情報通信基礎Ⅰ　第１章　情報とエントロピー

Information and Communication Basics I Chapter 1 Information and Entropy

2019/4/24　情シス　渡辺

ポイント

・情報源とは

・情報量、平均情報量、エントロピー

point

・What is an information source?

・Information volume, average information volume, entropy

１．１　情報量

1.1 Information volume

別紙

１．２　情報源

1.2 Information sources

ある情報源Xがある。この情報源から、n個の通報$s_1, s_2, , , s_n$が、$P_1, P_2, , , P_n$の確率で生起する。この情報源Xを

$$X = \begin{pmatrix} S_1 & S_2 & \cdots & S_n \\ P_1 & P_2 & \cdots & P_n \end{pmatrix} \qquad \sum_{i=1}^{n} P_i = 1$$

と表現する。

There is a source X. From this source, n reports s_1,s_2,,,s_n occur with a probability of P_1,P_2,,,P_n. This source X
To express.

（注：これは、iid 情報源の場合であり、より複雑な場合はこれでは表現できない）

(Note: this is for the iid source, and for more complex cases this cannot be represented)

例）サイコロ

Example) Dice

$X = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \end{pmatrix}$

情報を確率でとらえる＝＞情報源から、通報が多数回繰り返して発生する。

Capturing information with probability => Notifications are generated many times from the information source.

１．３　エントロピー

1.3 Entropy

ある単純な情報源

A simple source

$X = \begin{pmatrix} S_1 & S_2 \\ P_1 & P_2 \end{pmatrix}$　$(P_1 + P_2 = 1)$を考える。

S_iに対する情報量は、$I_i = -logP_i$ である。これらの平均を取る。すなわち、

The amount of information for S_i is I_i=-logP_i. Take these averages. That is,

$H =$ ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ (bit)

を、情報源Xの平均情報量、または、エントロピーと言う。

Is called the average information amount of the information source X, or entropy.

通報の数がn個の一般的な場合は、

In the general case of n notifications,

$X = \begin{pmatrix} S_1 & S_2 & \cdots & S_n \\ P_1 & P_2 & \cdots & P_n \end{pmatrix}$　　$\sum_i P_i = 1$　に対して、

$H = -\sum_{i=1}^{n} P_i * logP_i$　(bit)

となる。

情報量は、個別の通報が生起したときの量を表現しているのに対し、エントロピーは情報源

X全体の量を表現している。

The amount of information expresses the amount when an individual notification occurs, while the entropy expresses the amount of the entire information source X.

何回も繰り返す。１回毎ではなく全体としてどの程度の情報量があるか＝エントロピー

Repeat many times. How much information is there as a whole, not every time = entropy

例１）ある地方１の天気

Example 1) Weather in a certain region 1

$$X_1 = \begin{pmatrix} \textit{晴れ} & \textit{くもり} & \textit{雨} & \textit{雪} \\ \frac{1}{2} & \frac{1}{4} & \frac{1}{8} & \frac{1}{8} \end{pmatrix}$$

$$I\left(\textit{晴れ}\right) = -\log\left(\frac{1}{2}\right) = 1 \quad I\left(\textit{くもり}\right) = -\log\left(\frac{1}{4}\right) = 2 \quad I\left(\textit{雨}\right) = I\left(\textit{雪}\right) = -\log\left(\frac{1}{8}\right) = 3$$

エントロピー

$H_1 =$

$= 7/4(\text{bit})$

例２）ある地方２の天気

Example 2) Weather in a certain region 2

$$X_2 = \begin{pmatrix} \textit{晴れ} & \textit{くもり} & \textit{雨} & \textit{雪} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{pmatrix} \qquad H_2 = I\left(\textit{晴れ}\right) = 2\,(bit)$$

考察　$H_1 < H_2$　の原因は何か？　確率の偏り

Discussion What is the cause of H_1<H_2? Bias of probability

確率が偏っている－＞予想が立てやすい－＞エントロピーが小さい

Probability is biased --> Easy to make prediction --> Small entropy

例３）ある地方３の天気

Example 3) Weather in a certain region 3

$$X_3 = \begin{pmatrix} 晴れ & くもり & 雨 & 雪 \\ 1 & 0 & 0 & 0 \end{pmatrix} \qquad H_3 = 0\,(bit)$$

完全に予想が立つ。ただし、$\lim_{x\to 0} x\log x = 0$

Completely predictable.

例４）くじ　２５６本

Example 4) 256 lottery tickets

$$X_4 = \begin{pmatrix} 1等 & 2等 & 3等 & 4等 \\ \frac{1}{256} & \frac{2}{256} & \frac{4}{256} & \frac{249}{256} \end{pmatrix} \quad H_4 = \frac{46+996}{256} = 0.22\,(bit)$$

一等が当たったときは大きいが、全体の平均の情報量（エントロピー）は小さい。つまり、ほとんどはずれる、という予測が立つ。

It is large when the first prize hits, but the average information amount (entropy) of the whole is small. In other words, it is predicted that it will be almost out of alignment.

例５）アルファベット文字の生起確率

Example 5) Occurrence probability of alphabetic characters

１．４　エントロピーの最大値、最小値

1.4 Maximum and minimum entropy values

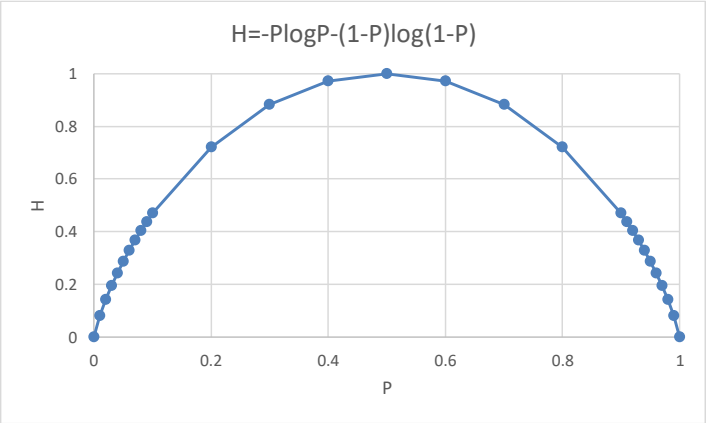
最も単純な２元情報源を考える。

Consider the simplest dual source.

$$X = \begin{pmatrix} S_1 & S_2 \\ P & 1-P \end{pmatrix} \qquad H = -PlogP - (1-P)\log(1-P)$$

【図】グラフ　横軸 P, 縦軸 H

[Figure] Horizontal axis P, Vertical axis H



（課題　このグラフを考察せよ）

(Exercise; Discuss this figure)

一般に、通報の数がn個の情報源を　$X=\begin{pmatrix} S_1 & S_2 & \cdots & S_n \\ P_1 & P_2 & \cdots & P_n \end{pmatrix}$　　$\sum_i P_i = 1$

とする時、エントロピーHが最大となるのは、$P_i = \frac{1}{n}$の時である。

Generally, if there are n sources, then, the maximum entropy is given when pi=1/n.

（課題　上記を証明せよ。）

(Exercise; Prove the above)

【証明】Lagrange の未定定数法を用いる。（P. 5 参照）

[Proof] The Lagrange undetermined constant method is used. (See page 5)

１．５　典型的系列 (typical sequence)

1.5 typical sequence

代表的系列とも言う。

It is also called a representative sequence.

エントロピーHの情報源Xから次々と文字が生起する。

Characters occur one after another from the information source X of entropy H.

文字をN 個連ねて文字系列を作成する。これをN文字系列と表現する。

Create a character sequence by connecting N characters. This is expressed as N character series.

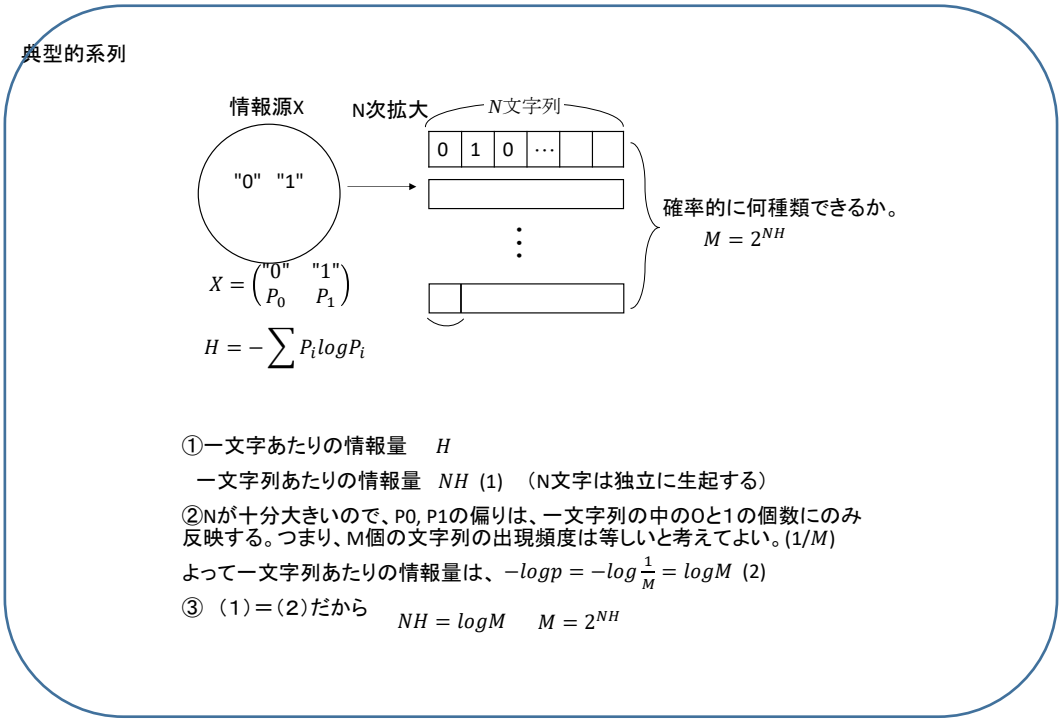
Nが十分大きい場合、ほぼ典型的な文字系列だけが生成される。

If N is large enough, only a typical sequence is generated.

典型的なN文字系列の種類は、$M = 2^{NH}$　である。

We have M=2^NH typical sequences for N.

【図】（簡略証明）



[Figure] (Simplified proof)

Nが十分大きい場合、ほぼ典型的な文字系列だけが生成される。

If N is large enough, only typical sequences are generated.

例１）p0=p1=1/2 の時、H=1。よって$M = 2^{NH} = 2^N$　　：全組合せが発生する

Example 1) When p0=p1=1/2, H=1. Therefore M=2^NH=2^N: All possible combinations are obtained.

例２）p0=1, p1=0 の時、H=0。よって$M = 2^{NH} = 1$　　：１種類しかできない

Example 2) H=0 when p0=1 and p1=0. Therefore, M=2^NH=1: It means only one type is available.

情報通信基礎Ⅰ　第１章　ラグランジェの未定乗数法

Method of Lagrange Multipliers

ラグランジェの未定乗数法　Lagrange's method of indeterminate coefficiency

２変数x, yの関数$f(x,y)$を制約条件$g(x,y)=0$の元で、最大化する。

Maximize the function f(x,y) of two variables x and y under the constraint, g(x,y)=0.

$F(x,y,\lambda)=f(x,y)+\lambda g(x,y)$ と置き、

$\frac{\partial F}{\partial x}=0$　$\frac{\partial F}{\partial y}=0$　$\frac{\partial F}{\partial \lambda}=0$

で得られる連立方程式を満たす$\hat{x}$, $\hat{y}$　が$f(x,y)$の最大値を与える。

X ^, y ^ that satisfies the simultaneous equations given by this equation gives the maximum value of f(x, y).

情報通信基礎Ⅰ　第２章　相互情報量

Information and Communication Fundamentals I Chapter 2 Mutual Information

2020/4/24　情シス　渡辺

ポイント

・情報源が複数

・結合エントロピー、条件付きエントロピー、相互情報量

Points

- Multiple information sources
- Joint entropy, conditional entropy, mutual information

２．１　結合エントロピー

2.1 Joint entropy

情報源XとYを考える。すなわち、

Let us consider two sources, X and Y.

$$X = \begin{pmatrix} x_1 & x_2 & \cdots & x_n \\ P(x_1) & P(x_2) & \cdots & P(x_n) \end{pmatrix} \qquad \sum_{i=1}^{n} P(x_i) = 1$$

$$Y = \begin{pmatrix} y_1 & y_2 & \cdots & y_m \\ P(y_1) & P(y_2) & \cdots & P(y_m) \end{pmatrix} \qquad \sum_{j=1}^{m} P(y_j) = 1$$

XとYを組み合わせた情報源$X \cdot Y$（結合情報源）を考える。(x_1, y_1)などを結合事象と呼ぶ。

Consider an information source X·Y (joint information source) that combines X and Y. (x_1, y_1) and so on are called joint events.

$X \cdot Y =$

$$\sum_{i=1}^{n} \sum_{j=1}^{m} P(x_i, y_j) = 1$$

情報源$X \cdot Y$のエントロピーは、以下となる。

The entropy of the source X·Y is

$$H(X,Y) = -\sum_{i=1}^{n} \sum_{j=1}^{m} P(x_i, y_j) * \log P(x_i, y_j)$$

これを結合エントロピーと呼ぶ。単位は bit。
This is called joint entropy. The unit is bit.

例２．１）コインを２回投げる
Example 2.1) Toss a coin twice

$$X \cdot Y = \begin{pmatrix} (H,H) & (H,T) & (T,H) & (T,T) \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{pmatrix}$$

$$H(X,Y) = \left(-\frac{1}{4}\log\frac{1}{4}\right) * 4 = 2 \text{ (bit)}$$

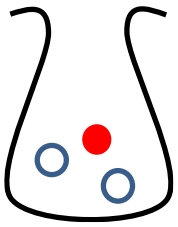
ちなみに、１回目のコイン投げを X とし、２回目のコイン投げを Y と表現すると、
If the first coin flip is X and the second coin flip is Y, then,

$$X = \begin{pmatrix} H & T \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix} \quad H(X) = \left(-\frac{1}{2}\log\frac{1}{2}\right) * 2 = 1$$

$$Y = \begin{pmatrix} H & T \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix} \quad H(Y) = \left(-\frac{1}{2}\log\frac{1}{2}\right) * 2 = 1$$

よって、この例では、$H(X,Y) = H(X) + H(Y)$が成立している。（後に見るように常に等号が成立するわけではない。）
Therefore, in this example, H(X,Y)=H(X)+H(Y) holds. (The equal does not always hold, as we will see later.)

例２．２）赤玉１個と白玉２個が袋に入っている。１回目の玉の色を X、２回目の玉の色を Y とする。ただし、１回目の玉は戻さない。X と Y からなる結合情報源のエントロピーを求めたい。
Example 2.2) One red ball and two white balls are in the bag. Let X be the color of the first ball and Y be the color of the second ball. The first ball is not returned. I want to find the entropy of a joint information source consisting of X and Y.



まず、以下の表のように結合確率を求める。（玉の色を赤R, 白Wと表現し、一回目に赤が出て２回目に赤が出る結合確率を $P(X=R,Y=R)$と表現する。）
First, the joint probability is calculated as shown in the table below. (The color of the red ball is expressed as R and the white ball as W. The joint probability that red appears at the first time and red appears at the second time is expressed as P(X=R, Y=R).)

【表】

$H(X,Y) =$ ≈ 1.58

ちなみに、

And,

$H(X) = -\frac{1}{3}log\frac{1}{3} - \frac{2}{3}log\frac{2}{3} = log3 - \frac{2}{3}$

$H(Y) =$

$H(X) + H(Y) = 2log3 - 4/3 \quad \approx 1.83$

２．２　条件付きエントロピー

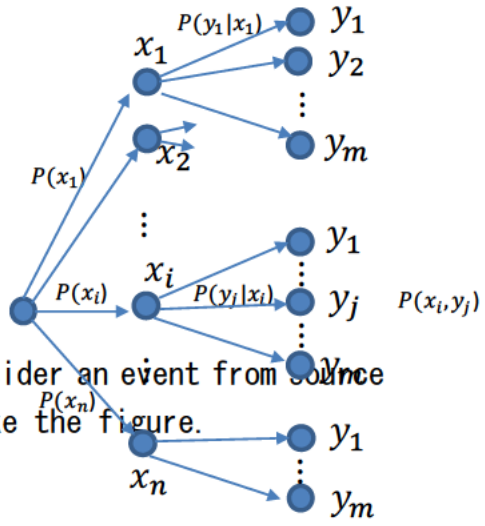
2.2 Conditional entropy

情報源XとYを考える。

Consider sources X and Y.

$$X = \begin{pmatrix} x_1 & x_2 & \cdots & x_n \\ P(x_1) & P(x_2) & \cdots & P(x_n) \end{pmatrix} \qquad \sum_{i=1}^{n} P(x_i) = 1$$

$$Y = \begin{pmatrix} y_1 & y_2 & \cdots & y_m \\ P(y_1) & P(y_2) & \cdots & P(y_m) \end{pmatrix} \qquad \sum_{j=1}^{m} P(y_j) = 1$$



今情報源 X を xi が生起したことを知った上で、情報源 Y で生起する事象を考える。図のような分岐図を書くことができる。

Now, knowing that xi occurred from source X, consider an event from source Y. It is possible to write a branch diagram like the figure.

【図】分岐図

x_iが分かったとき、Yに関してどの程度情報が分かるか。
もし、XとYが無関係（独立）
　　x_iが分かっても、Yに関して何も情報が得られない。
もし、XとYが関係あり（従属）
　　x_iが分かると、Yに関して何か情報が得られる。すなわち、曖昧さが減って予想が立ちやすくなる。

準備
１）ベイズ則
$$\mathrm{P}(x_i, y_j) = \mathrm{P}(y_j|x_i)P(x_i)$$
２）周辺分布
$$\mathrm{P}(x_i) = \sum_j \mathrm{P}(x_i, y_j)$$

ここで、x_iが出た条件の下での情報源$Y|x_i$を考える。

$$Y|x_i = \begin{pmatrix} (y_1|x_i) & (y_2|x_i) & \cdots & (y_j|x_i) & \cdots & (y_m|x_i) \\ P(y_1|x_i) & P(y_2|x_i) & \cdots & P(y_j|x_i) & \cdots & (y_m|x_i) \end{pmatrix}$$
ただし、$\sum_j P(y_j|x_i) = 1$
このエントロピーは、

$$H(Y|x_i) = -\sum_j P(y_j|x_i) log P(y_j|x_i)$$

である。（x_iが生起したと言う条件の下での、Yの予想の立てにくさ）
これをXについても平均する。

$$H(Y|X) = \sum_{i=1}^{n} P(x_i)H(Y|x_i) = -\sum_{i=1}^{n}\sum_{j=1}^{m} P(x_i)P(y_j|x_i) log P(y_j|x_i)$$
$$= -\sum_{i=1}^{n}\sum_{j=1}^{m} P(x_i, y_j) log P(y_j|x_i)$$

　これを条件付きエントロピーと呼ぶ。
同様に、

$$H(X|Y) = -\sum_{i=1}^{n}\sum_{j=1}^{m} P(x_i, y_j) log P(x_i|y_j)$$

例２．２では、２回目の玉の色から１回目の玉の色に関する情報量を考えていることになる。時間に逆行している。しかし、情報通信では重要。

例２．３）例２．２）と同様に、赤玉１個と白玉２個が袋に入っている。１回目の玉の色

を X、２回目の玉の色を Y とする。ただし、１回目の玉は戻さない。条件付きエントロピーを求めたい。

$$H(Y|X) = -\sum_{i=1}^{n}\sum_{j=1}^{m} P(x_i, y_j) log P(y_j|x_i)$$

$P(Y = R|X = R) = \frac{0}{2}$　$P(Y = W|X = R) = \frac{2}{2} = 1$

$P(Y = R|X = W) = \frac{1}{2}$　$P(Y = W|X = W) = \frac{1}{2}$

$H(Y|X) =$ [　　] $= 0.67$ (bit)

$H(Y) = \log 3 - \frac{2}{3} = 0.97$

$H(Y) > H(Y|X)$　　この差は何か。

注意　この例では、$H(X) = H(Y)$　$H(Y|X) = H(X|Y)$ であるが、常に成立するわけではない。演習問題２．１参照。

２．３　エントロピーの性質

① $H(X,Y) = H(X) + H(Y|X)$

左辺：XとYを結合した情報源のエントロピー（曖昧さ、予想の立てにくさ）

右辺第一項：Xのエントロピー（曖昧さ、予想の立てにくさ）

右辺第二項：Xに関して分かったときのYのエントロピー（曖昧さ、予想の立てにくさ）

証明　　数式で解ける

② $H(X,Y) = H(Y) + H(X|Y)$

③ $H(Y) \geq H(Y|X)$

シャノンの不等式と呼ぶ。

左辺：Yの曖昧さ

右辺：Xに関して分かったときのYの曖昧さ

同様に、$H(X) \geq H(X|Y)$

④ $H(X) + H(Y) \geq H(X,Y)$

証明は、①と③より自明。

２．４　相互情報量

シャノンの不等式の差分、すなわち

$$I(X;Y) = H(X) - H(X|Y) = \sum_{i=1}^{n}\sum_{j=1}^{m} P(x_i, y_j) log \frac{P(x_i, y_j)}{P(x_i)P(y_j)}$$

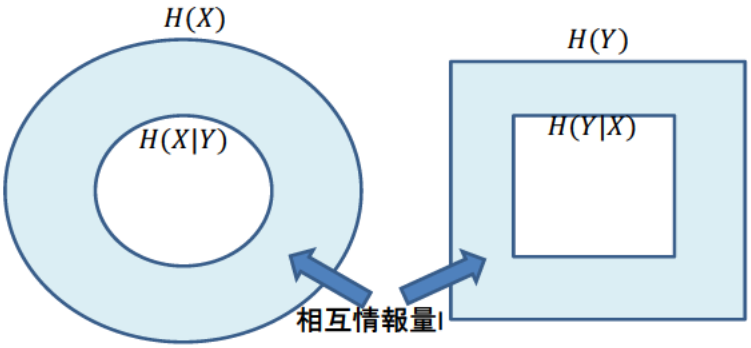
を相互情報量と呼ぶ。

$I(X;Y) = H(X) - H(X|Y)$

Yを知ることによって、どの程度Xの曖昧さが減ったか。どの程度予想が立てやすくなったか。つまり、Yから間接的にXについて得られる情報量。

【図】

$$I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X) = H(X) + H(Y) - H(X,Y) = I(Y;X)$$



（課題　演習問題２．１～２．３）

例２．４）試験の出来具合と表情

A 君

X/Y	にこにこy_1	しずんでいるy_2	無表情y_3	周辺分布
できたx_1	0.45	0	0.05	0.50
できないx_2	0	0.45	0.05	0.50
周辺分布	0.45	0.45	0.10	

$H(X) =$

$P(x_1|y_1) =$ $= 1$　$P(x_1|y_2) = 0$　$P(x_1|y_3) = \frac{0.05}{0.10} = 0.5$

$P(x_2|y_1) = 0$　$P(x_2|y_2) = \frac{0.45}{0.45} = 1$　$P(x_2|y_3) = \frac{0.05}{0.10} = 0.5$

$$H(X|Y) = -\sum_{i=1}^{n}\sum_{j=1}^{m} P(x_i, y_j) log P(x_i|y_j)$$

=

$I = H(X) - H(X|Y) = 0.9$　表情を見るとできたかおおよそ分かる。

B 君

X/Y	にこにこy_1	しずんでいるy_2	無表情y_3	周辺分布
できたx_1	0	0	0.50	0.50
できないx_2	0	0	0.50	0.50
周辺分布	0	0	1	

$H(X) = 1$　$H(X|Y) = 1$　$I = H(X) - H(X|Y) = 0$　表情を見ても分からない。

C 君

X/Y	にこにこy_1	しずんでいるy_2	無表情y_3	周辺分布

できたx_1	0.50	0	0	0.50
できないx_2	0	0.50	0	0.50
周辺分布	0.50	0.50	0	

$H(X)=1$　$H(X|Y)=0$　　$I=H(X)-H(X|Y)=1$　　表情を見ると完全に分かる。

２．５　離散的情報源の種類

２．５．１　記憶のない情報源 Memoryless Information Source　(IS)

情報源からの記号の発生が独立に生じる。

２．５．２　定常情報源　Stationary IS

時刻 t における記号の発生と、時刻 t+α における記号の発生が不偏（確率分布が同一）

２．５．３　記憶のない定常情報源　Independently and identically distributed IS (i.i.d.)

２．５．１＋２．５．２

２．５．４　エルゴード情報源 Ergodic IS

エルゴード性を持つ IS　【図】

十分長い系列に統計的な性質が完全に含まれている。

２．５．５　拡大情報源

$$X=\begin{pmatrix} S_1 & S_2 & S_3 \\ P_1 & P_2 & P_3 \end{pmatrix}$$

$$X^2=\begin{pmatrix} S_1S_1 & S_1S_2 & \cdots & S_3S_3 \\ P_1P_1 & P_1P_2 & \cdots & P_3P_3 \end{pmatrix}$$

２次拡大情報源　（課題　演習問題３．３）

２．５．６　マルコフ情報源

２．６　マルコフ情報源

情報源から、１秒に１回文字が生起することとする。ある時刻tに生起する文字が、時刻$t-1$，$t-2$,,, $t-M$に生起した文字に依存するとき、M 重マルコフ情報源という。

M=0　過去の文字に依存しない場合　０重マルコフ情報源、無記憶情報源

M=1　過去の１文字に依存する場合　１重マルコフ情報源、単純マルコフ情報源

例）英語のアルファベット

例２．５）１重マルコフ情報源X_1

【図】

このエントロピーH_1

①$P(0), P(1)$の定常状態確率を求める。

定常状態方程式 $\vec{\pi} = \vec{\pi}\boldsymbol{P} \quad |\vec{\pi}| = 1$

$\vec{\pi}$：定常状態確率ベクトル　$\vec{\pi} = [\pi_1, \pi_2, , , \pi_n]$　　π_i：状態iの定常状態確率

$\boldsymbol{P}$：遷移確率行列　$\boldsymbol{P} = [P(j|i)]$ 状態iから状態jに遷移する確率

上記の例では、$\vec{\pi} = [\pi_1, \pi_2] = [P(0), P(1)] \quad \boldsymbol{P} = \begin{bmatrix} P(0|0) & P(1|0) \\ P(0|1) & P(1|1) \end{bmatrix}$

これを解くと　$P(0) = \frac{1}{3} \quad P(1) = \frac{2}{3}$

②エントロピーH_1は、

$$H_1 = -\sum_{i=1}^{n}\sum_{j=1}^{m} P(x_1, x_2) log P(x_2|x_1) = -\frac{1}{3}(0log0 + 1log1) - \frac{2}{3}\left(\frac{1}{2}log\frac{1}{2} + \frac{1}{2}log\frac{1}{2}\right) = \frac{2}{3}$$

$$= 0.67$$

参考）0 と 1 がそれぞれ$P(0), P(1)$の確率で無記憶で発生する情報源X_0

$X_0 = \begin{pmatrix} 0 & 1 \\ \frac{1}{3} & \frac{2}{3} \end{pmatrix}$

$H_0 = -\sum \quad P_i * logP_i = -\frac{1}{3}log\frac{1}{3} - \frac{2}{3}log\frac{2}{3} = log3 - \frac{2}{3} = 0.918$

よって、$H_1 < H_0$

１重マルコフ情報源X_1の方が予想が立てやすい。例えば、０の次は必ず１であるから。

（発展　２重マルコフ情報源について調べよ。さらに、M 重マルコフ情報源のエントロピーを求める方法を調べよ。）

情報通信基礎Ⅰ　第３章　情報源符号化

Chapter 3 Information Source Coding

2020/4/26　情シス　渡辺

ポイント

・情報源符号化の定理（シャノンの第一定理）

・ハフマン符号化、シャノン符号化

Points

- Source Coding Theorem (Noiseless Coding Theorem)
- Huffman coding, Shannon coding

同じ情報量を送るなら少ない文字数の方がいい！　どうしたらいいのだろうか。

我々が日常で行っていることは何か？

スマートホンー＞スマホ

パーソナルコンピューターパソコン

情報源符号化定理ー＞情報源符号化定理　情定なんて言わない。なんでだろう？

一方で、それに潜む危険は何か。ー＞第４章に

If we send the same amount of information, it is better to have a small number of characters. What should we do? What do we do the similar things on a daily life?

Ex)

Cellular phone-> cell phone, Personal computer-> PC

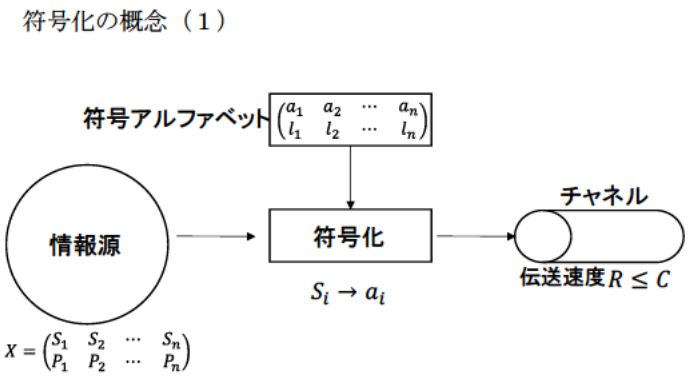
Source Coding Theory -> Source Coding Theorem.　We don't say SCT. Why? What are the disadvantage of SCT? -> Chapter 4

３．１　基礎

【図３．１】

符号化の概念（１）

Encoding concept (1)



符号化：S_iを別のa_iに一対一に割り当てる。

符号アルファベットa_i　の例【図３．２】

（０，１）、(アナログ信号)、(日本語のあいうえお)、(モールス信号）等（補足資料２）

Encoding: Assign S_i to another a_i one-to-one corresponding.
Example of code alphabet a_i
(analog signals), (Morse code), etc. (See Additional material 2)

情報源Xのエントロピーは、
$H(X) = -\sum_{i=1}^{n} P_i * \log P_i$　である。
一方で、一符号語の平均長（時間；平均符号語長）Lは、$\sum_{i=1}^{n} P_i * l_i$
従って、情報伝送速度Rは、（１符号語の持つ情報量）/（１符号語の伝送時間）であり、以下で表現できる。

$$R = \frac{H}{L}$$

単位は、(bit/sec)。

The entropy of Source X is

$$H(X) = -\sum_{i=1}^{n} P_i * \log P_i$$

On the other hand, the average length (a.k.a ,time or average code word length)
L of one code word is

$$\sum_{i=1}^{n} P_i * l_i$$

Therefore, the information transmission rate R can be (the amount of information carried by one code word) / (the transmission time of one code word) expressed as follows.

$$R = \frac{H}{L}$$

The unit is (bit / sec).

Rは、大きいほどよい。　つまり、

$$R = \frac{H}{L} = -$$

を最大化するにはどうしたらよいかを考える。

The larger R is, the better. Then, let's think about how to maximize the following.

$$R = \frac{H}{L} = \frac{-\sum_{i=1}^{n} P_i * \log P_i}{\sum_{i=1}^{n} P_i * l_i}$$

アプローチ１）l_i　が与えられたとき、Rを最大化するP_iを求める。
Approach 1) When l_i is given, find P_i that maximizes R.

［定理３．１］

[Theorem 3.1]

Rの最大値Cは、次式の正の根で与えられる。

The maximum value C of R is given by the positive root of the following equation.

$$\sum_{i=1}^{n} 2^{-Cl_i} = 1$$

このとき、以下となる。

At this time, the following holds.

$$P_i = 2^{-Cl_i}$$

（課題　証明せよ）

(Prove the theorem)

ヒント

ラグランジュの未定乗数法を用いる。

$F(P_1, P_2, , , P_n, \lambda) = R(P_1, P_2, , , , P_n) + \lambda(\sum P_i - 1)$　として、これを各変数で偏微分して、

$\frac{\partial F}{\partial P_i} = 0$　$\frac{\partial F}{\partial \lambda} = 0$　を解く。

Tips

Use Lagrange's undetermined multiplier method.

Partially differentiate the following equation with respect to each variable.

$F(P_1, P_2, , , P_n, \lambda) = R(P_1, P_2, , , , P_n) + \lambda(\sum P_i - 1)$

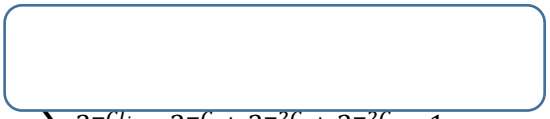
$$\frac{\partial F}{\partial P_i} = 0 \quad \frac{\partial F}{\partial \lambda} = 0$$

C:　通信路に誤りがない場合には、Rの最大値Cはそのまま通信路の最大伝送速度になる。このためCは通信路容量と呼ばれる。

C: If there is no error in the communication channel, the maximum value C of R becomes the maximum transmission rate of the channel. For this reason, C is called the channel capacity.

例３．１）　符号アルファベット集合が$\begin{pmatrix} a & b & c \\ 1 & 2 & 2 \end{pmatrix}$のときの通信路容量を求めよ。

Example 3.1) Find the channel capacity when the code alphabet set is $\begin{pmatrix} a & b & c \\ 1 & 2 & 2 \end{pmatrix}$.



$$\sum_{i=1} 2^{-Cl_i} = 2^{-C} + 2^{-2C} + 2^{-2C} = 1$$

$x = 2^{-C} > 0$ とすると、$x + x^2 + x^2 = 1$

これを解いて、$x = \frac{1}{2}$、$C = 1$を得る。よって、

$P_a = 2^{-C} = \frac{1}{2}$　$P_b = 2^{-2C} = \frac{1}{4}$　$P_c = 2^{-2C} = \frac{1}{4}$

とすれば、Rは最大値Cをとる。

この例からも分かるように、小さいl_iの符号アルファベットには、大きなP_i　のS_iを割り当てることがポイントである。

As you can see from this example, the point is assigning S_i with a large P_i to the alphabet with small l_i.

アプローチ２）P_i　が与えられたとき、Rを最大化するl_iを求める。

Approach 2) When P_i is given, find l_i that maximizes R.

l_iを変化させ、$P_i = 2^{-Cl_i}$　を満足するように決める。

Change l_i to satisfy P_i = 2 ^ (-Cl_i).

例３．２）

$X = \begin{pmatrix} S_1 & S_2 & S_3 & S_4 \\ \frac{1}{2} & \frac{1}{4} & \frac{1}{8} & \frac{1}{8} \end{pmatrix}$ のとき、

$S_1 \to a_1$　　0　$l_1 = 1$

$S_2 \to a_2$　　1 0　$l_2 = 2$

$S_3 \to a_3$　1 1 0　$l_3 = 3$

$S_4 \to a_4$　1 1 1　$l_4 = 3$

と符号化すると、

$\sum_{i=1}^{n} 2^{-Cl_i} = 2^{-C} + 2^{-2C} + 2^{-3C} + 2^{-3C} = 1$　より、

$C = 1$　であり、$P_i = 2^{-Cl_i}$　とうまく一致している。

アプローチ３）実際には、l_i　や　P_i には制約があって、必ずしも満足させることができない。

Approach 3) Actually, l_i and P_i have restrictions and cannot always be satisfied.

例えば、① P_i やl_iが何らかの理由で固定の場合、② l_iが０，１の個数で決定される離散値の場合、など。どうしたらよいか。

For example, (1) when P_i やl_i is fixed for some reason, (2) when l_i is a discrete value. What should we do?

cf.　　アプローチ１、２は、

S_i, P_i　→　a_i, l_i のように一対一で対応させている。

cf. Approaches 1 and 2 have a one-to-one correspondence such as $S_i, P_i \rightarrow a_i, l_i$.

$S_1S_2S_2 \rightarrow a_1a_5a_6a_8$　などとまとめて符号化する。

For example, the block $S_1S_2S_2$ are coded into $a_1a_5a_6a_8$.

（確率$P_1P_2P_2$で生起する事象に、$l_1 + l_5 + l_6 + l_8$の長さの符号を割り当てる。）

(A code with a length of $l_1 + l_5 + l_6 + l_8$ is assigned to the event that occurs with probability $P_1P_2P_2$.)

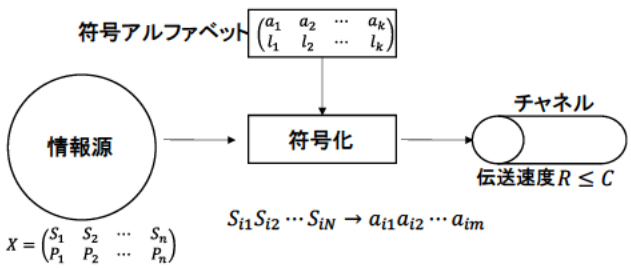
つまり、S_iの文字列をa_jの文字列に対応させ、$\prod P_i = 2^{-c\Sigma l_j}$ として自由度を上げれば、Rを最大化できる可能性がある。

In other words, if the sequence of S_i corresponds to the sequence of a_j, which means $\prod P_i = 2^{-c\Sigma l_j}$, then the degree of freedom is increased to maximize R.

３．２　情報源符号化の定理

3.2 Source Coding Theorem

【図３．３】　符号化の概念（２）



［シャノン第１定理（情報源符号化定理）］

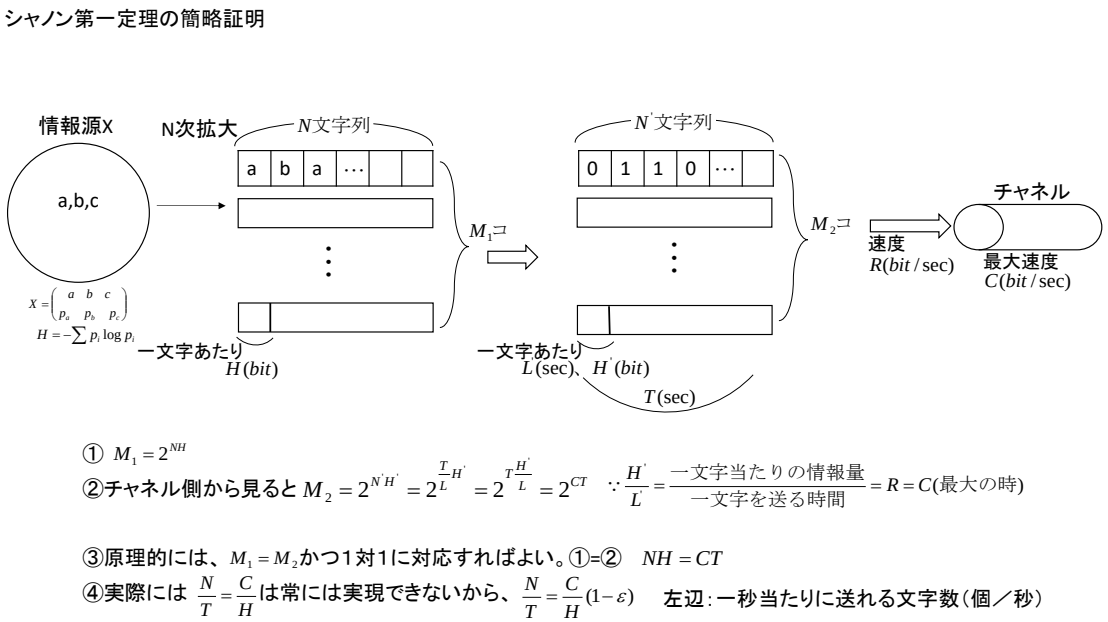
[Source Coding Theorem]

情報源のエントロピーをH(bit)、通路容量をC(bit/sec)とするとき、
$\frac{C}{H}(1-\varepsilon)$個/sec で伝送可能な符号が存在する。εは、任意に小さい正数。
When the entropy of the information source is H (bit) and the channel capacity is C (bit / sec), there are codes that can be transmitted at $\frac{C}{H}(1-\varepsilon)$ per second. ε is an arbitrarily small positive number.

（別形式　今井 p.76）平均符号語長を$\bar{m}$とするとき、$H \leq \bar{m} < H + \varepsilon$　となる符号化が存在する。（クラフトの不等式を満たす 2 元可分符号の場合）補足資料 1

(Another form , Imai p.76) There is a coding method such that$H \leq \bar{m} < H + \varepsilon$, where the average code word length is $\bar{m}$. (In the case of a binary separable code that satisfies the Kraft inequality) See supplement material 1

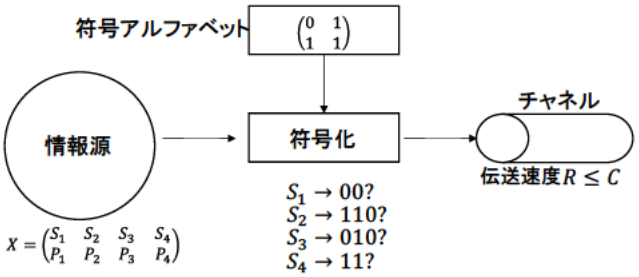
【図３．４】シャノン第 1 定理の証明
[Fig. 3.4] Proof of Source Coding Theory



３．３　符号の性質

3.3 Nature of code

【図３．５】　　符号の性質



３．３．１　一意に復号可能、瞬時に復号可能

3.3.1 Unique decodability, instant decodability

a)符号化方法１

a) Coding method 1

$S_1 \rightarrow$　０　　$m_1 = 1$（符号語長）

$S_2 \rightarrow$　１０　　$m_2 = 2$

$S_3 \rightarrow$　１１０　　$m_3 = 3$

$S_4 \rightarrow$　１１１０　$m_4 = 4$

このとき、０１０１１０１０・・を受信した。復号できるか。

When 0101010... was received, can we decode it correctly?

０　１０　１１０　１０・・・

S_1　S_2　S_3　・・・

一意に復号可能

Uniquely decodable

b)符号化方法２

b) Coding method 2

$S_1 \rightarrow$　０　　$m_1 = 1$（符号語長）

$S_2 \rightarrow$　１０　　$m_2 = 2$

$S_3 \rightarrow$　１１　　$m_3 = 2$

$S_4 \rightarrow$　１１０　　$m_4 = 3$

このとき、０１０１１０１０・・を受信した。復号できるか。

When 0101010... was received, can we decode it correctly?

０　１０　１１０　１０・・・

S_1　S_2　$S_3?S_4?$　・・・

一意に復号不可能

Uniquely undecodable

c)符号化方法３

c) Coding method 3

$S_1 \rightarrow$　０　　$m_1 = 1$（符号語長）

$S_2 \rightarrow$　０１　　$m_2 = 2$

$S_3 \rightarrow$　０１１　　$m_3 = 3$

$S_4 \rightarrow$　０１１１　$m_4 = 4$

このとき、０１０１１０１０・・を受信した。復号できるか。

When 0101010... was received, can we decode it correctly?

０１　０１１　０１　０・・・

S_2--- ↑　S_3--- ↑ S_2--- ↑ ・・・

一意に復号可能　ただし、瞬時に復号は不可能

Uniquely decodable, instantly undecodable

Kraft の不等式

Kraft inequality

一意に復号可能な条件

Condition of unique decodability

情報源Xを以下とする。

The information source X is as follows.

$$X = \begin{pmatrix} S_1 & S_2 & \cdots & S_n \\ P_1 & P_2 & \cdots & P_n \end{pmatrix} \qquad \sum_{i=1}^{n} P_i = 1$$

S_iに割り当てられた符号語の長さをm_iとする。

Let m_i be the length of the code word assigned to S_i.

符号アルファベットがk個（k進）の場合、一意に復号可能な符号化は、以下の不等式を満たす。

When there are k code alphabets (k-ary), the uniquely decodable coding satisfies the following inequality.

$$\sum_{i=1}^{n} k^{-m_i} \leq 1$$

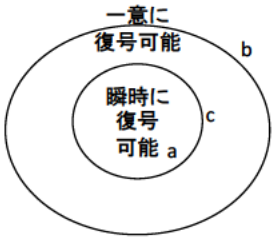
（課題　証明せよ。ヒント符号木において、すべての符号語を木の終端節点に対応するように構成する。）
(Prove the problem. Tip: In a code tree, all code words are configured to correspond to the end nodes of the tree.)

a) c)の場合、
a) and c)

$$\sum_{i=1}^{n} 2^{-m_i} = 2^{-1} + 2^{-2} + 2^{-3} + 2^{-4} = 15 \cdot 2^{-4} = \frac{15}{16} \leq 1$$

b)の場合、
b)

$$\sum_{i=1}^{n} 2^{-m_i} = 2^{-1} + 2^{-2} + 2^{-2} + 2^{-3} = 9 \cdot 2^{-3} > 1$$



【図３．６】一意に復号可能、瞬時に復号可能
[Fig. 3. 6] Uniquely decodable, instantly decodable

（発展　一意性、瞬時性に関しては、マクミランの不等式、サーディナスパターソン定理など）
(Advanced topics: For uniqueness and instantaneousness, see McMillan's inequality, Sardinas–Patterson algorithm, etc.)

