, c_f, c_g;

/*乱数生成器の分母部分を先に計算し，記憶する*/
void all_bunbo(double sum_bunbo[]){
    int i, j, l;
    int a;
    int b[4]; //x3 x5 x6 x7 のシンボル（今回の設定では x3 x5 x6 x7 が決まると x1 x2 x4 も決まる）
    int x[7];
    int m[2];
    double bunbo;

    for (i = 0; i < 4; i++){
        //メッセージごとに分母部分を記憶
        sum_bunbo[i] = 0;
        m[0] = i / 2;
        m[1] = i % 2;
        for (j = 0; j < 16; j++){
            a = j;
            //x3 x5 x6 x7 を決める
            for (l = 0; l < 4; l++){
                b[l] = a % 2;
                a = a / 2;
            }


            //Ax = c, Bx = m を満たす符号語の候補を生成
```

```
            x[0] = (b[0] + b[1] + b[3] + m[1]) % 2; //x1
            x[1] = (b[0] + b[2] + b[3] + m[0]) % 2; //x2
            x[3] = (b[1] + b[2] + b[3]) % 2;        //x4

            x[2] = b[0]; //x3
            x[4] = b[1]; //x5
            x[5] = b[2]; //x6
            x[6] = b[3]; //x7


            //最適な入力分布を用いて符号語の分布 P(x) を計算
            bunbo = 1.0;
            for (l = 0; l < 7; l++){
                bunbo *= (x[l] == 0)? 1 - c_p : c_p;

            }

            //P(x) の総和をとり分母部分をメッセージごとに記憶
            sum_bunbo[i] += bunbo;
        }
    }
    return;
}

double sum_bunbo[4];


/*符号器 (受け取ったメッセージから符号語を生成)*/
void encoder(int m[], int x[]){
    int i, j, l;
    int a;
    int decimal_m; //十進数で表されるメッセージ
    int b[4]; //x3 x5 x6 x7 のシンボル
    double F; //符号器に現われる乱数生成器の分布 F
    double bunsi; //符号語の分布
    double random_number; //乱数

    decimal_m = 0;

    //メッセージを十進数に変換
    for (i = 0; i < 2; i++){
        decimal_m = decimal_m * 2 + m[i];
    }


    F = 0.0;
    random_number = genrand_real2();

    //Ax = c, Bx = m を満たす符号語の候補を生成
    for (i = 0; i < 16; i++){
        a = i;
        //x3 x5 x6 x7 を決める
        for (j = 0; j < 4; j++){
            b[j] = a % 2;
            a = a / 2;
        }

        //Ax = c, Bx = m を満たす符号語の候補を生成
        x[0] = (b[0] + b[1] + b[3] + m[1]) % 2; //x1
        x[1] = (b[0] + b[2] + b[3] + m[0]) % 2; //x2
        x[3] = (b[1] + b[2] + b[3]) % 2;        //x4

        x[2] = b[0]; //x3
        x[4] = b[1]; //x5
        x[5] = b[2]; //x6
        x[6] = b[3]; //x7


        //最適な入力分布を用いて符号語の分布 P(x) を計算
```

```
        bunsi = 1.0;
        for (j = 0; j < 7; j++){
            bunsi *= (x[j] == 0)? 1 - c_p : c_p;
        }

        //事前に記憶しておいた sum_bunbo を利用し符号器に現われる乱数生成器の分布 F を計算
        F += bunsi / sum_bunbo[decimal_m];

        //符号語を決定
        if (random_number < F){
            break;
        }
    }
    return;
}


/*通信路（入力シンボル 0 と 1 が，それぞれ確率 c_f と c_g で反転)*/
void tusinro(int x[]){
    int i;
    double random_number; //乱数


    for(i = 0; i < 7; i++){
        random_number = genrand_real2();
        //printf("rana = %f\n", random_number);

        //入力シンボル 0 が反転確率 c_f で 1 に反転
        if(x[i] == 0){
            if(random_number < c_f){
                x[i] = 1;
            }
        }

        //入力シンボル 1 が反転確率 c_g で 0 に反転
        else if(x[i] == 1){
            if(random_number < c_g){
                x[i] = 0;
            }
        }
    }
}

double W[2][2];


/*復号器（受信語をブロック MAP 復号にて復号)*/
void decoder(int y[], int dec_m[]){
    int i, j, k, l;
    int a;
    int b[4];
    int x[7];
    int m[2];
    double bunsi;
    double u;
    double G;
    double G_max;

    G_max = 0.0;

    //Ax = c を満たす，すべての x に対して事後確率 P(x|y) を計算し最大値を求める
    for (i = 0; i < 4; i++){
        m[0] = i / 2;
        m[1] = i % 2;
        for (j = 0; j < 16; j++){
            a = j;
            for (k = 0; k < 4; k++){
                b[k] = a % 2;
                a = a / 2;
```

```
            }

            //Ax = c, Bx = m を満たす符号語の候補を生成
            x[0] = (b[0] + b[1] + b[3] + m[1]) % 2; //x1
            x[1] = (b[0] + b[2] + b[3] + m[0]) % 2; //x2
            x[3] = (b[1] + b[2] + b[3]) % 2;        //x4

            x[2] = b[0]; //x3
            x[4] = b[1]; //x5
            x[5] = b[2]; //x6
            x[6] = b[3]; //x7


            bunsi = 1.0;
            u = 1.0;

            for (l = 0; l < 7; l++){
                //最適な入力分布を用いて符号語の分布 P(x) を計算
                bunsi *= (x[l] == 0)? 1 - c_p : c_p;

                //通信路の条件付き確率 P(y|x) を計算
                u *= W[ y[l] ][ x[l] ];
            }

            //P(x|y) の最大化に必要な部分を計算
            G = u * (bunsi / sum_bunbo[i]);


            //最大値を探す
            if (G > G_max){
                G_max = G;

                //G が最大となるときの m を推定メッセージとして記憶
                dec_m[0] = m[0];
                dec_m[1] = m[1];
            }
        }
    }
    return;
}


//送受信シミュレーションを行い復号誤りが発生した場合 1 を返す
int error_count(){
    int i;
    double a;
    int m[2]; //送信メッセージ
    int dec_m[2]; //復元メッセージ
    int x[7]; //符号語
    int count = 0;

    //乱数を用いて送信メッセージを決める
    for (i = 0; i < 2; i++){
        a = genrand_real2() * 2;
        m[i] = a;

    }

    //符号化
    encoder(m, x);

    //通信路
    tusinro(x);

    //復号
    decoder(x, dec_m);

    //送信メッセージと復元メッセージが同じか判別
    for (i = 0; i < 2; i++){
```

```
        if (m[i] != dec_m[i]){
        count = 1;
        }
    }
    return(count);
}


int main(void){
    int i;
    int TRIAL;  //試行回数
    int error;  //誤り数

    c_f = 0;   //通信路における入力シンボル 0 の反転確率
    c_g = 0.1; //通信路における入力シンボル 1 の反転確率

    c_p = p(c_f, c_g); //最適な入力分布


    //通信路の条件付き確率
    W[0][0] = 1 - c_f;
    W[0][1] = c_g;
    W[1][0] = c_f;
    W[1][1] = 1 - c_g;


    error = 0;
    TRIAL = 1000000;

    //事前に乱数生成器の分母部分を計算
    all_bunbo(sum_bunbo);

    //送受信シミュレーションを繰り返す
    for (i = 0; i < TRIAL; i++){
        error += error_count_27();
    }

    //復号誤り確率
    printf("ayamri = %f", (double)error/TRIAL);


    return 0;

}
```