３．３．２　平均符号語長
3.3.2 Average code word length

情報源$X = \begin{pmatrix} S_1 & S_2 & \cdots & S_n \\ P_1 & P_2 & \cdots & P_n \end{pmatrix}$から生起する文字列が、符号アルファベットの集合$\begin{pmatrix} a_1 & a_2 & \cdots & a_n \\ l_1 & l_2 & \cdots & l_n \end{pmatrix}$を用いて一対一に符号化される時、符号語長の平均（平均符号語長）$\bar{m}$は、以下で計算される。

When a sequence generated from information source, $X = \begin{pmatrix} S_1 & S_2 & \cdots & S_n \\ P_1 & P_2 & \cdots & P_n \end{pmatrix}$ is encoded one-to-one using a set of code alphabets, $\begin{pmatrix} a_1 & a_2 & \cdots & a_n \\ l_1 & l_2 & \cdots & l_n \end{pmatrix}$, then, the average code word length $\bar{m}$ is calculated as follows.

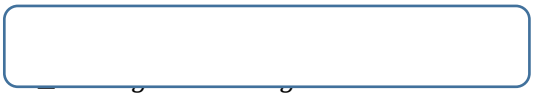
$$\bar{m} = \sum_{i=1}^{n} P_i m_i$$

（簡単のため、符号化の概念（１）と同じ条件としたが、符号化の概念（２）でも同様の議論ができる。）
(For the sake of simplicity, the same condition as encoding concept (1) in Fig. 3.1 is used, but the same argument can be made with the encoding concept (2) in Fig. 3.3.)

平均符号語長$\bar{m}$　は、小さいほどよい。
The smaller the average code word length $\bar{m}$, the better.

［定理３．２］
[Theorem 3.2]

Kraft の不等式を満たす符号は、以下を満たす。（課題　証明せよ。）
A code that satisfies the Kraft inequality satisfies the following. (Prove the theorem.)



証明中　　$-log_k P_i \leq m_i < -log_k P_i + 1$　　－＞シャノンの符号化

［定理３．２］の解釈（$k = 2$ とすると）
Interpretation of [Theorem 3.2] (assuming k = 2)

・$\bar{m}$をHより小さくはできない。
・$\bar{m}$を$H + 1$以下にはできる。
・ $\bar{m}$ cannot be smaller than H.
・ $\bar{m}$ can be less than $H + 1$.

a)符号化方法１の場合

a) Coding method 1

$S_1 \rightarrow$ 0 $m_1 = 1$ $P_1 = \frac{1}{2}$

$S_2 \rightarrow$ 1 0 $m_2 = 2$ $P_2 = \frac{1}{4}$

$S_3 \rightarrow$ 1 1 0 $m_3 = 3$ $P_3 = \frac{1}{8}$

$S_4 \rightarrow$ 1 1 1 0 $m_4 = 4$ $P_4 = \frac{1}{8}$

$\bar{m} = \frac{1}{2} \cdot 1 + \frac{1}{4} \cdot 2 + \frac{1}{8} \cdot 3 + \frac{1}{8} \cdot 4 = \frac{15}{8}$ $H = \frac{14}{8}$ $\therefore H < \bar{m} < H + 1$

d)符号化方法４

d) Coding method 4

$S_1 \rightarrow$ 0 $m_1 = 1$ $P_1 = \frac{1}{2}$

$S_2 \rightarrow$ 1 0 $m_2 = 2$ $P_2 = \frac{1}{4}$

$S_3 \rightarrow$ 1 1 0 $m_3 = 3$ $P_3 = \frac{1}{8}$

$S_4 \rightarrow$ 1 1 1 $m_4 = 3$ $P_4 = \frac{1}{8}$

一意に復号可能、瞬時に復号可能

Uniquely decodable, instantly decodable

$\bar{m} = \frac{1}{2} \cdot 1 + \frac{1}{4} \cdot 2 + \frac{1}{8} \cdot 3 + \frac{1}{8} \cdot 3 = \frac{14}{8}$ $H = \frac{14}{8}$ $\therefore H = \bar{m}$ 非常によい符号化

Excellent coding

３．４　具体的な符号化方法

3.4 Specific coding method

１）シャノンの符号化

1) Shannon coding

（１）生起確率P_iの高い順に上から並べる。

（２）累積確率 $q_i = \sum_{j=1}^{i-1} P_j \quad (i \geq 2)$ を求める。

（３）$-log\ P_i \leq m_i < -log\ P_i + 1$ となるm_iを求める。

（４）q_iの２進数展開の小数点以下m_i桁を取る。

(1) Sort the symbols from the top in descending order of probability P_i.

(2) Calculate cumulative probability $q_i = \sum_{j=1}^{i-1} P_j \quad (i \geq 2)$.

(3) Find m_i such that $-log\ P_i \leq m_i < -log\ P_i + 1$.

(4) Take the m_i digits after the decimal point of the binary expansion of q_i.

２）ハフマンの符号化

1) Huffman coding

（１）生起確率の高い順に上から並べる。

（２）下から２つ（すなわち最も生起確率の小さいもの２つ）を１グループとした枝にした木を作る。それらの確率を加算してそのグループの生起確率とする。

（３）新たなグループも含めて生起確率の高い順に上から並べる。

（４）グループが１つになる、すなわち生起確率が１になるまで、（２）（３）を繰り返す。

（５）全確率（=1）のルートから木をたどり分岐に対して０と１を割り付ける

(1) Sort the symbols from the top in descending order of probability.

(2) Make a tree with branches from the bottom two (that is, the two with the lowest probability) as one group. Add those probabilities to get the probability for that group.

(3) Sort again the symbols in descending order, including new groups.

(4) Repeat (2) and (3) until the number of groups becomes one, that is, the probability becomes one.

(5) Follow the tree from the route with all probabilities (= 1) and assign 0 and 1 to the branch.

例

Examples

１）シャノンの符号化

1) Shannon coding

S_i	P_i	q_i 二進数展開		m_i
S_i	P_i	q_i binary expansion		m_i
S_1	$\frac{5}{16}$	$0 = 0.0000$	$-log\frac{5}{16} = 1.68$	2
S_2	$\frac{4}{16}$	$\frac{5}{16} = 0.0101$	$-log\frac{4}{16} = 2$	2
S_3	$\frac{3}{16}$	$\frac{9}{16} = 0.1001$	$-log\frac{3}{16} = 4 - \log 3$	3
S_4	$\frac{2}{16}$	$\frac{12}{16} = 0.1100$	$-log\frac{2}{16}$	3
S_5	$\frac{1}{16}$	$\frac{14}{16} = 0.1110$	$-log\frac{1}{16}$	4
S_6	$\frac{1}{16}$	$\frac{15}{16} = 0.1111$	$-log\frac{1}{16}$	4

以上より、

Thus,

$S_1 \rightarrow$ ００

$S_2 \rightarrow$ ０１

$S_3 \rightarrow$ １００

$S_4 \rightarrow$ １１０

$S_5 \rightarrow$　　1 1 1 0

$S_6 \rightarrow$　　1 1 1 1

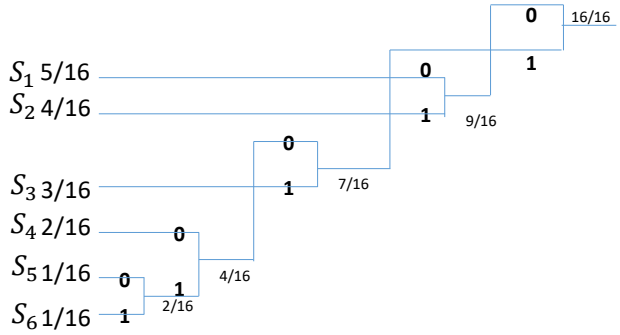
シャノン符号の平均符号語長

The average code word length

$\bar{m}_s =$

2）ハフマンの符号化

1) Huffman coding



以上より、

Thus,

$S_1 \rightarrow$　　0 0

$S_2 \rightarrow$　　0 1

$S_3 \rightarrow$　　1 1

$S_4 \rightarrow$　　1 0 0

$S_5 \rightarrow$　　1 0 1 0

$S_6 \rightarrow$　　1 0 1 1

ハフマン符号の平均符号語長

The average code word length

$\bar{m}_h =$

$$H = \frac{5}{16}(4 - log5) + \frac{4}{16}(2) + \frac{3}{16}(4 - log3) + \frac{2}{16}(3) + \frac{1}{16}(4) + \frac{1}{16}(4)$$

$$= \frac{1}{16}(20 + 8 + 12 + 6 + 4 + 4 - 5log5 - 3log3) = \frac{1}{16}(54 - 11.6 - 4.75)$$

$$= 2.35$$

$$H < \bar{m}_h < \bar{m}_s$$

ハフマン符号の注意点

Discussion of Huffman coding

・分岐（葉）に割り付ける０，１は、任意（上から０，１でも上から１，０でもよい。分岐毎に変えてもよい。）

- The assignment of 0 and 1 to the branch (leaf) is arbitrary.

・平均符号語長$\bar{m}$を最小にするという意味でコンパクト符号と呼ばれる。

- It is called a compact code in the sense that it minimizes the average code word length, $\bar{m}$.

・接頭符号(Prefix code)

- One of prefix codes.

（課題　ハフマン符号をコンピュータで実現するアルゴリズムを考えよ。データ構造を定義し、フローチャートを作成せよ。実際のプログラミングは行わなくてもよい。）

(Exercise: Develop an algorithm of Huffman coding by defining the data structure and a flowchart for the programming.)

（発展）

(Advanced topics)

・k 進のハフマン符号化はどのようにしたら実現するか。

・ハフマン符号のコンパクト性の証明

・ハフマン符号は、末端の葉（つまり最小の確率を持つ点）から開始した。逆に根（つまり全確率１）からほぼ等確率（２進の場合は１／２）になるように割り付ける符号化（ファノ符号）がある。

- How can Huffman coding in k-ary be achieved?

- Proof of compactness of Huffman code

- The Huffman code starts from the terminal leaves (that is, with the lowest probability). Whereas, Fano coding starts from the root to divide it into k leaves with almost the same probability (1/2 in the case of binary).

情報通信基礎Ⅰ　第４章　通信路符号化

Chapter 4 Channel Coding

2020/6/4　情シス　渡辺

ポイント

・第３章では、平均符号語長$\bar{m}$をできるだけHに近づけた。（冗長度をなくす）
　データ圧縮、誤りのない通信路での通信
・第４章では、雑音がある通信路で誤ったとしても正しく復号できる符号化を考える。
　（冗長度を加える）
・通信路符号化の定理（シャノンの第二定理）

Points

- In Chapter 3, the average code word length $\bar{m}$ was made as close to H as possible. (Redundancy elimination for data compression, communication on a communication path without errors)
- In Chapter 4, we will consider coding that can be correctly decoded even if errors are made via a noisy communication channel. (Redundancy addition for error-prone channel)
- Shannon's channel coding theorem

誤りがある場合、我々はどうしているのだろうか？

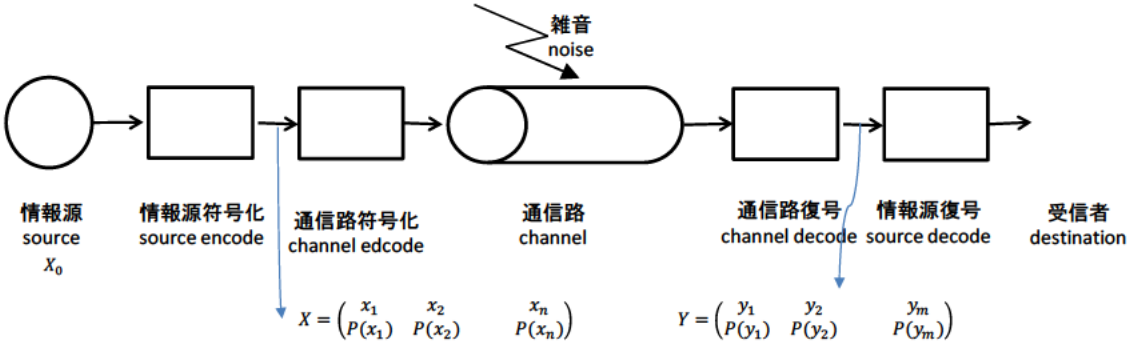
単純に繰り返したのでは時間がかかる。

想像を働かせて訂正する。一つの言葉が別の意味に解釈されるのを防ぐ。異なる語や言葉が同一の意味に解釈されるのを防ぐ。l と１、o と 0、れとわ。

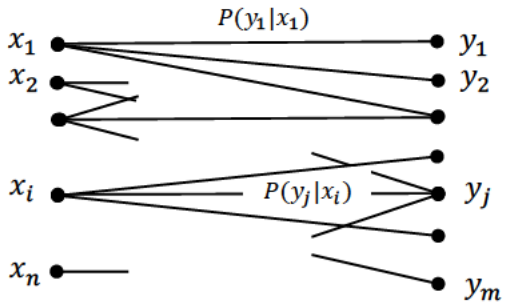
What do we do if we may have words with errors?
Simply repeat it. But, it takes time.
Imagine the correct words from words with errors.
Prevent one word from being interpreted into another.
Prevent different words from being interpreted as having the same meaning.
For example, we might not use l and 1, or o and 0 since the pairs are very confusing.

４．１　基礎

【図４．１】



【図４．２】



x_iを送るとy_jが受信される確率は条件付き確率　$P(y_j|x_i)$　で表現される。これを行列として並べたℙを通信路行列と呼ぶ。通信路行列は、通信路の誤りやすさの性質を表している。

The probability that y_i will be received after x_i is sent is expressed by the conditional probability $P(y_j|x_i)$.

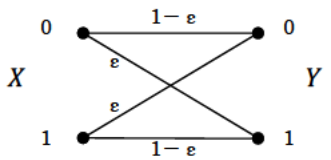
ℙ, where $P(y_j|x_i)$ for all i,j are arranged as a matrix, is called a channel matrix.

The channel matrix represents the nature of the channel error proneness.

$$\mathbb{P} = \begin{bmatrix} P(y_1|x_1) & P(y_2|x_1) & \cdots & P(y_m|x_1) \\ P(y_1|x_2) & \cdots & \cdots & P(y_m|x_2) \\ \vdots & \cdots & \cdots & \vdots \\ P(y_1|x_n) & \cdots & \cdots & P(y_m|x_n) \end{bmatrix}$$

例４．１　二元対称通信路(BSC; binary symmetric channel)

【図４．３】



$$\mathbb{P} = \begin{bmatrix} 1-\varepsilon & \varepsilon \\ \varepsilon & 1-\varepsilon \end{bmatrix}$$

４．２　通信誤り

【図４．４】　アナログ信号の誤り　ビットエラー率 BER (Bit Error Rate)

４．３　情報伝送速度

4.3 Information transmission rate

4.3 Taux de transmission des informations

図４．１に示したように、チャネルへの入力を情報源X、チャネルからの出力を情報源Yと考える。

As shown in Fig. 4.1, the input to the channel is considered to be the information source X, and the output from the channel is considered to be another information source Y. Thus,

すなわち、

$$X = \begin{pmatrix} x_1 & x_2 & \cdots & x_n \\ P(x_1) & P(x_2) & \cdots & P(x_n) \end{pmatrix} \qquad \sum_{i=1}^{n} P(x_i) = 1$$

$$Y = \begin{pmatrix} y_1 & y_2 & \cdots & y_m \\ P(y_1) & P(y_2) & \cdots & P(y_m) \end{pmatrix} \qquad \sum_{j=1}^{m} P(y_j) = 1$$

Yを情報「源」と呼ぶには抵抗感があるかもしれないが、十分長い時間Yを観測すれば、y_jの確率を計測でき情報「源」のように書ける。つまり、情報源と見なせる。

You may be embarrassed to call Y the information "source", but if we observe Y for a long enough time, we can measure the probability of y_j and write it like the information source format. In other words, we can regard it as a source of information.

入力側から送られるエントロピー（一文字あたりの情報量）は、以下である。

The entropy (amount of information per character) sent from the input side is as follows.

$$H(X) = -\sum_{i}^{n} P(x_i) log P(x_i)$$

出力側に現れるエントロピーは、以下である。

The entropy that appears on the output side is as follows.

$$H(Y) = -\sum_{j}^{m} P(y_j) log P(y_j)$$

XについてYから間接的に得られる情報量、すなわち相互情報量は、以下で表現される。

The amount of information implicitly obtained from Y about X, that is, the amount of mutual information, is expressed as follows.

$$I(X;Y) = H(Y) - H(Y|X) = \sum_{i=1}^{n}\sum_{j=1}^{m} P(x_i, y_j) log \frac{P(x_i, y_j)}{P(x_i)P(y_j)} = H(X) - H(X|Y)$$

で表現される。(送信側では$H(X)$の情報量を送っているが途中で誤るので$H(X|Y)$だけ曖昧さの減少量が減る。(情報量が減る。))

(The sender sends the amount of H (X) information. Since some part of the information may change, the amount of ambiguity is reduced by H (X | Y).)

一文字あたりの伝送時間をlとすると、情報伝送速度Rは、$R = \frac{I(X;Y)}{l}$ であり、Rの最大値をC（通信路容量）と呼ぶ。すなわち、

Assuming that the transmission time per character is l, the transmission rate R is R = (I (X; Y)) / l, and the maximum value of R is called C (channel capacity).

$$C = max \frac{I(X;Y)}{l}$$

である。今簡単のため、lを定数とすると、

For simplicity now, let l be a constant.

$$lC = max\, I(X;Y) = \max\{H(Y) - H(Y|X)\}$$

となる。これを最大化するには、以下が考えられる。

To maximize this, the followings can be considered.

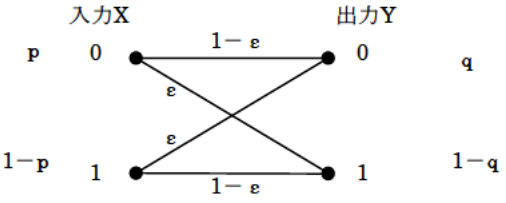
・$H(Y)$を大きくする。
・$H(Y|X)$を小さくする。

・ Increase H (Y).
・ Reduce H (Y | X).

例４．２　二元対称通信路の通信路容量

Example 4.2 Channel capacity of BSC

【図４．５】



$$\mathbb{P} = \begin{bmatrix} 1-\varepsilon & \varepsilon \\ \varepsilon & 1-\varepsilon \end{bmatrix}$$

$$X = \begin{pmatrix} 0 & 1 \\ p & 1-p \end{pmatrix} \qquad Y = \begin{pmatrix} 0 & 1 \\ q & 1-q \end{pmatrix}$$

$$I(X;Y) = H(Y) - H(Y|X) \qquad (式4-1)$$

$$H(Y) = -qlogq - (1-q)\log(1-q) \qquad (式4-2)$$

$$H(Y|X) = -\sum_{i=1}^{n}\sum_{j=1}^{m} P(x_i, y_j) * \log P(y_j|x_i) = -\sum_{i=1}^{n} P(x_i) \sum_{j=1}^{m} P(y_j|x_i) * \log P(y_j|x_i)$$

$$= -p\{(1-\varepsilon)log(1-\varepsilon) + \varepsilon log\varepsilon\} - (1-p)\{\varepsilon log\varepsilon + (1-\varepsilon)log(1-\varepsilon)\}$$

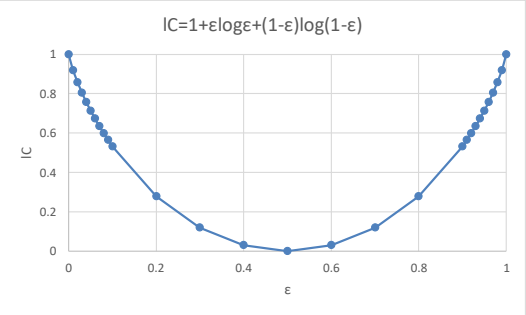
$$= \boxed{\qquad\qquad} \qquad (式4-3)$$

すなわち、$H(Y|X)$は、p, qに無関係。使う通信路によって決定される。
従って、(通信路が固定でεは変更できないとすると、) XについてYから間接的に得られる情報量、すなわち相互情報量$I(X;Y) = H(Y) - H(Y|X)$　が最大となるのは、$H(Y)$が最大になるとき、すなわち、$q = \frac{1}{2}$の時である。
よって、
That is, H (Y | X) is irrelevant to p and q. It is determined by the communication path used.
Therefore, the amount of information indirectly obtained from Y for X (assuming that the communication path is fixed and ε cannot be changed), that is, mutual information I (X; Y) = H (Y) -H (Y | X) can be maximized when H (Y) is maximum. At that time, q = 1/2.
Then,

$$lC = H(Y)_{|q=\frac{1}{2}} - H(Y|X) = -\frac{1}{2}log\frac{1}{2} - \frac{1}{2}log\frac{1}{2} + \varepsilon log\varepsilon + (1-\varepsilon)log(1-\varepsilon)$$

$$= 1 + \varepsilon log\varepsilon + (1-\varepsilon)log(1-\varepsilon) \qquad (式4-4)$$

【図４．６】縦軸lc、横軸ε



$\varepsilon = 0.5$ の時、実質的な通信が全く行えない。
When ε = 0.5, no actual information can be transmitted.

（より詳細な計算１）$I(X;Y) = H(Y) - H(Y|X)$で計算する場合

(More detailed calculation 1) When calculating with I (X; Y) = H (Y) -H (Y | X)

$P(Y=0) = P(X=0)P(Y=0|X=0) + P(X=1)P(Y=0|X=1) = p(1-\varepsilon) + (1-p)\varepsilon$

$P(Y=1) = P(X=0)P(Y=1|X=0) + P(X=1)P(Y=1|X=1) = p\varepsilon + (1-p)(1-\varepsilon)$

$$H(Y) = -\sum_{j=1}^{m} P(y_j) log P(y_j)$$

$$= -[\{p(1-\varepsilon) + (1-p)\varepsilon\} \log\{p(1-\varepsilon) + (1-p)\varepsilon\}$$

$$+ \{p\varepsilon + (1-p)(1-\varepsilon) \log\{p\varepsilon + (1-p)(1-\varepsilon)\}] \quad (式4-5)$$

$P(X=0,Y=0) = P(X=0)P(Y=0|X=0) = p(1-\varepsilon)$　　$P(X=0,Y=1) = P(X=0)P(Y=1|X=0) = p\varepsilon$

$P(X=1,Y=0) = P(X=1)P(Y=0|X=1) = (1-p)\varepsilon$　　$P(X=1,Y=1) = P(X=1)P(Y=1|X=1) = (1-p)(1-\varepsilon)$

$$H(Y|X) = -\sum_{i=1}^{n}\sum_{j=1}^{m} P(x_i, y_j) * \log P(y_j|x_i) = -p(1-\varepsilon)\log(1-\varepsilon) - p\varepsilon log\varepsilon - (1-p)\varepsilon log\varepsilon - (1-p)(1-\varepsilon)log(1-\varepsilon)$$

$$= -\varepsilon log\varepsilon - (1-\varepsilon)log(1-\varepsilon) \quad (式4-6)$$

以上より、

$$I(X;Y) = H(Y) - H(Y|X)$$

$$= -\{p(1-\varepsilon) + (1-p)\varepsilon\} \log\{p(1-\varepsilon) + (1-p)\varepsilon\} - \{p\varepsilon + (1-p)(1-\varepsilon) \log\{p\varepsilon + (1-p)(1-\varepsilon)\}$$

$$+ \varepsilon log\varepsilon + (1-\varepsilon)log(1-\varepsilon) \quad 式（4-7）$$

（より詳細な計算２）　$I(X;Y) = H(X) - H(X|Y)$で計算する場合

(More detailed calculation 2) When calculating with I (X; Y) = H (X) -H (X | Y)

$P(X=0|Y=0) = \frac{P(X=0,Y=0)}{P(Y=0)} = \frac{p(1-\varepsilon)}{p(1-\varepsilon)+(1-p)\varepsilon}$　　$P(X=0|Y=1) = \frac{P(X=0,Y=1)}{P(Y=1)} = \frac{p\varepsilon}{p\varepsilon+(1-p)(1-\varepsilon)}$

$P(X=1|Y=0) = \frac{P(X=1,Y=0)}{P(Y=0)} = \frac{(1-p)\varepsilon}{p(1-\varepsilon)+(1-p)\varepsilon}$　　$P(X=1|Y=1) = \frac{P(X=1,Y=1)}{P(Y=1)} = \frac{(1-p)(1-\varepsilon)}{p\varepsilon+(1-p)(1-\varepsilon)}$

$$H(X|Y) = -\sum_{i=1}^{n}\sum_{j=1}^{m} P(x_i, y_j) * \log P(x_i|y_j)$$

$$= -[P(X=0,Y=0)logP(X=0|Y=0) + P(X=0,Y=1)logP(X=0|Y=1)$$

$$+ P(X=1,Y=0)logP(X=1|Y=0) + P(X=1,Y=1)logP(X=1|Y=1)]$$

$$= -p(1-\varepsilon)\log\frac{p(1-\varepsilon)}{p(1-\varepsilon)+(1-p)\varepsilon} - p\varepsilon\log\frac{p\varepsilon}{p\varepsilon+(1-p)(1-\varepsilon)} - (1-p)\varepsilon\log\frac{(1-p)\varepsilon}{p(1-\varepsilon)+(1-p)\varepsilon}$$

$$- (1-p)(1-\varepsilon)\log\frac{(1-p)(1-\varepsilon)}{p\varepsilon+(1-p)(1-\varepsilon)}$$

$H(X) = -plogp - (1-p)\log(1-p)$

以上より、

$$I(X;Y) = H(X) - H(X|Y) = \cdots$$

$$= -\{\mathrm{p}(1-\varepsilon) + (1-p)\varepsilon\}\log\{p(1-\varepsilon) + (1-p)\varepsilon\} - \{p\varepsilon + (1-p)(1-\varepsilon)\log\{p\varepsilon + (1-p)(1-\varepsilon)\}$$

$$+ \varepsilon log\varepsilon + (1-\varepsilon)log(1-\varepsilon) \quad 式(4-7)と同じ$$

case 1)　$\varepsilon = 0$の場合（誤りがない場合）

case 1) When ε = 0 (when there is no error)

$$\mathbb{P} = \begin{bmatrix} 1.0 & 0 \\ 0 & 1.0 \end{bmatrix}$$

式４－５より、

From Equation 4-5,

$$H(Y) = -\{\mathrm{p}(1-\varepsilon) + (1-p)\varepsilon\}\log\{p(1-\varepsilon) + (1-p)\varepsilon\} - \{p\varepsilon + (1-p)(1-\varepsilon)\log\{p\varepsilon + (1-p)(1-\varepsilon)\} = -plogp - (1-p)\log(1-p) = H(X)$$

式４－６より、

From Equation 4-6,

$$H(Y|X) = -\varepsilon log\varepsilon - (1-\varepsilon)log(1-\varepsilon) = 0$$

よって、$I(X;Y) = H(Y) - H(Y|X) = H(X)$

つまり、完全にXの情報量を送っていることになる。

This means that the complete amount of X information can be received.

case 2)　$\varepsilon = 0.5$の場合

case 2) When ε = 0.5 (when there is no error)

$$\mathbb{P} = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}$$

式４－５より、

From Equation 4-5,

$$H(Y) = -\{\mathrm{p}(1-\varepsilon) + (1-p)\varepsilon\}\log\{p(1-\varepsilon) + (1-p)\varepsilon\} - \{p\varepsilon + (1-p)(1-\varepsilon)\log\{p\varepsilon + (1-p)(1-\varepsilon)\}$$

$$= -\left\{\frac{1}{2}p + \frac{1}{2}(1-p)\right\}\log\left\{\frac{1}{2}p + \frac{1}{2}(1-p)\right\}$$

$$-\left\{\frac{1}{2}p + \frac{1}{2}(1-p)\right\}\log\left\{\frac{1}{2}p + \frac{1}{2}(1-p)\right\} = -\frac{1}{2}log\frac{1}{2} - \frac{1}{2}log\frac{1}{2} = 1$$

式４－６より、

From Equation 4-6,

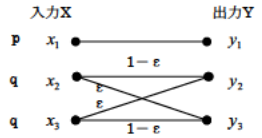
$$H(Y|X) = -\varepsilon log\varepsilon - (1-\varepsilon)log(1-\varepsilon) = -\frac{1}{2}log\frac{1}{2} - \frac{1}{2}log\frac{1}{2} = 1$$

であり、$I(X;Y) = H(Y) - H(Y|X) = 0$、つまり実質的には通信できない。

So, I (X; Y) = H (Y) -H (Y｜X) = 0. In this case, actual communication is impossible.

例４．３　以下の通信路の通信路容量を求めよ。

Example 4.3 Calculate the capacity of the channel.

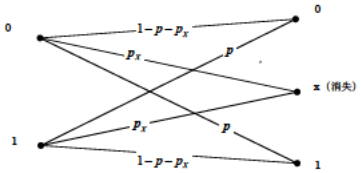


（演習問題４．７）

例４．４　以下の二元対称消失通信路の通信路容量を求めよ。

Example 4.4 Calculate the capacity of the binary erasure channel.

【図４．７】



入力アルファベットよりも出力アルファベットを多くする。軟判定復号（Soft decision decoding）とも呼ばれる。（演習問題４．３）

We have more output alphabets than input ones. It is also called soft decision decoding. (See Exercise 4.3)

４．４　通信路符号化

4.4 Channel coding

誤りを下げる方法

Strategies to reduce error rate are,

（１）εを小さくする。　S/N を上げる。

（２）冗長度を増す。

・繰り返す　　－＞単純に繰り返したのでは通信速度が下がる

・符号化に工夫をする　－＞通信速度を下げずに冗長度を増す方法がある（シャノンの第２定理）

(1) Make ε smaller. ex. Increase S / N.

(2) Increase redundancy.

・ Repeating -> Simple repeat decreases the transmission rate.

・ Coding tricks -> Shannon's second theorem (Channel coding theorem) indicates increasing redundancy without reducing the transmission rate.

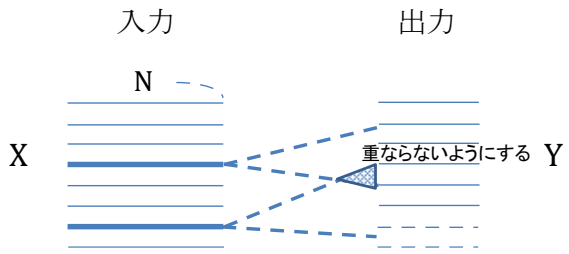
（繰返す方法）繰返し回数に対して、誤り率が低下するが、情報伝送速度Ｒも低下する。

Simple repeating reduces the error rate, however, does the transmission rate according to the number of repetitions.

実質的に誤りのない通信方法：送信符号語が誤ったとしても隣の領域に入らないようにする。すなわち、下図の領域が重ならないようにする。

Substantially error-free communication: Even if the transmitted code word is incorrect, it should not enter the adjacent area as shown in the figure. It can be achieved by un-overlapping.

【図４．８】



では、重ならないようにするには？

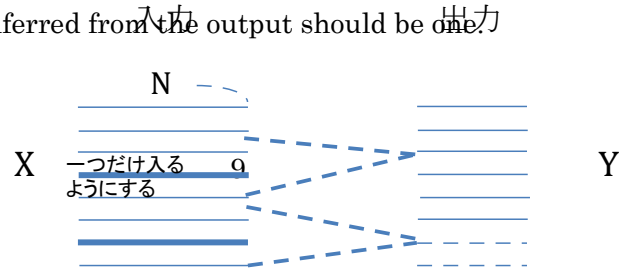
How to achieve it?

Xのすべての符号語を使うのではなく、まばらに使う。

Use sparsely, rather than all X code words.

逆に、出力から類推される入力が１つになるようにする。

Instead, the input inferred from the output should be one.



2020-08-20

［通信路符号化の定理］（シャノンの第２定理）
通信路容量がCの時、情報伝送速度Rが$R < C$であれば、任意の正の数εに対し、符号誤り率P_eが$P_e < \varepsilon$となる符号化が存在する。

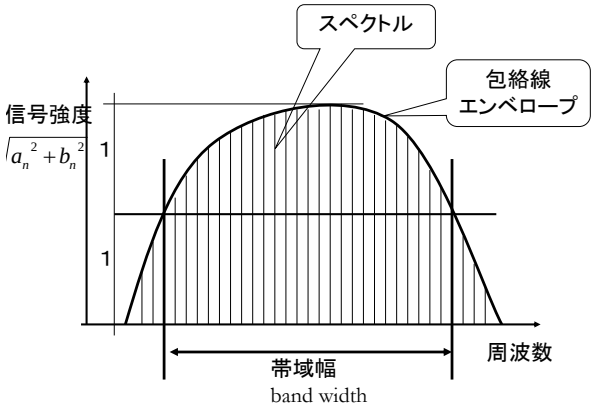
[Channel coding theorem] (Shannon's second theorem)
When the channel capacity is C and the information transmission rate R is R <C, there is coding where the error rate P_e is P_e <ε for any positive number ε.

（証明）　略　（補足資料１）
(Proof) See Supplementary material 1

４．５　その他の関連する話題
4.5 Other related topics

［シャノン-ハートレーの定理］
伝送路の帯域幅をH(Hz)、受信信号電力をS(W)、雑音電力をN(W)とすると、達成可能な最大伝送速度R(bit/sec)は、以下で表される。

[Shannon-Hartley theorem]
Assuming that the bandwidth of the transmission link is H (Hz), the received signal power is S (W), and the noise power is N (W), the maximum achievable transmission rate R (bit / sec) is expressed as follows.



$$R = H \log_2 \left(1 + \frac{S}{N}\right)$$

これをシャノン容量、あるいはシャノンの定理と表現することもある。

This is sometimes expressed as Shannon capacity or Shannon's theorem.

具体的な帯域幅

Specific bandwidth

無線 LAN 20MHz-80MHz

LTE　5MHz、10MHz、15MHz、20MHz

光ファイバ　数 THz

情報通信基礎Ⅰ　第５章　誤り訂正符号

Chapter 5 Error Correction Code

2021/12/7　情シス　渡辺

ポイント

・第４章で論じた通信路符号化の具体的な手法を学ぶ。ポイントは多くの符号語を用意しそのうちの一部（何らかの条件を満たす符号語）を使うことである。（冗長度を増す）

・符号語をベクトルとして扱う方法（第５章）、符号語を多項式で扱う方法（第６章）を学ぶ。

・パリティ符号、ハミング符号

Points

- Learn the specific method of channel coding discussed in Chapter 4. The point is to prepare many code words and use some of them (code words that satisfy some conditions). (Increase redundancy)
- Learn how to handle code words as vectors (Chapter 5) and how to handle code words with polynomials (Chapter 6).
- Parity code, Hamming code

５．１　基礎

5. 1 Introduction

５．１．１　誤り訂正の方法

１）連送方式：同じ符号語を繰り返して送る

２）返送照合方式：受信した符号語を送信側に返送して送信側で照合する。誤りがあれば再送する。

３）自動再送要求方式（ARQ; Automatic Repeat reQuest）受信側で誤りを「検出」し、誤りがあれば再送を要求する。

5.1.1 Error correction methods

1) Repetitive method: Send the same code word repeatedly

2) Loop checking method: The received code word is returned to the sender and check the identity by the sender. If it finds an error, resend the word.

3) Automatic repeat request method (ARQ): The receiving side detects an error, and if there is an error, requests the retransmission.

ACK 方式：送信側は送信後タイマを起動する。受信側は、正しく受信できた場合はACK(acknowledgement)を送る。誤った場合は何も送らない。送信側は、ACK が来たら次の符号語を送る。タイマが切れたら再度符号語を送信する。

NAK 方式：受信側は、正しく受信できた場合は何も送らない。誤っていた場合は

NAK(Negative ACK)を送る。送信側は NAK を受け取らない限り次の符号語を送る。NAK が来たら再送する。

ACK と NAK の併用等より深い考察が必要。インターネットは Selective ACK 方式。

ACK method: The transmitting side activates the timer after transmission of a code word. The receiving side sends an ACK (acknowledgement) if it can received the code correctly. Otherwise, it sends nothing. The sender sends the next code word after an ACK arrives. When the timer expires while waiting for the ACK, the sender sends the code word again.

NAK method: The receiving side sends nothing if it can received the code correctly. Otherwise, it sends NAK (Negative ACK). The sender sends the next code word unless a NAK is received. It sends the code again if NAK arrives.

More sophisticated discussion is required if we use ACK and NAK. The Internet uses the Selective ACK method.

4）FEC（Forward Error Correction）

誤り訂正符号によって、受信側が「訂正」する。

5）混成などその他

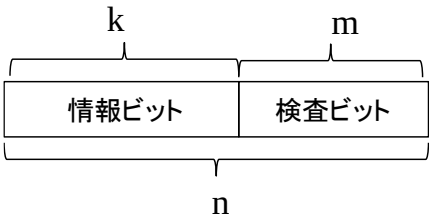
4) FEC (Forward Error Correction)

The receiving side corrects using error correction code.

5) Others

５．１．２　組織符号

右図のような構造をもった符号を組織符号と呼ぶ。



【図５．１】(n, k)符号

符号長n、情報ビットk、検査ビット$m = n - k$

また、組織符号の中で固定長のものをブロック符号と呼ぶ。

5.1.2 Systematic code

A code having a structure as shown in the right figure is called a systematic code.

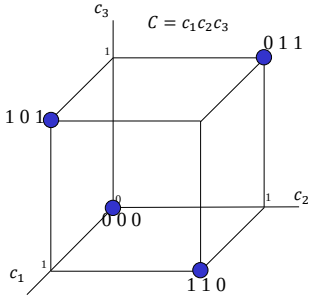


Fig. 5.1: (n, k) code

Code length n, information bit k, check bit m = n-k

Among the systematic codes, those having a fixed length are called block codes.

例５．１　(3,2)符号　最も簡単なパリティ符号

符号ℂ＝符号語Cの集合　$\mathbb{C} = \{000, 011, 101, 110\}$

【図５．２】

符号語$C = c_1c_2c_3$ とすると、$c_3 = c_1 + c_2$となっている。

+は排他的論理和(XOR)

$0 + 0 = 0$　$0 + 1 = 1$　$1 + 0 = 1$　$1 + 1 = 0$

これを法２に関する加算　とも呼ぶ。

（今後+は、特に断らない限り排他的論理和を表すこととする）

$c_3 = c_1 + c_2$は、$c_1 + c_2 + c_3 = 0$と等価である。

Example 5.1 (3,2) code: The simplest parity code

Code $\mathbb{C}$ = set of code words C $\mathbb{C}$ ={000,011,101,110}

FIG. 5.2.

If the code word $C = c_1c_2c_3$, then $c_3 = c_1 + c_2$.

+ means the exclusive OR (XOR)

0 + 0 = 0 0 + 1 = 1 1 + 0 = 1 1 + 1 = 0

This is also called modulo 2 addition.

(From now on, + represents the exclusive OR unless otherwise specified)

Note: $c_3 = c_1 + c_2$ is equivalent to $c_1 + c_2 + c_3 = 0$.

５．１．３　パリティ符号

例５．１の情報ビットを増やして拡張する。

ここでは、情報ビットa_5 a_4 a_3 a_2　に１個の検査記号a_1を付加した符号長 5 の符号を考える。各符号語の記号１の数が偶数になるように構成する。すなわち、

となるように、a_1と定める。これをパリティと呼ぶ。

すべての符号語は、【表５．１】となる。

（発展　検査記号を複数に拡張するとどうなるか。－＞ハミング符号）

5.1.3 Parity code

Let's expand Example 5.1 by increasing the information bits.

Here, consider a code having a code length of 5 in which one check bit a_1 is added to the information bits a_5 a_4 a_3 a_2.

Each code word is configured as the number of symbols 1 of the code word be even.

a_5 + a_4 + a_3 + a_2 + a_1 = 0

This is called parity code. All code words are shown in [Table 5.1].

(Advance: What happens if we have more check bits?-> Hamming code)

$(b_5\ b_4\ b_3\ b_2\ b_1) = (1\ 0\ 0\ 1\ 1)$を受信したとする。

$b_5 + b_4 + b_3 + b_2 + b_1 \neq 0$　誤り検出可能。

１ビット誤り検出可能。２ビット誤りは検出不可能。

Suppose that (b_5 b_4 b_3 b_2 b_1) = (1 0 0 1 1) is received.

b_5 + b_4 + b_3 + b_2 + b_1 ≠ 0 Thus, error is detected.

1-bit error detection is possible, while 2-bit error is not.

表5･1　単一誤り検出(5,4)符号の例

	a_5	a_4	a_3	a_2	a_1		a_5	a_4	a_3	a_2	a_1
1つ1つが符号語	0	0	0	0	0		1	0	0	0	1
	0	0	0	1	1		1	0	0	1	0
	0	0	1	0	1		1	0	1	0	0
	0	0	1	1	0		1	0	1	1	1
	0	1	0	0	1		1	1	0	0	0
	0	1	0	1	0		1	1	0	1	1
	0	1	1	0	0		1	1	1	0	1
	0	1	1	1	1		1	1	1	1	0

全体が符号　16符号語からなる符号

５．１．４　ハミング距離と最小ハミング距離

今、２つの符号語、$A = (a_1 a_2 a_3 \ldots a_n)$　と$B = (b_1 b_2 b_3 \ldots b_n)$　を考える。

各ビット毎の差を加算したものをハミング距離と呼ぶ。

$d = \sum_{i=1}^{n} (a_i - b_i)$　　ただし、ここでの加算は排他的論理和ではなく算術加算である。

例）$A = (0\,0\,0\,1\,1)$、$B = (0\,1\,1\,0\,1)$　$d = 0 + 1 + 1 + 1 + 0 = 3$　　ここで、+は通常の算術加算を示す。

5.1.4 Hamming distance and minimum Hamming distance

Now consider two code words, A = (a_1 a_2 a_3 ... a_n) and B = (b_1 b_2 b_3 ... b_n).

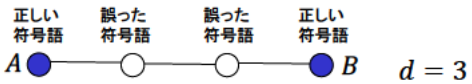
The sum of the differences for each bit is called the Hamming distance.

D = Σ_ (i = 1) ^ n (a_i-b_i)

Note that the addition here is not an XOR but an arithmetic addition.

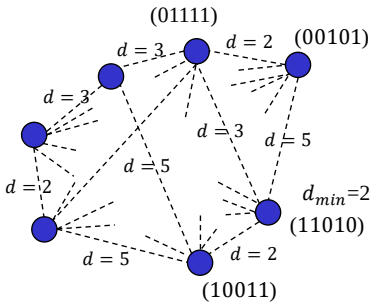
Example) A = (0 0 0 1 1), B = (0 1 1 0 1) d = 0 + 1 + 1 + 1 + 0 = 3 Here, + indicates arithmetic addition.

【図５．３】



符号に含まれるすべての符号語間のハミング距離のうち、最小のものを最小ハミング距離（最小距離）と呼ぶ。

The minimum Hamming distance between all the code words included in the code is called the minimum Hamming distance (minimum distance).



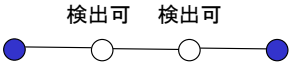
【図５．４】

一般に、

・最小距離$d_{min} \geq d+1$ のとき、d以下の誤り検出が可能。

【図５．５】ex) $d_{min}=3$　$d=2$

２以下の誤り検出可能



Generally,

- d_min ≥ d+1, error of distance d or less is detectable.

[Fig. 5.5] ex) d_min = 3 d = 2　　2 or less error detection is possible

・最小距離$d_{min} \geq 2t+1$ のとき、t以下の誤り訂正が可能。

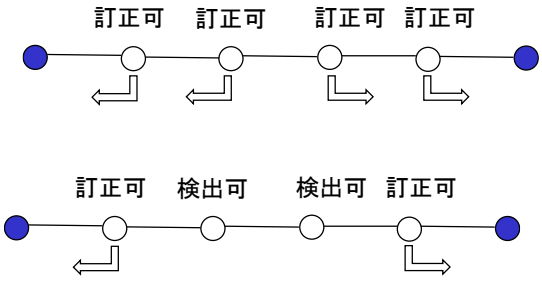
【図５．６】ex) $d_{min}=5$ $t=2$

２以下の誤り訂正可能

・最小距離$d_{min} \geq t+d+1$ $(d>t)$ のとき、t以下の誤り訂正とd以下の誤り検出が可能。

【図５．７】ex) $d_{min}=5$　$d=3$ $t=1$

１誤り訂正可能、２・３誤り検出可能



- When d_min ≥ 2t+1, error of distance t or less is correctable.

[Fig. 5.6] ex) d_min = 5 t = 2　　2 or less error correction is possible

- When d_min ≥ t + d + 1 (d> t), error of distance t or less is correctable and error of distance d or less is detectable.

[Fig. 5.7] ex) d_min = 5 d = 3 t = 1　1 error correction and, 2.3 error detection are possible

５．１．５　よい(n,k)符号

・符号能率　$\frac{k}{n}$　が大きい

・誤り検出、訂正の能力が高い。　最小距離が大きい。

・誤り検出、訂正の処理が容易。

5.1.5 Good (n, k) code

- Code efficiency, k / n is large.
- High ability to detect and correct errors. The minimum distance is large.
- Easy error detection and correction processing.

５．２　ハミング符号

５．２．１　概要

・1950 年にベル研 Richard Hamming によって発案された。

・パリティ検査を拡張。検査記号を複数にし、単一誤り訂正を可能とする。

・符号長　$n = 2^m - 1$

・検査ビット数　m

・情報ビット　$k = n - m$

・最小距離　3　（単一誤り訂正可能）

・数学的構造が明確。処理が容易。ハードウエア化。

・(3,1)符号、 (7, 4)符号、 (15, 11)符号、 (31, 26)符号　等

・計算機メモリ用 ECC

$m = 7$ (127, 120)->1 ビット拡大化-> (128, 120)–>短縮化->(72, 64)8 バイト単位で扱う。

5.2 Hamming code

5.2.1 Overview

- Invented by Bell Labs Richard Hamming in 1950.
- Expanded parity check. Multiple check bits enable single error correction.
- Code length n = 2 ^ m-1
- Number of check bits is m
- Information bit k = n-m
- Minimum distance is 3 (single error correction is possible)
- The mathematical structure is clear. Easy to process by hardware.
- (3,1), (7,4), (15, 11), (31, 26) , etc.

・ ECC for computer memory

m = 7 (127, 120)-> 1-bit extension -> (128, 120)-> shortening-> (72, 64) (can be handled in 8-byte.)

５．２．２　ハミング符号の具体例

【表５．２】　ハミング(7, 4)符号

情報ビット$a_7\ a_6\ a_5\ a_3$

検査ビット$a_4\ a_2\ a_1$

5.2.2 Example of Hamming code

[Table 5.2] Humming (7, 4) code

Information bit a_7 a_6 a_5 a_3

Check bit a_4 a_2 a_1

表5･2 単一誤り訂正ハミング(7,4)符号の例

a_7	a_6	a_5	a_4	a_3	a_2	a_1		a_7	a_6	a_5	a_4	a_3	a_2	a_1
0	0	0	0	0	0	0		1	0	0	1	0	1	1
0	0	0	0	1	1	1		1	0	0	1	1	0	0
0	0	1	1	0	0	1		1	0	1	0	0	1	0
0	0	1	1	1	1	0		1	0	1	0	1	0	1
0	1	0	1	0	1	0		1	1	0	0	0	0	1
0	1	0	1	1	0	1		1	1	0	0	1	1	0
0	1	1	0	0	1	1		1	1	1	1	0	0	0
0	1	1	0	1	0	0		1	1	1	1	1	1	1

符号化規則

This example has the following coding rules

$$\begin{aligned} a_7 + a_6 + a_5 + a_4 &= 0 \\ a_7 + a_6 + a_3 + a_2 &= 0 \\ a_7 + a_5 + a_3 + a_1 &= 0 \end{aligned}$$

上の式を、$u = (a_7\ a_6\ a_5\ a_4\ a_3\ a_2\ a_1)$として、以下のように記述できる。

The above equation can be described as follows with u = (a_7 a_6 a_5 a_4 a_3 a_2 a_1).

$$(a_7\ a_6\ a_5\ a_4\ a_3\ a_2\ a_1)\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = u \cdot H^T = 0$$

ただし、

, where

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

符号語=(0 0 0 0 1 1 1) ベクトルと考える。

Codeword = (0 0 0 0 1 1 1) We think of it as a vector.

符号　符号語の集合　　ベクトル空間

Code = set of codewords = vector space

$u = (a_7\ a_6\ a_5\ a_4\ a_3\ a_2\ a_1) = (0\,0\,1\,1\,0\,0\,1)$ を送信し、$v = (b_7\ b_6\ b_5\ b_4\ b_3\ b_2\ b_1) = (0\,1\,1\,1\,0\,0\,1)$を受信したとする。符号化規則に照らして以下を計算する。

When u = (a_7 a_6 a_5 a_4 a_3 a_2 a_1) = (0 0 1 1 0 0 1) was sent and v = (b_7 b_6 b_5 b_4 b_3 b_2 b_1) = (0 1 1 1 0 0 1) is received. Then, we calculate the following according to coding rules:

$$\begin{aligned} s_4 &= \square = 1 \\ s_2 &= \square = 1 \\ s_1 &= \square = 0 \end{aligned}$$

The above equation can also be written as follows.

上の式を、以下のように記述することもできる。

$$s = (s_4\ s_2\ s_1) = v \cdot H^T = (b_7\ b_6\ b_5\ b_4\ b_3\ b_2\ b_1) \cdot H^T = (1\,1\,0)$$

符号化規則によれば、vが正しい符号語ならば、$s = (s_4\ s_2\ s_1) = (0\,0\,0)$となるはずである。しかし、そうはならない。従って、誤りがあることがわかる。s をシンドロームと呼ぶ。また、Hは、誤りがあるかどうかを検査する行列であるため、検査行列と呼ばれる。

（この例では、$s = (s_4\ s_2\ s_1) = (1\,1\,0) = 6$であり、$b_6$が誤りであることを決定できるがここでは置いておく）

According to the coding rules, if v is the correct codeword, then s = (s_4 s_2 s_1) = (0 0 0) should be. But this is not the case.
Therefore, we find that there is an error in v. s is called Syndrome.

Further, H is called Check Matrix (or Parity Check Matrix) because it is a matrix for detecting the errors.
(In this example, s = (s_4 s_2 s_1) = (1 1 0) = 6, and we can determine that b_6 is incorrect. We will see the reason later.)

５．３　線形符号
5.3 Linear code

５．３．１　符号理論のための群、環、体
　集合Ｇ、Ｒ、Ｆを考える。
5.3.1 Group, Ring, Field for coding theory
Let us consider the set G, R, and F.

１）群 group
・ある集合Ｇの元に対して、一つの演算○が定義されている
・以下の条件を満たす時、Ｇを群と言う。
Ｇ１（閉塞性）：Ｇの任意の元 a, b に対して、a○b はＧの元である。
Ｇ２（結合則）：Ｇの任意の元 a, b, c に対して、(a○b)○c=a○(b○c)を満たす。
Ｇ３（恒等元）：Ｇの任意の元 a に対して、a○I=I○a=a なるＩ（恒等元）がＧにある。
Ｇ４（逆元）：　Ｇの全ての元 a に対して、a○a’=a’○a=I となる a’（逆元）がＧにある。

1) Group
- One operation ○ is defined for all elements of a set G.
- When the following conditions are held, G is called a Group.
G1 (closure): for any element a, b of G, a ○ b is an element of G.
G2 (associative law): For any element a, b, c of G, (a ○ b) ○ c = a ○ (b ○ c) is satisfied.
G3 (identity element): For any element a of G, I (identity element) such that a ○ I = I ○ a = a is in G.
G4 (inverse element): For all elements a of G, there is a'(inverse element) in which a ○ a'= a'○ a = I.

１－１）加法群
・群Ｇの演算が＋であったとき、加法群と呼ぶ。

・恒等元 I=0, 逆元 a’=－a と表現する。

1-1) Additive group
- When the operation of the group G is +, it is called an additive group.
- Identity element I is 0 and inverse element a’ is -a.

１－２）アーベル群
・群ＧがＧ５を満たす時、アーベル群、あるいは可換群と呼ぶ。
Ｇ５（交換則）：Ｇの任意の元 a, b に対して、a○b=b○a を満たす。
1-2) Abelian group
-When the group G satisfies G5, it is called an abelian group or a commutative group.
G5 (commutative law): For any element a, b of G, a ○ b = b ○ a is satisfied.

１－３）例
・実数すべての集合、すべての整数の集合は加法群でありアーベル群である。
・G={0 1}は、以下の演算＋（これを法２の元の加法と呼ぶ）の元でアーベル群である。
0+0=0　0+1=1　1+0=1　1+1=0

1-3) Example
- The set of all real numbers and the set of all integers are additive groups and abelian groups.
- G = {0 1} is an abelian group under the following operation + (this is called addition under the modulus of 2).
0 + 0 = 0 0 + 1 = 1 1 + 0 = 1 1 + 1 = 0

２）環 ring
・ある集合Ｒの元に対して、二つの演算（加法＋　乗法・）が定義されている。
・以下の条件を満たす時、Ｒを環と言う。
Ｒ１（加法群）：Ｒは加法の元でアーベル群である。
Ｒ２（閉塞性）：Ｒの任意の元 a, b に対し、a・b はＲの元である。
Ｒ３（結合則）：Ｒの任意の元 a, b, c に対し、(a・b)・c=a・(b・c)が成り立つ。

Ｒ４（分配則）：Ｒの任意の元 a, b, c に対し、a・(b＋c)=a・b＋a・c および
(a＋b)・c=a・c＋b・c が成り立つ。

2) Ring
- Two operations (addition + multiplication・) are defined for all elements of a set R.
- When the following conditions are held, R is called a Ring.
R1 (additive group): R is an abelian group under additive.
R2 (closure): for any element a, b of R, a ・ b is an element of R.
R3 (associative law): (a・b)・c = a・(b・c) holds for any element a, b, c of R.
R4 (distributive law): for any element a, b, c of R, a・(b + c) = a・b + a・c and (a + b)・c = a・c + b・c hold.

２－１）可換環
・環ＲがＲ５を満たす時、可換環と呼ぶ。
Ｒ５（乗法に関する交換則）：Ｒの任意の元 a,b に対して、a・b=b・a を満たす。
2-1) Commutative Ring
When the ring R satisfies R5, it is called a commutative ring.
R5 (commutative law over multiplication): For any element a, b of R, a ・b = b・a is satisfied.

２－２）剰余環
・正の整数 r に対して、法 r に関する整数の剰余集合（{0 1 2 ,,, r-1}）は、法 r に関する加法と乗法のもとに環をなす。これを剰余環と呼ぶ。

2-2) Quotient Ring (Residue Ring)
- For a positive integer r, the remainder set of integers under modulus of r ({0 1 2 ,,, r-1}) forms a ring based on addition and multiplication under modulus of r. This is called a quotient ring.

２－３）例
・R={0 1}は、以下の演算＋
0+0=0　0+1=1　1+0=1　1+1=0

および以下の乗法の元で可換環である。

0・0=0　0・1=0　1・0=0　1・1=1

Ｒは法２の剰余環と考えることもできる。

2-3) Example

- R = {0 1} is a commutative ring under the following addition +

0 + 0 = 0 0 + 1 = 1 1 + 0 = 1 1 + 1 = 0, and under the following multiplication ・

0 ・ 0 = 0 0 ・ 1 = 0 1 ・ 0 = 0 1 ・ 1 = 1 .

R can also be thought of as the quotient ring under modulus of 2.

補足２－４）イデアル

・環Ｒの部分集合Ｊが加法のもとにＲの部分群であり、Ｊの任意の元ａとＲの任意の元ｒの積ａ・ｒがＪの元であるとき、ＪをＲのイデアルと呼ぶ。

Supplement 2-4) Ideal

- When a subset J of the ring R is a subgroup of R under addition and the multiplication of any element a of J and any element r of R, that is,　a・r, is an element of J, then J is called an Ideal or R.

３）体 field

・ある集合Ｆが以下の条件を満たす時、Ｆを体と言う。

Ｆ１：Ｆは可換環である。

Ｆ２：Ｆの０以外の元の集合が乗法群をなす。

・有限集合の体を有限体またはガロア体(Galois Field; GF)と呼ぶ。

3) Field

- When a set F satisfies the following conditions, F is called a Field.

F1: F is a commutative ring.

F2: Elements except zero of F forms a multiplicative group.

- The field of a finite set is called Finite Field or Galois Field (GF).

３－１）例

・有理数全ての集合、実数すべての集合、複素数すべての集合はそれぞれ体である。

・F={0 1}は、以下の加法+

0+0=0　0+1=1　1+0=1　1+1=0

+	0	1
0	0	1
1	1	0

および以下の乗法・

0・0=0　0・1=0　1・0=0　1・1=1

*	0	1
0	0	0
1	0	1

の元で、体となる。これをガロア体 GF(2)と呼ぶ。

・p を素数とするとき、法 p に関する整数の剰余の集合$\{0,1,2,,,,p-1\}$は、法 p に関する加法と乗法の元に体 GF(p)をなす。

3-1) Example

- The set of all rational numbers, the set of all real numbers, and the set of all complex numbers are fields.
- F = {0 1} is a Field under the following addition +,

0 + 0 = 0 0 + 1 = 1 1 + 0 = 1 1 + 1 = 0,

and the following multiplication,

0 ・ 0 = 0 0 ・ 1 = 0 1 ・ 0 = 0 1 ・ 1 = 1

The field is also called Galois field GF(2).

- When p is a prime number, the set of integer remainders {0,1,2 ,,,, p-1} forms Field GF (p) for addition and multiplication under the modulus of p.

５．３．２　ベクトル空間と符号

GF(2)上の元a_i　$a_i \in GF(2)$　＜ー要は、{0, 1} と　＋ ・の演算

GF(2)上の元のn記号列 $\mathbb{a} = (a_n\ a_{n-1} \cdots a_1)\ a_i \in GF(2)$

この記号列をベクトルと呼ぶ。

記号列は2^n個ある。これらを要素とする集合をベクトル空間と呼びここではUと表現する。

5.3.2 Vector space and code

We consider an element, a_i on GF (2) a_i ∈ GF (2) (In short, {0, 1} and + ・ operations).

Then n symbol string on GF (2) a = (a_n a_ (n-1) … a_1) a_i ∈ GF (2) is called a vector.

There are 2 ^ n symbol strings. The set of all elements is called a vector space expressed as U.

【図５．８】
Figure. 5.8.

$\mathbb{a} \in U \quad \mathbb{b} \in U \;\; c \in GF(2)$を考える。

$$\mathbb{a} = (a_n\, a_{n-1} \cdots a_1) \;\; a_i \in GF(2)$$
$$\mathbb{b} = (b_n\, b_{n-1} \cdots b_1) \;\; b_i \in GF(2)$$

ここで、積と和を以下のように定義する。

Let us consider a ∈ U b ∈ U c ∈ GF (2).
a = (a_n a_ (n-1) ⋯ a_1) a_i ∈ GF (2)
b = (b_n b_ (n-1) ⋯ b_1) b_i ∈ GF (2)
We define multiplication and addition as follows.

積　$c\mathbb{a} = c(a_n\, a_{n-1} \cdots a_1) = (ca_n\, ca_{n-1} \cdots ca_1)$
multiplication　$c\mathbb{a} = c(a_n\, a_{n-1} \cdots a_1) = (ca_n\, ca_{n-1} \cdots ca_1)$

和　$\mathbb{a} + \mathbb{b} = (a_n\, a_{n-1} \cdots a_1) + (b_n\, b_{n-1} \cdots b_1) = (a_n + b_n\, a_{n-1} + b_{n-1} \cdots a_1 + b_1)$
addtion　$\mathbb{a} + \mathbb{b} = (a_n\, a_{n-1} \cdots a_1) + (b_n\, b_{n-1} \cdots b_1) = (a_n + b_n\, a_{n-1} + b_{n-1} \cdots a_1 + b_1)$

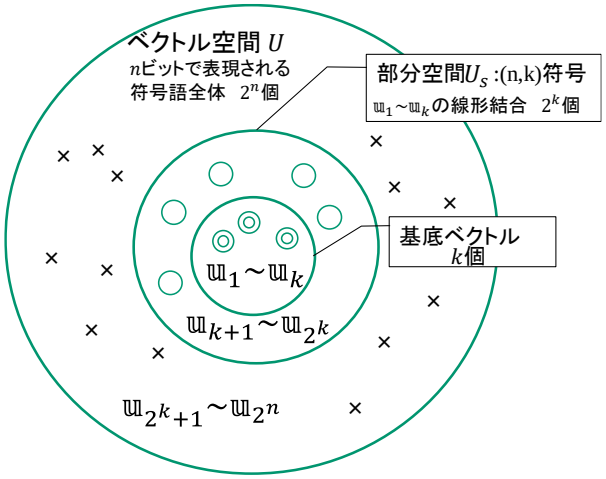
ベクトル$\mathbb{u}$がベクトル$\mathbb{u}_1, \mathbb{u}_2, \cdots \mathbb{u}_k$　の線形結合、すなわち、$\mathbb{u} = c_1\mathbb{u}_1 + c_2\mathbb{u}_2 + \cdots + c_k\mathbb{u}_k$　であるとき、

c_1~c_kがすべて０　⇔　$\mathbb{u} = 0$

ならば、$\mathbb{u}_1$~$\mathbb{u}_k$は独立であるという。また、$\mathbb{u}_1$~$\mathbb{u}_k$を基底ベクトルという。

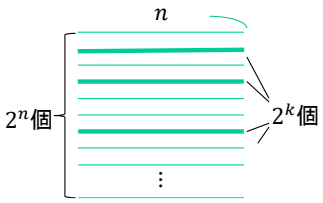
When a vector u is a linear combination of vectors u_1, u_2, ⋯u_k, that is, u = c_1 u_1 + c_2 u_2 + ⋯ + c_k u_k, if the following holds, u_1 ~ u_k are said to be independent. Also, u_1 ~ u_k are called basis vectors.
c_1 ~ c_k are all 0 ⇔ u = 0

(n,k)符号は、nビットの符号語のうち、k個の基底ベクトルの線形結合で表された符号語を用いる。すなわち、2^n個のnビットの符号語のうち、2^k個だけを用いる。



(n, k) code uses the codeword represented by a linear combination of k basis vectors among the n-bit codewords. That is, only 2 ^ k out of the 2 ^ n n-bit codewords are used.

【図５．９】



$\mathbb{u}_1$～$\mathbb{u}_k$の線形結合によって部分空間U_sが生成されることを、基底ベクトル$\mathbb{u}_1$～$\mathbb{u}_k$が部分空間U_sを張る、と表現する。

The fact that the subspace U_s is generated by the linear combination of u_1 ~ u_k is also said the basis vector u_1 ~ u_k span the subspace U_s.

注意：今後は、ベクトル$\mathbb{u}_1$などをu_1と表現する。

５．３．３　生成行列と検査行列

5.3.3 Generator matrix and Parity Check matrix

送信側は、送るべき情報ビットxに生成行列Gをかけて、送信符号語uを求める。

Sender side calculates multiplication of information bits x and generator matrix G to obtain the code word u to transmit is obtained.

$$u = x \cdot G$$

一方、受信側は受信した符号語vに検査行列Hを使ってシンドロームsを求め、sが０であるかを調べる。

On the other hand, the receiving side obtains the syndrome s from the received codeword v by using the parity check matrix H, and checks whether s is 0.

$$s = v \cdot H^T$$

$s = 0$ならば誤りがない。($u \cdot H^T = 0$である)

$s \neq 0$ならば誤りがある。(さらにこれを用いて誤りを訂正できる可能性がある)

If s = 0, there is no error. (U·H ^ T = 0)
If s ≠ 0, there is an error. (Furthermore, this may be used to correct errors)

５．３．４　ハミング(7,4)符号の例

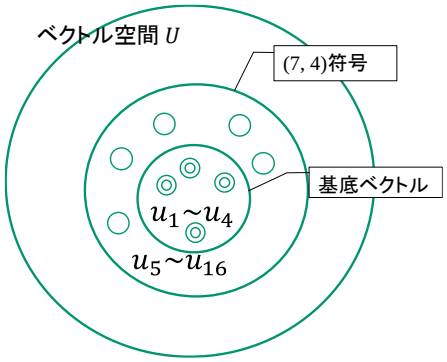
5.3.4 Example of Hamming (7,4) code

１）　表５．２から独立な４行を選びベクトルとする。(選び方は一通りではない)

1) Select 4 independent rows from Table 5.2 and set them as a vector. (There are several ways to choose independent rows.)

$u_1 = (1\,0\,0\,1\,0\,1\,1)$
$u_2 = (0\,1\,0\,1\,0\,1\,0)$
$u_3 = (0\,0\,1\,1\,0\,0\,1)$
$u_4 = (0\,0\,0\,0\,1\,1\,1)$

他の符号語は、u_1~u_4の線形結合で表現が可能。また、u_1~u_4は独立であり、u_1~u_4は基底ベクトルである。u_1~u_4を線形結合した符号語の集合U_sは、(7,4)符号を構成する。【図５．１０】



Other codewords can be represented by linear combinations of u_1 to u_4. u_1 ~ u_4 are independent, and u_1 ~ u_4 are basis vectors.
The set of codewords U_s, which is a linear combination of u_1 to u_4, is the (7,4) code. (Figure 5.10.)

２）送信側
基底ベクトルを並べた行列が生成行列となる。上記の例のハミング(7,4)符号の生成行列Gは、以下となる。

2) Sender side
Sender side uses a generator matrix that includes the basis vectors of the code. For example, the generator matrix G of Hamming (7,4) code in the above mentioned example is as follows.

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

情報ビット$x = (x_4\ x_3\ x_2\ x_1)$を送るとき、送るべき符号語uを、$u = x \cdot G$で生成する。

When sending an information bits x = (x_4 x_3 x_2 x_1), the codeword u to be sent is generated by u = x·G.

例）$x = (1\,0\,1\,1)$
Example) x = (1 0 1 1)

$$u = x \cdot G = (1\,0\,1\,1) \cdot \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix} = (1 \quad\quad\quad\quad + u_3 + u_4$$

３）受信側

表５．２の符号化規則から、検査行列Hは以下である。

3) Receiving side

From the coding rules in Table 5.2, the check matrix H is as follows.

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

vを受信すると、$s = v \cdot H^T$によりシンドロームを求める。

When codeword v is received, the syndrome is calculated by s = v·H ^ T.

例　$v = (1\,1\,1\,0\,1\,0\,1)$の時

Example) x = (1 1 1 0 1 0 1)

$$s = v \cdot H^T = (1\,1\,1\,0\,1\,0\,1) \cdot \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \quad (\qquad\qquad) 0$$

従って、誤りがあることが分かる。

Therefore, we can tell that there is an error.

４）生成行列Gと検査行列Hの関係

もし、誤りが起こらず、送信語uと受信語vが一致していれば

$$s = v \cdot H^T = u \cdot H^T = (x \cdot G)H^T = x \cdot (G \cdot H^T) = 0 \; for \; any \; x$$

となる。従って$G \cdot H^T = 0$となるようにG、Hを構成する。

（一般的には、HからGを求める場合は、Hを行基本変形し Reduced Row Echelon Form を求め、零空間の基底ベクトルからGを定める。$H = (A|I)$ [Iは単位行列]の形式の場合はより簡単に求められる。補足資料２（補足資料１を読んだ後の方が理解しやすい））

4) Relationship between generator matrix G and check matrix H

If no error occurs and the sent word u and the received word v are identical, then,

s = v·H ^ T = u·H ^ T = (x·G) H ^ T = x· (G·H ^ T) = 0 for any x.

Therefore, G and H are configured so that G·H ^ T = 0.

(In general, for example, when finding G from H, H is transformed to find the Reduced Row Echelon Form by elementary transform.
G is obtained from the basis vectors of the null space of H.
If H has the form of H = (A | I) [I is the identity matrix], we can easily find G. See Supplementary Material 2.

５．４　ハミング符号詳細－ハミング(7,4)符号を例に－

別紙　補足資料１

5.4 Hamming code details -Specially for Hamming (7,4) code-

See Supplementary Material 1.

５．５　線形符号のさらなる議論

5.5 Further discussion of linear code

５．５．１　最小距離と最小重み

5.5.1 Minimum distance and minimum weight

１）ハミング重みω

符号語を$u = (a_n\, a_{n-1} \cdots a_1)$　とするとき、0 ではないa_iの数をハミング重みという。

1) Hamming weight w

When the codeword is u = (a_n a_ (n-1) … a_1), the number of a_i that is not 0 is called the Hamming weight.

２）最小重みω_{min}

符号Uの全符号語（すべて 0 の語以外）で、最小のハミング重みをUの最小重みという。

例）表５．２のハミング(7,4)符号の最小重みは３

2) Minimum weight w_min

Minimum Hamming weight of all codewords in code U (other than all 0 words) is called the minimum weight of U.

Example) The minimum weight of Hamming (7,4) code in Table 5.2 is 3.

３）ハミング距離dとハミング重みω

u_1とu_2のハミング距離は、u_1-u_2のハミング重みに等しい

例　$u_1=(100)$、$u_2=(001)$　距離は２。$u_1-u_2=(101)$、重みは２。

3) Hamming distance d and Hamming weight w

Hamming distance between u_1 and u_2 is equal to Hamming weight of u_1-u_2.

Example u_1 = (100), u_2 = (001) Distance is 2. u_1-u_2 = (101) whose weight is 2.

４）最小距離d_{min}と最小重みω_{min}

ベクトル空間Uの符号語間の最小距離をd_{min}、Uの最小重みをω_{min}とすると、

$d_{min}=\omega_{min}$

4) Minimum distance d_min and minimum weight w_min

If the minimum distance between codewords in the vector space U is d_min and the minimum weight of U is w_min, then,

D_min = ω_min.

[証明]

[Proof]

０）準備

Uの基底ベクトルをu_1~u_kとする。Uの符号語v_iは、以下のように表現できる。

$$v_i=c_{i_1}u_1+c_{i_2}u_2+\cdots+c_{i_k}u_k=\sum_{m=1}^{k}c_{i_m}u_m\in U$$

また、Uの２つの符号語v_iとv_jの差はまたUの符号語である。すなわち、　$v_i-v_j\in U$

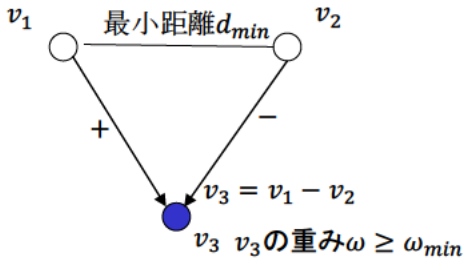
0) Preparation

Let u_1 ~ u_k be the basis vector of U. any codeword v_i of U can be expressed as follows.

v_i = c_ (i_1) u_1 + c_ (i_2) u_2 + … + c_ (i_k) u_k = Σ_ (m = 1) ^ k ▒〖c_ (i_m) u_m〗 ∈ U

Also, the difference between two codewords v_i and v_j of U is also a codeword in U. That is, v_i-v_j ∈ U.

1）今、v_1とv_2を最小距離d_{min}離れている符号語とする。



1) Now, let v_1 and v_2 be codewords that are separated by the minimum distance d_min.

$$d_{min} = |v_1 - v_2| = \left|\sum c_{1_m} u_m - \sum c_{2_m} u_m\right| = \left|\sum (c_{1_m} - c_{2_m}) u_m\right| = \sum (c_{1_m} - c_{2_m})$$

$v_3 = v_1 - v_2$　とすると、$v_3 \in U$であるから、$v_3 = \sum_{m=1}^{k} c_{3_m} u_m$と表現できる。

If v_3 = v_1-v_2, then since v_3 ∈ U, so it can be expressed as v_3 = ∑_ (m = 1) ^ k▒〖c_ (3_m) u_m〗.

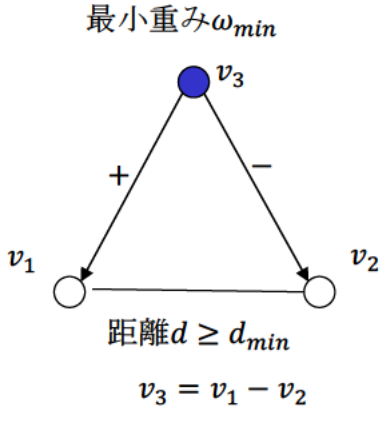
$$v_3 = \sum_{m=1}^{k} c_{3_m} u_m = \sum_{m=1}^{k} c_{1_m} u_m - \sum_{m=1}^{k} c_{2_m} u_m = \sum_{m=1}^{k} (c_{1_m} - c_{2_m}) u_m$$

v_3の重みは、以下となる。
The weight of v_3 is as follows.

$$\omega = \sum_{m=1}^{k} c_{3_m} = \sum_{m=1}^{k} (c_{1_m} - c_{2_m}) = d_{min}$$

ωはω_{min}以上であるので、$d_{min} = \omega \geq \omega_{min}$
すなわち、$d_{min} \geq \omega_{min}$

Since w is greater than or equal to w_min,
d_min = w ≥ w_min
That is, d_min ≥ w_min



2）v_3は最小重みω_{min}を持つ符号語とする。ただし、$v_3 \neq 0$

2) v_3 is a codeword with the minimum weight w_min. v_3 ≠ 0.

$$v_3 = \sum_{m=1}^{k} c_{3_m} u_m$$

従って、$\omega_{min} = \sum_{m=1}^{k} c_{3_m}$　である。
一方、$v_3 = v_1 - v_2$　となるv_1, v_2が存在する。
すなわち、

$$v_3 = \sum_{m=1}^{k} c_{3_m} u_m = \sum_{m=1}^{k} c_{1_m} u_m - \sum_{m=1}^{k} c_{2_m} u_m = \sum_{m=1}^{k} (c_{1_m} - c_{2_m}) u_m$$

よって、

$$c_{3_m} = c_{1_m} - c_{2_m}$$

v_1とv_2の距離 d　は、以下となる。

Therefore, $\omega_{min} = \sum_{m=1}^{k} c_{3_m}$.
On the other hand, there are v_1 and v_2 such that v_3 = v_1-v_2.
That is,

$$v_3 = \sum_{m=1}^{k} c_{3_m} u_m = \sum_{m=1}^{k} c_{1_m} u_m - \sum_{m=1}^{k} c_{2_m} u_m = \sum_{m=1}^{k} (c_{1_m} - c_{2_m}) u_m$$

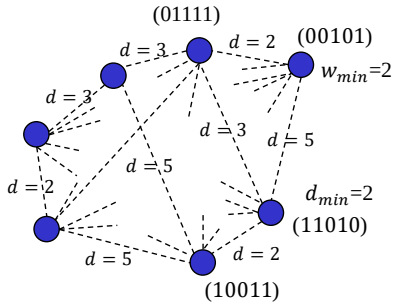
Therefore,

$$c_{3_m} = c_{1_m} - c_{2_m}$$

The distance d between v_1 and v_2 is as follows.

$$d = |v_1 - v_2| = \left| \sum c_{1_m} u_m - \sum c_{2_m} u_m \right| = \left| \sum (c_{1_m} - c_{2_m}) u_m \right| = \sum c_{3_m} = \omega_{min}$$

dは、d_{min}以上であるので、$d_{min} \leq d = \omega_{min}$
すなわち、$d_{min} \leq \omega_{min}$



Since d is greater than or equal to d_min, d_min ≤ d = w_min
That is, d_min ≤ w_min

３）１）２）より、$d_{min} = \omega_{min}$
3) From 1) 2), d_min = w_min

５．５．２　最小距離の求め方
5.5.2 How to find the minimum distance

１）生成行列Gから求める。

$d_{min} = \omega_{min}$　より、ω_{min}　から求められる。

ただし、Gの中にω_{min}の符号語があるように線形結合で求める。

1) Obtain from the generator matrix G.
We can obtain from w_min from d_min = w_min.
Note that G must contain a codeword of w_min. If it does not,
transform G by a linear combination.

２）検査行列Hから求める。

Hの列ベクトルで独立な物の数から求める。

Hの列ベクトル（n個あるとする）を考える。

(1)　全列ベクトルの中から　d_m個の列ベクトルを取り出す。この全組み合わせ（$\binom{n}{d_m}$通り）の中に線形従属なものがあれば、　$d_{min} \leq d_m$

(2)　全列ベクトルの中から　$d_m - 1$個の列ベクトルを取り出す。この全組み合わせ（$\binom{n}{d_m - 1}$通り）がすべて独立ならば、　$d_{min} \geq d_m$

(3) (1)(2)よりこのようなd_mを見つけられれば、　$d_{min} = d_m$

2) Obtain from Check matrix H.
We can obtain from the number of independent column vectors in H.
Let us consider we have n independent vectors in H.
(1) Take d_m column vectors from all column vectors.
If there is a linear dependency among all the combinations $\binom{n}{d_m}$, then d_min ≤ d_m.
(2) Taks d_m-1 column vectors from all column vectors.
If all combinations $\binom{n}{d_m - 1}$ are independent, then d_min ≥ d_m.
(3) If you can find such d_m, d_min = d_m

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

(1) 7つある列ベクトルから、$d_m=3$　個の列ベクトルを取り出す。

(1) Extract d_m = 3 column vectors out of 7.

$$\begin{bmatrix}0\\1\\1\end{bmatrix}+\begin{bmatrix}0\\1\\0\end{bmatrix}+\begin{bmatrix}0\\0\\1\end{bmatrix}=0$$

よってこの３つの列ベクトルは従属。従って、$d_{min}\leq 3$

These three column vectors are dependent. Therefore, d_min ≤ 3

(2) $d_m-1=2$　個を取り出す。

どの２つの列ベクトルをとっても独立。

2) Take d_m-1 = 2 column vectors.

Any two column vectors are independent.

例えば、$c_1\begin{bmatrix}1\\1\\1\end{bmatrix}+c_2\begin{bmatrix}1\\1\\0\end{bmatrix}=0$　　となるためには、　$c_1=0$、$c_2=0$

従って、$d_{min}\geq 3$

For example, in order for $c_1\begin{bmatrix}1\\1\\1\end{bmatrix}+c_2\begin{bmatrix}1\\1\\0\end{bmatrix}=0$, $c_1=0$、$c_2=0$.

Therefore, d_min ≥ 3

(3)(1)(2)より、$d_{min}=3$

(3) From (1) (2), d_min = 3.

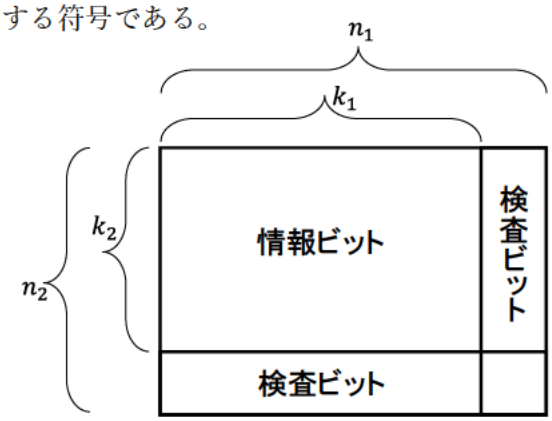
５．６　積符号・拡張ハミング・短縮ハミング

5.6 Horizontal-Vertical Diagonal Check code / Extended Hamming / Shortened Hamming

５．６．１　積符号

5.6.1 Horizontal-Vertical Diagonal Check code

(n_1, k_1)符号と(n_2, k_2)符号を２次元に配列する符号である。



情報ビット数　$k_1 k_2$
検査ビット数　$n_1 n_2 - k_1 k_2$
符号長　　$n_1 n_2$
最小距離　　$d_{min_1} + d_{min_2}$

It is a code that arranges
(n_1, k_1) code horizontally and
the (n_2, k_2) code vertically in two dimensions.

Number of information bits k_1 k_2
Number of inspection bits n_1 n_2-k_1 k_2
Code length n_1 n_2
Minimum distance d_ (min_1) + d_ (min_2)

ただし、d_{min_1}、d_{min_2}は、それぞれ(n_1, k_1)符号、(n_2, k_2)符号の最小距離
where, d_ (min_1) and d_ (min_2) are minimum distances of the (n_1, k_1) code and (n_2, k_2) code, respectively.

例１ (9,4)符号　　演習問題５．２も参照のこと
Example 1 (See also Exercise 5.2)

x_1	x_2	c_1
x_3	x_4	c_2
c_3	c_4	c_5

２つの(3,2)符号の組み合わせ
符号語 $u = (x_1\ x_2\ x_3\ x_4\ c_1\ c_2\ c_3\ c_4\ c_5)$
情報ビット数２×2=4　検査ビット数３×3-2×2=5　符号長３×3=9
検査規則は、以下である。

Combination of two codes.
Codeword: u = (x_1 x_2 x_3 x_4 c_1 c_2 c_3 c_4 c_5)
Number of information bits: 2 x 2 = 4
Number of check bits: 3 x 3-2 x 2 = 5
Code length: 3 x 3 = 9
The checking rules are as follows.

$$
\begin{aligned}
x_1 + x_2 \qquad\quad + c_1 \qquad\qquad\quad &= 0 \\
\quad x_3 + x_4 \quad + c_2 \qquad\quad &= 0 \\
x_1 \quad + x_3 \qquad\quad + c_3 \qquad\quad &= 0 \\
\quad x_2 \quad + x_4 \qquad\quad + c_4 \quad &= 0 \\
x_1 + x_2 + x_3 + x_4 \qquad\quad + c_5 &= 0
\end{aligned}
$$

従って、検査行列は以下となる。

Therefore, the check matrix is as follows.

$$
\mathrm{H} = \begin{bmatrix}
1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
$$

(3,2)符号の最小距離は、２であるので、最小距離 4。

従って、１誤り訂正と２誤り検出が可能。

Since the minimum distance of each (3, 2) code is 2, the minimum distance of (9, 4) code is 4.

Therefore, 1 error correction and 2 error detection are possible.

例２ (35,24)符号　　HDLC で実用化された

Example 2 (35, 24) code

horizontal-vertical diagonal (HVD) parity check in HDLC

x_1	x_2	x_3	x_4	c_1
x_5	x_6	x_7	x_8	c_2
x_9	x_{10}	x_{11}	x_{12}	c_3
x_{13}	x_{14}	x_{15}	x_{16}	c_4
x_{17}	x_{18}	x_{19}	x_{20}	c_5
x_{21}	x_{22}	x_{23}	x_{24}	c_6
c_7	c_8	c_9	c_{10}	c_{11}

(5,4)符号と(7,6)符号の組み合わせ

情報ビット数 4×6=24

検査ビット数 5×7-4×6=11　符号長 5×7=35

(5,4)符号、(7,6)符号の最小距離は、

ともに２であるので、最小距離 4。

Combination of (5, 4) code and (7, 6) code.

Number of information bits: 4 x 6 = 24

Number of check bits: 5 x 7-4 x 6 = 11
Code length: 5 x 7 = 35
Since the minimum distance of (5, 4) code and (7, 6) code are 2, the minimum distance of (35, 24) code is 4.

５．６．２　拡張ハミング(Extended Hamming Code)

5.6.2 Extended Hamming Code

拡大ハミングとも呼ぶ。
例えば、ハミング(7,4)符号にさらに１個のパリティを加えて、(8,4)符号を作る。
つまり、ハミング(7,4)符号の符号語を$(a_7\ a_6\ a_5\ a_4\ a_3\ a_2\ a_1)$とするとき、
$a_7 + a_6 + a_5 + a_4 + a_3 + a_2 + a_1 + a_0 = 0$　となるようにa_0を定める。
$(a_7\ a_6\ a_5\ a_4\ a_3\ a_2\ a_1\ a_0)$で構成された符号を拡張ハミング(8,4)符号と言う。
ハミング(7,4)符号の最小距離は３であるのに対し、(8,4)符号の最小距離は４となる。

For example, add one more parity bit to Hamming (7,4) code to create (8, 4) code.
If a code word of the Hamming (7,4) code is (a_7 a_6 a_5 a_4 a_3 a_2 a_1), set a_0 so that a_7 + a_6 + a_5 + a_4 + a_3 + a_2 + a_1 + a_0 = 0.
The code composed of (a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0) is called an extended Hamming (8,4) code.
The minimum distance of (7, 4) code is 3, while the one of (8, 4) code is 4.

５．６．３　短縮ハミング(Shortened Hamming Code)

5.6.3 Shortened Hamming Code

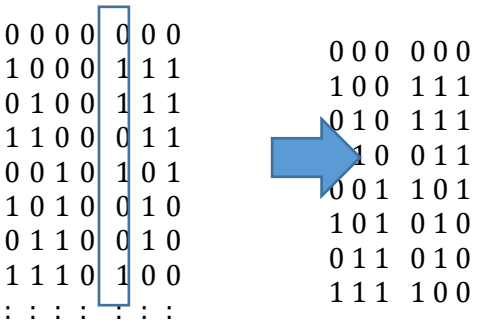
短縮化ハミングとも呼ぶ。
ハミング(n,k)符号からl個の情報記号がすべて０である符号語を取り出し、l個の０を除く。これによって構成される$(n-l,k-l)$符号を短縮化ハミング符号と呼ぶ。
最小重みは変わらない。

Extract codewords from which l information symbols are all 0 from the Hamming (n, k) code, and remove l 0s.
This (n-l, k-l) code is called a shortened Hamming code.
The minimum weight does not change.

例　短縮ハミング(6,3)符号　$l=1$

Example: Shortened Hamming (6, 3) code l = 1

ハミング(7,4)符号	短縮ハミング(6,3)符号
Hamming (7, 4) code	Shortened Hamming (6, 3) code



情報通信基礎Ⅰ　第６章　誤り訂正符号２（符号の多項式表現、巡回符号）

2021/5/25　情シス　渡辺

ポイント

・第５章は符号語をベクトルで扱った。ここでは多項式で扱う方法を学ぶ。

・生成多項式$g(x)$をどのように作ったらよいかが重要。

・巡回符号、ＢＣＨ符号、ＲＳ（リードソロモン）符号　応用としてＱＲコードなど

・多くの符号語の中である基準を満たす符号語だけを利用する（シャノンの第２定理）

第５章：全ビットを加算すると０（パリティ）、検査行列を掛けると０（ハミング等）

第６章：符号語や生成多項式等を多項式で表現する。

生成多項式$g(x)$で割り切れる符号語だけを使う。これは、$g(x)$の根と関係が深い。

符号語を$u(x)$とする。

$g(x)$で割り切れるとは、$u(x) = A(x) \cdot g(x)$　となることである。

ここで、$g(x)$の根αを考えると、$g(\alpha) = 0$。よって、$u(\alpha) = A(\alpha) \cdot g(\alpha) = 0$

すなわち、αは$u(x)$の根でもある。

「生成多項式$g(x)$で割り切れる符号語だけを使う」＝「生成多項式$g(x)$の根を持つ符号語$u(x)$を使う」

Point

- Chapter 5 dealt with codewords as vectors. Here you will learn how to handle by polynomials.
- It is important how to create the generate polynomial g (x).
- The technique is a basis for Cyclic code, BCH code, RS (Reed-Solomon) code, QR code, etc.
- Shannon's second law means using only codewords that hold certain criteria among many codewords.
  - In Chapter 5: the criteria for parity code was 0 when all bits are added, while for Hamming code it was 0 when multiplied by the check matrix.
  - In Chapter 6: Codewords are represented by polynomials and generated by generate polynomials, g(x). Then, code contains only codewords that are divisible by the generate polynomial.

This is closely related to the root of g (x).

Let the codeword be u (x).

Dividing by g (x) means that u (x) = A (x) · g (x).

Here, considering the root α of g (x), g (α) = 0.

Therefore, u (α) = A (α) ·g (α) = 0.

That means, α is also the root of u (x).

"Using only codewords divisible by the generate polynomial g (x)" equals to "Using codewords u (x) with roots in the generate polynomial g (x)"

６．１　基礎

符号の分類　別紙　補足資料１前半

誤りの種類　ランダム誤り、バースト誤り

6.1 Basics

For code classification, see the first half of supplement 1.

Types of errors: Random errors, burst errors.

６．２　多項式表現

６．２．１　符号の多項式表現

符号長nの符号語　$u = (a_{n-1}\, a_{n-2} \cdots a_0)$　$a_i \in GF(2)$　を　$GF(2)$　上の多項式で表現する。（係数が０と１、演算・＋）

6.2 Polynomial expression

6.2.1 Polynomial expression of code

The binary code word u is expressed by polynomials on GF(2). That means coefficient is 0 or 1 and multiplication and sum (XOR) is defined.

u = (a_ (n-1) a_ (n-2) … a_0) a_i ∈ GF (2)

$$u(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots + a_1 x + a_0$$

例　(１１０１)　－＞

Example

$GF(2)$上の多項式の演算

Polynomial operations on GF (2)

$0+0=0$　$0+1=1+0=1$　$1+1=0$　$0-1=1$　$0\cdot 0=0\cdot 1=1\cdot 0=0$　$1\cdot 1=1$

$$ax^i + bx^i = (a+b)x^i \qquad eg.\ x^5 + x^5 = 0$$

$$\frac{ax^i}{bx^j} = \frac{a}{b}x^{i-j} \quad b \neq 0 \qquad eg.\ \frac{x^5}{x^3} = x^2$$

$f(x)+g(x)=h(x)$ -> $f(x)=g(x)+h(x)$, $g(x)=f(x)+h(x)$, $f(x)+g(x)+h(x)=0$

６．２．２　生成多項式

生成多項式$g(x)$：第５章の生成行列に相当するもの

6.2.2 Generation polynomial

Generation polynomial g (x): Corresponds to the generation matrix in Chapter 5.

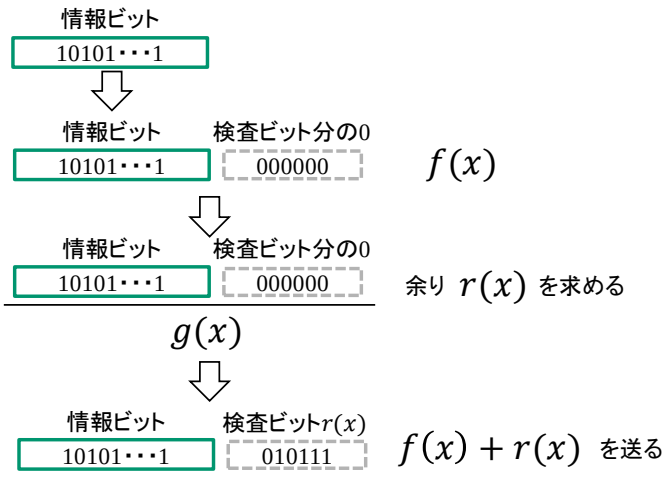
６．２．３　送信・受信の手順

送信側では、検査ビットの初期値を０とし、

（情報ビット＋検査ビット）／生成多項式$g(x)$　の余り（剰余）を検査ビットとして加えて符号化する。

$g(x)$がm次とすると、$r(x)$は$m-1$次である。（つまり、検査ビット数はmビット）

$g(x)$で生成される符号は、(n,k)符号$k=n-m$となる。



6.2.3 Transmission / reception procedure

On the transmitting side, the initial value of the check bits are set to 0.

The remainder of (information bit + check bit) / generated polynomial g (x) are check bits.

The corresponding code word is generated by adding the information bits with the check bits.

If g (x) is m-th order polynomial, r (x) is (m-1)-th order. That is, the number of check bits is m bits.

The code generated by generate polynomial g (x) is the (n, k) code where k = n-m.

$n = 7$、$k = 4$の場合
情報ビット$(a_6\ a_5\ a_4\ a_3)$を　$a_6x^3 + a_5x^2 + a_4x + a_3$とする。次に、$x^3$をかけて、
$f(x) = a_6x^6 + a_5x^5 + a_4x^4 + a_3x^3 + 0x^2 + 0x + 0$　とし、
$f(x)$　を $g(x)$で割った余り$r(x)$を求め、$f(x) + r(x)$　を符号語とする。つまり、
$f(x) = A(x) \cdot g(x) + r(x)$　　$f(x) + r(x) = A(x) \cdot g(x)$ となり、
$f(x) + r(x)$は、$g(x)$で割り切れる。（剰余が０、整除する、とも表現する）

For example, when n = 7, k = 4,
if the information bit (a_6 a_5 a_4 a_3) be
a_6 x ^ 3 + a_5 x ^ 2 + a_4 x + a_3.
Next, multiply by x ^ 3 to create f(x) .
f (x) = a_6 x ^ 6 + a_5 x ^ 5 + a_4 x ^ 4 + a_3 x ^ 3 + 0x ^ 2 + 0x + 0.
Then, divide f (x) by g (x) to find the remainder r (x) to obtain the codeword as f (x) + r (x).

In other words, f(x) can be
f (x) = A (x) ·g (x) + r (x) f (x) + r (x) = A (x) ·g (x).
This means that f(x) + r (x) is divisible by g (x).
(The remainder is 0)

受信側は、受信した符号語を生成多項式$g(x)$で割って余りが０になるかを調べる。
言い換えると、生成多項式$g(x)$で整除される符号語だけを用いていることになる。
The receiving side divides the received codeword by the generate polynomial g (x) and check whether the remainder becomes 0.
If it is 0, the received codeword is correct, otherwise wrong.
In other words, we are using only codewords that are divisible by the generate polynomial g (x).

例６．１　$g(x) = x^3 + x^2 + 1 = (1\ 1\ 0\ 1)$は、$(7, 4)$符号を生成する。（としよう）
Example 6.1
g (x) = x ^ 3 + x ^ 2 + 1 = (1 1 0 1) generates the (7, 4) code.

１）送信側

今、情報ビット$x^3+x+1=(1\,0\,1\,1)$を送るとする。

1) Sender side

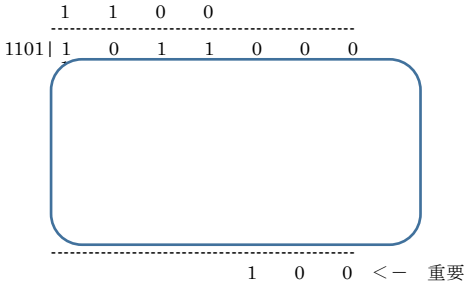
Suppose that the information bit x ^ 3 + x + 1 = (1 0 1 1) is sent.

$f(x)=(1\,0\,1\,1\,|\,0\,0\,0)=x^6+x^4+x^3$

$f(x)/g(x)$ を計算する。

Calculate f (x) / g (x).

【図６．１】　多項式の割り算



以上より、$r(x)=x^2=(1\,0\,0)$

よって、送信すべき符号語は、

$u(x)=f(x)+r(x)=x^6+x^4+x^3+x^2=(1\,0\,1\,1\,1\,0\,0)$

From the above, r (x) = x ^ 2 = (1 0 0)

Therefore, the codeword to be transmitted is

U (x) = f (x) + r (x) = x ^ 6 + x ^ 4 + x ^ 3 + x ^ 2 = (1 0 1 1 1 0 0)

２）受信側

$v(x)=(1\,1\,1\,1\,1\,0\,0)$を受け取ったとする。

$v(x)/g(x)$ を計算する。

2) Receiving side

Suppose that v (x) = (1 1 1 1 1 0 0) is received.

Calculate v (x) / g (x).

【図６．２】

```
     ------------------------------------------------
1101| 1    1    1    1    1    0    0
      1    1    0    1
     ------------------------------------------------
           0    1    0    1
                1    1    0    1
     ------------------------------------------------
                     1    1    1    0
                     1    1    0    1
     ------------------------------------------------
                          0    1    1
```

以上より、シンドロームは、

$s(x) = x + 1 \neq 0$

であり、少なくとも誤りがあることが分かる。

このシンドロームが全ての誤りパタンで異なれば、誤りを特定でき、訂正できる。

１誤りのパタンは７（符号長が７ビットだから）。３ビットでは８パタン表現可能。正しいものも含めて異なるパタンで特定できる可能性がある。（実際、$g(x) = x^3 + x^2 + 1$　は可能である。）

From the above, the syndrome is
s (x) = x + 1 ≠ 0
A least we found that there is an error.
If this syndrome is different in all error patterns, the error can be identified and corrected.
The number of patterns of one bit error is 7 (because the code length is 7 bits).
8 patterns can be expressed with 3 bits.
It may be possible to identify with different patterns,
including the correct one.
(Actually, g (x) = x ^ 3 + x ^ 2 + 1 is possible.)

参考書等：

電子情報通信学会『知識の森』（http://www.ieice-hbkb.org/portal/doc_608.html/）1 群（信号・システム）2 編（符号理論）編主任　藤原融 1 章符号理論の基礎（鎌部、松嶋、鴻巣、高田）、2 章代数的符号（藤原、常磐）

６．３　巡回符号
6.3 Cyclic code

６．３．１　巡回符号
6.3 Cyclic code

符号長nの符号Uのある符号語　$u = (a_{n-1}\, a_{n-2}\, \cdots\, a_0) \in U$　（ただし$a_i \in GF(2)$）を考える。この符号語uを左に１ビットシフトした符号語u'は

$$u' = (a_{n-2}\, a_{n-3}\, \cdots\, a_0\, a_{n-1})$$

である。この語がやはり　U　に属すとき、すなわち $u' \in U$　の時、Uを巡回符号と呼ぶ。

巡回符号は、BCH, RS 等重要な符号の基礎となる。

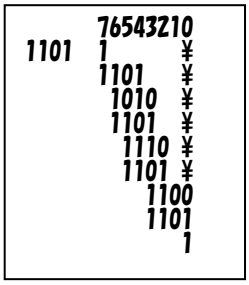
Let’s consider a codeword u = (a_ (n-1) a_ (n-2) … a_0) ∈ U (where a_i ∈ GF (2)) with a code U of code length n.
The codeword u^’, which is the shifted codeword of u to the left by 1 bit, is u^’= (a_ (n-2) a_ (n-3) … a_0 a_ (n-1)).
When this codeword also belongs to U, that is, when u^’∈ U, U is called a cyclic code.
Cyclic code is the basis of important codes such as BCH and RS.

６．３．２　巡回符号の生成多項式

6.3.2 Cyclic code generation polynomial

x^n+1を整除する多項式は、巡回符号の生成多項式となりうる。（全てではないことに注意）

x^n+1を GF(2)上でそれ以上分解できないところまで、因数分解する。構成する各多項式を既約多項式と呼ぶ。



The polynomial that divides x ^ n + 1 can be a cyclic code generation polynomial. (Note that not all)
Factor x ^ n + 1 to the point where it cannot be further decomposed on GF (2).
Each of the polynomials is called an irreducible polynomial.

例６．２　$x^7+1=(x+1)(x^3+x^2+1)(x^3+x+1)$

^^^^^^^^^ ^^^^^^^^

これらが、特に重要

生成多項式$g(x)=x^3+x^2+1$　は、x^7+1を整除する。確認－＞

$g(x)=x^3+x^2+1$　は、$n=7$、$m=3$、$k=4$の巡回$(7,4)$符号を生成する。

情報ビットは、(0 0 0 0)~(1 1 1 1)であり、それぞれにx^3をかけた多項式を$g(x)$で割って余り$r(x)$を求める。（実際の計算は※参照）

These are especially important

The generation polynomial g (x) = x ^ 3 + x ^ 2 + 1 divides x ^ 7 + 1.
See right for confirmation->

g (x) = x ^ 3 + x ^ 2 + 1 generates a cyclic (7, 4) code of n = 7, m = 3, k = 4.
The information bits are possible combinations from (0 0 0 0) to (1 1 1 1).
The remainder r(x) is obtained by multiplying by x ^ 3 then being divided by g (x). (Refer to * for actual calculation)

	情報ビット	余り$r(x)$	符号語$u(x)$	巡回のパタン
0	0 0 0 0	0 0 0	0 0 0 0 0 0 0	p1
1	0 0 0 1	1 0 1	0 0 0 1 1 0 1	p2
2	0 0 1 0	1 1 1	0 0 1 0 1 1 1	p3
3	0 0 1 1	0 1 0	0 0 1 1 0 1 0	p2
4	0 1 0 0	0 1 1	0 1 0 0 0 1 1	p2
5	0 1 0 1	1 1 0	0 1 0 1 1 1 0	p3
6	0 1 1 0	1 0 0	0 1 1 0 1 0 0	p2
7	0 1 1 1	0 0 1	0 1 1 1 0 0 1	p3
8	1 0 0 0	1 1 0	1 0 0 0 1 1 0	p2
9	1 0 0 1	0 1 1	1 0 0 1 0 1 1	p3
10	1 0 1 0	0 0 1	1 0 1 0 0 0 1	p2
11	1 0 1 1	1 0 0	1 0 1 1 1 0 0	p3
12	1 1 0 0	1 0 1	1 1 0 0 1 0 1	p3
13	1 1 0 1	0 0 0	1 1 0 1 0 0 0	p2
14	1 1 1 0	0 1 0	1 1 1 0 0 1 0	p3
15	1 1 1 1	1 1 1	1 1 1 1 1 1 1	p4

※除算の例

* Example of division

```
1101 1000000    1101 1100000    1101 1010000    1101 1001000    1101 1110000    1101 1011000    1101 1101000
     1101            1101            1101            1101            1101            1101            1101
      101             001             111             100             011             110             000
     1101            0000            1101            1101            0000            1101
      111             010             011             101             110             100
     1101            0000            0000            1101            1101
      011             100             110             111             001
     0000            1101            1101            1101            0000
      110             101             001             011             010
```

全てを割り算で求めずに、いくつかを計算しそれらの線形結合で求めることもできる。例えば、6 の符号語は、2 と 4 の符号語の加算で求められる。

Instead of dividing everything, you can calculate by linear combinaiton.
For example, the codeword of 6 is obtained by adding the codewords of 2 and 4.

巡回符号Uが生成多項式$g(x)$で生成されるとする。ただし、$g(x)$は、x^n+1を整除する$GF(2)$上の多

項式である。このとき、Uの任意の符号語uを左にkビットシフトした符号語$u^{(k)}$は、Uの符号語であることを示そう。

Suppose the cyclic code U is generated by the generation polynomial g (x), where g (x) is a polynomial on GF (2) that divides x ^ n + 1.
Then, let's show that the codeword u ^ ((k)), which is the shifted codeword u in U to the left by k bits, is also the codeword of U.

(1) $u(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots a_1 x + a_0 \in U$ とする。

(2) $k = 1$の時

$u^{(1)} = a_{n-2}x^{n-1} + \cdots + a_1 x^2 + a_0 x + a_{n-1}$

$= (a_{n-1}x^{n-1} + \cdots + a_1 x + a_0)x + a_{n-1} + a_{n-1}x^n = u(x)x + a_{n-1}(1 + x^n) = g(x) \cdot A(x)$

$u(x)$は$g(x)$で整除可、$(1 + x^n)$は$g(x)$で整除可

よって、$u^{(1)} \in U$

(3) $k = l(l \geq 1)$で成立すると仮定する。すなわち、 $u^{(l)} = g(x)A(x) \in U$ とすれば、

$u^{(l+1)} = xu^{(l)} + a_{n-l}(1 + x^n) = g(x)B(x)$

よって、$u^{(l+1)} \in U$

(4)(2)(3)より証明された。

(1) u (x) = a_ (n-1) x ^ (n-1) + a_ (n-2) x ^ (n-2) + … a_1 x + a_0 ∈ U.
(2) When k = 1
u ^ ((1)) = a_ (n-2) x ^ (n-1) + … + a_1 x ^ 2 + a_0 x + a_ (n-1)
= (a_ (n-1) x ^ (n-1) + … + a_1 x + a_0) x + a_ (n-1) + a_ (n-1) x ^ n = u (x) x + a_ (n-1) (1 + x ^ n) = g (x) · A (x)
u (x) can be divided by g (x), and (1 + x ^ n) can be divided by g (x)
Therefore, u ^ ((1)) ∈ U
(3) It is assumed that k = l (l ≥ 1) holds. That is, if u ^ ((l)) = g (x) A (x) ∈ U, then
u ^ ((l + 1)) = xu ^ ((l)) + a_ (n-l) (1 + x ^ n) = g (x) B (x)
Therefore, u ^ ((l + 1)) ∈ U
(4) It is proved from (2) (3).

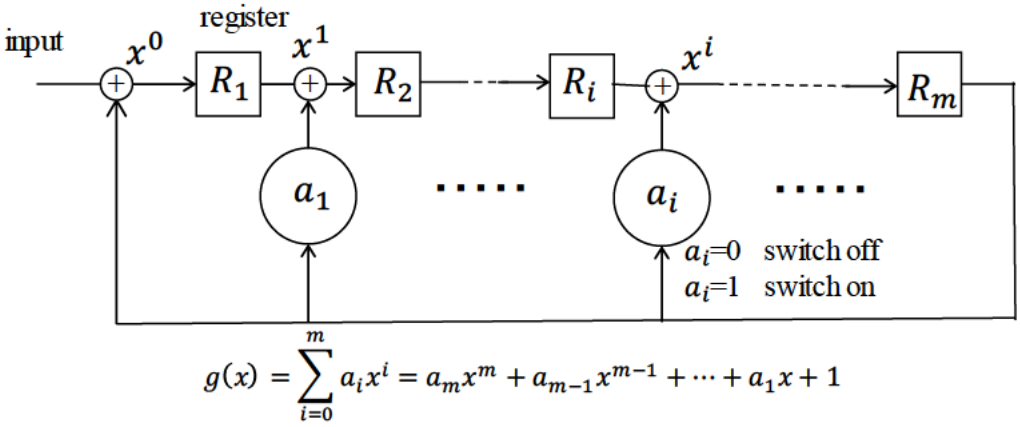
６．３．３　割り算回路

送信側も受信側も生成多項式$g(x)$での割り算を行う。割り算回路はシフトレジスタを用いれば容易にハードウエアで実現できる。

Both the sender and receiver use division by the generated polynomial g (x).

The division circuit can be easily realized by hardware by using shift registers.

【図６．２】　割り算回路

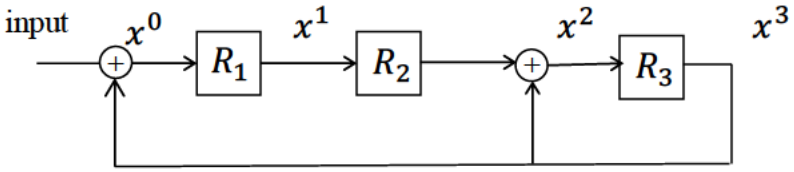


[Fig. 6.2] Division circuit

例６．３　割り算回路の例

Example 6.3 Example of division circuit

【図６．３】　割り算回路　$g(x) = x^3 + x^2 + 1$



レジスタの初期値を 000 とし、１００１０００が入力として与えられると、

入力	R1	R2	R3	
1	1	0	0	
0	0	1	0	
0	0	0	1	
1	0	0	1	＜－　下の［１］に相当
0	1	0	1	＜－　下の［２］に相当
0	1	1	1	＜－　下の［３」に相当
0	1	1	0	＜－　下の［４］に相当

従って、余りは(０１１)。よって(１００１０１１)を送信する。

If the initial value of the registers is 000 and 1 0 0 1 0 0 0 is given as an input sequence, then

Input R1 R2 R3
1 1 0 0
0 0 1 0
0 0 0 1
100 0 1 <-corresponds to [1] below
0 1 0 1 <-corresponds to [2] below
0 1 1 1 <-corresponding to [3] below
0 1 1 0 <-corresponds to [4] below
Therefore, the remainder is (0 1 1), and (1 0 0 1 0 1 1) is transmitted.

確認のため、(１００１０００)を (１１０１)で割ると、
For confirmation, let's divide (1 0 0 1 0 0 0) by (1 1 0 1) as follows.

```
1101| 1   0   0   1   0   0   0
     1   1   0   1
    ---------------------------------------
          1   0   0   0         <－［1］
          1   1   0   1
    ---------------------------------------
              1   0   1   0     <－［2］
              1   1   0   1
    ---------------------------------------
                  1   1   1   0       <－［3］
                  1   1   0   1
    ---------------------------------------
                      0   1   1       <－［4］
```

となり、最終的なレジスタの内容と余りが一致している。
The remainder matches the final contents of the registers.

６．３．４　巡回符号の符号語の求め方、生成多項式と生成行列、検査行列
6.3.4 Calculation of cyclic code words, generator polynomial and generator matrix, check matrix

（１）巡回符号の符号語の求め方
(1) Calculation of code words of the cyclic code

方法としては
（A）情報符号語を生成多項式$g(x)$で割った余り$r(x)$を計算し求める方法
（B）$g(x)$から生成行列Gを求め、線形結合で求める方法
などがある。６．３．２で示したのは、Aの方法

（B）の方法を例で示す。

We have 2 methods;
(A) Calculating the remainder polynomial r(x) obtained by dividing the information code word by the generation polynomial g(x)
(B) Calculating the generator matrix G from generator polynomial g(x) and finding the code word by a linear combination
The method A is shown in 6.3.2, while (B) is shown below with examples.

全ての符号語は、$u(x) = A(x) \cdot g(x)$ と表される。$g(x)$ の次数をm、生成される符号の符号長をnとする。

まず、$A(x) = 1$とした$u(x) = g(x)$は符号語であり、$g(x)$のベクトル表現（長さはn）をg_0とすると、g_0は生成行列の行となる。次に、$A(x) = x$とした$u(x) = xg(x)$も符号語であり、かつ$g(x)$とは一次独立である。また、$xg(x)$のベクトル表現g_1は、g_0を 1 ビット左シフトしたものである。g_1も生成行列の行となる。これを、$n - m - 1$ビットシフトしたベクトルg_{n-m-1}まで繰り返すと以下のように生成行列が求められる。

All codewords are expressed as u (x) = A (x) ·g (x). Let m be the order of g (x) and n be the code length of the generated code.
First, u (x) = g (x) with A (x) = 1 is a codeword.
Let the vector expression of g (x) (length is n) be g_0,
g_0 is a row of the generator matrix.
Next, u (x) = xg (x) with A (x) = x is also a codeword and is linearly independent of g (x).
The vector expression g_1 of xg (x) is the vector obtained by shifting g_0 to the left by 1 bit.
g_1 is also a row of the generator matrix.
If we repeat up to the vector g_ (n-m-1) shifted by n-m-1 bits, the generator matrix is obtained as follows.

$$G = \begin{pmatrix} g_{n-m-1} \\ g_{n-m-2} \\ \vdots \\ g_2 \\ g_1 \\ g_0 \end{pmatrix}$$

例として、生成多項式を$g(x) = x^3 + x^2 + 1$　とする。$n = 7$、$m = 3$、$g_0 = (0001101)$であるから生成行列は以下となる。

As an example, let the generated polynomial be g (x) = x ^ 3 + x ^ 2 + 1. Since n = 7, m = 3, g_0 = (0001101), the generator matrix is as follows.

$$G = \begin{pmatrix} g_3 \\ g_2 \\ g_1 \\ g_0 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}$$

g_0～g_3が一次独立であることは以下のように証明できる。

It can be proved that g_0 to g_3 are linearly independent as follows.

証明　$c_1x^3g(x)+c_2x^2g(x)+c_3xg(x)+c_41g(x)=0$となるのは、$c_1=c_2=c_3=c_4=0$のときであることを示せばよい。（見やすくするために転置して列ベクトルで考える）

$$c_1x^3g(x)+c_2x^2g(x)+c_3xg(x)+c_41g(x)=c_1(1\,1\,0\,1\,0\,0\,0)^T+c_2(0\,1\,1\,0\,1\,0\,0)^T+c_3(0\,0\,1\,1\,0\,1\,0)^T+c_4(0\,0\,0\,1\,1\,0\,1)^T$$

$$=\begin{pmatrix} c_1 & & & \\ c_1 & +c_2 & & \\ & c_2 & +c_3 & \\ c_1 & & +c_3 & +c_4 \\ & c_2 & & +c_4 \\ & & c_3 & \\ & & & c_4 \end{pmatrix}=0$$

これより、$c_1=c_2=c_3=c_4=0$。よって一次独立である。

Proof: Let’s show c_1 x ^ 3 g (x) + c_2 x ^ 2 g (x) + c_3 xg (x) + c_4 1g (x) = 0 holds if and only if c_1 = c_2 = c_3 = c_4 = 0.
(We use transposed column vectors for easy viewing)
c_1 x ^ 3 g (x) + c_2 x ^ 2 g (x) + c_3 xg (x) + c_4 1g (x) = c_1 (1 1 0 1 0 0 0) ^ T + c_2 (0 1 1 0 1) 0 0) ^ T + c_3 (0 0 1 1 0 1 0) ^ T + c_4 (0 0 0 1 1 0 1) ^ T = (■ (c_1 &&&@ c_1 & + c_2 &&@& c_2 & + c_3 & @ c_1 && + c_3 & + c_4 @ & c_2 && + c_4 @&& c_3 &@&&&c_4))) = 0
The equation implies that c_1 = c_2 = c_3 = c_4 = 0 must be held. Therefore, the corresponding vectors are linearly independent.

扱いを簡単にするためにGを行列の基本変形で変形すると、

For easy handling, the matrix G can be transformed by elementary operations of linear algebra.

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

これは既約梯陣形である。Ｇでも、Ｇ’でも同じ符号語ができる。

This is an irreducible ladder form of the matrix G. The same sets of code word can be generated by both G and G'.

G'で生成されるすべての符号語を線形結合によって求める。
Let's find all code words generated by G' by linear combination.

行の番号 Line number	線形結合 Linear combination	重み Weight	パタン Pattern	
①	1 0 0 0 1 1 0	3	1101	p1
②	0 1 0 0 0 1 1	3	1101	p1
③	0 0 1 0 1 1 1	4	10111	p2
④	0 0 0 1 1 0 1	3	1101	p1
①＋②	1 1 0 0 1 0 1	4	10111	p2
①＋③	1 0 1 0 0 0 1	3	1101	p1
①＋④	1 0 0 1 0 1 1	4	10111	p2
②＋③	0 1 1 0 1 0 0	3	1101	p1
②＋④	0 1 0 1 1 1 0	4	10111	p2
③＋④	0 0 1 1 0 1 0	3	1101	p1
①＋②＋③	1 1 1 0 0 1 0	4	10111	p2
①＋②＋④	1 1 0 1 0 0 0	3	1101	p1
①＋③＋④	1 0 1 1 1 0 0	4	10111	p2
②＋③＋④	0 1 1 1 0 0 1	4	10111	p2
①＋②＋③＋④	1 1 1 1 1 1 1	7	11111111	p3
①＋①	0 0 0 0 0 0 0	0	0000000	p4

以上１６個が符号語。p1 は７個、p2 は７個、p3 は１個、p4 は１個がそれぞれ巡回している。
The above 16 are all code words with seven p1, seven p2, one p3, and one p4 patterns. Each pattern is cyclic.

別解　G'が巡回符号を生成することを前提とすれば、G'の一行目、二行目、四行目は同じ符号を巡回したものであり、従って、一行目と三行目の巡回を考えて求めてもよい。
また、Gでも同じ符号ができる。既約梯陣形G'の方が計算が容易である。
Another solution: Assuming that G' generates a cyclic code, the first, second, and fourth lines of G' are cyclic codes. Thus, we can obtain codes by circulate the first and third lines.
Also, the same code words can be obtained with G.
The irreducible ladder form, G' is easier to calculate.

（２）生成多項式と生成行列、検査行列
(2) Generation polynomial, generation matrix and check matrix

例えば、符号語$(a_7\ a_6\ a_5\ a_4\ a_3\ a_2\ a_1)$を生成多項式$g(x) = x^3 + x^2 + 1 = (1101)$で割ることは、以下の計算をしていることとなる。（以下、7とはa_7を、67とは、$a_6 + a_7$を意味する。）

For example, dividing the code word (a_7 a_6 a_5 a_4 a_3 a_2 a_1) by the generation polynomial g (x) = x ^ 3 + x ^ 2 + 1 = (1101) means that the following calculation is performed. (Hereinafter, 7 means a_7 and 67 means a_6 + a_7.)

```
        7    67   567   456
      ------------------------------------------------
1101| 7     6     5     4     3     2     1
      7     7     0     7
            67    5    47
```

よって余りは、$(a_3 + a_4 + a_5 + a_7)x^2 + (a_2 + a_5 + a_6 + a_7)x + (a_1 + a_4 + a_5 + a_6)$となる。

Therefore, the remainder is (a_3 + a_4 + a_5 + a_7) x ^ 2 + (a_2 + a_5 + a_6 + a_7) x + (a_1 + a_4 + a_5 + a_6).

従って、シンドロームは

Therefore, the syndrome is calculated as below.

$$\begin{aligned} s_1 &= a_7 + a_5 + a_4 + a_3 \\ s_2 &= a_7 + a_6 + a_5 + a_2 \\ s_3 &= a_6 + a_5 + a_4 + a_1 \end{aligned}$$

で計算される。よって、検査行列は、以下となる。

The check matrix is as follows.

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

これからGを求められる。

Finally, we obtain the generator matrix G.

６．３．５　巡回ハミングと非巡回ハミング

6.3.5 Cyclic Humming code and Non-cyclic Humming code

別紙　補足資料１中盤

See the supplement 1.

６．３．６　巡回ハミング符号から BCH 符号への拡張
（BCH 符号の後で読み返した方がより深く理解できる。）
6.3.6 Extension from Cyclic Hamming code to BCH code
(It is easier to understand if you read it back after the BCH code.)

参考文献　岩垂好裕　符号理論入門　昭晃堂 p.39-44　より
（１）巡回ハミング(7,4)符号
(1) Cyclic Humming (7,4) code

Let's consider,

$x^7+1=(x+1)(x^3+x^2+1)(x^3+x+1)$を考える。

原始多項式を　$p(x)=x^3+x+1$とする。この根をαとすれば、$\alpha^3+\alpha+1=0$
Let the primitive polynomial be p (x) = x ^ 3 + x + 1 and the root is α, then α ^ 3 + α + 1 = 0

$x^7+1=X(x)(x^3+x+1)$であるので、$\alpha^7+1=X(\alpha)p(\alpha)=0$
Since x ^ 7 + 1 = X (x) (x ^ 3 + x + 1), α ^ 7 + 1 = X (α) p (α) = 0

よって、$\alpha^7=1$　（$\alpha^7=1$　は、αは１の７乗根と言い換えられる。）
Therefore, α ^ 7 = 1 (It means that α is the 7th root of 1).

生成多項式を　$p(x)$と同じとする。
Here, let the generator polynomial be the same as p (x). Thus,

つまり、$g(x)=p(x)=x^3+x+1=(x+\alpha)(x+\alpha^2)(x+\alpha^4)$
生成多項式は、x^7+1を整除する。また３次であるので、符号長$n=7$、検査ビット数 3 の巡回ハミング(7, 4)符号を生成する。単一誤り訂正可能
The generator polynomial divides x ^ 7 + 1. Moreover, since it is the third order polynomial, it generates a cyclic Humming (7, 4) code having a code

length of n = 7 and a number of check bits of 3. Single error correction is possible.

（２）巡回ハミング(15,11)符号

(1) Cyclic Humming (15,11) code

Let's consider,

$x^{15}+1=(x+1)(x^4+x^3+1)(x^4+x+1)(x^4+x^3+x^2+x+1)(x^2+x+1)$を考える。

原始多項式を　$p(x)=x^4+x+1$とする。

Let the primitive polynomial be p (x) = x ^ 4 + x + 1 and the root is α, then α ^ 4 + α + 1 = 0

この根をαとすれば、$\alpha^4+\alpha+1=0$、$\alpha^{15}=1$　（αは１の１５乗根）

(α is the 15th root of 1).

The generator polynomial is

生成多項式　$g(x)=x^4+x+1=(x+\alpha)(x+\alpha^2)(x+\alpha^4)(x+\alpha^8)$

生成多項式は、$x^{15}+1$を整除する。また４次であるので、符号長$n=15$、検査ビット数 4 の巡回ハミング(15, 11)符号を生成する。単一誤り訂正可能

The generator polynomial divides x^15+1. Moreover, since it is the forth order polynomial, it generates a cyclic Humming (15, 11) code having a code length of n =15 and a number of check bits of 4. Single error correction is possible.

（３）（２）巡回ハミング(15,11)符号を拡張して、２誤り訂正可能になるようにしたい。

(3) We want to extend the circular Humming (15,11) shown in (2) so that 2 errors can be corrected.

３－１）準備

3-1) Preparations

(A) 符号語多項式$u(x)$は生成多項式$g(x)$で整除される。すなわち、$u(x)=X(x)g(x)$

従って、$g(x)$の根をαとすると、$u(\alpha)=X(\alpha)g(\alpha)$=0

すなわち、すべての符号語$u(x)$はαを根として持つ。

（別の表現：根αを持つ多項式で表現される符号語だけを使う。）

(A) The code word polynomial u(x) is divided by the generator polynomial g (x). That is, u (x) = X(x)g(x)
Therefore, if the root of g(x) is α, u (α)=X(α)g(α)= 0
That is, all code words u(x) have α as the root.
(In other words, we use only code words expressed by polynomials with root α.)

(B)GF(2)上の任意の多項式$p(x)$について、 $[p(x)]^2 = p(x^2)$ （証明せよ）である。
$p(x)$の根をαとする（$p(\alpha)$＝0）と、$[p(x)]^2|_{x=\alpha} = p(x^2)|_{x=\alpha} = 0$より、
α^2も$p(x)$の根である。

(B) For any polynomial p(x) on GF (2), [p(x)]^2=p(x^2).
If the root of p(x) is α (i.e., p(α)= 0), then [p(x)]^2|_(x=α) = p(x^2)|_(x=α)) = 0. Thus, A^2 is also the root of p (x).

３－２）巡回ハミング(15, 11)の生成多項式は、α 、α^2、α^4、α^8を根に持つ。ならば、、生成多項式$g(x)$がα^3も根に持てばいいかも。。（持つと仮定しよう）

3-2) The generator polynomial of cyclic Humming (15, 11) has α, α^2, α^4, and α^8 as roots. Then, it would be good if the generator polynomial g(x) also has α^3 as its root. Let's assume it is true.

送信符号語$u(x)$ を送信し、i ビット目とj ビット目が誤って、$v(x)$ を受信したとする。

Suppose that code word u(x) is transmitted, and the i-th and j-th bits are erred, and then v(x) is received.

2 ビットの誤りは $e(x) = x^i + x^j$ と表現できる。

The 2-bit error can be denoted as e(x)=x^i+x^j.

送信符号語は、$u(x) = X(x)g(x)$

The transmitted code word is u (x) = X (x) g (x)

受信符号語は、$v(x) = u(x) + e(x) = X(x)g(x) + x^i + x^j$

The received code word is v(x)=u(x)+e(x)=X(x)g(x)+x^i+x^j

受信符号語多項式に、αを代入する。準備(A)を考慮すると、以下のように変形できる。

Substitute α for the received code word polynomial. From preparation (A), it can be transformed as follows.

$v(\alpha) = u(\alpha) + e(\alpha) = e(\alpha) = \alpha^i + \alpha^j = S_1$　・・・（式１）

つぎに、$v(\alpha^3)$を計算すると、（＜－なぜ$v(\alpha^2)$を計算しないのか。$v(\alpha^2)$は役に立つか？）

Next, if we calculate v(α^3), we obtain the followings. (Why not calculate v(α^2). Is it useful?)

$v(\alpha^3) = u(\alpha^3) + e(\alpha^3) = e(\alpha^3) = \alpha^{3i} + \alpha^{3j} = S_3$　（$v(\alpha^3)$をS_3とする。）

(Let v(α^3) be S_3.)

(式１)を変形して、$S_1{}^2 = \alpha^{2i} + \alpha^{2j}$

By transforming Equation 1, S_1^2 =α^2i+α^2j

一般に$a^3 + b^3 = (a+b)(a^2 + ab + b^2)$ であり、$a = \alpha^i$ 、$b = \alpha^j$ とおけば、

Generally, a^3+b^3=(a+b)(a^2+ab+b^2). If a=α^i and b=α^j,then,

$$S_3 = \alpha^{3i} + \alpha^{3j} = (\alpha^i + \alpha^j)(\alpha^{2i} + \alpha^i\alpha^j + \alpha^{2j}) = (\alpha^i + \alpha^j)\{(\alpha^{2i} + \alpha^{2j}) + \alpha^i\alpha^j\}$$

$$= (\alpha^i + \alpha^j)\left\{(\alpha^i + \alpha^j)^2 + \alpha^i\alpha^j\right\} = S_1(S_1{}^2 + \alpha^i\alpha^j)$$

よって、$\alpha^i\alpha^j = \frac{S_3}{S_1} + S_1{}^2$　・・・(式２)

Therefore, (Equation 2)

（式１）と（式２）を連立方程式として解けば、α^iとα^jを求められる。つまり、iとjが求められ、２誤り訂正が可能となる。

By solving (Equation 1) and (Equation 2) as simultaneous linear equations, α^i and α^j can be obtained.
In other words, i and j are determined, thus 2 errors can be corrected.

以上、生成多項式$g(x)$がα^3も根に持てば２誤り訂正が可能であることがわかった。

From the above, we found that 2 error correction is possible if the generation polynomial g(x) also has α^3 as the root.

３－３）生成多項式$g(x)$がα^3も根に持つようにする。

3-3) Let the generator polynomial g(x) also have α^3 as its root.

原始多項式　$p(x) = x^4 + x + 1$　　$\alpha^4 + \alpha + 1 = 0$

Primitive polynomial is p(x)=x^4+x+1　α^4+α+1=0

$q(x)$をα^3を根に持つ最小次元の多項式とし、生成多項式を　$g(x) = p(x)q(x)$　とする。

Let q (x) be the smallest order polynomial with α^3 as the root, and let the generator polynomial be g(x)=p(x)q(x).

準備(B)より、$q(x)$がα^3を根に持つならば、α^6、α^{12}、$\alpha^{24} = \alpha^9$、（$\alpha^{48} = \alpha^3$）も根に持つ。

From preparation (B), if q(x) has α^3 as the root, then it also has α^6, α^12, α^24=α^9, (α^48=α^3) as the roots.

従って、$q(x) = (x + \alpha^3)(x + \alpha^6)(x + \alpha^9)(x + \alpha^{12}) = x^4 + x^3 + x^2 + x + 1$

Therefore,

以上より、生成多項式は、以下となる。

Finally, we get the generator polynomial as follows.

$$g(x) = p(x)q(x) = (x^4 + x + 1)(x^4 + x^3 + x^2 + x + 1) = x^8 + x^7 + x^6 + x^4 + 1$$
$$= (x + \alpha^1)(x + \alpha^2)(x + \alpha^3)(x + \alpha^4)(x + \alpha^6)(x + \alpha^8)(x + \alpha^9)(x + \alpha^{12})$$

この生成多項式は、べきが連続した根　$\alpha^1, \alpha^2, \alpha^3, \alpha^4$　を持つため、$2t_0 = 4$より２誤り訂正可能である（後の BCH 符号の生成多項式を参照）。また、$x^{15} + 1$を整除し、8 次であるので、符号長$n =$ 15、検査ビット数 8 の 2 誤り訂正可能な BCH(15, 7)符号を生成する。

Since this generator polynomial has continuous roots α^1, α^2, α^3, α^4, it is possible to correct 2 errors from 2t_0=4 (see the later chapter of BCH code).

Also, since it divide x^15+1 and 8th order, it generates a BCH (15, 7) code that can correct 2 errors with a code length of n=15 and a number of check bits of 8.

（４）(15, 7)符号を元にして、３誤り訂正可能な符号を生成したい。

(4) We want to generate a code that can correct 3 errors based on the (15, 7) code.

３－３）と同様に、生成多項式がα^5も根として持つように構成する。
Similar to 3-3), let’s configure a generator polynomial to have α^5 as a root.

結果は、(15,5)符号となる。（確認せよ。）
The result is the (15, 5) code.

（５）４，５，６誤り訂正可能な符号は？　７誤り訂正可能な符号は？
(5) Can we obtain a code which can correct 4 errors, 5 errors, 6 errors? The answer is no, but we can obtain a code which can correct 7 errors.

結果は７誤り訂正可能な(15,1)符号。(00・・・0)(11・・・1)の２語しかない符号。
It is the 7 error-correctable (15, 1) code. It has only two code words, i.e., (00 ... 0) (11 ... 1).

（６）まとめ
(6) Summary

符号名 Code name	生成多項式の根 Root of the generator polynomial	生成多項式の次数 Degree of the generator polynomial	符号語数 Number of code words
１誤り訂正可能(15, 11)符号 1 Error correctable (15, 11) code	$\alpha^1, \alpha^2, \alpha^4, \alpha^8$	4 次	$2^{11} = 2048$
２誤り訂正可能(15, 7)符号 2 Error correctable (15, 7) code	$\alpha^1, \alpha^2, \alpha^3, \alpha^4, \alpha^6, \alpha^8, \alpha^9, \alpha^{12}$	8 次	$2^7 = 128$
３誤り訂正可能(15, 5)符号 3 Error correctable (15, 5) code	$\alpha^1, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6, \alpha^8, \alpha^9, \alpha^{10}, \alpha^{12}$	10 次	$2^5 = 32$
７誤り訂正可能(15, 1)符号 7 Error correctable (15, 1) code	$\alpha^1, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6, \alpha^7, \alpha^8, \alpha^9, \alpha^{10}, \alpha^{11}, \alpha^{12}, \alpha^{13}, \alpha^{14}$	14 次	$2^1 = 2$

情報通信基礎Ⅰ　第７章　誤り訂正符号３（BCH, RS, Goppa）

2021/12/８　情シス　渡辺

ポイント

・第６章では、２誤り以上の訂正が可能となるように巡回ハミングを拡張した。その際、生成多項式の根のべき乗の連続性がポイントとなった。ここではこの性質に基づいた BCH 符号、RS 符号、Goppa 符号を学ぶ。

・生成多項式$g(x)$をどのように作るかが重要。

Point

- In Chapter 6, Cyclic Hamming code was extended so that more than one error can be corrected. The basic feature was the continuity of the power of the root of the generator polynomial. In this chapter, we learn BCH code and RS code based on the feature. Then, Goppa code is also presented.
- What is important is how to make the generator polynomial g(x).

７．１　BCH 符号

7.1 BCH code

７．１．１　概要

7.1.1 Overview

・1959 年頃 Bose, Chaudhuri, Hocquenghem が考案

・巡回符号の一種

・生成多項式の$g(x)$の根、すなわち$g(\alpha) = 0$となるαによって最小距離が決定される。

逆に言うと、必要な誤り訂正能力を実現するために最小距離から生成多項式を決定でき、符号を設計できる。

- Invented by Bose, Chaudhuri, Hocquenghem around 1959
- One of Cyclic codes
- Minimum distance is determined by the root of generator polynomial g(x), that is α where g (α) = 0.

In other words, the generator polynomial can be determined from the desired minimum distance and the code can be designed to realize the required error correction capability.

例）原始多項式$p(x) = x^4 + x + 1$、生成多項式$g(x) = p(x) = x^4 + x + 1$　とする BCH 符号
・$g(x) = 0$の根は４つ。
・$g(x)$は、$x^{15} + 1$を整除する。
従って、$n = 15$。すなわち、$x^{15} + 1 = (x + 1)(x^4 + x + 1)$…と書ける。結果的には、$g(x)$は、符号長１５、検査ビット４の巡回（15, 11）符号を生成する。

Example) BCH code with primitive polynomial p(x) = x ^ 4 + x + 1, and generator polynomial g (x) = p (x) = x ^ 4 + x + 1
- g(x) has four roots.
- g(x) divides x ^ 15 + 1.

Therefore, x ^ 15 + 1 = (x + 1) (x ^ 4 + x + 1)A(x), A(X) is an arbitral polynomial. As a result, g(x) generates a cyclic (15, 11) code with a code length of 15 and check bit 4.

次に、$g(x)$の根の性質を調べてみる。一般には、$g(x)$は複数の根を持つが、そのうちの一つをαとする。このとき、以下の定理が成り立つ。（α^0～α^{14}は、α^0～α^3で表現可能。）

Let us examine the properties of roots of g(x). Generally, g (x) has multiple roots, one of which is α. At this time, the following theorem holds. (α^0 to α^14 can be expressed by α^0 to α^3.)

［定理７．１］
巡回符号を生成する生成多項式$g(x)$が

$$\alpha^1, \alpha^2, \alpha^3, \alpha^4, \cdots, \alpha^{2t_0}$$

のように連続したべき乗の根を持つとき、$g(x)$で生成される符号の最小距離は、$2t_0 + 1$となる。この生成多項式$g(x)$で生成される符号を BCH 符号と呼ぶ。（フーリエ変換などを使った証明がある。別紙　補足資料１後半）

[Theorem 7.1]

If generator polynomial g(x) has roots of continuous power, such as
α ^ 1, α ^ 2, α ^ 3, α ^ 4, …, α ^ (2t_0),
the minimum distance of the cyclic code generated by g (x) is 2t_0 + 1.
The code is called the BCH code.
(The proof can be done using Fourier transform of codes, etc. See supplementary material 1.)

例７．１　生成多項式を$g(x) = x^4 + x + 1$　とするBCH符号

生成多項式$g(x)$の一つの根をαとする。すなわち、$\alpha^4 + \alpha + 1 = 0$

$g(x)$が、α^2を根に持つかを調べる。

$g(\alpha^2) = \alpha^8 + \alpha^2 + 1 = (\alpha + 1)^2 + \alpha^2 + 1 = 0$　　従って、根に持つ。

$g(x)$が、α^3を根に持つかを調べる。

$g(\alpha^3) = \alpha^{12} + \alpha^3 + 1 = (\alpha + 1)^3 + \alpha^3 + 1 = \alpha^2 + \alpha \neq 0$　よって、根に持たない。

以上より、生成多項式$g(x)$は、α^1, α^2をべき乗が連続する根として持つ。これより、
$2t_0 = 2$ であり、最小距離$2t_0 + 1$は３、１誤り訂正可能な符号を生成する。

Example 7.1 BCH code with a generator polynomial, g (x) = x ^ 4 + x + 1
Let α be one root of g (x). That is, α ^ 4 + α + 1 = 0.
At first we check if g (x) has α ^ 2 as its root.
G (α ^ 2) = α ^ 8 + α ^ 2 + 1 = (α + 1) ^ 2 + α ^ 2 + 1 = 0.
Therefore, it is the root of g(x).
Secondly we check if g (x) has α ^ 3 as its root.
G (α ^ 3) = α ^ 12 + α ^ 3 + 1 = (α + 1) ^ 3 + α ^ 3 + 1 = α ^ 2 + α $\neq$ 0
Therefore, it is not the root of g(x).
Then, the generator polynomial g (x) has α ^ 1 and α ^ 2 as roots of continuous powers.
Then, since 2t_0 = 2, and the minimum distance 2t_0 + 1 is 3.,which means the code is 1 error-correctable code.

$g(x)$の周期は１５（$x^{15} + 1$を整除する）　－＞符号長$n = 15$

$g(x)$の次数は４　－＞検査ビット数$m = 4$

これより、$g(x) = x^4 + x + 1$は、最小距離３のBCH(15, 11)符号を生成する。

（BCH(15, 11, 3)と書くこともある。）

The period of g (x) is 15 so that it divides x ^ 15 + 1). Then the code length n is 15.
The order of g (x) is 4 so that number of check bits m is 4.
Thus, g (x) = x ^ 4 + x + 1 generates BCH (15, 11) code with a minimum distance of 3. It is also written as BCH (15, 11, 3).

例７．２　生成多項式を$g(x) = (x^4 + x + 1)(x^4 + x^3 + x^2 + 1) = x^8 + x^7 + x^6 + x^4 + 1$
とする。ただし、原始多項式を$p(x) = x^4 + x + 1$　とし、その根をα　とする。
・$g(x)$は、連続した根$\alpha^1, \alpha^2, \alpha^3, \alpha^4$　を持つ。$2t_0 = 4$ であり、最小距離$2t_0 + 1$は 5。従って、２誤り訂正可能
・$g(x)$は$x^{15} + 1$を整除する。　よって符号長は１５
・$g(x)$は８次。検査ビット数は８。
以上より、$g(x) = x^8 + x^7 + x^6 + x^4 + 1$は、BCH(15, 7, 5)を生成する。

Example 7.2 Let us assume that generator polynomial g (x) = (x ^ 4 + x + 1) (x ^ 4 + x ^ 3 + x ^ 2 + 1) = x ^ 8 + x ^ 7 + x ^ 6 + x ^ 4 + 1
and the primitive polynomial p (x) = x ^ 4 + x + 1, and its root is α ^.
 - g (x) has α ^ 1, α ^ 2, α ^ 3, and α ^ 4 as roots of continuous powers. 2t_0 = 4, and the minimum distance 2t_0 + 1 is 5.
Therefore, 2 errors can be correctable.
 - g(x) divides x ^ 15 + 1. Thus, the code length is 15.
 - g(x) is 8th order. Then, the number of check bits is 8.
From the above, g (x) = x ^ 8 + x ^ 7 + x ^ 6 + x ^ 4 + 1 generates BCH (15, 7, 5).

これらの例は、生成多項式が与えられて最小距離を求めている。実際の符号を設計する時は、<u>最小距離から生成多項式を求めたい</u>。これは７．１．３で議論する。

In these examples, a generator polynomial is given to find the minimum distance. When designing an actual code, we want to find the generator polynomial from a desirable minimum distance. We will discuss later in 7.1.3.

７．１．２　拡大体

7.1.2 Extension field

・体に何らかの元を追加することによって、体を拡大することを考える。基礎となった体を基礎体、拡大された後の体を拡大体と呼ぶ。（より正確には、基礎体上の多項式の根を追加して拡大体を構成する。）
・以下では符号理論に特に関係が深い$GF(2)$を基礎体とする。

- Let us consider extension of field by adding some element to an original field.
The original field is called the subfield.
More precisely, the subfield is extended by adding roots of the polynomials to the subfield.
- In the following, we will use GF (2) as the subfield, which is closely related to coding theory.

（１）符号の多項式表現の復習
・多項式：$GF(2)$の元を係数とする多項式を考える。係数が 0, 1 の多項式。x^3+x+1など
　これを$GF(2)$上の多項式と表現する。
・既約多項式：$GF(2)$上の多項式でこれ以上因数分解できない多項式を既約多項式と呼ぶ。
　例 $x^7+1=(x+1)(x^3+x^2+1)(x^3+x+1)$　x^7+1は非既約。$x+1$、x^3+x^2+1、x^3+x+1は既約。
・周期：ある多項式がx^n+1を整除し、かつ$x^i+1\ (i<n)$を整除しない時、その多項式の周期がnであると言う。
・原始多項式：既約多項式のうち、周期が最大となるものを原始多項式と言う。
　例 $x^7+1=(x+1)(x^3+x^2+1)(x^3+x+1)$　原始多項式は、x^3+x^2+1、x^3+x+1
　例 $x^{15}+1=(x+1)(x^4+x^3+1)(x^4+x+1)(x^4+x^3+x^2+x+1)(x^2+x+1)$
　　この原始多項式は、(x^4+x^3+1)、(x^4+x+1)、$(x^4+x^3+x^2+x+1)$
　　(x^2+x+1)は、(x^3+1)を整除するので原始多項式ではない。

(1) Review of polynomial representation of code

- Polynomial: A polynomial whose coefficient is the element of GF (2), that is, 0 and 1. Ex) 1. x ^ 3 + x + 1.
- Ireducible polynomials: Polynomials on GF (2) that cannot be further reduced.

Example

x ^ 7 + 1 = (x + 1) (x ^ 3 + x ^ 2 + 1) (x ^ 3 + x + 1).

x ^ 7 + 1 is not irreducible. x + 1, x ^ 3 + x ^ 2 + 1, x ^ 3 + x + 1 are irreducible on GF(2).

- Period: When a polynomial divides x ^ n + 1 and does not divide x ^ i + 1, where i<n, the period of the polynomial is n.
- Primitive polynomials: Of the irreducible polynomials, the one with the maximum period is called a primitive polynomial.

Example x ^ 7 + 1 = (x + 1) (x ^ 3 + x ^ 2 + 1) (x ^ 3 + x + 1).

Primitive polynomials are x ^ 3 + x ^ 2 + 1, and x ^ 3 + x + 1

Example x ^ 15 + 1 = (x + 1) (x ^ 4 + x ^ 3 + 1) (x ^ 4 + x + 1) (x ^ 4 + x ^ 3 + x ^ 2 + x + 1) (x ^ 2 + x + 1)

Primitive polynomials are (x ^ 4 + x ^ 3 + 1), (x ^ 4 + x + 1), and (x ^ 4 + x ^ 3 + x ^ 2 + x + 1).

x ^ 2 + x + 1 is not a primitive polynomial because it divides (x ^ 3 + 1).

Example of primitive polynomials on GF(2) (mainly polynomials consisting of three terms)

$GF(2)$上の原始多項式の例（主に３項からなる多項式）

1	$x+1$	9	x^9+x^4+1	17	$x^{17}+x^3+1$	25	$x^{25}+x^3+1$
2	x^2+x+1	10	$x^{10}+x^3+1$	18	$x^{18}+x^7+1$	26	$x^{26}+x^6+x^2+x+1$
3	x^3+x+1 x^3+x^2+1	11	$x^{11}+x^2+1$	19	$x^{19}+x^5+x^2+x+1$	27	$x^{27}+x^5+x^2+x+1$
4	x^4+x+1 x^4+x^3+1	12	$x^{12}+x^6+x^4+x+1$	20	$x^{20}+x^3+1$	28	$x^{28}+x^3+1$
5	x^5+x^2+1	13	$x^{13}+x^4+x^3+x+1$	21	$x^{21}+x^2+1$	29	$x^{29}+x^2+1$
6	x^6+x+1	14	$x^{14}+x^{10}+x^6+x+1$	22	$x^{22}+x+1$	30	$x^{30}+x^{23}+x^2+x+1$
7	x^7+x+1	15	$x^{15}+x+1$	23	$x^{23}+x^5+1$	31	$x^{31}+x^3+1$
8	$x^8+x^4+x^3+x^2+1$	16	$x^{16}+x^{12}+x^3+x+1$	24	$x^{24}+x^7+x^2+x+1$	32	$x^{32}+x^{22}+x^2+x+1$

・原始多項式の根を原始根という。原始多項式の周期をnとすると、原始根は１のn乗根である。
例 x^3+x^2+1の根α。x^3+x^2+1はx^7+1を整除するので、$\alpha^7+1=0$。よって、$\alpha^7=1$

・$GF(2)$上の任意の多項式$u(x)$を多項式$g(x)$で割り算した余りの多項式$r(x)$を考える。
例 $u(x)=x^6+x^5 \quad g(x)=x^3+x+1 \quad (x^6+x^5)\ mod\ (x^3+x+1)=x$
$g(x)$の次元をmとすると、余りの多項式$r(x)$の次元は$m-1$以下となる。
$GF(2)$上の任意の多項式$u(x)$をm次の多項式$g(x)$で除した余りの多項式（剰余多項式）の集合$\{r(x)\}$は、$m-1$次以下の多項式すべてが含まれる。$u(x)=f(x)g(x)+r(x)$であり、$u(x)$は任意であるので、$r(x)$はどのような$m-1$次以下の多項式にもなる。

- Root of a primitive polynomial is called a primitive root. If the period of the primitive polynomial is n, the primitive root is 1 to the n-th root.
  Example
  α is a root of x ^ 3 + x ^ 2 + 1. Since x ^ 3 + x ^ 2 + 1 divides x ^ 7 + 1, α ^ 7 + 1 = 0. Therefore, α ^ 7 = 1
- Let us assume that the remainder polynomial r (x) obtained by dividing an arbitrary polynomial u (x) by a polynomial g (x) on GF(2).
  Example
  u (x) = x ^ 6 + x ^ 5 g (x) = x ^ 3 + x + 1
  which means (x^6 + x ^ 5) mod (x ^ 3 + x + 1) = x.
- If the dimension of g (x) is m, the dimension of the remainder polynomial r (x) is m-1 or less.
- The set {r (x)} of the remainder polynomials obtained by dividing any polynomial u (x) by the polynomial g (x) of degree of m includes all possible polynomials of degree m-1 or less.

In other words, since u (x) = f (x) g (x) + r (x) and u (x) is arbitrary, r (x) can be any polynomial of degree m-1 or less.

（２）拡大体　$GF(2^m)$

・$GF(2)$上の多項式$u(x)$の根（つまり$u(\alpha)=0$となるα）は、一般には$GF(2)$上にはない。このαを$GF(2)$に追加して、拡大体を構成する。$GF(2)$は加法や乗法が定義されているので、αを加えることはαのべき乗などα以外の元も加えることになる。

・$GF(2)$上の任意の多項式$u(x)$をm次の原始多項式$g(x)$で除した余りの多項式の集合$\{r(x)\}$は、$m-1$次以下の多項式すべてが含まれ、０以外は$g(x)$の根（αとする）のべき乗で表現できる。集合$\{r(x)\}=\{0,\alpha^0,\alpha^1,\alpha^2,,,\}$は拡大体$GF(2^m)$をなす。

(2) Field extension GF (2 ^ m)

- The root of the polynomial u (x) on GF (2), that is, α such that u (α) = 0) is generally not on GF (2).

To form an extension of the field, α is added to GF (2).

Since field defines addition and product operations, adding α also adds elements other than α, such as the power of α.

- The set {r (x)} of the remainder polynomials obtained by dividing any polynomial u (x) on GF (2) by the primitive polynomial g (x) of degree m contains all polynomials of degree m-1 and less.

All elements except 0 can be expressed by the power of the root of g (x).

- The extension filed GF(2^m) is the set of {r (x)} = {0, α ^ 0, α ^ 1, α ^ 2, ,,,}.

---例（2 次の原始多項式の例）

・$GF(2)$上の任意の多項式を二次の多項式で除した余りの集合（剰余多項式集合と呼ぶ）は、１次以下の多項式すべてが含まれる。剰余多項式集合の要素を列挙すると以下となる。

多項式	ベクトル表現
0	00
1	01
x	10
$x+1$	11

・$GF(2)$上の既約多項式（原始多項式）$g(x)=x^2+x+1$の根をα（つまり$g(\alpha)=\alpha^2+\alpha+1=0$）とする。剰余多項式集合の各多項式にαを代入した値は、以下となる。

----- Example (Example of a quadratic primitive polynomial)

- The remainder set (called the remainder polynomial set) obtained by dividing any polynomial on GF (2) by a quadratic polynomial includes all polynomials of degree 1 or less.

The elements of the remainder polynomial set are listed below.

polynomial	vector expression
0	00
1	01
x	10
x + 1	11

- Let α be the root of the irreducible polynomial (primitive polynomial) g (x) = x ^ 2 + x + 1 on GF (2). (that is, g (α) = α ^ 2 + α + 1 = 0)

The value obtained by substituting α for each polynomial in the remainder polynomial set is as follows.

polynomial vector expression α power expression

多項式	ベクトル表現	αべき表現	
0	00	0	
1	01	1	$=\alpha^0$
x	10	α	$=\alpha^1$
$x+1$	11	$\alpha+1$	$=\alpha^2$

従って、0 以外の要素をαのべき乗で表現できる。
以上のことを、$GF(2)$上の任意の多項式を原始多項式$g(x)=x^2+x+1$で割った余りの多項式（剰余多項式）の集合は、拡大体$GF(2^2)=\{0,1,\alpha,\alpha^2\}$　を構成する、と言う。
・符号理論からすると、基礎体$GF(2)$の二要素から構成される符号の世界{0, 1}から、2 個の組合せでできる$GF(2^2)$の４要素で構成される世界{00, 01, 10, 11}に拡大した、と言える。

Therefore, all elements except 0 can be expressed by the power of α.
The set of the remainder polynomial (remainder polynomial) obtained by dividing any polynomial on GF (2) by the primitive polynomial g (x) = x ^ 2 + x + 1 is said to be the extension field GF (2 ^ 2).) = {0,1, α, α ^ 2}.
- From the point of coding theory view, the code space of {0, 1} of GF(2) is extended to the space of {00, 01, 10, 11} of GF (2 ^ 2).

---例（３次の原始多項式の例）
・$GF(2)$上の任意の多項式を原始多項式x^3+x+1（ベクトル表現では 1011）　で除した余りの集合は、以下となる。

多項式	ベクトル表現	αべき表現	
0	000	0	
1	001	1	$=\alpha^0$
x	010	α	$=\alpha^1$
$x+1$	011	$\alpha+1$	$=\alpha^3$
x^2	100	α^2	$=\alpha^2$
x^2+1	101	α^2+1	$=\alpha^6$
x^2+x	110	$\alpha^2+\alpha$	$=\alpha^4$
x^2+x+1	111	$\alpha^2+\alpha+1$	$=\alpha^5$

ここで、$\alpha^3+\alpha+1=0$　より

$\alpha^3=\alpha+1$

$\alpha^4=\alpha(\alpha+1)=\alpha^2+\alpha$

$\alpha^5=\alpha(\alpha^2+\alpha)=\alpha^3+\alpha^2=\alpha+1+\alpha^2$

$\alpha^6=\alpha(\alpha^2+\alpha+1)=\alpha^3+\alpha^2+\alpha=\alpha+1+\alpha^2+\alpha=\alpha^2+1$

を使っている。拡大体は、$GF(2^3)=\{0,1,\alpha,\alpha^2,\alpha^3,\alpha^4,\alpha^5,\alpha^6\}$である。

--- Example (Example of a cubic primitive polynomial)
- The set of remainders obtained by dividing any polynomial on GF (2) by the primitive polynomial x ^ 3 + x + 1 (1011 in vector expression) is as follows.

Polynomial vector expression	α power expression		
0	000	0	
1	001	1	$=\alpha^0$
x	010	α	$=\alpha^1$
$x+1$	011	$\alpha+1$	$=\alpha^3$
x^2	100	α^2	$=\alpha^2$
x^2+1	101	α^2+1	$=\alpha^6$
x^2+x	110	$\alpha^2+\alpha$	$=\alpha^4$

$x^2 + x + 1$ 111 $\alpha^2 + \alpha + 1$ $=\alpha^5$

From $\alpha^3 + \alpha + 1 = 0$, we use

$\alpha^3 = \alpha + 1$

$\alpha^4 = \alpha(\alpha + 1) = \alpha^2 + \alpha$

$\alpha^5 = \alpha(\alpha^2 + \alpha) = \alpha^3 + \alpha^2 = \alpha + 1 + \alpha^2$

$\alpha^6 = \alpha(\alpha^2 + \alpha + 1) = \alpha^3 + \alpha^2 + \alpha = \alpha + 1 + \alpha^2 + \alpha = \alpha^2 + 1$

The extension field is GF (2 ^ 3) = {0,1, α, α ^ 2, α ^ 3, α ^ 4, α ^ 5, α ^ 6}.

---例（４次の原始多項式の例）

・$GF(2)$上の任意の多項式を原始多項式$x^4 + x + 1$ (ベクトル表現 10011)で除した余りの集合は、以下となる。

--- Example (Example of a 4th-order primitive polynomial)

- The set of remainders obtained by dividing any polynomial on GF (2) by the primitive polynomial x ^ 4 + x + 1 (vector expression is 10011) is as follows.

Polynomial expression Vector expression α exponentiation expression

多項式表現	ベクトル表現	αべき乗表現
0	0000	0
1	0001	$=\alpha^0$
x	0010	α $=\alpha^1$
$x + 1$	0011	$\alpha + 1$ $=\alpha^4$
x^2	0100	α^2 $=\alpha^2$
$x^2 + 1$	0101	$\alpha^2 + 1$ $=\alpha^8$
$x^2 + x$	0110	$\alpha^2 + \alpha$ $=\alpha^5$
$x^2 + x + 1$	0111	$\alpha^2 + \alpha + 1$ $=\alpha^{10}$
x^3	1000	α^3 $=\alpha^3$
$x^3 + 1$	1001	$\alpha^3 + 1$ $=\alpha^{14}$
$x^3 + x$	1010	$\alpha^3 + \alpha$ $=\alpha^9$
$x^3 + x + 1$	1011	$\alpha^3 + \alpha + 1$ $=\alpha^7$
$x^3 + x^2$	1100	$\alpha^3 + \alpha^2$ $=\alpha^6$
$x^3 + x^2 + 1$	1101	$\alpha^3 + \alpha^2 + 1$ $=\alpha^{13}$
$x^3 + x^2 + x$	1110	$\alpha^3 + \alpha^2 + \alpha$ $=\alpha^{11}$
$x^3 + x^2 + x + 1$	1111	$\alpha^3 + \alpha^2 + \alpha + 1$ $=\alpha^{12}$

拡大体は、$GF(2^4)=\{0,1,\alpha,\alpha^2,\alpha^3,\alpha^4,\alpha^5,\alpha^6,\alpha^7,\alpha^8,\alpha^9,\alpha^{10},\alpha^{11},\alpha^{12},\alpha^{13},\alpha^{14}\}$である。

The extension field is
$GF(2^4)=\{0,1,\alpha,\alpha^2,\alpha^3,\alpha^4,\alpha^5,\alpha^6,\alpha^7,\alpha^8,\alpha^9,\alpha^{10},\alpha^{11},\alpha^{12},\alpha^{13},\alpha^{14}\}$

７．１．３　BCH 詳細
7.1.3. BCH details

（１）BCH の定義（以下は２元に限定して説明する）
・生成多項式の根に誤り訂正能力が依存する性質を利用。
・$GF(2^m)$上で演算を行う。
・m次の原始多項式を１つ選ぶ。その根を αとする。符号長は$n=2^m-1$である。（cf. primitive）
・達成したい最小距離d_{min}を$n=2^m-1$以下の整数で選ぶ。
・べき乗が連続した根（$\alpha^l,\alpha^{l+1},\cdots,\alpha^{l+d_{min}-2}$）を持つ最小次数の多項式を生成多項式$g(x)$とする。
・この生成多項式$g(x)$により、最小距離がd_{min}である符号が構成される。

$$g(x)=\left[\prod_{i=l}^{l+d_{min}-2}\left(x-\alpha^i\right)\right]A(x) \quad （式 7.1.3.1）$$

ただし、$A(x)$は根α^l～$\alpha^{l+d_{min}-2}$の 2 乗、4 乗などを根として持つ多項式である。（$(x-\alpha^{2l})(x-\alpha^{4l})\cdots\left(x-\alpha^{2(l+1)}\right)\left(x-\alpha^{4(l+1)}\right)\cdots$）。また、$g(x)$の中に重複した項（重根）がないように構成する。
・lは任意の非負整数。普通は 0 または 1 が選ばれる（cf. narrow）
・最小距離がd_{min}の場合、t（$t=\left\lfloor\frac{d_{min}-1}{2}\right\rfloor$）個以下の誤りを訂正できる

(1) Definition of BCH (The following is limited to binary code)
- It uses the property that error correction capability depends on the roots of the generator polynomial.
- All operations are performed on GF (2 ^ m).
- We choose one primitive polynomial of order m. Let α be its root. The code length is n = 2 ^ m-1. (Cf. primitive)
- Then, we select the minimum distance d_min you want to achieve with any positive integer less than or equal to n = 2 ^ m-1.
- The polynomial of the minimum degree with continuous roots of powers (α ^ l, α ^ (l + 1), …, α ^ (l + d_min-2)) is the generator polynomial g (x).
-The generator polynomial g (x) generates a code whose minimum distance is d_min.
g (x) = [Π_ (i = l) ^ (l + d_min-2) ▒ (x-α ^ i)] A (x) (Equation 7.1.3.1)
where A (x) is a polynomial whose roots are the roots α ^ l to α ^ (l + d_min-2) squared, 4-th ordered, etc.

That is,((X-α ^ 2l) (x-α ^ 4l) … (x-α ^ (2 (l + 1))) (x-α ^ (4 (l + 1))) …).
Also, g(x) should not include multiple roots.
- l is an arbitrary non-negative integer. Usually 0 or 1 is chosen (cf. narrow)
- If the minimum distance is d_min, then it can correct errors of t (t = ⌈(d_min-1) / 2⌉) or less.

（２）具体的な BCH 符号
(2) Example of BCH code

１誤り訂正：$d_{min}=3$　(3,1)(7,4)(15,11),,,(255,247)　巡回ハミング符号と同一
２誤り訂正：$d_{min}=5$　(15,7)(31,21),,,(255,239)
３誤り訂正：$d_{min}=7$　(15,5),,,(255,231)
４誤り訂正：$d_{min}=9$　(63,39),,,(255,223)

1 Error correction code: $d_{min}=3$　(3,1)(7,4)(15,11),,,(255,247) Same as Cyclic Hamming code
2 Error correction code: $d_{min}=5$　(15,7)(31,21),,,(255,239)
3 Error correction code: $d_{min}=7$　(15,5),,,(255,231)
4 Error correction code: $d_{min}=9$　(63,39),,,(255,223)

（３）BCH 符号の構成方法
(3) BCH code configuration procedure

３－１　最小多項式による方法
3-1 Method by minimal polynomials

・α^iを根とする多項式の内、最小の次数を持つものを $M_i(x)$とする。
・最小距離をd_{min}とする場合、生成多項式$g(x)$を以下の積の形とする。

$$g(x)=LCM\left(M_l(x),M_{l+1}(x),\cdots,M_{l+d_{min}-2}(x)\right)$$　（式 7.1.3.2）

ここで、LCM は、最小公倍多項式を意味する。すなわち $g(x)$は$M_l(x)$～$M_{l+2t-2}(x)$で割り切れる最小の次数をもつ多項式である。

- Of the polynomials whose root is α ^ i, let us M_i(x) be one with the smallest degree.
- To obtain the minimum distance of d_min, we configure the generator polynomial g (x) be as follows.
g (x) = LCM (M_l (x), M_ (l + 1) (x), …, M_ (l + d_min-2) (x)) (Equation 7.1.3.2), where LCM means least common multiple polynomial.
That is, g (x) is a polynomial with the smallest degree divisible by M_l (x) to M_ (l + d_min-2) (x).

３－２　べきが連続した根の要素の積による方法
3-2 Method by product of continuous root elements

式 7.1.3.1 に従って、まず、生成多項式をべきが必要な数連続するように構成する。例えば、3 誤り訂正可能な符号としたい場合は、べきの初期値 l を１として、$\alpha,\alpha^2,\alpha^3,\alpha^4,\alpha^5,\alpha^6$ を根として持つ多項式を作る。

$$a(x)=(x+\alpha)(x+\alpha^2)(x+\alpha^3)(x+\alpha^4)(x+\alpha^5)(x+\alpha^6)\cdot$$
$$(x+\alpha)(x+\alpha^2)(x+\alpha^4)(x+\alpha^8)(x+\alpha^{16})(x+\alpha^{32})\cdots$$
$$(x+\alpha^3)(x+\alpha^6)(x+\alpha^{12})(x+\alpha^{24})(x+\alpha^{48})\cdots$$
$$(x+\alpha^5)(x+\alpha^{10})(x+\alpha^{20})(x+\alpha^{40})(x+\alpha^{80})\cdots$$

２段目以降は、それぞれ、α,α^3,α^5 の２乗に対応する項であり、$\alpha^{n+1}=\alpha^1$, $\alpha^{3(n+1)}=\alpha^3$, $\alpha^{5(n+1)}=\alpha^5$ となるまで項をかけている。また、α^2,α^4などの偶数に対応する項がないのは、それ以前の奇数のべき乗の部分に含まれているからである。

最後に、$a(x)$ から重複する項（この例では、$(x+\alpha)$や$(x+\alpha^2)$ など）を削除する。これで最小次元の生成多項式$g(x)$が得られる。２－１で LCM を使っているのは、$g(x)$ に重複する項が含まれないようにするためである。

According to Equation 7.1.3.1, first, the generator polynomial is built so as to include the required number of continuous powers.
For example, if you want to make a code that can correct 3 errors, create a polynomial with α, α ^ 2, α ^ 3, α ^ 4, α ^ 5, α ^ 6 as its root, with the initial value l as 1.
a (x) = (x + α) (x + α ^ 2) (x + α ^ 3) (x + α ^ 4) (x + α ^ 5) (x + α ^ 6) ·
(x + α) (x + α ^ 2) (x + α ^ 4) (x + α ^ 8) (x + α ^ 16) (x + α ^ 32) ····
(x + α ^ 3) (x + α ^ 6) (x + α ^ 12) (x + α ^ 24) (x + α ^ 48) ····

(x + α ^ 5) (x + α ^ 10) (x + α ^ 20) (x + α ^ 40) (x + α ^ 80) ····
The second and subsequent lines are terms corresponding to the squares of α, α ^ 3, and α ^ 5, respectively.
The terms are continued until they reach α ^ (n + 1) = α ^ 1, α ^ (3 (n + 1)) = α ^ 3, α ^ (5 (n + 1)) = α ^ 5.
Note that there are no terms corresponding to even numbers such as α ^ 2 and α ^ 4 since they are included in the odd power part before that.

Finally, we delete the duplicated terms (in this example, (x + α), (x + α ^ 2), etc.) from a (x). This gives the minimum dimension generator polynomial g (x).
Fot this reason, LCM is used in 3-1 to eliminate duplicate terms.

（４）構成例
(4) Configuration example

原始多項式x^4+x+1（この根をα）を用いて、２誤り訂正能力を持つ２元(15, 7)BCH を構成する。
１）生成多項式$g(x)$の構成
式 7.1.3.2 で $l=1$とする。最小距離 5 とすれば、$l+d_{min}-2=1+5-2=4$より、$g(x)=LCM(M_1(x),M_2(x),M_3(x),M_4(x))$である。

$M_1(x)$, $M_2(x)$, $M_3(x)$, $M_4(x)$を議論する。
$M_1(x)$: αを根とする最小多項式－＞原始多項式そのもの　x^4+x+1
$M_2(x)$: α^2を根とする最小多項式－＞α^2も原始多項式の根。よって原始多項式そのもの（※１）
$M_3(x)$: α^3を根とする最小多項式－＞$x^4+x^3+x^2+x+1$　（※２）
$M_4(x)$: α^4を根とする最小多項式－＞$M_2(x)$と同様に原始多項式そのもの
従って、
$g(x)=LCM\big(M_1(x),M_2(x),M_3(x)\big)=LCM\big(M_1(x),M_3(x)\big)=(x^4+x+1)(x^4+x^3+x^2+x+1)$

Using primitive polynomial x ^ 4 + x + 1 (whose root is α), we generate a binary (15, 7) BCH with two error correction capabilities.
1) Configuration of the generator polynomial g (x)
Let l = 1 in Equation 7.1.3.2.

To enable the minimum distance is 5, since l + dmin-2 = 1 + 5-2 = 4, g (x) = LCM (M_1 (x), M_2 (x), M_3 (x), M_4 (x)).
Firstly, we find M_1 (x), M_2 (x), M_3 (x), and M_4 (x).
M_1 (x): Minimal polynomial whose root is α-> Equals to the primitive polynomial x ^ 4 + x + 1
M_2 (x): Minimal polynomial whose root is α ^ 2-> α ^ 2 is also the root of the primitive polynomial. Therefore, the primitive polynomial itself (See * 1)
M_3 (x): Minimal polynomial whose root is α ^ 3-> x ^ 4 + x ^ 3 + x ^ 2 + x + 1 (See * 2)
M_4 (x): Minimal polynomial whose root is α ^ 4-> Same to M_2(x).

Therefore,
g (x) = LCM (M_1 (x), M_2 (x), M_3 (x), M_4(x)) = LCM (M_1 (x), M_3 (x)) = (x ^ 4 + x + 1) (x ^ 4 + x ^ 3 + x ^ 2 + x + 1)

２）各符号語は、$g(x)$を用いて構成する。
$g(x)$は、$\alpha,\alpha^2,\alpha^3,\alpha^4$を根に持つ。$g(x)$が$x^{15}+1$を整除し、また、$g(x)$の次元が８次であることから、２誤り訂正可能な(15,7)符号を構成する。
---
2) Each codeword is constructed using g (x).

g (x) has α, α ^ 2, α ^ 3, and α ^ 4 as roots. Since g (x) divides x ^ 15 + 1, and the dimension of g (x) is eighth, it generates (15, 7) code that can correct two errors.

※１　$M_2(x)=M_1(x)=x^4+x+1$　の証明（$M_1(x)$がα^2を根に持つことの証明）
$M_1(\alpha^2)=\alpha^8+\alpha^2+1=(\alpha+1)^2+\alpha^2+1=\alpha^2+1+\alpha^2+1=0$　　$\because \alpha^4+\alpha+1=0$
一般に、$M_{2j}(x)$は、$M_j(x)$と同一である。

*1 Proof of M_2 (x) = M_1 (x) = x ^ 4 + x + 1 (proof that M_1 (x) has α ^ 2 as its root)
M_1 (α ^ 2) = α ^ 8 + α ^ 2 + 1 = (α + 1) ^ 2 + α ^ 2 + 1 = α ^ 2 + 1 + α ^ 2 + 1 = 0 , since α ^ 4 + α + 1 = 0
Generally, M_2j (x) is identical to M_j (x).

※２　$M_3(x)$の導出
*2 Derivation of M_3 (x)

$M_3(x)$　はα^3を根に持つ。従って、α^6、α^{12}、$\alpha^{24}=\alpha^9$、（$\alpha^{48}=\alpha^3$）も根に持つため、以下となる。

$$M_3(x)=(x+\alpha^3)(x+\alpha^6)(x+\alpha^9)(x+\alpha^{12})=x^4+x^3+x^2+x+1$$

M_3 (x) has α ^ 3 as its root.
Therefore, since α ^ 6, α ^ 12, α ^ 24 = α ^ 9, and (α ^ 48 = α ^ 3) are also roots.
Then,
M_3 (x) = (x + α ^ 3) (x + α ^ 6) (x + α ^ 9) (x + α ^ 12) = x ^ 4 + x ^ 3 + x ^ 2 + x + 1

Another metod

$M_3(x)=a_4x^4+a_3x^3+a_2x^2+a_1x^1+1$とする。（できるだけ次数の少ない多項式とする）
この多項式がα^3を根に持つように、a_4～a_1を定めればよい。つまり、
$M_3(\alpha^3)=a_4\alpha^{12}+a_3\alpha^9+a_2\alpha^6+a_1\alpha^3+1=0$
となるa_4～a_1を求める。ここで、$\alpha^4+\alpha+1=0$を用いると、

*2 Derivation of M_3 (x)
M_3 (x) = a_4 x ^ 4 + a_3 x ^ 3 + a_2 x ^ 2 + a_1 x ^ 1 + 1.
(Use a polynomial with as few degrees as possible)
a_4 to a_1 should be defined so that this polynomial has α ^ 3 as its root.
Then, a_4 to a_1 should satisfy,
M_3 (α ^ 3) = a_4 α ^ 12 + a_3 α ^ 9 + a_2 α ^ 6 + a_1 α ^ 3 + 1 = 0
Then, by using α ^ 4 + α + 1 = 0, we obtain the followings.

$\alpha^{12}=(\alpha+1)^3=\alpha^3+\alpha^2+\alpha^1+1$
$\alpha^9=(\alpha+1)^2\alpha=(\alpha^2+1)\alpha=\alpha^3+\alpha$
$\alpha^6=(\alpha+1)\alpha^2=\alpha^3+\alpha^2$

従って、
Then,

$a_4(\alpha^3+\alpha^2+\alpha+1)+a_3(\alpha^3+\alpha)+a_2(\alpha^3+\alpha^2)+a_1\alpha^3+1=0$
$(a_4+a_3+a_2+a_1)\alpha^3+(a_4+a_2)\alpha^2+(a_4+a_3)\alpha+(a_4+1)=0$
ここから、$a_4=1,a_3=1,a_2=1,a_1=1$　を得る。
Finally, we obtain $a_4=1,a_3=1,a_2=1,a_1=1$.

（５）BCH 符号の検査行列（発展）
(5) BCH code check matrix

符号語多項式$u(x)$は生成多項式$g(x)$で整除される。すなわち、$u(x)=X(x)g(x)$である。　従って、$g(x)$の根をαとすると、$u(\alpha)=X(\alpha)g(\alpha)=0$　であり、これを利用して検査行列を作ることができる。受信語を$v=(a_{n-1}\quad a_{n-2}\quad\cdots\quad a_2\quad a_1\quad a_0)$、多項式表現を$v(x)=a_{n-1}x^{n-1}+a_{n-2}x^{n-2}+\cdots+a_2x^2+a_1x+a_0$とすると、以下を確認する。

The codeword polynomial u (x) is divided by the generator polynomial g (x). That is, u (x) = X (x) g (x). Therefore, if the root of g (x) is α, u (α) = X (α) g (α) = 0, and this can be used to create the corresponding check matrix. If the received word is v = (a_ (n-1) & a_ (n-2) & … & a_2 & a_1 & a_0), and its polynomial expression is v (x) = a_ (n-1) x ^ (n-1) + a_ (n-2)x ^ (n-2) + … + a_2 x ^ 2 + a_1 x + a_0, then the following is calculated.

$$v(\alpha)=a_{n-1}\alpha^{n-1}+a_{n-2}\alpha^{n-2}+\cdots+a_2\alpha^2+a_1\alpha+a_0=0$$

これは以下の式と同等である。
This is equivalent to the following equation.

$$vH^T=(a_{n-1}\quad a_{n-2}\quad\cdots\quad a_2\quad a_1\quad a_0)\begin{pmatrix}\alpha^{n-1}\\ \alpha^{n-2}\\ \vdots\\ \alpha^2\\ \alpha^1\\ 1\end{pmatrix}=0$$

従って、$H=(\alpha^{n-1}\quad\alpha^{n-2}\quad\cdots\quad\alpha^2\quad\alpha^1\quad 1)$である。その他に根があれば、それも列挙することとなる。

Therefore, H = (α ^ (n-1) & α ^ (n-2) & … & α ^ 2 & α ^ 1 & 1). If the generator polynomial has other roots, they are also listed.

例１　$GF(2^3)$上の原始多項式$p(x)=x^3+x+1$の場合。原始根をαとすれば、$\alpha^3+\alpha+1=0$。生成多項式がαを根として持つ場合の符号化規則（検査規則）は、$u(\alpha)=a_6\alpha^6+a_5\alpha^5+a_4\alpha^4+a_3\alpha^3+a_2\alpha^2+a_1\alpha+a_0=0$である。$\alpha^6$などを２元で展開すれば、検査行列を以下のように書ける。（７．１．２　３次の例を参照）

Example 1

Let us assume the primitive polynomial p (x) = x ^ 3 + x + 1 on GF (2 ^ 3).
If its primitive root is α, then α ^ 3 + α + 1 = 0.
For the case when the generate polynomial has α as the root, the coding rule (checking rule) is u (α) = a_6 α ^ 6 + a_5 α ^ 5 + a_4 α ^ 4 + a_3 α ^ 3 + a_2 α ^ 2 + a_1 α + a_0 = 0.
By replacing α ^ 6 by its binary vector expression, the check matrix can be written as follows. (See the example of a cubic primitive polynomial in 7.1.2)

$$H=(\alpha^6\quad\alpha^5\quad\alpha^4\quad\alpha^3\quad\alpha^2\quad\alpha^1\quad 1)=\begin{pmatrix}1&1&1&0&1&0&0\\0&1&1&1&0&1&0\\1&1&0&1&0&0&1\end{pmatrix}$$

例２　$GF(2^3)$上の原始多項式$p(x)=x^3+x^2+1$の場合。原始根をαとすれば、$\alpha^3+\alpha^2+1=0$。
①生成多項式がαを根として持つ場合の符号化規則（検査規則）は、$u(\alpha)=a_6\alpha^6+a_5\alpha^5+a_4\alpha^4+a_3\alpha^3+a_2\alpha^2+a_1\alpha+a_0=0$である。$\alpha^6$などを２元で展開すれば、検査行列を以下のように書ける。

Example 2

Let us assume the primitive polynomial p (x) = x ^ 3 + x ^ 2 + 1 on GF (2 ^ 3).
For the case when the generator polynomial has α as the root, the coding rule (checking rule) is u (α) = a_6 α ^ 6 + a_5 α ^ 5 + a_4 α ^ 4 + a_3 α ^ 3 + a_2 α ^ 2 + a_1 α + a_0 = 0.
By replacing α ^ 6 by its binary vector expression, the check matrix can be written as follows.

$$H=(\alpha^6\quad\alpha^5\quad\alpha^4\quad\alpha^3\quad\alpha^2\quad\alpha^1\quad 1)=\begin{pmatrix}1&0&1&1&1&0&0\\1&1&1&0&0&1&0\\0&1&1&1&0&0&1\end{pmatrix}$$

②生成多項式がα, α^3を根として持つ場合は、以下となる。

(2) If the generator polynomial has α and α ^ 3 as roots, it means the follows.

$$u(\alpha) = a_6\alpha^6 + a_5\alpha^5 + a_4\alpha^4 + a_3\alpha^3 + a_2\alpha^2 + a_1\alpha + a_0 = 0$$
$$u(\alpha^3) = a_6\alpha^{18} + a_5\alpha^{15} + a_4\alpha^{12} + a_3\alpha^9 + a_2\alpha^6 + a_1\alpha^3 + a_0 = 0$$

α^6などを２元で展開すれば、検査行列を以下のように書ける。

By replacing α ^ 6 by its binary vector expression, the check matrix can be written as follows.

$$H = \begin{pmatrix} \alpha^6 & \alpha^5 & \alpha^4 & \alpha^3 & \alpha^2 & \alpha^1 & 1 \\ \alpha^{18} & \alpha^{15} & \alpha^{12} & \alpha^9 & \alpha^6 & \alpha^3 & 1 \end{pmatrix} = \begin{pmatrix} \alpha^6 & \alpha^5 & \alpha^4 & \alpha^3 & \alpha^2 & \alpha^1 & 1 \\ \alpha^4 & \alpha^1 & \alpha^5 & \alpha^2 & \alpha^6 & \alpha^3 & 1 \end{pmatrix}$$
$$= \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 \end{pmatrix}$$

例３　$GF(2^4)$上の原始多項式$p(x) = x^4 + x + 1$の場合。原始根をαとすれば、$\alpha^4 + \alpha + 1 = 0$。
①生成多項式がαを根として持つ場合（$\alpha, \alpha^2, \alpha^4, \alpha^8$を根に持つ）の符号化規則（検査規則）は、
$u(\alpha) = a_{14}\alpha^{14} + a_{13}\alpha^{13} + a_{12}\alpha^{12} + a_{11}\alpha^{11} + a_{10}\alpha^{10} + a_9\alpha^9 + a_8\alpha^8 + a_7\alpha^7 + a_6\alpha^6 + a_5\alpha^5 + a_4\alpha^4 + a_3\alpha^3 + a_2\alpha^2 + a_1\alpha + a_0$=0
であるので、検査行列は以下となる。

Example 3 When the primitive polynomial p (x) = x ^ 4 + x + 1 on GF (2 ^ 4). If the primitive root is α, then α ^ 4 + α + 1 = 0.

(1) When the generated polynomial has α as the root (also having α, α ^ 2, α ^ 4, α ^ 8 as the roots), the coding rule (checking rule) is

u（α）= a_14 α ^ 14 + a_13 α ^ 13 + a_12 α ^ 12 + a_11 α ^ 11 + a_10 α ^ 10 + a_9 α ^ 9 + a_8 α ^ 8 + a_7 α ^ 7 + a_6 α ^ 6 + a_5 α ^ 5 + a_4 α ^ 4 + a_3 α ^ 3 + a_2 α ^ 2 + a_1 α + a_0 = 0

Therefore, the check matrix is as follows.

$$H = (\alpha^{14} \quad \alpha^{13} \quad \alpha^{12} \quad \alpha^{11} \quad \alpha^{10} \quad \alpha^{9} \quad \alpha^{8} \quad \alpha^{7} \quad \alpha^{6} \quad \alpha^{5} \quad \alpha^{4} \quad \alpha^{3} \quad \alpha^{2} \quad \alpha^{1} \quad 1)$$
$$= \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

αとα^3を根として持つ場合（$\alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^6, \alpha^8, \alpha^9, \alpha^{12}$も根に持つ）の検査行列は、以下である。

(2) When α and α ^ 3 are rooted (α, α ^ 2, α ^ 3, α ^ 4, α ^ 6, α ^ 8, α ^ 9, α ^ 12 are also roots), The check matrix is as follows.

$u(\alpha) = a_{14}\alpha^{14} + a_{13}\alpha^{13} + a_{12}\alpha^{12} + a_{11}\alpha^{11} + a_{10}\alpha^{10} + a_9\alpha^9 + a_8\alpha^8 + a_7\alpha^7 + a_6\alpha^6 + a_5\alpha^5 + a_4\alpha^4 + a_3\alpha^3 + a_2\alpha^2 + a_1\alpha + a_0$=0

$u(\alpha^3) = a_{14}\alpha^{42} + a_{13}\alpha^{39} + a_{12}\alpha^{36} + a_{11}\alpha^{33} + a_{10}\alpha^{30} + a_9\alpha^{27} + a_8\alpha^{24} + a_7\alpha^{21} + a_6\alpha^{18} + a_5\alpha^{15} + a_4\alpha^{12} + a_3\alpha^9 + a_2\alpha^6 + a_1\alpha^3 + a_0$

$=a_{14}\alpha^{12} + a_{13}\alpha^9 + a_{12}\alpha^6 + a_{11}\alpha^3 + a_{10}\alpha^0 + a_9\alpha^{12} + a_8\alpha^9 + a_7\alpha^6 + a_6\alpha^3 + a_5\alpha^0 + a_4\alpha^{12} + a_3\alpha^9 + a_2\alpha^6 + a_1\alpha^3 + a_0$=0

$$H = \begin{pmatrix} \alpha^{14} & \alpha^{13} & \alpha^{12} & \alpha^{11} & \alpha^{10} & \alpha^{9} & \alpha^{8} & \alpha^{7} & \alpha^{6} & \alpha^{5} & \alpha^{4} & \alpha^{3} & \alpha^{2} & \alpha^{1} & 1 \\ \alpha^{42} & \alpha^{39} & \alpha^{36} & \alpha^{33} & \alpha^{30} & \alpha^{27} & \alpha^{24} & \alpha^{21} & \alpha^{18} & \alpha^{15} & \alpha^{12} & \alpha^{9} & \alpha^{6} & \alpha^{3} & 1 \end{pmatrix}$$
$$= \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

（６）最小多項式（発展）

(6) Minimal polynomial

BCH を構成する際には最小多項式を用いる。

The minimal polynomial is used when constructing BCH.

［最小多項式］

$GF(p^m)$上のm次の原始多項式を$p(x)$とする。この根（原始根）をαとする。すなわち、$p(\alpha)$=0 である。$q(\beta) = 0\ (\beta \in GF(p^m))$　を満たす$GF(p)$上の多項式の中で次数が最小のもの$q(x)$を$GF(p)$上のβに対する最小多項式と呼ぶ。

[Minimal polynomial]

Let p (x) be a primitive polynomial of order m on GF (p ^ m).
Let α be its root (primitive root). That is, p (α) = 0. Among the polynomials on GF (p) that satisfy q (β) = 0 ($\beta \in$ GF (p ^ m)), the polynomial with the smallest degree q (x) is called the minimal polynomial for β on GF (p).

例１　$GF(2^3)$上の原始多項式$p(x) = x^3 + x + 1$を考える。周期７。原始根をαとすれば、$\alpha^3 + \alpha + 1 = 0$、$\alpha^7 = 1$。ここで、$q(\beta) = 0\ (\beta \in GF(2^3))$を満たす$GF(2)$上の最小多項式を求める。
$GF(2^3) = \{0, 1, \alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6\}$であるから、それぞれが$\beta$の候補となり得る。

Example 1 Consider the primitive polynomial p (x) = x ^ 3 + x + 1 on GF (2 ^ 3). Its cycle is 7. If the primitive root is α, then α ^ 3 + α + 1 = 0, and α ^ 7 = 1. Here, let us find the minimal polynomial on GF (2) that satisfies q (β) = 0 ($\beta \in$ GF (2 ^ 3)).
Since GF (2 ^ 3) = {0, 1, α, α ^ 2, α ^ 3, α ^ 4, α ^ 5, α ^ 6}, each element can be a candidate for β.

$\beta = 0: q(x)|_{x=0} = 0$　　$q(x) = x$
$\beta = 1: q(x)|_{x=1} = 0$　　$q(x) = x + 1$
$\beta = \alpha: q(x)|_{x=\alpha} = 0$　　$q(x) = x^3 + x + 1 = p(x)$
$\beta = \alpha^2: q(x)|_{x=\alpha^2} = 0$　　$q(x) = x^3 + x + 1 = p(x)$　　※2
$\beta = \alpha^3: q(x)|_{x=\alpha^3} = 0$　　$q(x) = x^3 + x^2 + 1$　　※3
$\beta = \alpha^4: q(x)|_{x=\alpha^4} = 0$　　$q(x) = x^3 + x + 1 = p(x)$　　※4
$\beta = \alpha^5: q(x)|_{x=\alpha^5} = 0$　　$q(x) = x^3 + x^2 + 1$　　※5
$\beta = \alpha^6: q(x)|_{x=\alpha^6} = 0$　　$q(x) = x^3 + x^2 + 1$　　※6
※2　$\alpha^6 + \alpha^2 + 1 = (\alpha + 1)^2 + \alpha^2 + 1 = 0$
※3　$\alpha^9 + \alpha^6 + 1 = \alpha^2 + \alpha^2 + 1 + 1 = 0$
※4　$\alpha^{12} + \alpha^4 + 1 = \alpha^5 + \alpha^4 + 1 = \alpha^2(\alpha + 1) + \alpha^4 + 1 = \alpha^3 + \alpha^2 + \alpha^2 + \alpha + 1 = 0$
※5　$\alpha^{15} + \alpha^{10} + 1 = \alpha^1 + \alpha^3 + 1 = 0$
※6　$\alpha^{18} + \alpha^{12} + 1 = \alpha^4 + \alpha^5 + 1 = 0$

例２　$GF(2^3)$上の原始多項式$p(x) = x^3 + x^2 + 1$を考える。周期７。原始根をαとすれば、$\alpha^3 + \alpha^2 + 1 = 0$。ここで、$q(\beta) = 0\ (\beta \in GF(2^3))$を満たす$GF(2)$上の最小多項式を求める。
$GF(2^3) = \{0, 1, \alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6\}$であるから、それぞれが$\beta$の候補となり得る。

Example 2 Consider the primitive polynomial p (x) = x ^ 3 + x ^ 2 + 1 on GF (2 ^ 3). Its cycle is 7. If the primitive root is α, then α ^ 3 + α ^ 2 + 1 = 0. Here, let us find the minimal polynomial on GF (2) that satisfies q (β) = 0 (β $\in$ GF (2 ^ 3)).
Since GF (2 ^ 3) = {0, 1, α, α ^ 2, α ^ 3, α ^ 4, α ^ 5, α ^ 6}, each can be a candidate for β.

$\beta = 0: q(x)|_{x=0} = 0$　　$q(x) = x$
$\beta = 1: q(x)|_{x=1} = 0$　　$q(x) = x + 1$
$\beta = \alpha: q(x)|_{x=\alpha} = 0$　　$q(x) = x^3 + x^2 + 1 = p(x)$
$\beta = \alpha^2: q(x)|_{x=\alpha^2} = 0$　　$q(x) = x^3 + x^2 + 1 = p(x)$　　※2
$\beta = \alpha^3: q(x)|_{x=\alpha^3} = 0$　　$q(x) = x^3 + x + 1$　　※3
$\beta = \alpha^4: q(x)|_{x=\alpha^4} = 0$　　$q(x) = x^3 + x^2 + 1 = p(x)$　　※4
$\beta = \alpha^5: q(x)|_{x=\alpha^5} = 0$　　$q(x) = x^3 + x + 1$　　※5
$\beta = \alpha^6: q(x)|_{x=\alpha^6} = 0$　　$q(x) = x^3 + x + 1$　　※6
※2　$\alpha^6 + \alpha^4 + 1 = (\alpha^2 + 1)^2 + \alpha^4 + 1 = \alpha^4 + 1 + \alpha^4 + 1 = 0$
※3　$\alpha^9 + \alpha^3 + 1 = \alpha^2 + \alpha^3 + 1 = 0$
※4　$\alpha^{12} + \alpha^8 + 1 = \alpha^5 + \alpha^1 + 1 = \alpha^2(\alpha^2 + 1) + \alpha^1 + 1 = \alpha^4 + \alpha^2 + \alpha^1 + 1 = \alpha(\alpha^2 + 1) + \alpha^2 + \alpha + 1 = 0$
※5　$\alpha^{15} + \alpha^5 + 1 = \alpha^1 + \alpha^5 + 1 = 0$
※6　$\alpha^{18} + \alpha^6 + 1 = \alpha^4 + \alpha^6 + 1 = 0$

例３　４次の例
例　$GF(2^4)$上の原始多項式$p(x) = x^4 + x + 1$を考える。周期１５。原始根をαとすれば、$\alpha^4 + \alpha + 1 = 0$、$\alpha^{15} = 1$。ここで、$q(\beta) = 0\ (\beta \in GF(2^4))$を満たす$GF(2)$上の最小多項式を求める。
$GF(2^4) = \{0, 1, \alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6, \alpha^7, \alpha^8, \alpha^9, \alpha^{10}, \alpha^{11}, \alpha^{12}, \alpha^{13}, \alpha^{14}\}$であるから、それぞれが$\beta$の候補となり得る。（７．１．２　４次の例を参照）
Example 3 4th-order primitive polynomial
Example: Consider the primitive polynomial p (x) = x ^ 4 + x + 1 on GF (2 ^ 4). Its cycle is 15. If the primitive root is α, then α ^ 4 + α + 1 = 0,

α ^ 15 = 1. Here, let us find the minimal polynomial on GF (2) that satisfies q (β) = 0 (β ∈ GF (2 ^ 4)).
Since GF (2 ^ 4) = {0,1, α, α ^ 2, α ^ 3, α ^ 4, α ^ 5, α ^ 6, α ^ 7, α ^ 8, α ^ 9, α ^ 10, α ^ 11, α ^ 12, α ^ 13, α ^ 14}, each can be a candidate for β. (See 7.1.2. 4 Example of a 4th-order primitive polynomial)

$\beta = 0: q(x)|_{x=0} = 0$ $q(x) = x$
$\beta = 1: q(x)|_{x=1} = 0$ $q(x) = x + 1$
$\beta = \alpha: q(x)|_{x=\alpha} = 0$ $q(x) = x^4 + x + 1 = p(x)$ ※1
$\beta = \alpha^2: q(x)|_{x=\alpha^2} = 0$ $q(x) = x^4 + x + 1 = p(x)$ ※2
$\beta = \alpha^3: q(x)|_{x=\alpha^3} = 0$ $q(x) = x^4 + x^3 + x^2 + x + 1$ ※3
$\beta = \alpha^4: q(x)|_{x=\alpha^4} = 0$ $q(x) = x^4 + x + 1 = p(x)$ ※4
$\beta = \alpha^5: q(x)|_{x=\alpha^5} = 0$ $q(x) = x^2 + x + 1$ ※5
$\beta = \alpha^6: q(x)|_{x=\alpha^6} = 0$ $q(x) = x^4 + x^3 + x^2 + x + 1$ ※6
$\beta = \alpha^7: q(x)|_{x=\alpha^7} = 0$ $q(x) = x^4 + x^3 + 1$ ※7
$\beta = \alpha^8: q(x)|_{x=\alpha^8} = 0$ $q(x) = x^4 + x + 1 = p(x)$ ※8
$\beta = \alpha^9: q(x)|_{x=\alpha^9} = 0$ $q(x) = x^4 + x^3 + x^2 + x + 1$ ※9
$\beta = \alpha^{10}: q(x)|_{x=\alpha^{10}} = 0$ $q(x) = x^2 + x + 1$ ※10
$\beta = \alpha^{11}: q(x)|_{x=\alpha^{11}} = 0$ $q(x) = x^4 + x^3 + 1$ ※11
$\beta = \alpha^{12}: q(x)|_{x=\alpha^{12}} = 0$ $q(x) = x^4 + x^3 + x^2 + x + 1$ ※12
$\beta = \alpha^{13}: q(x)|_{x=\alpha^{13}} = 0$ $q(x) = x^4 + x^3 + 1$ ※13
$\beta = \alpha^{14}: q(x)|_{x=\alpha^{14}} = 0$ $q(x) = x^4 + x^3 + 1$ ※14
※2 $\alpha^8 + \alpha^2 + 1 = (\alpha + 1)^2 + \alpha^2 + 1 = \alpha^2 + 1 + \alpha^2 + 1 = 0$ $(0101) + (0100) + (0001) = 0$
※3 $\alpha^{12} + \alpha^9 + \alpha^6 + \alpha^3 + 1 = (1111) + (1010) + (1100) + (1000) + (0001) = 0$
※4 $\alpha^{16} + \alpha^4 + 1 = \alpha^1 + \alpha^4 + 1 = 0$
※5 $\alpha^{10} + \alpha^5 + 1 = (0111) + (0110) + (0001) = 0$
※6 $\alpha^{24} + \alpha^{18} + \alpha^{12} + \alpha^6 + 1 = \alpha^9 + \alpha^3 + \alpha^{12} + \alpha^6 + 1 = (1010) + (1000) + (1111) + (1100) + (0001) = 0$
※7 $\alpha^{28} + \alpha^{21} + 1 = \alpha^{13} + \alpha^6 + 1 = (1101) + (1100) + (0001) = 0$
※8 $\alpha^{32} + \alpha^8 + 1 = \alpha^2 + \alpha^8 + 1 = (0100) + (0101) + (0001) = 0$
※9 $\alpha^{36} + \alpha^{27} + \alpha^{18} + \alpha^9 + 1 = \alpha^6 + \alpha^{12} + \alpha^3 + \alpha^9 + 1 = 0$
※10 $\alpha^{20} + \alpha^{10} + 1 = \alpha^5 + \alpha^{10} + 1 = 0$
※11 $\alpha^{44} + \alpha^{33} + 1 = \alpha^{14} + \alpha^3 + 1 = (1001) + (1000) + (0001) = 0$
※12 $\alpha^{48} + \alpha^{36} + \alpha^{24} + \alpha^{12} + 1 = \alpha^3 + \alpha^6 + \alpha^9 + \alpha^{12} + 1 = 0$
※13 $\alpha^{52} + \alpha^{39} + 1 = \alpha^7 + \alpha^9 + 1 = (1011) + (1010) + (0001) = 0$
※14 $\alpha^{56} + \alpha^{42} + 1 = \alpha^{11} + \alpha^{12} + 1 = (1110) + (1111) + (0001) = 0$

（７）復号（発展）

BCH符号の復号には、ピーターソン、バーレカンプーマッシー、杉山の方法などがある。以下はピーターソンの方法に沿っている。

まず、第６章６．３．６に示した２誤り訂正可能BCH(15, 7)符号を例に説明する。

(7) Decode BCH code

There are some methods such as Peterson, Barre Kampoo Massy, and Sugiyama for decoding BCH code. The following follows Peterson's method.

The two error-correctable BCH (15, 7) codes shown in Chapter 6, 6.3.6 is an example.

原始多項式　$p(x) = x^4 + x + 1$　この根をαとすれば、$\alpha^4 + \alpha + 1 = 0$
生成多項式　$g(x) = (x^4 + x + 1)(x^4 + x^3 + x^2 + x + 1)$

Primitive polynomial p (x) = x ^ 4 + x + 1 If this root is α, then α ^ 4 + α + 1 = 0
Generator polynomial g (x) = (x ^ 4 + x + 1) (x ^ 4 + x ^ 3 + x ^ 2 + x + 1)

送信符号語$u(x)$を送信し、iビット目とjビット目が誤って、$v(x)$を受信したとする。
2ビットの誤りは　$e(x) = x^i + x^j$　と表現できる。
受信符号語を$v(x)$とし、シンドロームS_1、S_3を以下のように計算する。
$S_1 = v(\alpha) = u(\alpha) + e(\alpha) = e(\alpha) = \alpha^i + \alpha^j$　　・・・（式１）
$S_3 = v(\alpha^3) = u(\alpha^3) + \alpha^{3i} + \alpha^{3j} = \alpha^{3i} + \alpha^{3j}$
これらより、$\alpha^i \alpha^j = \alpha^{(i+j)} = \frac{S_3}{S_1} + {S_1}^2$　　・・・（式２）
式１，式２を連立して、i,jを求める。

Suppose that a codeword u (x) is transmitted and v (x) is received where the i-th and j-th bits get flipped to be errors.
The 2-bit error can be expressed as e (x) = x ^ i + x ^ j.
The syndromes S_1 and S_3 are calculated as follows.
S_1 = v (α) = u (α) + e (α) = e (α) = α ^ i + α ^ j ・ ・ ・ (Equation 1)
S_3 = v (α ^ 3) = u (α ^ 3) + α ^ 3i + α ^ 3j = α ^ 3i + α ^ 3j
From these, α ^ i α ^ j = α ^ ((i + j)) = S_3 / S_1 + 〚S_1〛 ^ 2 ・ ・ ・ (Equation 2)
Equations 1 and 2 are combined to obtain i and j.

今、$v(x) = (111000011110010) = x^{14} + x^{13} + x^{12} + x^7 + x^6 + x^5 + x^4 + x$を受信したとする。シンドロームを計算する。（７．１．２　４次の例を参照）

For example, suppose that v (x) = (111000011110010) = x ^ 14 + x ^ 13 + x ^ 12 + x ^ 7 + x ^ 6 + x ^ 5 + x ^ 4 + x is received. Calculate the syndromes. (Also see 7.1.2. Example of a 4-th order primitive polynomial)

$S_1 = v(\alpha) = \alpha^{14} + \alpha^{13} + \alpha^{12} + \alpha^7 + \alpha^6 + \alpha^5 + \alpha^4 + \alpha = (1001) + (1101) + (1111) + (1011) + (1100) + (0110) + (0011) + (0010) = (1011) = \alpha^7$

$S_3 = v(\alpha^3) = \alpha^{42} + \alpha^{39} + \alpha^{36} + \alpha^{21} + \alpha^{18} + \alpha^{15} + \alpha^{12} + \alpha^3 = \alpha^{12} + \alpha^9 + \alpha^6 + \alpha^6 + \alpha^3 + 1 + \alpha^{12} + \alpha^3 = \alpha^9 + 1 = (1010) + (0001) = (1011) = \alpha^7$

$$\frac{S_3}{S_1} + {S_1}^2 = \frac{\alpha^7}{\alpha^7} + \alpha^{7\cdot 2} = 1 + \alpha^{14} = (0001) + (1001) = (1000) = \alpha^3$$

よって、式１は、$\alpha^i + \alpha^j = \alpha^7$。式２は、$\alpha^i \alpha^j = \alpha^{(i+j)} = \alpha^3$　となり、$i + j = 3$ である。
これを解いて、$i = 5$、$j = 13$　と求められる。（総当たりしてもこの程度ならば容易）

Thus, Equation 1 is α ^ i + α ^ j = α ^ 7. Equation 2 is α ^ i α ^ j = α ^ ((i + j)) = α ^ 3, and i + j = 3.
By solving this, i = 5 and j = 13 are obtained. (As far as this example, the exhaustive search can be possible.)

一般的には、シンドロームと以下の誤り位置方程式を立てて求める。（京大　西田より）

In general, solve the following error position equation.

$$\sigma(z) = (1 - \alpha^{j_1} z)(1 - \alpha^{j_2} z) \cdots (1 - \alpha^{j_l} z)$$

ただし、$0 \le j_1 < j_2 \cdots < j_l$、$l \le t$。ここから$l$本の方程式が立ち$l$個の$j_i$を特定する。
　先ほどの２誤り訂正可能 BCH(15, 7)符号では、２誤り訂正可能であるので、$t = 2$。$j_1 = i$, $j_2 = j$ とすれば、誤り位置方程式は以下となる。

where, 0 ≤ j_1 <j_2 … <j_l, l ≤ t. The l equations enable to identify j_i.

For the 2-error correctable BCH (15, 7) code, t = 2. Let j_1 = i and j_2 = j, the error position equation is as follows.

$$\sigma(z) = (1 - \alpha^i z)(1 - \alpha^j z) = 1 - (\alpha^i + \alpha^j) z + \alpha^{(i+j)} z^2$$

シンドロームS_1、S_3から、誤り位置方程式は、

From syndromes S_1 and S_3, the equation is derived as follows.

$$\sigma(z) = 1 - S_1 z + (\frac{S_3}{S_1} + {S_1}^2)z^2$$

$v(x) = (111000011110010) = x^{14} + x^{13} + x^{12} + x^7 + x^6 + x^5 + x^4 + x$を受信した場合は、以下となる。

When v (x) = (111000011110010) = x ^ 14 + x ^ 13 + x ^ 12 + x ^ 7 + x ^ 6 + x ^ 5 + x ^ 4 + x is received, the error position equation is as follows.

$$\sigma(z) = 1 - \alpha^7 z + \alpha^3 z^2$$

この式のzにαのべき乗を代入すると、以下となる。

Substituting each power of α into z in this equation gives:

$\sigma(1) = 1 - \alpha^7 + \alpha^3 = \alpha$
$\sigma(\alpha) = 1 - \alpha^7\alpha + \alpha^3\alpha^2 = 1 + \alpha^8 + \alpha^5 = \alpha$
$\sigma(\alpha^2) = 1 - \alpha^7\alpha^2 + \alpha^3\alpha^4 = 1 + \alpha^9 + \alpha^7 = (0001) + (1010) + (1011) = 0$
$\sigma(\alpha^3) = 1 - \alpha^7\alpha^3 + \alpha^3\alpha^6 = 1 + \alpha^{10} + \alpha^9 = (0001) + (0111) + (1010) = (1100) = \alpha^6$
$\sigma(\alpha^4) = 1 - \alpha^7\alpha^4 + \alpha^3\alpha^8 = 1 + \alpha^{11} + \alpha^{11} = 1$
$\sigma(\alpha^5) = 1 - \alpha^7\alpha^5 + \alpha^3\alpha^{10} = 1 + \alpha^{12} + \alpha^{13} = (0001) + (1111) + (1101) = (0011) = \alpha^4$
$\sigma(\alpha^6) = 1 - \alpha^7\alpha^6 + \alpha^3\alpha^{12} = 1 + \alpha^{13} + 1 = \alpha^{13}$
$\sigma(\alpha^7) = 1 - \alpha^7\alpha^7 + \alpha^3\alpha^{14} = 1 + \alpha^{14} + \alpha^2 = (0001) + (1001) + (0100) = (1100) = \alpha^6$
$\sigma(\alpha^8) = 1 - \alpha^7\alpha^8 + \alpha^3\alpha^{16} = 1 + 1 + \alpha^4 = \alpha^4$
$\sigma(\alpha^9) = 1 - \alpha^7\alpha^9 + \alpha^3\alpha^{18} = 1 + \alpha^1 + \alpha^6 = (0001) + (0010) + (1100) = (1111) = \alpha^{12}$
$\sigma(\alpha^{10}) = 1 - \alpha^7\alpha^{10} + \alpha^3\alpha^{20} = 1 + \alpha^2 + \alpha^8 = (0001) + (0100) + (0101) = 0$
$\sigma(\alpha^{11}) = 1 - \alpha^7\alpha^{11} + \alpha^3\alpha^{22} = 1 + \alpha^3 + \alpha^{10} = (0001) + (1000) + (0111) = (1110) = \alpha^{11}$
$\sigma(\alpha^{12}) = 1 - \alpha^7\alpha^{12} + \alpha^3\alpha^{24} = 1 + \alpha^4 + \alpha^{12} = (0001) + (0011) + (1111) = (1101) = \alpha^{13}$
$\sigma(\alpha^{13}) = 1 - \alpha^7\alpha^{13} + \alpha^3\alpha^{26} = 1 + \alpha^5 + \alpha^{14} = (0001) + (0110) + (1001) = (1110) = \alpha^{11}$
$\sigma(\alpha^{14}) = 1 - \alpha^7\alpha^{14} + \alpha^3\alpha^{28} = 1 + \alpha^6 + \alpha^1 = (0001) + (1100) + (0010) = (1111) = \alpha^{12}$

ここから、$\sigma(z) = 1 - \alpha^7 z + \alpha^3 z^2 = (1 - \alpha^{-2}z)(1 - \alpha^{-10}z) = (1 - \alpha^{13}z)(1 - \alpha^5 z)$

と分解でき、誤り位置が特定できる。

From here,
σ (z) = 1-α ^ 7 z + α ^ 3 z ^ 2 = (1-α ^ (-2) z) (1-α ^ (-10) z) = (1-α ^ 13 z) (1-α ^ 5 z),
we can specify the error positions.

（８）BCH の最小距離（発展）
(8) Minimum distance of BCH

設計時に設定した最小距離d_{min_d}と実際に作成される符号の最小距離d_{min_a}は、初期値 l に依存し、$d_{min_d} \leq d_{min_a}$となる。これはlによって、最終的にできる生成多項式の連続するべき乗の根の数が変わるためである。

The minimum distance at design d_min_d and the minimum distance of the code finally created d_min_a are sometimes different depending on the initial value l, since l changes the number of roots of continuous power.
(d_min_d <= d_min_a)

最小距離$d_{min} = 3$として設計する場合を考える。原始多項式を$x^4 + x + 1$とする。
生成多項式を以下に示す。

Let us consider the case of designing with the minimum distance d_min = 3.
Let the primitive polynomial be x ^ 4 + x + 1.
The generator polynomial is shown below.

$$g(x) = \left[\prod_{i=l}^{l+d_{min}-2} \left(x - \alpha^i\right) \right] A(x) \quad \text{（式 1）}$$

$$g(x) = LCM\left(M_l(x), M_{l+1}(x), \cdots, M_{l+d_{min}-2}(x)\right) \quad \text{（式 2）}$$

$$l + d_{min} - 2 = l + 3 - 2 = l + 1$$

いくつかのlに対して、式１に入る根と、式２の項、最終的に達成される最小距離を以下に示す。

For some ls, roots in Equation 1, the terms in Equation 2, and the minimum distance finally achieved are shown below.

l	$l + d_{min} - 2$	式１の根	式２の項	最終的な最小距離
l	l + d_min-2	roots of eq.1	terms of eq.2	final min distance
0	1	$1, \alpha, (\alpha^2)$	M_0, M_1	$d_{min} = 4$
1	2	α, α^2	M_1, M_2	$d_{min} = 3$
2	3	$(\alpha), \alpha^2, \alpha^3, (\alpha^4)$	M_2, M_3	$d_{min} = 5$
3	4	$(\alpha), (\alpha^2), \alpha^3, \alpha^4$	M_3, M_4	$d_{min} = 5$

（）は、結果的に加わる根でべきが連続しているものである。

Roots in () show the ones finally added with continuous power.

$l = 2$の場合を詳しく説明する。式１に従って、必要な根の項とそのべきの２乗の項の積を求める。

Let us see the case of l = 2 in detail.

According to Equation 1, the product of the terms with of required roots and the term with the squared roots is as follows.

$$a(x) = (x + \alpha^2)(x + \alpha^3) \cdot$$
$$(x + \alpha^2)(x + \alpha^4)(x + \alpha^8)(x + \alpha^{16}) \cdot$$
$$(x + \alpha^3)(x + \alpha^6)(x + \alpha^{12})(x + \alpha^{24})(x + \alpha^{48})$$

よって、生成多項式は以下となる。

Therefore, we obtain the generator polynomial.

$$g(x) = (x+\alpha)(x+\alpha^2)(x+\alpha^3)(x+\alpha^4)(x+\alpha^6)(x+\alpha^8)(x+\alpha^9)(x+\alpha^{12}) = x^8+x^7+x^6+x^4+1$$

α, α^2, α^3, α^4 が連続しており、最小距離d_{min}は 5 となる。つまり、$d_{min}=3$を目指して設計したが、$d_{min}=5$の符号、すなわち、２誤り訂正可能 BCH(15,7)符号ができる。
式２の$g(x) = LCM\big(M_2(x), M_3(x)\big)$　でも同じ結果が得られる。

α, α ^ 2, α ^ 3, α ^ 4 are continuous, so that, the minimum distance d_min is 5.
Although aiming at d_min = 3, the final d_min = 5, which makes 2-error-correctable BCH (15,7).
The same result can be obtained with g (x) = LCM (M_2 (x), M_3 (x)) (eq.2).

$M_2(x)$: α^2を根とする最小多項式－＞原始多項式$M_1(x)$
$M_3(x)$: α^3を根とする最小多項式－＞$x^4+x^3+x^2+x+1$

M_2 (x): Minimal polynomial whose root is α ^ 2-> Primitive polynomial M_1 (x)
M_3 (x): Minimal polynomial whose root is α ^ 3-> x ^ 4 + x ^ 3 + x ^ 2 + x + 1

$$g(x) = LCM\big(M_2(x), M_3(x)\big) = LCM\big(M_1(x), M_3(x)\big) = (x^4+x+1)(x^4+x^3+x^2+x+1)$$
$$= x^8+x^7+x^6+x^4+1$$

$l=3$の場合は、$g(x) = LCM\big(M_3(x), M_4(x)\big) = LCM\big(M_1(x), M_3(x)\big)$　であり、$l=2$と同一の符号が得られる。
$l=0$の場合は、$g(x) = (x+1)(x+\alpha)(x+\alpha^2)(x+\alpha^4)(x+\alpha^8)$であり、$d_{min}=4$の(15, 10)符号となる。この符号は、$l=1$の$d_{min}=3$の符号の生成多項式に$(x+1)$の項が加わって、パリティビットが付いた符号となる。

In the case of l = 3, g (x) = LCM (M_3 (x), M_4 (x)) = LCM (M_1 (x), M_3 (x)), and the same code as l = 2 can be obtained.
When l = 0, g (x) = (x + 1) (x + α) (x + α ^ 2) (x + α ^ 4) (x + α ^ 8), and d_min = 4.
This is a (15, 10) code, that is added by a parity bit to a code with l=1, d_min=3.
The generator polynomial is also added by a term (x + 1).

まとめると以下となる。
The summary

l	$l+d_{min}-2$	べきが連続する根	LCM の項	最終的な最小距離
l	l + d_min-2	roots with continuous power	LCM terms	final min distance
0	1	$1,\alpha,\alpha^2$	M_0,M_1	$d_{min}=4$
1	2	α,α^2	M_1,M_2	$d_{min}=3$
2	3	$\alpha,\alpha^2,\alpha^3,\alpha^4$	M_2,M_3	$d_{min}=5$
3	4	$\alpha,\alpha^2,\alpha^3,\alpha^4$	M_3,M_4	$d_{min}=5$
4	5	$\alpha,\alpha^2 \mid \alpha^4,\alpha^5$	M_4,M_5	$d_{min}=3$
5	6	$\alpha^5,\alpha^6 \mid \alpha^9,\alpha^{10}$	M_5,M_6	$d_{min}=3$
6	7	$\alpha^{11},\alpha^{12},\alpha^{13},\alpha^{14}$	M_6,M_7	$d_{min}=5$
7	8	$\alpha,\alpha^2 \mid \alpha^7,\alpha^8 \mid \alpha^{13},\alpha^{14}$	M_7,M_8	$d_{min}=3$
8	9	$\alpha,\alpha^2,\alpha^3,\alpha^4$	M_8,M_9	$d_{min}=5$
9	10	$\alpha^5,\alpha^6 \mid \alpha^9,\alpha^{10}$	M_9,M_{10}	$d_{min}=3$
10	11	$\alpha^{10},\alpha^{11} \mid \alpha^{13},\alpha^{14}$	M_{10},M_{11}	$d_{min}=3$
11	12	$\alpha^{11},\alpha^{12},\alpha^{13},\alpha^{14}$	M_{11},M_{12}	$d_{min}=5$
12	13	$\alpha^{11},\alpha^{12},\alpha^{13},\alpha^{14}$	M_{12},M_{13}	$d_{min}=5$
13	14	α^{13},α^{14}	M_{13},M_{14}	$d_{min}=3$
14	15	$\alpha^{13},\alpha^{14},\alpha^{15}=1$	M_{14},M_0	$d_{min}=4$

なお、$LCM\big(M_3(x),M_4(x)\big)=LCM\big(M_1(x),M_3(x)\big)$ から分かるように、最初からべきが連続している必要はない。例えば、$LCM\big(M_4(x),M_6(x)\big)=LCM\big(M_1(x),M_3(x)\big)$ であり、α^4,α^6を根として持つ生成多項式からも、$d_{min}=5$の２誤り訂正可能 BCH(15,7)符号ができる。

Note that as you can see from LCM (M_3 (x), M_4 (x)) = LCM (M_1 (x), M_3 (x)), powers of roots are not requisite to be continuous.
For example, since LCM (M_4 (x), M_6 (x)) = LCM (M_1 (x), M_3 (x)), the generator polynomial whose roots are α ^ 4, α ^ 6 also generates 2 error correctable BCH (15,7) code with dmin=5.

また、生成多項式に原始多項式が含まれるのは、べきが連続する区間に、$\alpha,\alpha^2,\alpha^4,\alpha^8$のいずれかが含まれる場合である。例えば、$l=6$の場合、$\alpha^{11},\alpha^{12},\alpha^{13},\alpha^{14}$が連続する根であり、$\alpha,\alpha^2,\alpha^4,\alpha^8$は含まれていない。従って、生成多項式には、原始多項式は含まれない。

In addition, the primitive polynomial is included in the generated polynomial as long as any of α, α ^ 2, α ^ 4, and α ^ 8 is included in the interval where the powers of roots are continuous. For example, when l = 6, α ^ 11, α ^ 12, α ^ 13, and α ^ 14, are continuous roots, and either α, α ^ 2, α ^ 4, or α ^ 8 is not included. Therefore, the generated polynomial does not include the primitive polynomial.

$$g(x) = LCM\big(M_6(x), M_7(x)\big) = LCM\big(M_3(x), M_7(x)\big) = (x^4 + x^3 + x^2 + x + 1)(x^4 + x^3 + 1)$$

７．２　リードソロモン符号

7.2 Reed-Solomon code

７．２．１　ＲＳ基礎

7.2.1 Introduction to Reed-Solomon code

・1960 年頃に Irving Reed と Gustave Solomon が考案

・巡回符号の一種

・笠原正雄：リード・ソロモン符号の半世紀

https://www.jstage.jst.go.jp/article/essfr/5/1/5_1_28/_pdf

・$GF(2^m)$上で演算を行う。

・多項式の係数にも拡大体を用いる。（情報記号も原始多項式の根αで表現する。）

・情報バイト数k、符号長nバイト、最小距離d_{min}、誤り訂正能力t　の関係

$$k = n - d_{min} + 1 \qquad d_{min} = 2t + 1 \qquad n = k + d_{min} - 1 = k + 2t$$

- Invented by Irving Reed and Gustave Solomon around 1960.
- A type of cyclic code.
- Perform operations on GF (2 ^ m).
- Use an extension field for coefficients of polynomials. All coefficients are expressed by using root α of a primitive polynomial.
- When number of information bytes k, code length n (byte), minimum distance d_min, and error correction capability t, relationship between these numbers is as follows.

$$k = n - d_{min} + 1 \qquad d_{min} = 2t + 1 \qquad n = k + d_{min} - 1 = k + 2t$$

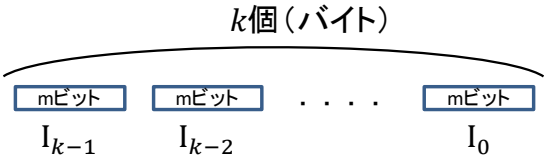
・生成多項式

$$G(y) = \prod_{i=l}^{l+d_{min}-2} (y - \alpha^i) = \prod_{i=l}^{l+2t-1} (y - \alpha^i) \qquad \text{（式 7.2.1）}$$

lは任意の非負整数。普通は 0 または 1 が選ばれる。（今井秀樹著「符号理論」電子情報通信学会 p.156 では、実装上$l = 0$が若干の優位性があるとされている。）

・非２元 BCH 符号　$n = q - 1$の場合とも言える。ただし、qは素数のべき乗である。ex. $2^3 = 8$

・k個の情報記号$I_{k-1}, \cdots, I_1, I_0$を符号化することを考える。ただし、これらの情報記号I_iは$GF(2^m)$の元とする。（各元は、原始多項式の根αのべき乗で表現可能であることに注意）。ここで、$I_i = (c_{m-1}, \cdots, c_1, c_0)$とし、この$m$ビットをバイトと表現する。$m$が 8 以外であっても RS 符号ではバイトと呼ぶ。バイト中に誤りが



生じた場合は、1バイト誤りと言う。（誤りが1ビットでもmビットでも）。RS符号はバイト誤り訂正を効率的に検出訂正する符号であるとも言える。一方、I_kを多進数と捉えることもできる。

以下では、具体的な符号化を主として、生成多項式で除算する方法で説明する。

- Generator polynomial of RS code is ;

$G(y) = \prod_{i=l}^{l+d_{min}-2}(y-\alpha^i) = \prod_{i=l}^{l+2t-1}(y-\alpha^i)$　　Equation 7.2.1)

Where, l is an arbitrary non-negative integer. Usually 0 or 1 is chosen for l.

- Consider encoding k information symbols I_ (k-1), …, I_1, I_0. These information symbols are elements on GF (2 ^ m). (Note that each element can be represented by some power of a root α of a primitive polynomial). That is, I_i = (c_ (m-1), …, c_1, c_0), and this m bit is referred to as a byte. Even if m is other than 8, it is called a byte in RS code. If an error occurs in a byte, it is said to be a 1-byte error. (Whether the error is 1 bit or m bits). RS code efficiently detects and corrects byte errors.

In the following, we see RS coding mainly by the method of dividing by the generator polynomial.

７．２．２　リード・ソロモン符号　具体的な例

7.2.2 Reed-Solomon code; An example

以下では、$n = 2^m - 1$、$l = 0$を考える。

（１）$m = 3$の場合　　$n = 2^m - 1 = 7$　（7バイトに注意）

１－１）

・$GF(2)$の上の原始多項式$p(x)$を$p(x) = x^3 + x + 1$とし、この根をαとする。すなわち$\alpha^3 + \alpha + 1 = 0$。

In the following, consider n = 2 ^ m-1 and l = 0.

(1) When m = 3 n = 2 ^ m-1 = 7 (note 7 bytes)

1-1)

- Let p (x) = x ^ 3 + x + 1 be a primitive polynomial on GF (2), and let the root of p(x) be α. That is, α ^ 3 + α + 1 = 0.

多項式	ベクトル表現	αべき表現	
0	000	0	
1	001	1	$=\alpha^0$
x	010	α	$=\alpha^1$
$x+1$	011	$\alpha+1$	$=\alpha^3$
x^2	100	α^2	$=\alpha^2$
x^2+1	101	α^2+1	$=\alpha^6$
x^2+x	110	$\alpha^2+\alpha$	$=\alpha^4$
x^2+x+1	111	$\alpha^2+\alpha+1$	$=\alpha^5$

拡大体は$GF(2^3)=\{0,1,\alpha,\alpha^2,\alpha^3,\alpha^4,\alpha^5,\alpha^6\}$

Polynomial vector expression α power expression

0	000	0	
1	001	1	$=\alpha^0$
x	010	α	$=\alpha^1$
$x+1$	011	$\alpha+1$	$=\alpha^3$
x^2	100	α^2	$=\alpha^2$
x^2+1	101	α^2+1	$=\alpha^6$
x^2+x	110	$\alpha^2+\alpha$	$=\alpha^4$
x^2+x+1	111	$\alpha^2+\alpha+1$	$=\alpha^5$

From $\alpha^3+\alpha+1=0$, we use

$\alpha^3=\alpha+1$

$\alpha^4=\alpha(\alpha+1)=\alpha^2+\alpha$

$\alpha^5=\alpha(\alpha^2+\alpha)=\alpha^3+\alpha^2=\alpha+1+\alpha^2$

$\alpha^6=\alpha(\alpha^2+\alpha+1)=\alpha^3+\alpha^2+\alpha=\alpha+1+\alpha^2+\alpha=\alpha^2+1$

The extension field is GF (2 ^ 3) = {0, 1, α, α ^ 2, α ^ 3, α ^ 4, α ^ 5, α ^ 6}.

１－２）$n=2^m-1=7$　であるので、n,k,tの組み合わせは、以下がある。

n	k	$2t$	t	
7	5	2	1	(7, 5)符号
7	3	4	2	(7, 3)符号
7	1	6	3	(7, 1)符号

1-2) Since n = 2 ^ m-1 = 7, there are the following combinations of n, k, and t.

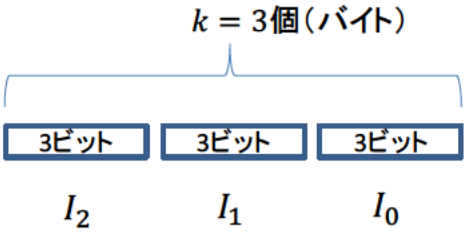
n k 2t t

7 5 2 1 (7, 5) code

7 3 4 2 (7, 3) code
7 1 6 3 (7, 1) code

1－3）具体的な符号化の方法
1-3) Specific coding method

■例１　(7,3)符号　　$m=3, n=7, k=3, t=2$, 原始多項式が$p(x)=x^3+x+1$の場合
①情報ビット系列を 3 ビット($m=3$)ずつの 3 個($k=3$)のブロック（バイト）に分割する。例えば
情報ビット 001 010 100
情報バイト　1　α^1　α^2
バイト番号　1　2　3
②情報バイトを多項式で表現する。
$I(y)=I_{k-1}y^{k-1}+\cdots+I_1y+I_0=1y^2+\alpha^1y^1+\alpha^2y^0$



Example 1: RS(7,3) code
m = 3, n = 7, k = 3, t = 2, when primitive polynomial is p (x) = x ^ 3 + x + 1
(1) Divide information bit sequence into 3 blocks (bytes) of 3 bits (m = 3) each.
Information bit　001 010 100
Information byte　1 α ^ 1 α ^ 2
Byte number　1　2　3
(2) Express the information byte as a polynomial.
$I(y)=I_{k-1}y^{k-1}+\cdots+I_1y+I_0=1y^2+\alpha^1y^1+\alpha^2y^0$

③生成多項式を求める。
$G(y)=\prod_{i=l}^{l+2t-1}(y-\alpha^i)=\prod_{i=0}^{3}(y-\alpha^i)=(y-\alpha^0)(y-\alpha^1)(y-\alpha^2)(y-\alpha^3)$
$=y^4+\alpha^2y^3+\alpha^5y^2+\alpha^5y^1+\alpha^6$

④情報多項式$I(y)*y^{n-k}$を生成多項式$G(y)$　で除算した余りを求める。
$I(y)=I_{k-1}y^{k-1}+\cdots I_1y+I_0=1y^2+\alpha^1y^1+\alpha^2y^0$
$r(y)=I(y)*y^{n-k}\ mod\ G(y)=(1y^2+\alpha^1y^1+\alpha^2y^0)*y^4\ mod\ y^4+\alpha^2y^3+\alpha^5y^2+\alpha^5y^1+\alpha^6$

$= \alpha^4 y^3 + \alpha^6 y^2 + \alpha^5 y + \alpha^3$

(3) Calculate the generator polynomial.

$G(y) = \prod_{i=l}^{l+2t-1}(y - \alpha^i) = \prod_{i=0}^{3}(y - \alpha^i) = (y - \alpha^0)(y - \alpha^1)(y - \alpha^2)(y - \alpha^3)$

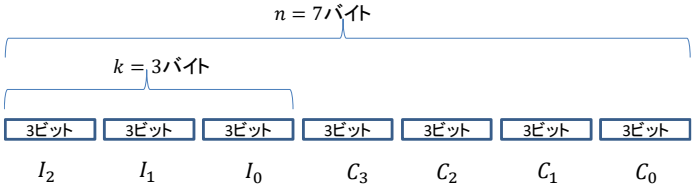
$= y^4 + \alpha^2 y^3 + \alpha^5 y^2 + \alpha^5 y^1 + \alpha^6$

(4) Find the remainder of dividing the information polynomial I (y) * y ^ (n-k) by the generator polynomial G (y).

$I(y) = I_{k-1} y^{k-1} + \cdots I_1 y + I_0 = 1y^2 + \alpha^1 y^1 + \alpha^2 y^0$

$r(y) = I(y) * y^{n-k} \; mod \; G(y) = (1y^2 + \alpha^1 y^1 + \alpha^2 y^0) * y^4 \; mod \; y^4 + \alpha^2 y^3 + \alpha^5 y^2 + \alpha^5 y^1 + \alpha^6$

$= \alpha^4 y^3 + \alpha^6 y^2 + \alpha^5 y + \alpha^3$



⑤以上より、RS 符号を得る。

(5) Finally, we obtain the RS code.

情報ビット	001	010	100				
情報バイト	1	α^1	α^2				
符号化後	1	α^1	α^2	α^4	α^6	α^5	α^3
RS 符号語	001	010	100	110	101	111	011
バイト番号	1	2	3	4	5	6	7

Information bit	001	010	100				
Information byte	1	α ^ 1	α ^ 2				
RS code in roots	1	α^1	α^2	α^4	α^6	α^5	α^3
RS code in vectors	001	010	100	110	101	111	011
Byte number	1	2	3	4	5	6	7

⑥③～⑤の別の方法　（生成多項式を陽に求めない方法）

符号語wを$w = (w_1, w_2, \cdots, w_n) = (I(\alpha^{n-1}), I(\alpha^{n-2}), \cdots, I(1))$で求める。

(6) Another method from (3) to (5) (method without explicit generator polynomial)
Find the codeword w = (w_1, w_2, …, w_n) by w = (I (α ^ (n-1)), I (α ^ (n-2)), …, I (1)).

$I(y) = I_{k-1}y^{k-1} + \cdots I_1 y + I_0 = 1y^2 + \alpha^1 y^1 + \alpha^2 y^0$
$w_7 = I(y = \alpha^0) = I(y = 1) = I_{k-1} + \cdots I_1 + I_0 = 1 + \alpha^1 + \alpha^2 = \alpha^5 = (111)$
$w_6 = I(y = \alpha^1) = 1\alpha^2 + \alpha^1\alpha^1 + \alpha^2 = \alpha^2 + \alpha^2 + \alpha^2 = \alpha^2 = (100)$
$w_5 = I(y = \alpha^2) = 1\alpha^4 + \alpha^1\alpha^2 + \alpha^2 = \alpha^4 + \alpha^3 + \alpha^2 = 1 = (001)$
$w_4 = I(y = \alpha^3) = 1\alpha^6 + \alpha^1\alpha^3 + \alpha^2 = \alpha^6 + \alpha^4 + \alpha^2 = \alpha^5 = (111)$
$w_3 = I(y = \alpha^4) = 1\alpha^8 + \alpha^1\alpha^4 + \alpha^2 = \alpha^1 + \alpha^5 + \alpha^2 = 1 = (001)$
$w_2 = I(y = \alpha^5) = 1\alpha^{10} + \alpha^1\alpha^5 + \alpha^2 = \alpha^3 + \alpha^6 + \alpha^2 = \alpha = (010)$
$w_1 = I(y = \alpha^6) = 1\alpha^{12} + \alpha^1\alpha^6 + \alpha^2 = \alpha^5 + 1 + \alpha^2 = \alpha = (010)$

以上より、
Thus,

情報ビット	001	010	100				
情報バイト	1	α^1	α^2				
符号化後	α^1	α^1	α^0	α^5	α^0	α^2	α^5
RS 符号語	010	010	001	111	001	100	111
バイト番号	1	2	3	4	5	6	7

Information bit	001	010	100				
Information byte	1	α ^ 1	α ^ 2				
RS code in roots	α^1	α^1	α^0	α^5	α^0	α^2	α^5
RS code in vectors	010	010	001	111	001	100	111
Byte number	1	2	3	4	5	6	7

（2）$m = 4$の場合　　$n = 2^m - 1 = 15$
2－1）
・情報記号は、$GF(2^4)$の上の元である。原始多項式を$p(x) = x^4 + x + 1$ とする。すなわち$\alpha^4 + \alpha + 1 = 0$。

多項式表現　　ベクトル表現　　αべき乗表現

Polynomial expression Vector expression α exponentiation expression

0	0000	0	
1	0001	1	$=\alpha^0$
x	0010	α	$=\alpha^1$
$x+1$	0011	$\alpha+1$	$=\alpha^4$
x^2	0100	α^2	$=\alpha^2$
x^2+1	0101	α^2+1	$=\alpha^8$
x^2+x	0110	$\alpha^2+\alpha$	$=\alpha^5$
x^2+x+1	0111	$\alpha^2+\alpha+1$	$=\alpha^{10}$
x^3	1000	α^3	$=\alpha^3$
x^3+1	1001	α^3+1	$=\alpha^{14}$
x^3+x	1010	$\alpha^3+\alpha$	$=\alpha^9$
x^3+x+1	1011	$\alpha^3+\alpha+1$	$=\alpha^7$
x^3+x^2	1100	$\alpha^3+\alpha^2$	$=\alpha^6$
x^3+x^2+1	1101	$\alpha^3+\alpha^2+1$	$=\alpha^{13}$
x^3+x^2+x	1110	$\alpha^3+\alpha^2+\alpha$	$=\alpha^{11}$
x^3+x^2+x+1	1111	$\alpha^3+\alpha^2+\alpha+1$	$=\alpha^{12}$

$GF(2^4)=\{0,1,\alpha,\alpha^2,\alpha^3,\alpha^4,\alpha^5,\alpha^6,\alpha^7,\alpha^8,\alpha^9,\alpha^{10},\alpha^{11},\alpha^{12},\alpha^{13},\alpha^{14}\}$

２－２）$n=2^m-1=15$

n	k	$2t$	t	
15	13	2	1	(15,13)符号
15	11	4	2	(15,11)符号
15	9	6	3	(15, 9)符号
15	7	8	4	(15, 7)符号
15	5	10	5	(15, 5)符号
15	3	12	6	(15, 3)符号

n	k	2t	t	
15	13	2	1	(15,13) code
15	11	4	2	(15,11) code
15	9	6	3	(15, 9) code
15	7	8	4	(15, 7) code
15	5	10	5	(15, 5) code
15	3	12	6	(15, 3) code

２－３）具体的な符号化の方法

2-3) Specific coding method

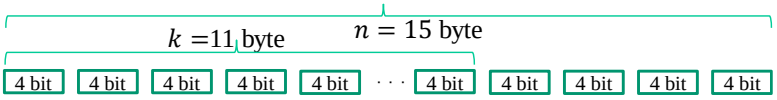
■例　(15, 11)符号　$m = 4, n = 15, k = 11, t = 2$, 原始多項式が$p(x) = x^4 + x + 1$の場合

①情報ビット系列を 4 ビット$(m = 4)$ずつの 11 個$(k = 11)$のブロック（バイト）に分割する。例えば、

情報ビット 1010 0011 0011 1101 1110 0101 1101 1000 1010 0101 0000

情報バイト	α^9	α^4	α^4	α^{13}	α^{11}	α^8	α^{13}	α^3	α^9	α^8	0
バイト番号	1	2	3	4	5	6	7	8	9	10	11

②情報バイトを多項式で表現する。



$$I(y) = I_{k-1}y^{k-1} + \cdots I_1 y + I_0$$
$$= \alpha^9 y^{10} + \alpha^4 y^9 + \alpha^4 y^8 + \alpha^{13} y^7 + \alpha^{11} y^6 + \alpha^8 y^5 + \alpha^{13} y^4 + \alpha^3 y^3 + \alpha^9 y^2 + \alpha^8 y^1 + 0y^0$$

Example 2: RS(15, 11) code

m = 4, n = 15, k = 11, t = 2, when primitive polynomial is p (x) = x ^ 4 + x + 1

(1) Divide information bit sequence into 11 blocks (bytes) (k=11) of 4 bits (m = 4) each.

Information bit 1010 0011 0011 1101 1110 0101 1101 1000 1010 0101 0000

Information byte	α^9	α^4	α^4	α^{13}	α^{11}	α^8	α^{13}	α^3	α^9	α^8	0
Byte numbr	1	2	3	4	5	6	7	8	9	10	11

(2) Express the information byte as a polynomial.

$$I(y) = I_{k-1}y^{k-1} + \cdots I_1 y + I_0$$
$$= \alpha^9 y^{10} + \alpha^4 y^9 + \alpha^4 y^8 + \alpha^{13} y^7 + \alpha^{11} y^6 + \alpha^8 y^5 + \alpha^{13} y^4 + \alpha^3 y^3 + \alpha^9 y^2 + \alpha^8 y^1 + 0y^0$$

③生成多項式を求める。

$$G(y) = \prod_{i=l}^{l+2t-1}(y - \alpha^i) = \prod_{i=0}^{3}(y - \alpha^i) = (y - \alpha^0)(y - \alpha^1)(y - \alpha^2)(y - \alpha^3)$$
$$= y^4 + \alpha^{12} y^3 + \alpha^4 y^2 + y^1 + \alpha^6$$

④情報多項式$I(y) * y^{n-k}$を生成多項式$G(y)$　で除算した余りを求める。

$$I(y) = I_{k-1}y^{k-1} + \cdots I_1 y + I_0$$
$$= \alpha^9 y^{10} + \alpha^4 y^9 + \alpha^4 y^8 + \alpha^{13} y^7 + \alpha^{11} y^6 + \alpha^8 y^5 + \alpha^{13} y^4 + \alpha^3 y^3 + \alpha^9 y^2 + \alpha^8 y^1 + 0y^0$$
$$r(y) = I(y) * y^{n-k} \; mod \; G(y)$$
$$= (\alpha^9 y^{10} + \alpha^4 y^9 + \alpha^4 y^8 + \alpha^{13} y^7 + \alpha^{11} y^6 + \alpha^8 y^5 + \alpha^{13} y^4 + \alpha^3 y^3 + \alpha^9 y^2 + \alpha^8 y^1 + 0y^0) * y^4$$
$$mod \; y^4 + \alpha^{12} y^3 + \alpha^4 y^2 + y^1 + \alpha^6$$

$= \alpha^{13}y^3 + \alpha^{13}y^2 + \alpha^0 y + \alpha^{10}$

(3) Calculate the generator polynomial.

$G(y) = \prod_{i=l}^{l+2t-1}(y - \alpha^i) = \prod_{i=0}^{3}(y - \alpha^i) = (y - \alpha^0)(y - \alpha^1)(y - \alpha^2)(y - \alpha^3)$

$= y^4 + \alpha^{12}y^3 + \alpha^4 y^2 + y^1 + \alpha^6$

(4) Find the remainder of dividing the information polynomial I (y) * y ^ (n-k) by the generator polynomial G (y).

$I(y) = I_{k-1}y^{k-1} + \cdots I_1 y + I_0$

$= \alpha^9 y^{10} + \alpha^4 y^9 + \alpha^4 y^8 + \alpha^{13} y^7 + \alpha^{11} y^6 + \alpha^8 y^5 + \alpha^{13} y^4 + \alpha^3 y^3 + \alpha^9 y^2 + \alpha^8 y^1 + 0y^0$

$r(y) = I(y) * y^{n-k} \; mod \; G(y)$

$= (\alpha^9 y^{10} + \alpha^4 y^9 + \alpha^4 y^8 + \alpha^{13} y^7 + \alpha^{11} y^6 + \alpha^8 y^5 + \alpha^{13} y^4 + \alpha^3 y^3 + \alpha^9 y^2 + \alpha^8 y^1 + 0y^0) * y^4$

$mod \; y^4 + \alpha^{12}y^3 + \alpha^4 y^2 + y^1 + \alpha^6$

$= \alpha^{13}y^3 + \alpha^{13}y^2 + \alpha^0 y + \alpha^{10}$

⑤以上より

(5) Finally, we obtain the RS code.

情報ビット	1010	0011	0011	1101	1110	0101	1101	1000	1010	0101	0000				
情報バイト	α^9	α^4	α^4	α^{13}	α^{11}	α^8	α^{13}	α^3	α^9	α^8	0				
符号化後	α^9	α^4	α^4	α^{13}	α^{11}	α^8	α^{13}	α^3	α^9	α^8	0	α^{13}	α^{13}	α^0	α^{10}
RS 符号語	1010	0011	0011	1101	1110	0101	1101	1000	1010	0101	0000	1101	1101	0001	0111
バイト番号	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Information bit 1010 0011 0011 1101 1110 0101 1101 1000 1010 0101 0000

Information byte α^9 α^4 α^4 α^{13} α^{11} α^8 α^{13} α^3 α^9 α^8 0

RS code in roots α^9 α^4 α^4 α^{13} α^{11} α^8 α^{13} α^3 α^9 α^8 0 α^{13} α^{13} α^0 α^{10}

RS code in vectors 1010 0011 0011 1101 1110 0101 1101 1000 1010 0101 0000 1101 1101 0001 0111

Byte number 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

⑥③～⑤の別の方法　（生成多項式を陽に求めない方法）

符号語wを$w = (w_1, w_2, \cdots, w_n) = (I(\alpha^{n-1}), I(\alpha^{n-2}), \cdots, I(1))$で求める。

(6) Another method from (3) to (5) (method without explicit generator polynomial)

Find the codeword w = (w_1, w_2, …, w_n) by w = (I (α ^ (n-1)), I (α ^ (n-2)), …, I (1)).

$$I(y) = I_{k-1}y^{k-1} + \cdots I_1 y + I_0$$
$$= \alpha^9 y^{10} + \alpha^4 y^9 + \alpha^4 y^8 + \alpha^{13} y^7 + \alpha^{11} y^6 + \alpha^8 y^5 + \alpha^{13} y^4 + \alpha^3 y^3 + \alpha^9 y^2 + \alpha^8 y^1 + 0y^0$$
$$w_{15} = I(y = \alpha^0) = I(y = 1) = I_{k-1} + \cdots I_1 + I_0$$
$$= \alpha^9 + \alpha^4 + \alpha^4 + \alpha^{13} + \alpha^{11} + \alpha^8 + \alpha^{13} + \alpha^3 + \alpha^9 + \alpha^8 + 0$$
$$= \alpha^{11} + \alpha^3 = \alpha^3 + \alpha^2 + \alpha + \alpha^3 = \alpha^2 + \alpha = \alpha^5 = (0110)$$
$$w_{14} = I(y = \alpha^1) = I_{k-1}\alpha^{(n-i)(k-1)} + \cdots + I_1\alpha^{n-i} + I_0$$
$$= I_{10}\alpha^{(15-14)(11-1)} + I_9\alpha^{(15-14)(11-2)} + \cdots + I_1\alpha^{(15-14)(11-10)} + I_0$$
$$= I_{10}\alpha^{10} + I_9\alpha^9 + I_8\alpha^8 + I_7\alpha^7 + I_6\alpha^6 + I_5\alpha^5 + I_4\alpha^4 + I_3\alpha^3 + I_2\alpha^2 + I_1\alpha^1 + I_0$$
$$= \alpha^9\alpha^{10} + \alpha^4\alpha^9 + \alpha^4\alpha^8 + \alpha^{13}\alpha^7 + \alpha^{11}\alpha^6 + \alpha^8\alpha^5 + \alpha^{13}\alpha^4 + \alpha^3\alpha^3 + \alpha^9\alpha^2 + \alpha^8\alpha^1 + 0\alpha^0 = (0010)$$

$w_{13} = I(y = \alpha^2) = (0100)$　　$w_{12} = I(y = \alpha^3) = (1101)$　　$w_{11} = I(y = \alpha^4) = (1110)$

$w_{10} = I(y = \alpha^5) = (1011)$　　$w_9 = I(y = \alpha^6) = (1000)$　　$w_8 = I(y = \alpha^7) = (1011)$

$w_7 = I(y = \alpha^8) = (1110)$　　$w_6 = I(y = \alpha^9) = (0111)$　　$w_5 = I(y = \alpha^{10}) = (0011)$

$w_4 = I(y = \alpha^{11}) = (1111)$　　$w_3 = I(y = \alpha^{12}) = (1011)$　　$w_2 = I(y = \alpha^{13}) = (0111)$

$w_1 = I(y = \alpha^{14}) = (1001)$

情報ビット	1010	0011	0011	1101	1110	0101	1101	1000	1010	0101	0000				
情報バイト	α^9	α^4	α^4	α^{13}	α^{11}	α^8	α^{13}	α^3	α^9	α^8	0				
符号化後	α^{14}	α^{10}	α^7	α^{12}	α^4	α^{10}	α^{11}	α^7	α^3	α^7	α^{11}	α^{13}	α^2	α^1	α^5
RS 符号語	1001	0111	1011	1111	0011	0111	1110	1011	1000	1011	1110	1101	0100	0010	0110
バイト番号	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Information bit	1010	0011	0011	1101	1110	0101	1101	1000	1010	0101	0000				
Information byte	α^9	α^4	α^4	α^{13}	α^{11}	α^8	α^{13}	α^3	α^9	α^8	0				
RS code in roots	α^{14}	α^{10}	α^7	α^{12}	α^4	α^{10}	α^{11}	α^7	α^3	α^7	α^{11}	α^{13}	α^2	α^1	α^5
RS code in vectors	1001	0111	1011	1111	0011	0111	1110	1011	1000	1011	1110	1101	0100	0010	0110
Byte number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

２－４）復号方法の概要

2-4) Decryption method

種々の方法があるが、基本的には連立方程式を立てて解くこととなる。
まず、以前 BCH 符号で扱った方法を思いだそう。例えば、２誤り訂正可能な BCH 符号では、2 ビットの誤りを誤り多項式　$e(x) = x^i + x^j$　で表現し、受信符号語$v(x)$から、シンドロームS_1、S_3を以下のように計算した。

Basically, simultaneous equations are solved.
First, let's think of the method we used to handle with BCH code.
For example, in the BCH code that can correct 2 errors, the 2-bit error is expressed by an error polynomial e (x) = x ^ i + x ^ j, and the syndromes S_1 and S_3 are calculated from the received codeword v (x) as follows.

$S_1 = v(\alpha) = u(\alpha) + e(\alpha) = e(\alpha) = \alpha^i + \alpha^j$　　・・・（eq. 1）
$S_3 = v(\alpha^3) = u(\alpha^3) + \alpha^{3i} + \alpha^{3j} = \alpha^{3i} + \alpha^{3j}$

$\alpha^i \alpha^j = \alpha^{(i+j)} = \frac{S_3}{S_1} + {S_1}^2$　　・・・（eq. 2）

式１，式２を連立して、i、jを求めた。リードソロモンでは、誤り多項式は、拡大体の要素を係数に持つから、

$$e(x) = e_i x^i + e_j x^j$$

と表現される。従って、４つの変数e_i, e_i, i, j　を求める必要がある。つまり、位置i、jだけでなく、その大きさe_i, e_jを決定しなければならない。2 誤り訂正可能な RS 符号の生成多項式は、べきが連続した４つの根を持つので、
$S_0 = v(1) = u(1) + e(1) = e(1) = e_i + e_j$
$S_1 = v(\alpha) = u(\alpha) + e(\alpha) = e(\alpha) = e_i\alpha^i + e_j\alpha^j$
$S_2 = v(\alpha^2) = e(\alpha^2) = e_i\alpha^{2i} + e_j\alpha^{2j}$
$S_3 = v(\alpha^3) = e(\alpha^3) = e_i\alpha^{3i} + e_j\alpha^{3j}$
の４つの方程式が立てられ、これを解けば求めることが可能である。（注意：BCH の時は、$S_2 = v(\alpha^2)$は意味がなかった。）より詳細については、７．２．４で述べる。

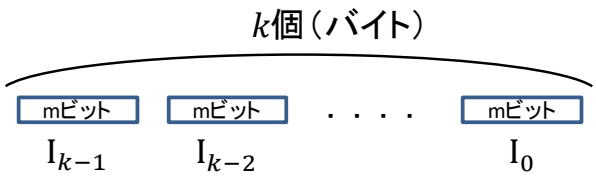
Equations 1 and 2 can be combined to get i and j.
In Reed-Solomon, an error polynomial has an element in the extension field as a coefficient, so it can be expressed as follows.
e (x) = e_i x ^ i + e_j x ^ j
Therefore, it is necessary to find the four variables e_i, e_i, i, j. That is, not only the positions i and j but also their values e_i and e_j must be determined. A 2-error-correctable RS code generator polynomial has four roots

of consecutive powers, so the following four equations can be established and solved. (For more details, see 7.2.4.)

$S_0 = v(1) = u(1) + e(1) = e(1) = e_i + e_j$
$S_1 = v(\alpha) = u(\alpha) + e(\alpha) = e(\alpha) = e_i\alpha^i + e_j\alpha^j$
$S_2 = v(\alpha^2) = e(\alpha^2) = e_i\alpha^{2i} + e_j\alpha^{2j}$
$S_3 = v(\alpha^3) = e(\alpha^3) = e_i\alpha^{3i} + e_j\alpha^{3j}$

７．２．３　ＲＳ詳細（発展）

7.2.3 RS code details



（１）定義

k個の情報記号$I_{k-1},\cdots,I_1,I_0$を符号化する。これらの情報記号I_iは$GF(2^m)$の元。すなわち、m個の$GF(2^m)$の元を用いて$I_i = (c_{m-1},\cdots,c_1,c_0)$と表現する。

k個の情報記号$I_{k-1},\cdots,I_1,I_0$を係数とする$GF(2^m)$上の多項式を考える。

$$I(y) = I_{k-1}y^{k-1} + \cdots + I_1y + I_0$$

nを$k < n \leq 2^m$とする。n個の$GF(2^m)$の互いに異なる元を、$\alpha_1,\alpha_2,\cdots,\alpha_n$とし、下記のように構成する符号を、符号長$n$の$GF(2^m)$上の RS 符号と言う。

$$w = (\,I(\alpha_1),I(\alpha_2),\cdots,I(\alpha_n)\,)$$

（この手法とフーリエ変換の関係は、今井秀樹著「符号理論」電子情報通信学会 p.158。$I(y)$を$G(y)$で割った余りで求めてもよい。）

(1) Definition

Let us consider encoding k information symbols, I_ (k-1), …, I_1, I_0.
These information symbols I_i are elements of GF (2 ^ m).
Then it is expressed as I_i = (c_ (m-1), …, c_1, c_0) using m elements of GF (2 ^ m).
Consider a polynomial on GF (2 ^ m) whose coefficients are k information symbols I_ (k-1), …, I_1, I_0.

I (y) = I_ (k-1) y ^ (k-1) + … + I_1 y + I_0

Let n be k <n ≤ 2 ^ m.
Let n Different elements of GFs (2 ^ m) be α_1, α_2, …, α_n.
Then, a code configured as shown below is called RS code on the GF (2 ^ m) with a code length of n.

w = (I (α_1), I (α_2), …, I (α_n))

(So far, apart from this definition we have discussed the method where I(y) is divided to find the remainder. The relationship between this definition and the method is based on Fourier transform.)

（2）$n = 2^m - 1$の RS 符号

(2) RS code whose length n is 2 ^ m-1

αを$GF(2^m)$上の原始元とし、$n = 2^m - 1$とするとき、$GF(2^m)$上の$n-1$次以下の多項式$W(x)$が$\alpha, \alpha^2, \cdots, \alpha^{n-k}$を根として持つならば、$W(x)$は符号長$n$、情報記号$k$、最小距離$d_{min} = n - k + 1$の RS 符号の符号語となる。

符号語$w = (I(\alpha_1), I(\alpha_2), \cdots, I(\alpha_n))$の $\alpha_1, \alpha_2, \cdots, \alpha_n$は、原始多項式$p(x)$の元（原始元）を$\alpha$とすると、$\alpha$のべき乗で表現できる。従って、

$$w = (w_1, w_2, \cdots, w_n) = (I(\alpha^{n-1}), I(\alpha^{n-2}), \cdots, I(1))$$

$$w_n = I(y = \alpha^0) = I(y = 1) = I_{k-1} + \cdots I_1 + I_0$$
$$\vdots$$
$$w_i = I\left(\alpha^{n-i}\right) = I_{k-1}\alpha^{(n-i)(k-1)} + \cdots + I_1\alpha^{n-i} + I_0$$
$$\vdots$$
$$w_1 = I(\alpha^{n-1}) = I_{k-1}\alpha^{(n-1)(k-1)} + \cdots + I_1\alpha^{n-1} + I_0$$

と符号化する。これは、元の情報記号$I_{k-1}, \cdots, I_1, I_0$を$\alpha$のべき乗で重み付けした符号語に変換しているとも言える。あるいは、$I(y) = I_{k-1}y^{k-1} + \cdots + I_1 y + I_0$　は、元の情報記号$I_{k-1}, \cdots, I_1, I_0$を係数とする多項式曲線を表しているので、$w = (w_1, w_2, \cdots, w_n)$の各要素を情報記号$I_{k-1}, \cdots, I_1, I_0$で定義される曲線上に配置しているとも言える。

$w = (w_1, w_2, \cdots, w_n)$を多項式表現したもの（符号語多項式）は以下となる。

$$W(x) = w_1x^n + w_2x^{n-1} + \cdots + w_{n-1}x + w_n$$
$$= I(y = \alpha^{n-1})x^n + I(y = \alpha^{n-2})x^{n-1} + \cdots + I(y = \alpha^1)x + I(y = \alpha^0)$$

When α is a primitive element on GF (2 ^ m) and n = 2 ^ m-1, if a polynomial W (x) of degree n-1 or less on GF (2 ^ m) has α, α ^ 2,…, α ^ (n-k) as its roots, W (x) is a codeword of RS code with code length n, information symbol k, and minimum distance d_min = n-k + 1.

Where α is a primitive element of the primitive polynomial p (x), then a codewords w = (I (α_1), I (α_2), …, I (α_n)), where α_1, α_2, …, α_n can be expressed by the power of root of α.
w = (w_1, w_2, …, w_n) = (I (α ^ (n-1)), I (α ^ (n-2)), …, I (1))

w_n = I (y = α ^ 0) = I (y = 1) = I_ (k-1) + …I_1 + I_0
::
w_i = I (α ^ (n-i)) = I_ (k-1) α ^ ((n-i) (k-1)) + … + I_1 α ^ (n-i) + I_0
::
w_1 = I (α ^ (n-1)) = I_ (k-1) α ^ ((n-1) (k-1)) + … + I_1 α ^ (n-1) + I_0

It can be also said that the method converts an original information symbols I_ (k-1), …, I_1, I_0 into codewords weighted by the power of α.
Alternatively, I (y) = I_ (k-1) y ^ (k-1) + … + I_1 y + I_0 is a polynomial whose coefficients are the original information symbols I_ (k-1), …, I_1, I_0. Since it represents a k-th order curve, each element of w = (w_1, w_2, …, w_n) is arranged on the curve defined by the information symbols I_ (k-1), …, I_1, I_0.
The polynomial of w = (w_1, w_2, …, w_n) (codeword polynomial) is as follows.

（３）最小距離
(3) Minimum distance

今、$w=(w_1,w_2,\cdots,w_n)$が０でなく、重みがdであったとする。（この場合の重みとは、非 0 の項を重み１とカウントする。）すなわち、n個のw_1~w_nのうちd個が非０である。逆に言うと、$n-d$個のw_iが０となっている。これは、

$$\left.\begin{array}{l} w_n = I(y=\alpha^0) = I(1) = I_{k-1} + \cdots I_1 + I_0 \\ : \end{array}\right\}$$

$w_i = I(\alpha^{n-i}) = I_{k-1}\alpha^{(n-i)(k-1)} + \cdots + I_1\alpha^{n-i} + I_0$　　このうちの$n-d$個が 0

:

$w_1 = I(\alpha^{n-1}) = I_{k-1}\alpha^{(n-1)(k-1)} + \cdots + I_1\alpha^{n-1} + I_0$

を考えると、$I(y)$は、$n-d$個の根を持つ。$I(y) = I_{k-1}y^{k-1} + \cdots + I_1y + I_0$の次元は、$k-1$次以下であるので、根の数（$I(y) = 0$となる$y$の数）は$k-1$個以下でなければならない。従って、$n-d \leq k-1$が成立する。よって、$d \geq n-k+1$である。すべての$d$に対して成立するので、最小距離$d_{min}$は$d_{min} \geq n-k+1$となる。

一方、$I(y) = I_{k-1}y^{k-1} + \cdots + I_1y + I_0 = (y-\alpha_1)(y-\alpha_2)\cdots(y-\alpha_{k-1})$とすると、$I(y)$は、$k-1$次の多項式であり、$I(\alpha^j) \neq 0\ (j = k, k+1, \cdots, n)$（この個数が$n-k+1$）となる。つまり、この$I(y)$で生成される符号語の 0 ではない成分は$n-k+1$個あり、最小重みは$n-k+1$であり、$d_{min} \leq n-k+1$となる。

以上より、よって、最小距離は$d_{min} = n-k+1$となる。RS 符号は最小距離の等号が成り立つ符号（最大距離分離符号）である。

Now suppose w = (w_1, w_2, …, w_n) is not 0 and the weight is d. (The weight counts one non-zero term as the weight 1.) That is, d terms out of n (w_1 to w_n) are non-zero. In other words, n-d terms are 0. this is,

$$\left.\begin{array}{l} w_n = I(y = \alpha^0) = I(1) = I_{k-1} + \cdots I_1 + I_0 \\ : \\ w_i = I(\alpha^{n-i}) = I_{k-1}\alpha^{(n-i)(k-1)} + \cdots + I_1\alpha^{n-i} + I_0 \\ : \\ w_1 = I(\alpha^{n-1}) = I_{k-1}\alpha^{(n-1)(k-1)} + \cdots + I_1\alpha^{n-1} + I_0 \end{array}\right\} n-d \text{ terms are } 0$$

Thus, I (y) has n-d roots. Since the dimension of I (y) = I_ (k-1) y ^ (k-1) + … + I_1 y + I_0 is less than or equal to k-1 order, the number of roots must be k-1 or less. Therefore, n-d ≤ k-1 holds. That is, d ≥ n-k + 1. Since it holds for all d, the minimum distance d_min is d_min ≥ n-k + 1.

On the other hand, I (y) = I_ (k-1) y ^ (k-1) + … + I_1 y + I_0 = (y-α_1) (y-α_2) … (y-α_ (k-1)) Then, I (y) is a polynomial of degree k-1, and I (α ^ j) ≠ 0 (j = k, k + 1, …, n) (this number is n-k + 1). .. That is, there are n-k + 1 non-zero terms of the codeword generated by this I (y), the minimum weight is n-k + 1, and d_min ≤ n-k + 1.

From the above, the minimum distance is d_min = n-k + 1. The RS code is a code that holds the equal sign of the minimum distance (maximum distance separable code).

（４）誤り検出・訂正の基本

(4) Basics of error detection / correction

符号語$w = (w_1, w_2, \cdots, w_n)$の各要素は、

$w_n = I(\alpha^0) = I(1) = I_{k-1} + \cdots I_1 + I_0$

：

$w_i = I(\alpha^{n-i}) = I_{k-1}\alpha^{(n-i)(k-1)} + \cdots + I_1\alpha^{n-i} + I_0$

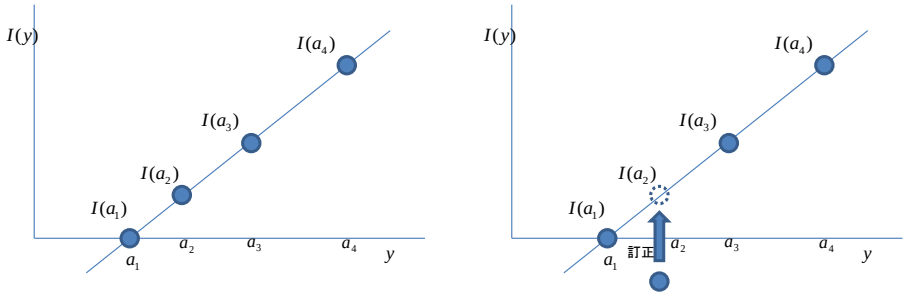
：

$w_1 = I(\alpha^{n-1}) = I_{k-1}\alpha^{(n-1)(k-1)} + \cdots + I_1\alpha^{n-1} + I_0$

で計算される。

つまり、$w = (w_1, w_2, \cdots, w_n)$の各要素は、$k-1$次元の曲線上に乗っていることになる。従って、復号時に曲線上にあるか否かで誤りを検出、訂正ができる。

例として、$n = 4$、$k = 2$でI_0、I_1を符号化することを考える。$I(y)$は$I(y) = I_1 y + I_0$であり、$w = (w_1, w_2, w_3, w_4)$の各要素は、下左図のように直線上に並ぶ。０でない符号語で$I(y) = 0$となるのは高々１であり、最小距離は３$(d_{min} = n - k + 1 = 4 - 2 + 1)$となる。つまり単一誤り訂正符号となる。下右図のように（バイトに）誤りが生じた場合に訂正できる。（今井秀樹著「情報・符号・暗号の理論」電子情報通信学会編　コロナ社 p.133）



Each element of the codeword w = (w_1, w_2, …, w_n) is calculated as follows.

$w_n = I(\alpha^0) = I(1) = I_{k-1} + \cdots I_1 + I_0$

：

$w_i = I(\alpha^{n-i}) = I_{k-1}\alpha^{(n-i)(k-1)} + \cdots + I_1\alpha^{n-i} + I_0$

：

$w_1 = I(\alpha^{n-1}) = I_{k-1}\alpha^{(n-1)(k-1)} + \cdots + I_1\alpha^{n-1} + I_0$

In other words, each element of w = (w_1, w_2, …, w_n) is on a k-1 dimensional curve. Therefore, an error can be detected and corrected depending on whether or not it is on a curve at the time of decoding.
As an example, consider encoding I_0 and I_1 with n = 4 and k = 2. I (y) is I (y) = I_1 y + I_0, and each element of w = (w_1, w_2, w_3, w_4) is arranged on a straight line as shown in the left figure. For codewords that are not 0, I (y) = 0 is at most 1, and the minimum distance is 3 (d_min = n-k + 1 = 4-2 + 1). That is, it becomes a single error correction code. It can be corrected when an error occurs (in bytes) as shown in the figure on the right. (For more detail see Hideki Imai, "Theory of Information, Code, and Cryptography", edited by Institute of Electronics, Information and Communication Engineers, Corona Publishing Co., Ltd. p.133)

（５）例題１

(5) Example 1

問 1）$GF(2^3)$において、情報バイト記号$(I_2, I_1, I_0) = (0, 1, \alpha)$の時、符号長$n = 2^3 - 1 = 7$、最小距離$d_{min} = 5$（すなわち２誤り訂正可能）のリードソロモン符号語$w = (w_1, w_2, \cdots, w_n)$を求めよ。ただし、$\alpha$は、原始多項式$p(x) = x^3 + x + 1$の根であり、$GF(2^3)$の原始元であるとする。

Q1) In GF (2 ^ 3), when the information byte symbol (I_2, I_1, I_0) = (0, 1, α), find the Reed-Solomon codeword w = (w_1, w_2, …, w_n) of the code length n = 2 ^ 3-1 = 7, and the minimum distance d_min = 5 (2 error correctable). It is assumed that α is the root of the primitive polynomial p (x) = x ^ 3 + x + 1 and is the primitive element of GF (2 ^ 3).

解答)

情報バイト数kは$k = n - d_{min} + 1 = 3$である。また、$I(y) = I_2 y^2 + I_1 y + I_0 = y + \alpha$である。

$w = (I(\alpha_1), I(\alpha_2), I(\alpha_3), I(\alpha_4), I(\alpha_5), I(\alpha_6), I(\alpha_7)) = (w_1, w_2, w_3, w_4, w_5, w_6, w_7)$とする。$\alpha^3 + \alpha + 1 = 0$であるから、

$w_1 = I(\alpha_1) = I(\alpha^6) = \alpha^6 + \alpha = \alpha^2 + 1 + \alpha = \alpha^3 + \alpha^2 = \alpha^5$　　同様に、

$w_2 = I(\alpha_2) = I(\alpha^5) = \alpha^6$　$w_3 = I(\alpha_3) = I(\alpha^4) = \alpha^2$　$w_4 = I(\alpha_4) = I(\alpha^3) = 1$

$w_5 = I(\alpha_5) = I(\alpha^2) = \alpha^4$　$w_6 = I(\alpha_6) = I(\alpha^1) = 0$　$w_7 = I(\alpha_7) = I(\alpha^0) = \alpha^3$

以上より、$w = (\alpha^5, \alpha^6, \alpha^2, 1, \alpha^4, 0, \alpha^3)$

Answer1)
The number of information bytes k is k = n-d_min + 1 = 3. I (y) = I_2 y ^ 2 + I_1 y + I_0 = y + α.
Let us w = (I (α_1), I (α_2), I (α_3), I (α_4), I (α_5), I (α_6), I (α_7)) be (w_1, w_2, w_3, w_4, w_5, w_6, w_7).
Because $\alpha^3 + \alpha + 1 = 0$,
w_1 = I (α_1) = I (α ^ 6) = α ^ 6 + α = α ^ 2 + 1 + α = α ^ 3 + α ^ 2 = α ^ 5 Similarly,
w_2 = I (α_2) = I (α ^ 5) = α ^ 6 w_3 = I (α_3) = I (α ^ 4) = α ^ 2 w_4 = I (α_4) = I (α ^ 3) = 1
w_5 = I (α_5) = I (α ^ 2) = α ^ 4 w_6 = I (α_6) = I (α ^ 1) = 0 w_7 = I (α_7) = I (α ^ 0) = α ^ 3
From the above, we get the codeword w = (α ^ 5, α ^ 6, α ^ 2,1, α ^ 4,0, α ^ 3).

問２）問１の符号語$W(y)$は、$\alpha, \alpha^2, \cdots, \alpha^{n-k} = \alpha^4$を根として持つことを証明せよ。
Q2) Prove that the codeword W (y) in Q1 has α, α ^ 2, …, α ^ (n-k) = α ^ 4 as its roots.

$W(y) = \alpha^5 y^6 + \alpha^6 y^5 + \alpha^2 y^4 + y^3 + \alpha^4 y^2 + \alpha^3$ であるので、
$W(\alpha) = \alpha^{11} + \alpha^{11} + \alpha^6 + \alpha^3 + \alpha^6 + \alpha^3 = 0$
$W(\alpha^2) = \alpha^{17} + \alpha^{16} + \alpha^{10} + \alpha^6 + \alpha^8 + \alpha^3 = \alpha^3 + \alpha^2 + \alpha^3 + \alpha^6 + \alpha^1 + \alpha^3 = 0$
$W(\alpha^3) = \alpha^{23} + \alpha^{21} + \alpha^{14} + \alpha^9 + \alpha^{10} + \alpha^3 = 0$
$W(\alpha^4) = \alpha^{29} + \alpha^{26} + \alpha^{18} + \alpha^{12} + \alpha^{12} + \alpha^3 = 0$

問３）問１の符号語$W(y)$は生成多項式$G(y)$で整除されることを確認せよ。
Q3) Confirm that the codeword W (y) in Q1 is divided by the generated polynomial G (y).

生成多項式$G(y)$は、$\alpha, \alpha^2, \alpha^3, \alpha^{n-k} = \alpha^4$を根として持つから、

$$G(y) = (y-\alpha)(y-\alpha^2)(y-\alpha^3)(y-\alpha^4) = y^4 + \alpha^3 y^3 + 1y^2 + \alpha y + \alpha^3$$

$$W(y) = \alpha^5 y^6 + \alpha^6 y^5 + \alpha^2 y^4 + y^3 + \alpha^4 y^2 + \alpha^3 = (y^4 + \alpha^3 y^3 + 1y^2 + \alpha y + \alpha^3)(\alpha^5 y^2 + \alpha^5 y + 1)$$

$$= G(y)(\alpha^5 y^2 + \alpha^5 y + 1)$$

従って、$W(y)$は$G(y)$で整除される。

Since the generation polynomial G (y) has α, α ^ 2, α ^ 3, α ^ (n-k) = α ^ 4 as roots,
G (y) = (y-α) (y-α ^ 2) (y-α ^ 3) (y-α ^ 4) = y ^ 4 + α ^ 3 y ^ 3 + 1y ^ 2 + αy + α ^ 3
W (y) = α ^ 5 y ^ 6 + α ^ 6 y ^ 5 + α ^ 2 y ^ 4 + y ^ 3 + α ^ 4 y ^ 2 + α ^ 3 = (y ^ 4 + α ^ 3 y ^ 3 + 1y ^ 2 + αy + α ^ 3) (α ^ 5 y ^ 2 + α ^ 5 y + 1) = G (y) (α ^ 5 y ^ 2 + α ^ 5 y + 1)
Therefore, W (y) is divided by G (y).

（６）例題２（岩垂好裕著「符号理論入門」昭晃堂 p.75 より）
問）３誤り訂正可能な$GF(2^4)$上の RS 符号生成多項式を求めよ。ただし、αを$x^4 + x + 1$の原始元とする。
(6) Example 2 (From "Introduction to Coding Theory" by Yoshihiro Iwadare, Shokodo p.75)
Q) Find RS code generator polynomial on GF (2 ^ 4) that can correct 3 errors. Let α be the primitive element of x ^ 4 + x + 1.

解）３誤りを訂正するためには、生成多項式はべき数が連続する６つの根を持たねばならない。従って、生成多項式（の一つ）$G(x)$は、$\alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6$を根として持つので以下となる。

$$G(x) = (x-\alpha)(x-\alpha^2)(x-\alpha^3)(x-\alpha^4)(x-\alpha^5)(x-\alpha^6)$$

$$= x^6 + \alpha^{10} x^5 + \alpha^{14} x^4 + \alpha^4 x^3 + \alpha^6 x^2 + \alpha^9 x + \alpha^6$$

これによって生成される符号の符号長は$n = 2^4 - 1 = 15$、情報記号数は$k = 9$である。

Answer) In order to correct 3 errors, a generator polynomial must have 6 roots with consecutive powers. Therefore, one of generator polynomials, G (x) has α, α ^ 2, α ^ 3, α ^ 4, α ^ 5, and α ^ 6 as roots. Thus,

G (x) = (x-α) (x-α ^ 2) (x-α ^ 3) (x-α ^ 4) (x-α ^ 5) (x-α ^ 6) = x ^ 6 + α ^ 10 x ^ 5 + α ^ 14 x ^ 4 + α ^ 4 x ^ 3 + α ^ 6 x ^ 2 + α ^ 9 x + α ^ 6

The code length of the code generated by this G(x) is n = 2 ^ 4-1 = 15, and the number of information symbols is k = 9.

７．２．４　リードソロモン符号の誤り訂正
7.2.4 Correction of Reed-Solomon symbols

１）概要
1) Overview

リードソロモンでは、誤りの位置だけでなく、誤りの大きさも決定する必要がある。
（cf. BCH では、誤りの大きさは１であったので位置を検出すればよかった。６．３．６および７．１．３（７）参照）

In Reed-Solomon codes, it is necessary to determine not only the position of the error, but also the magnitude of the error.
(cf. In BCH codes, the error size was 1, so that it was sufficient to detect the position only. See 6.3.6 and 7.1.3 (7))

今、送信語$u(x)$ を送り、w個の誤りが生じて受信語$v(x)$ を受け取ったとする。
w個の誤りは、それぞれの誤りの位置をl_0、l_1、、、l_{w-1}とし、大きさをe_0、e_1、、、e_{w-1} とすれば、以下の式で表現できる。

Suppose now that we have sent a word u(x) and received a word v(x) with w errors.
W errors can be represented as the following.

$$e(x) = \sum_{c=0}^{w-1} e_c x^{l_c} \quad （式 7.2.2）$$

シンドロームは、
Syndromes are calculated below.

$$S_i = v(\alpha^i) = u(\alpha^i) + e(\alpha^i) = u(\alpha^i) + \sum_{c=0}^{w-1} e_c (\alpha^i)^{l_c} \quad （式 7.2.3）$$

で計算される。

t個の誤り訂正能力を持つ符号では、$2t$個のべきが連続した根を持つので、

For a code with t error correction capability, the 2t roots have consecutive powers, so the following holds.

$$u(\alpha^i) = 0 \; for \; i = 0,1,,,2t-1$$

である。従って、w（$\leq t$）個の誤りの訂正を行う場合は、

Therefore, when correcting w (≤t) errors, we obtain the following equations.
Thus, 2t simultaneous equations are established.

$$S_i = v(\alpha^i) = u(\alpha^i) + e(\alpha^i) = u(\alpha^i) + \sum_{c=0}^{w-1} e_c(\alpha^i)^{l_c}$$
$$= \sum_{c=0}^{w-1} e_c(\alpha^i)^{l_c} \; for \; i = 0,1,,,2t-1 \quad (式\ 7.2.4)$$

となり、$2t$ 本の連立方程式が立てられる。
誤り多項式の未知変数は、位置l_0、l_1、、、l_{w-1}、大きさe_0、e_1、、、e_{w-1} の$2w$個あるので、連立方程式を解けば、誤り多項式を決定できる。この決定に関連する方法としては、ユークリッド互除法、Berlekamp-Massey, Chein Search, Forney 等がある。以下では、基本的な方法を述べる。

There are 2w unknown variables in the error polynomial at positions l_0, l_1, ..., l_(w-1), and magnitudes e_0, e_1, ..., e_(w-1).
By solving the simultaneous equations, we can decide these variables.
Related methods for this determination include Euclidean Algorithm, Berlekamp-Massey, Chein Search, Forney, and others.
The basic method is described below.

２）復号の手順　（2 誤り訂正の場合）

2) Decoding procedure (for 2-error correction)

2 誤りの場合を例に挙げる。ただし、2 以上にも適応可能である。（参考　岩垂好裕著「符号理論入門」昭晃堂 p. 75-77 では３誤りを挙げている）
誤り多項式を以下のように表現する。

Let us take a case of 2 errors as an example.
However, it can be easily extended 3 or more.

(Reference Yoshihiro Iwadare, "Introduction to Coding Theory", Shokodo, p.75-77 lists three errors.)
We express the error polynomial as follows.

$$e(x) = e_i x^i + e_j x^j$$

受信語は、
The received words are as follows.

$$v(x) = u(x) + e(x)$$

である。2誤り訂正可能な場合は、生成多項式のべきの連続する根が４つ連続している。
シンドロームを以下で求める

If 2 errors are correctable, the generator polynomial has four roots with consecutive powers.
Find the syndrome below.

$h = 0$ $S_0 = v(\alpha^0) = u(1) + e(1) = e(1) = e_i + e_j$ 式 2-1
$h = 1$ $S_1 = v(\alpha^1) = u(\alpha) + e(\alpha) = e(\alpha) = e_i\alpha^i + e_j\alpha^j$ 式 2-2
$h = 2$ $S_2 = v(\alpha^2) = u(\alpha^2) + e(\alpha^2) = e(\alpha^2) = e_i\alpha^{2i} + e_j\alpha^{2j}$ 式 2-3
$h = 3$ $S_3 = v(\alpha^3) = u(\alpha^3) + e(\alpha^3) = e(\alpha^3) = e_i\alpha^{3i} + e_j\alpha^{3j}$ 式 2-4
（一般のhに対しては、 $S_h = e_i\alpha^{hi} + e_j\alpha^{hj} = e_i(\alpha^i)^h + e_j(\alpha^j)^h$である。）

(For general h, S_h=e_i α^hi+e_j α^hj=e_i (α^i)^h+e_j (α^j)^h.)

σ_0, σ_1 を係数とする多項式（誤り位置多項式）$\sigma(x)$ を以下のように定義する。
A polynomial (called error locator polynomial) σ(x) with σ_0 and σ_1 as coefficients is defined as follows.

$\sigma(x) = (x - \alpha^i)(x - \alpha^j) = x^2 + \sigma_1 x + \sigma_0$ 式 2-5
この式に、$x = \alpha^i$、$x = \alpha^j$ を代入する。

Substitute x = α^i and x = α^j into this formula.

$0 = \alpha^{2i} + \sigma_1\alpha^i + \sigma_0$　式 2-6
$0 = \alpha^{2j} + \sigma_1\alpha^j + \sigma_0$　式 2-7

式 2-6 の左右辺を入れ替え、両辺に、$e_i(\alpha^i)^h$をかける。

Swap the left and right sides of Equation 2-6 and multiply both sides by e_i (α^i)^h.

$e_i(\alpha^i)^h\alpha^{2i} + e_i(\alpha^i)^h\sigma_1\alpha^i + e_i(\alpha^i)^h\sigma_0$=0
$e_i(\alpha^i)^{h+2} + e_i(\alpha^i)^{h+1}\sigma_1 + e_i(\alpha^i)^h\sigma_0$=0　　式 2-8

式 2-7 の左右辺を入れ替え、両辺に、$e_j(\alpha^j)^h$をかける。
Swap the left and right sides of Equation 2-7 and multiply both sides by e_j (α^j)^h.

$e_j(\alpha^j)^h\alpha^{2j} + e_j(\alpha^j)^h\sigma_1\alpha^j + e_j(\alpha^j)^h\sigma_0$=0
$e_j(\alpha^j)^{h+2} + e_j(\alpha^j)^{h+1}\sigma_1 + e_j(\alpha^j)^h\sigma_0$=0　　式 2-9

式 2-8、2-9 の両辺を縦に加算して、整理する。

Vertically add both sides of Equations 2-8 and 2-9 to rearrange them.

$$\left[e_i(\alpha^i)^{h+2} + e_j(\alpha^j)^{h+2}\right] + \sigma_1\left[e_i(\alpha^i)^{h+1} + e_j(\alpha^j)^{h+1}\right] + \sigma_0\left[e_i(\alpha^i)^h + e_j(\alpha^j)^h\right] = 0$$

この式は、
$S_{h+2} + \sigma_1 S_{h+1} + \sigma_0 S_h = 0$ を意味している。

This formula means that
S_(h+2)+σ_1 S_(h+1)+σ_0 S_h=0.

$h = 0,1$に対して計算すると、以下となる。
Calculating for h=0,1 yields:

$h = 0$　$S_2 + \sigma_1 S_1 + \sigma_0 S_0 = 0$　式 2-10

$h=1 \quad S_3+\sigma_1 S_2+\sigma_0 S_1=0$　式 2-11

復号の手順

Decoding procedure

ステップ１）式 2-1～2-4（の前半部；$S_h=v(\alpha^h)$）により、S_0～S_3　を求める。
ステップ２）式 2-10～2-11 により、σ_0、σ_1　を求める。
ステップ３）式 2-5 を用いて、α^i、α^j　を求める。
　つまり、$\sigma(x)=x^2+\sigma_1 x+\sigma_0=(x-\alpha^i)(x-\alpha^j)$ となるように因数分解して、α^i、α^j　を定め、i、jを求める。（式 2-6～2-7 から求めていることと同じ）
ステップ４）式 2-1～2-2 により、e_i、e_j　を求める。

Step 1) Calculate S_0 to S_3 by formulas 2-1 to 2-4 (the first half of them: S_h=v(α^h)).
Step 2) Obtain σ_0 and σ_1 from equations 2-10 to 2-11.
Step 3) Use Equation 2-5 to find α^i and α^j.
In other words, factorize so that σ(x)=x^2+σ_1 x+σ_0=(x-α^i)(x-α^j), determine α^i and α^j, and i , j. (same as obtained from formulas 2-6 to 2-7)
Step 4) Obtain e_i and e_j from formulas 2-1 and 2-2.

$h=0 \quad S_0=v(\alpha^0)=v(1)=u(1)+e(1)=e(1)=e_i+e_j$　式 2-1
$h=1 \quad S_1=v(\alpha^1)=v(\alpha)=u(\alpha)+e(\alpha)=e(\alpha)=e_i\alpha^i+e_j\alpha^j$　式 2-2

　なお、式 2-3, 2-4 は使わないが満たされる。
ステップ５）i、j、e_i、e_j　から、誤り多項式　$e(x)=e_i x^i+e_j x^j$　を決定し、$u(x)=v(x)-e(x)$ で誤り訂正を行う。

Equations 2-3 and 2-4 are satisfied even though they are not used.
Step 5) From i, j, e_i, e_j, determine the error polynomial e(x)=e_i x^i+e_j x^j and correct the error with u(x)=v(x)-e(x) conduct.

３）具体的な例

3) Specific examples

３－１）準備

3-1) Preparation

以下では、$n=2^m-1$、$l=0$、$m=3$　を考える。

$n=2^m-1=7$　（7 バイトに注意）

Below, let us consider n=2^m-1, l=0, m=3.

n=2^m-1=7 (Note that this means 7 bytes.)

$GF(2)$の上の原始多項式$p(x)$を$p(x)=x^3+x+1$とし、この根をαとする。すなわち$\alpha^3+\alpha+1=0$。

多項式　　ベクトル表現　　αべき表現

Polynomial vector expression α power expression

0	000	0	
1	001	1	$=\alpha^0$
x	010	α	$=\alpha^1$
$x+1$	011	$\alpha+1$	$=\alpha^3$
x^2	100	α^2	$=\alpha^2$
x^2+1	101	α^2+1	$=\alpha^6$
x^2+x	110	$\alpha^2+\alpha$	$=\alpha^4$
x^2+x+1	111	$\alpha^2+\alpha+1$	$=\alpha^5$

拡大体は$GF(2^3)=\{0,1,\alpha,\alpha^2,\alpha^3,\alpha^4,\alpha^5,\alpha^6\}$

The extension field is GF (2 ^ 3) = {0,1, α, α ^ 2, α ^ 3, α ^ 4, α ^ 5, α ^ 6}.

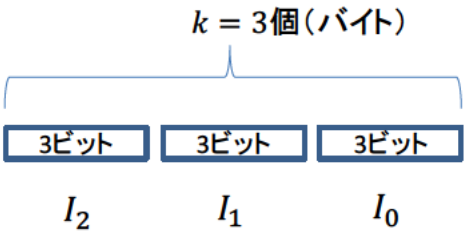
３－２）(7,3)符号　　$m=3,n=7,k=3,t=2$

3-2) (7,3) code m = 3, n = 7, k = 3, t = 2

① 情報ビット系列を 3 ビット($m=3$)ずつの 3 個($k=3$)のブロック（バイト）に分割する。例えば

(1) Divide the information bit sequence into three (k=3) blocks (bytes) of 3 bits (m=3) each. For example,

情報ビット 001 010 100
情報バイト 1 α^1 α^2
バイト番号 1 2 3



Info. bit 001 010 100
Info. byte 1 α^1 α^2
Byte # 1 2 3

② 情報バイトを多項式で表現する。
(2) Represent information bytes as polynomials.

$$I(y) = I_{k-1}y^{k-1} + \cdots + I_1 y + I_0 = 1y^2 + \alpha^1 y^1 + \alpha^2 y^0$$

③ 生成多項式を求める。
(3) Obtain the generator polynomial.

$$G(y) = \prod_{i=l}^{l+2t-1}(y - \alpha^i) = \prod_{i=0}^{3}(y - \alpha^i) = (y - \alpha^0)(y - \alpha^1)(y - \alpha^2)(y - \alpha^3)$$
$$= y^4 + \alpha^2 y^3 + \alpha^5 y^2 + \alpha^5 y^1 + \alpha^6$$

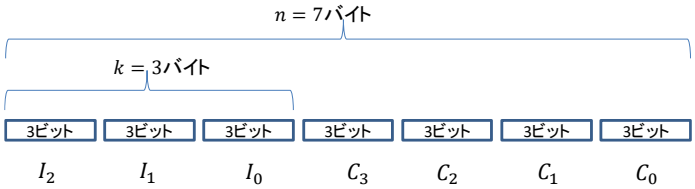
④ 情報多項式$I(y) * y^{n-k}$を生成多項式$G(y)$　で除算した余りを求める。
(4) Find the remainder after dividing the information polynomial I(y)*y^(n-k) by the generator polynomial G(y).

$I(y) = I_{k-1}y^{k-1} + \cdots I_1 y + I_0 = 1y^2 + \alpha^1 y^1 + \alpha^2 y^0$

$r(y) = I(y) * y^{n-k} \; mod \; G(y) = (1y^2 + \alpha^1 y^1 + \alpha^2 y^0) * y^4 \; mod \; y^4 + \alpha^2 y^3 + \alpha^5 y^2 + \alpha^5 y^1 + \alpha^6$

$= \alpha^4 y^3 + \alpha^6 y^2 + \alpha^5 y + \alpha^3$

⑤ 以上より



(5) From the above, finally we obtain the encoded result.

情報ビット	001	010	100				
情報バイト	1	α^1	α^2				
符号化後	1	α^1	α^2	α^4	α^6	α^5	α^3
RS 符号語	001	010	100	110	101	111	011
バイト番号	1	2	3	4	5	6	7

Info. bit	001	010	100				
Info. byte	1	α^1	α^2				
RS code	1	α^1	α^2	α^4	α^6	α^5	α^3
RS code	001	010	100	110	101	111	011
Byte #	1	2	3	4	5	6	7

３－３）誤りがある受信語の誤り訂正

3-3) Error correction of erroneous received words

$u(x)$ を送信し、2 バイト誤り $e(x)$ が生じた $v(x)$ を受信したとする。

Suppose u(x) is sent and v(x) with a 2-byte-error e(x) is received.

送信語 $u(x) = 1x^6 + \alpha^1 x^5 + \alpha^2 x^4 + \alpha^4 x^3 + \alpha^6 x^2 + \alpha^5 x + \alpha^3$

受信語 $v(x) = 1x^6 + 0x^5 + \alpha^2 x^4 + \alpha^5 x^3 + \alpha^6 x^2 + \alpha^5 x + \alpha^3$

誤り多項式　$e(x) = e_i x^i + e_j x^j$　　iバイト目の誤りe_i、jバイト目の誤りe_j

Sent word $u(x) = 1x^6 + \alpha^1 x^5 + \alpha^2 x^4 + \alpha^4 x^3 + \alpha^6 x^2 + \alpha^5 x + \alpha^3$

Received word $v(x) = 1x^6 + 0x^5 + \alpha^2 x^4 + \alpha^5 x^3 + \alpha^6 x^2 + \alpha^5 x + \alpha^3$

Error polynomial　$e(x) = e_i x^i + e_j x^j$　　e_i: error at i-th byte e_j: error at j-th byte

ステップ１）式 2-1～2-4（の前半部；$S_h = v(\alpha^h)$）により、S_0～S_3　を求める。

Step 1) Calculate S_0 to S_3 by formulas 2-1 to 2-4 (the first half of them: S_h=v(α^h)).

$S_0 = v(1) = 1 + 0 + \alpha^2 + \alpha^5 + \alpha^6 + \alpha^5 + \alpha^3 = (001) + (100) + (101) + (011) = (011) = \alpha^3$

$S_1 = v(\alpha) = 1\alpha^6 + 0 + \alpha^2\alpha^4 + \alpha^5\alpha^3 + \alpha^6\alpha^2 + \alpha^5\alpha + \alpha^3 = \alpha^6 + \alpha^6 + \alpha^8 + \alpha^8 + \alpha^6 + \alpha^3 = \alpha^6 + \alpha^3 = (101) + (011) = (110) = \alpha^4$

$S_2 = v(\alpha^2) = 1\alpha^{12} + 0 + \alpha^2\alpha^8 + \alpha^5\alpha^6 + \alpha^6\alpha^4 + \alpha^5\alpha^2 + \alpha^3 = \alpha^{12} + \alpha^{10} + \alpha^{11} + \alpha^{10} + \alpha^7 + \alpha^3 = \alpha^5 + \alpha^4 + 1 + \alpha^3 = (111) + (110) + (001) + (011) = (011) = \alpha^3$

$S_3 = v(\alpha^3) = 1\alpha^{18} + 0 + \alpha^2\alpha^{12} + \alpha^5\alpha^9 + \alpha^6\alpha^6 + \alpha^5\alpha^3 + \alpha^3 = \alpha^{18} + \alpha^{14} + \alpha^{14} + \alpha^{12} + \alpha^8 + \alpha^3 = \alpha^4 + \alpha^5 + \alpha^1 + \alpha^3 = (110) + (111) + (010) + (011) = (000) = 0$

ステップ２）式 2-10～2-11 により、σ_0、σ_1　を求める。

Step 2) Obtain σ_0 and σ_1 from equations 2-10 to 2-11.

$S_2 + \sigma_1 S_1 + \sigma_0 S_0 = 0$　　式 2-10

$\alpha^3 + \sigma_1\alpha^4 + \sigma_0\alpha^3 = 0$　　式 2-2-1

$S_3 + \sigma_1 S_2 + \sigma_0 S_1 = 0$　　式 2-11

$0 + \sigma_1\alpha^3 + \sigma_0\alpha^4 = 0$　　式 2-2-2

式 2-2-2 と式 2-2-1 より以下を得る。

The followings are obtained from Equations 2-2-2 and 2-2-1.

$\alpha^3 + \sigma_0\alpha^1\alpha^4 + \sigma_0\alpha^3 = 0$

$\alpha^3 + \sigma_0(\alpha^5 + \alpha^3) = 0$

$\alpha^5 + \alpha^3 = (111) + (011) = (100) = \alpha^2$

$\alpha^3 + \sigma_0\alpha^2 = 0$　　$\sigma_0 = \alpha^1$

$\sigma_1 = \sigma_0\alpha^1 = \alpha^2$

以上より、$\sigma_0 = \alpha^1$　　$\sigma_1 = \alpha^2$

Then, $\sigma_0 = \alpha^1$　　$\sigma_1 = \alpha^2$

ステップ３）式 2-5 を用いて、α^i、α^j　を求める。

Step 3) Use Equation 2-5 to find α^i and α^j.

つまり、$\sigma(x) = x^2 + \sigma_1 x + \sigma_0 = (x - \alpha^i)(x - \alpha^j)$ となるように因数分解して、α^i、α^j　を定め、i、jを求める。

In other words, factorize so that σ(x)=x^2+σ_1 x+σ_0=(x-α^i)(x-α^j) to determine α^i and α^j, and then, i , j.

$\sigma(x) = x^2 + \sigma_1 x + \sigma_0 = x^2 + \alpha^2 x + \alpha^1 = (x + \alpha^3)(x + \alpha^5)$

（xにα　のべき乗を入れて確認する。α^3を入れると 0 となることが分かれば、$\alpha^3 * y = \alpha^1$　から、$y = \alpha^5$ と求められる。その後、$\alpha^3 + \alpha^5 = (011) + (111) = (100) = \alpha^2$　を確認する。）

(Try putting some power of α into x. Once you find α^3 produces 0, you can get y=α^5 from α^3*y=α^1. Also check ^3+α^5=(011)+(111)=(100)=α^2.)

これより、$\alpha^i = \alpha^3$、$\alpha^j = \alpha^5$　となり、$i = 3$、$j = 5$ が決定される。

Finally we found α^i=α^3, α^j=α^5, and then i=3, j=5.

ステップ４）式 2-1～2-2 により、e_i、e_j　を求める。

Step 4) Obtain e_i and e_j from formulas 2-1 and 2-2.

$S_0 = v(\alpha^0) = v(1) = u(1) + e(1) = e(1) = e_i + e_j = e_3 + e_5 = \alpha^3$　　式 2-4-1

$S_1 = v(\alpha^1) = v(\alpha) = u(\alpha) + e(\alpha) = e(\alpha) = e_i\alpha^i + e_j\alpha^j = e_3\alpha^3 + e_5\alpha^5 = \alpha^4$　式 2-4-2

式 2-4-1 と式 2-4-2 から以下を得る。
From Equations 2-4-1 and 2-4-2, we obtain the following.

$e_3\alpha^3 + (\alpha^3 + e_3)\alpha^5 = \alpha^4$
$(\alpha^3 + \alpha^5)e_3 + \alpha^8 = \alpha^4$
$\alpha^3 + \alpha^5 = (011) + (111) = (100) = \alpha^2$
$\alpha^8 + \alpha^4 = \alpha^1 + \alpha^4 = (010) + (110) = (100) = \alpha^2$

よって
Therefore,

$\alpha^2 e_3 = \alpha^2 \qquad e_3 = 1$
$e_5 = \alpha^3 + e_3 = \alpha^3 + 1 = (011) + (001) = (010) = \alpha^1$

$e_3 = 1 \qquad e_5 = \alpha$

ステップ５）i、j、e_i、e_j　から、誤り多項式　$e(x) = e_i x^i + e_j x^j$　を決定し、$u(x) = v(x) - e(x)$ で誤り訂正を行う。

Step 5) From i, j, e_i, e_j, determine the error polynomial e(x)=e_i x^i+e_j x^j and correct the errors with u(x)=v(x)-e(x).

$e(x) = e_i x^i + e_j x^j = e_3 x^3 + e_5 x^5 = 1x^3 + \alpha x^5$
$v(x) = 1x^6 + 0x^5 + \alpha^2 x^4 + \alpha^5 x^3 + \alpha^6 x^2 + \alpha^5 x + \alpha^3$
$u(x) = v(x) - e(x) = (1x^6 + 0x^5 + \alpha^2 x^4 + \alpha^5 x^3 + \alpha^6 x^2 + \alpha^5 x + \alpha^3) - (1x^3 + \alpha x^5)$
$= 1x^6 + (0 + \alpha^1)x^5 + \alpha^2 x^4 + (\alpha^5 + 1)x^3 + \alpha^6 x^2 + \alpha^5 x + \alpha^3$
$= 1x^6 + \alpha^1 x^5 + \alpha^2 x^4 + \alpha^4 x^3 + \alpha^6 x^2 + \alpha^5 x + \alpha^3$
$\because 1 + \alpha^5 = (001) + (111) = (110) = \alpha^4$

以上より、正しく復号された。
It was decoded correctly.

以下、式 2-3, 2-4 が満たされるか確認する。
Below, we confirm whether the formulas 2-3 and 2-4 are satisfied.

$h = 2 \quad S_2 = v(\alpha^2) = u(\alpha^2) + e(\alpha^2) = e(\alpha^2) = e_i\alpha^{2i} + e_j\alpha^{2j}$ 式 2-3

$h = 3 \quad S_3 = v(\alpha^3) = u(\alpha^3) + e(\alpha^3) = e(\alpha^3) = e_i\alpha^{3i} + e_j\alpha^{3j}$ 式 2-4

$S_2 = v(\alpha^2) = e(\alpha^2) = e_i\alpha^{2i} + e_j\alpha^{2j} = e_3\alpha^6 + e_5\alpha^{10} = 1\alpha^6 + \alpha^1\alpha^{10} = \alpha^6 + \alpha^{11} = \alpha^6 + \alpha^4 = (101) + (110) = (011) = \alpha^3$

$S_2 = v(\alpha^2) = 1\alpha^{12} + 0\alpha^{10} + \alpha^2\alpha^8 + \alpha^5\alpha^6 + \alpha^6\alpha^4 + \alpha^5\alpha^2 + \alpha^3 = \alpha^{12} + \alpha^{10} + \alpha^{11} + \alpha^{10} + \alpha^7 + \alpha^3 = \alpha^5 + \alpha^4 + 1 + \alpha^3 = (111) + (110) + (001) + (011) = (011) = \alpha^3$

$S_3 = v(\alpha^3) = e(\alpha^3) = e_i\alpha^{3i} + e_j\alpha^{3j} = e_3\alpha^9 + e_5\alpha^{15} = 1\alpha^9 + \alpha^1\alpha^{15} = \alpha^9 + \alpha^{16} = \alpha^2 + \alpha^2 = 0$

$S_3 = v(\alpha^3) = 1\alpha^{18} + 0\alpha^{15} + \alpha^2\alpha^{12} + \alpha^5\alpha^9 + \alpha^6\alpha^6 + \alpha^5\alpha^3 + \alpha^3 = \alpha^{18} + \alpha^{14} + \alpha^{14} + \alpha^{12} + \alpha^8 + \alpha^3 = \alpha^4 + \alpha^5 + \alpha^1 + \alpha^3 = (110) + (111) + (010) + (011) = (000) = 0$

７．２．５　BCH と RS のまとめ

7.2.5 Summary of BCH and RS

・生成多項式$g(x)$がポイント

・巡回符号：$g(x)$はx^n-1を整除する。

・情報記号多項式$I(x)$／生成多項式$g(x)$＝$A(x)$ 余り $r(x)$

$u(x)=I(x)\cdot x^m+r(x)$　$u(x)$は生成多項式$g(x)$で整除される。$u(x)$を送信する。

受信側は受信語$v(x)$が$g(x)$で整除されるかを検査

・$u(x)$が$g(x)$で整除される　＜－＞　$u(\alpha)=0$　　α：$g(x)$の根

$u(x)$が$g(x)$で整除されるから、$u(x)=X(x)g(x)$

αは$g(x)$の根なので、$u(\alpha)=X(\alpha)g(\alpha)$=0

・すべての符号語$u(x)$はαを根として持つ。根αを持つ符号語だけを使う。

・生成多項式$g(x)$

BCH の生成多項式（式 7.1.3.1）と RS の生成多項式（式 7.2.1）の違い。BCH は$GF(2)$上、RS は$GF(2^m)$上（mは検査バイト数）

・BCH と RS 練習問題　補足資料２

- Generator polynomial g(x) is the point
- Cyclic code: g(x) divides x^n-1.
- Information polynomial I(x)/generator polynomial g(x) = A(x) remainder r(x).

Since u(x)=I(x)·x^m + r(x), u(x) is divided by the generator polynomial g(x). Send u(x).

The receiving side checks whether the received word v(x) is divisible by g(x).

- u(x) is divided by g(x) if and only if u(α)=0 α: root of g(x)

Since u(x) is divided by g(x), u(x)=X(x)g(x)

Since α is the root of g(x), u(α)=X(α)g(α)=0

- All codewords u(x) have α as a root. Only codewords with root α are used.
- Generator polynomial g(x)

See the difference between the BCH generator polynomial (formula 7.1.3.1) and the RS generator polynomial (formula 7.2.1).

BCH is on GF(2), RS is on GF(2^m) (m is the number of check bytes)

- Solve supplemental problems to see more BCH and RS for practical use.

GF 計算のためのページ

$GF(2^4)$　原始多項式を$p(x) = x^4 + x + 1$ とする。すなわち$\alpha^4 + \alpha + 1 = 0$。

多項式表現	ベクトル表現	αべき乗表現	
0	0000	0	
1	0001	1	$=\alpha^0$
x	0010	α	$=\alpha^1$
$x + 1$	0011	$\alpha + 1$	$=\alpha^4$
x^2	0100	α^2	$=\alpha^2$
$x^2 + 1$	0101	$\alpha^2 + 1$	$=\alpha^8$
$x^2 + x$	0110	$\alpha^2 + \alpha$	$=\alpha^5$
$x^2 + x + 1$	0111	$\alpha^2 + \alpha + 1$	$=\alpha^{10}$
x^3	1000	α^3	$=\alpha^3$
$x^3 + 1$	1001	$\alpha^3 + 1$	$=\alpha^{14}$
$x^3 + x$	1010	$\alpha^3 + \alpha$	$=\alpha^9$
$x^3 + x + 1$	1011	$\alpha^3 + \alpha + 1$	$=\alpha^7$
$x^3 + x^2$	1100	$\alpha^3 + \alpha^2$	$=\alpha^6$
$x^3 + x^2 + 1$	1101	$\alpha^3 + \alpha^2 + 1$	$=\alpha^{13}$
$x^3 + x^2 + x$	1110	$\alpha^3 + \alpha^2 + \alpha$	$=\alpha^{11}$
$x^3 + x^2 + x + 1$	1111	$\alpha^3 + \alpha^2 + \alpha + 1$	$=\alpha^{12}$

$GF(2^4) = \{0, 1, \alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6, \alpha^7, \alpha^8, \alpha^9, \alpha^{10}, \alpha^{11}, \alpha^{12}, \alpha^{13}, \alpha^{14}\}$

$GF(2)$の上の原始多項式$p(x)$を$p(x) = x^3 + x + 1$とし、この根をαとする。すなわち$\alpha^3 + \alpha + 1 = 0$。

多項式	ベクトル表現	αべき表現	
0	000	0	
1	001	1	$=\alpha^0$
x	010	α	$=\alpha^1$
$x + 1$	011	$\alpha + 1$	$=\alpha^3$
x^2	100	α^2	$=\alpha^2$
$x^2 + 1$	101	$\alpha^2 + 1$	$=\alpha^6$
$x^2 + x$	110	$\alpha^2 + \alpha$	$=\alpha^4$
$x^2 + x + 1$	111	$\alpha^2 + \alpha + 1$	$=\alpha^5$

拡大体は$GF(2^3) = \{0, 1, \alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6\}$

7．2．6　リード・ソロモン符号の追加例と計算方法

以下は暫定、2020 は授業 HP にアップした

1）RS 符号の例２

(15, 9)符号　$m=4, n=15, k=9, t=3$, 原始多項式が$p(x)=x^4+x+1$の場合

多項式表現	ベクトル表現	αべき乗表現	
0	0000	0	
1	0001	1	$=\alpha^0$
x	0010	α	$=\alpha^1$
$x+1$	0011	$\alpha+1$	$=\alpha^4$
x^2	0100	α^2	$=\alpha^2$
x^2+1	0101	α^2+1	$=\alpha^8$
x^2+x	0110	$\alpha^2+\alpha$	$=\alpha^5$
x^2+x+1	0111	$\alpha^2+\alpha+1$	$=\alpha^{10}$
x^3	1000	α^3	$=\alpha^3$
x^3+1	1001	α^3+1	$=\alpha^{14}$
x^3+x	1010	$\alpha^3+\alpha$	$=\alpha^9$
x^3+x+1	1011	$\alpha^3+\alpha+1$	$=\alpha^7$
x^3+x^2	1100	$\alpha^3+\alpha^2$	$=\alpha^6$
x^3+x^2+1	1101	$\alpha^3+\alpha^2+1$	$=\alpha^{13}$
x^3+x^2+x	1110	$\alpha^3+\alpha^2+\alpha$	$=\alpha^{11}$
x^3+x^2+x+1	1111	$\alpha^3+\alpha^2+\alpha+1$	$=\alpha^{12}$

$GF(2^4)=\{0,1,\alpha,\alpha^2,\alpha^3,\alpha^4,\alpha^5,\alpha^6,\alpha^7,\alpha^8,\alpha^9,\alpha^{10},\alpha^{11},\alpha^{12},\alpha^{13},\alpha^{14}\}$

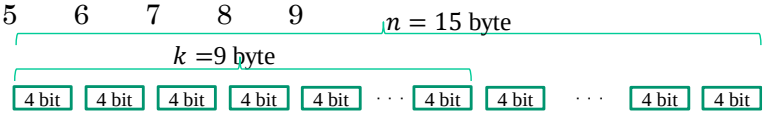
■例２　(15, 9)符号　$m=4, n=15, k=9, t=3$, 原始多項式が$p(x)=x^4+x+1$の場合

①情報ビット系列を 4 ビット$(m=4)$ずつの 9 個$(k=9)$のブロック（バイト）に分割する。例えば、

情報ビット　1010 0011 0011 1101 1110 0101 1101 1000 1010

情報バイト　α^9　α^4　α^4　α^{13}　α^{11}　α^8　α^{13}　α^3　α^9

バイト番号　1　2　3　4　5　6　7　8　9



②情報バイトを多項式で表現する。

$$I(y)=I_{k-1}y^{k-1}+\cdots I_1y+I_0$$
$$=\alpha^9y^8+\alpha^4y^7+\alpha^4y^6+\alpha^{13}y^5+\alpha^{11}y^4+\alpha^8y^3+\alpha^{13}y^2+\alpha^3y^1+\alpha^9$$

③生成多項式を求める。

$$G(y)=\prod_{i=l}^{l+2t-1}(y-\alpha^i)=\prod_{i=0}^{5}(y-\alpha^i)=(y-\alpha^0)(y-\alpha^1)(y-\alpha^2)(y-\alpha^3)(y-\alpha^4)(y-\alpha^5)$$
$$=y^6+\alpha^9y^5+\alpha^{12}y^4+\alpha^1y^3+\alpha^2y^2+\alpha^4y^1+\alpha^0$$

④情報多項式$I(y)*y^{n-k}$を生成多項式$G(y)$　で除算した余りを求める。

$$I(y)=I_{k-1}y^{k-1}+\cdots I_1y+I_0$$
$$=\alpha^9y^8+\alpha^4y^7+\alpha^4y^6+\alpha^{13}y^5+\alpha^{11}y^4+\alpha^8y^3+\alpha^{13}y^2+\alpha^3y^1+\alpha^9$$
$$r(y)=I(y)*y^{n-k}\ mod\ G(y)$$
$$=(\alpha^9y^8+\alpha^4y^7+\alpha^4y^6+\alpha^{13}y^5+\alpha^{11}y^4+\alpha^8y^3+\alpha^{13}y^2+\alpha^3y^1+\alpha^9)*y^6$$
$$mod\ y^6+\alpha^9y^5+\alpha^{12}y^4+\alpha^1y^3+\alpha^2y^2+\alpha^4y^1+\alpha^0$$
$$=\alpha^0y^5+\alpha^8y^4+\alpha^{14}y^3+\alpha^{11}y^2+\alpha^3y+\alpha^3$$

⑤以上より

情報ビット　1010 0011 0011 1101 1110 0101 1101 1000 1010

情報バイト　α^9　α^4　α^4　α^{13}　α^{11}　α^8　α^{13}　α^3　α^9

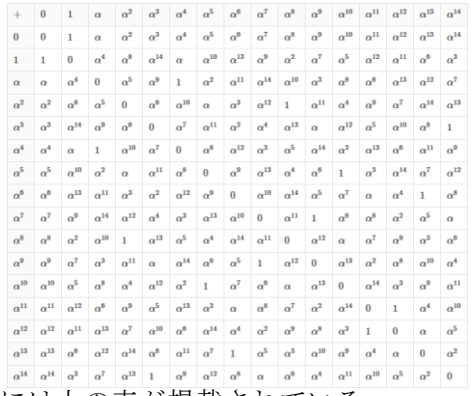
符号化後	α^9	α^4	α^4	α^{13}	α^{11}	α^8	α^{13}	α^3	α^9	α^0	α^8	α^{14}	α^{11}	α^3	α^3
RS 符号語	1010	0011	0011	1101	1110	0101	1101	1000	1010	0001	0101	1001	1110	1000	1000
バイト番号	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

となる。

剰余計算の方法１：ストレートな方法

```
                  9    7   13
                  ---------------------
0 9 12 1 2 4 0|9    4    4    13    11    8    13    3    9    x    x    x    x    x    x
              9   18   21   10    11   13    9
                  =3   =6
   ---------------------------------------------------------------
                  4+3  4+6   13+10  x   8+13  13+9
                  =7   =12   =9     x   =3    =10    3
                  7    16    19     8    9     11     7
                       =1    =4
   ---------------------------------------------------------------
                       12+1  9+4     8   3+9   10+11  3+7
                       =13   =14     8   =1    =14    =4   9
```

：（以下省略）



この計算には、GF(2^4)の加算表があると便利である。
例えば、https://hg.hatenablog.jp/entry/2018/03/11/011748　には上の表が掲載されている。
（表中の１はα^0であることに注意）

剰余計算の方法２：ユークリッド互除法を用いる

剰余計算の方法３：計算機を利用する。

１）カナダの Department of Electrical and Computer Engineering - University of New Brunswick のサイト

http://www.ee.unb.ca/cgi-bin/tervo/calc2.pl?num=10+3+3+13+14+5+13+8+10+0+0+0+0+0+0&den=1+10+15+2+4+3+1&f=d&p=4&d=1&y=1&m=1

このサイトでは、$GF(2^m)$上の計算を行える。
根のべき乗はベクトル表現を１０進で読んで入力する。

例：α^9 -> (1010) -> [10]

例２の計算

・情報多項式$I(y) * y^{n-k}$を生成多項式$G(y)$　で除算した余りを求める。

$r(y) = I(y) * y^{n-k} \ mod \ G(y)$

$= (\alpha^9 y^8 + \alpha^4 y^7 + \alpha^4 y^6 + \alpha^{13} y^5 + \alpha^{11} y^4 + \alpha^8 y^3 + \alpha^{13} y^2 + \alpha^3 y^1 + \alpha^9) * y^6$

$mod \ y^6 + \alpha^9 y^5 + \alpha^{12} y^4 + \alpha^1 y^3 + \alpha^2 y^2 + \alpha^4 y^1 + \alpha^0$

$= \alpha^1 y^5 + \alpha^1 y^4 + \alpha^7 y^3 + \alpha^3 y^2 + \alpha^{10} y + \alpha^{10}$

・　A 欄　$(\alpha^9 y^8 + \alpha^4 y^7 + \alpha^4 y^6 + \alpha^{13} y^5 + \alpha^{11} y^4 + \alpha^8 y^3 + \alpha^{13} y^2 + \alpha^3 y^1 + \alpha^9) y^6$

=> $(\alpha^9 \ \alpha^4 \ \alpha^4 \ \alpha^{13} \ \alpha^{11} \ \alpha^8 \ \alpha^{13} \ \alpha^3 \ \alpha^9 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0)$　=> [10 3 3 13 14 5 13 8 10 0 0 0 0 0 0]

・　B 欄　$y^6 + \alpha^9 y^5 + \alpha^{12} y^4 + \alpha^1 y^3 + \alpha^2 y^2 + \alpha^4 y^1 + \alpha^0$　=>　$(\alpha^0 \ \alpha^9 \ \alpha^{12} \ \alpha^1 \ \alpha^2 \ \alpha^4 \ \alpha^0)$

=> [1 10 15 2 4 3 1]

・　結果　[1 5 9 14 8 8] -> $\alpha^0\ \alpha^8\ \alpha^{14}\ \alpha^{11}\ \alpha^3\ \alpha^3$　-> $\alpha^0 y^5 + \alpha^8 y^4 + \alpha^{14} y^3 + \alpha^{11} y^2 + \alpha^3 y + \alpha^3$

・生成多項式も unb のサイトを使って計算できる。

$G(y) = \prod_{i=l}^{l+2t-1}(y - \alpha^i) = \prod_{i=0}^{5}(y - \alpha^i) = (y - \alpha^0)(y - \alpha^1)(y - \alpha^2)(y - \alpha^3)(y - \alpha^4)(y - \alpha^5)$

$= y^6 + \alpha^9 y^5 + \alpha^{12} y^4 + \alpha^1 y^3 + \alpha^2 y^2 + \alpha^4 y^1 + \alpha^0$

unb のサイトで

$(y - \alpha^0)$ => [1 1]、$(y - \alpha^1)$ => [1 2] 、$(y - \alpha^2)$ => [1 4] 、$(y - \alpha^3)$ => [1 8]

$(y - \alpha^4)$ => [1 3] 、$(y - \alpha^5)$ => [1 6] として順次かけ算を行う。

結果　[1 10 15 2 4 3 1] => $\alpha^0\ \alpha^9\ \alpha^{12}\ \alpha^1\ \alpha^2\ \alpha^4\ \alpha^0$

２）microjamjar

RS 符号の符号化、復号をデモンストレーションしている。

http://www.ujamjar.com/demo/ocaml/2014/06/18/reed-solomon-demo.html

確認１）３次の例　(7, 3)符号

$I(y) = I_{k-1}y^{k-1} + \cdots + I_1 y + I_0 = 1y^2 + \alpha^1 y^1 + \alpha^2 y^0$

情報ビット 001 010 100

[1 2 4]

m=3, t=2

Primitive polynomial 11 = (1011) = x^3+x+1　　Primitive element=2

Initial root b =0　(l のこと)

Message 4 2 1　　<- 逆順に入れる

Errors　全て 0

とし、calculate を押すと、

$1x^6 + 2x^5 + 4x^4 + 6x^3 + 5x^2 + 7x + 3$

が得られる。すなわち

RS 符号語 001　010　100　110　101　111　011

と一致する。

確認２）４次の例１　(15, 11)符号

$I(y) = I_{k-1}y^{k-1} + \cdots I_1 y + I_0$

$= \alpha^9 y^{10} + \alpha^4 y^9 + \alpha^4 y^8 + \alpha^{13} y^7 + \alpha^{11} y^6 + \alpha^8 y^5 + \alpha^{13} y^4 + \alpha^3 y^3 + \alpha^9 y^2 + \alpha^8 y^1 + 0y^0$

情報ビット 1010 0011 0011 1101 1110 0101 1101 1000 1010 0101 0000

[10 3 3 13 14 5 13 8 10 5 0]

m=4, t=2

Primitive polynomial 19 = (10011) =x^4+x+1　　Primitive element=2

Initial root b =0　(l のこと)

Message　0 5 10 8 13 5 14 13 3 3 10　　<- 逆順に入れる

Errors　全て 0
とし、calculate を押すと、
10x14+3x13+3x12+13x11+14x10+5x9+13x8+8x7+10x6+5x5+13x3+13x2+x+7
が得られる。すなわち
　RS 符号語　1010 0011 0011 1101 1110 0101 1101 1000 1010 0101 0000 1101 1101 0001 0111
と一致する。

確認３）4 次の例２　(15, 9)符号

$$I(y) = I_{k-1}y^{k-1} + \cdots I_1 y + I_0$$
$$= \alpha^9 y^8 + \alpha^4 y^7 + \alpha^4 y^6 + \alpha^{13} y^5 + \alpha^{11} y^4 + \alpha^8 y^3 + \alpha^{13} y^2 + \alpha^3 y^1 + \alpha^9$$

　情報ビット 1010 0011 0011 1101 1110 0101 1101 1000 1010
[10 3 3 13 14 5 13 8 10]

m=4, t=3
Primitive polynomial 19 = (10011) =x^4+x+1　　Primitive element=2
Initial root b =0　(l のこと)
Message 10 8 13 5 14 13 3 3 10　　<- 逆順に入れる
Errors　全て 0
とし、calculate を押すと、
10x14+3x13+3x12+13x11+14x10+5x9+13x8+8x7+10x6+x5+5x4+9x3+14x2+8x+8
が得られる。すなわち
　RS 符号語　1010 0011 0011 1101 1110 0101 1101 1000 1010 0001 0101 1001 1110 1000 1000
と一致する。

７．３　Goppa 符号

7.3 Goppa code

７．３．１　概要

7.3.1 Overview

・1970 年頃 V. D. Goppa が考案（古典 Goppa 符号）。その後 1980 年頃に拡張（代数幾何符号）
・一般的には非巡回符号。巡回符号 BCH, RS を包含する。
・BCH, RS の符号長に関する制約を緩和。
・有理式を用いる。留数型符号
・VG 限界を超える高効率な Goppa 符号があることが報告されている。

Tsfasman, Vladut, Zink, Modular curves, Shimura curves, and Goppa codes, better than Varshamov-Gilbert bound, Mathematische Nachrichten, vol 109, 1982

・MacEliece 暗号の基礎

以下では、古典 Goppa 符号について述べる。

- It is invented by V. D. Goppa around 1970 (classical Goppa code). Later it is expanded around 1980 (algebraic geometric code)
- Typically an acyclic code. It contains cyclic codes BCH, RS.
- Restrictions on code length of BCH and RS are relaxed.
  -- Based on rational functions. It is also called residue type code.
  -- There is a highly efficient Goppa code that exceeds the VG limit.

Tsfasman, Vladut, Zink, Modular curves, Shimura curves, and Goppa codes, better than Varshamov-Gilbert bound, Mathematische Nachrichten, vol. 109, 1982.

- Fundamentals of MacEliece cryptography.

In the following, we discuss classical Goppa codes.

参考：

References:

・Ellen Jochemsz: Goppa Codes and the McEliece Cryptosystem, Vrije Universiteit Amasterdam https://klevas.mif.vu.lt/~skersys/vsd/crypto_on_codes/goppamceliece.pdf

・Elwyn Berlekamp, Goppa Codes, IEEE Transactions on Information Theory, Vol. IT-19, No. 5, Sep. 1973

https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1055088

・上原剛：代数幾何符号に関する研究

http://www.jssac.org/Editor/Suushiki/V09/No2/V9N2_104.pdf

・E. Berlekamp, "Goppa codes," in IEEE Transactions on Information Theory, vol. 19, no. 5, pp. 590-592, September 1973, doi: 10.1109/TIT.1973.1055088.

https://ieeexplore.ieee.org/abstract/document/1055088

７．３．２　定義

7.3.2 Definition

ある原始多項式で生成される拡大体$GF(q^m)$を考える。ただし、qは素数である。
$GF(q^m)$は、原始元αのべき乗と０を要素として持つ。すなわち、

Consider an extension field GF(q^m) generated by a primitive polynomial, where q is a prime number.
GF(q^m) has any power of the primitive element α and 0 as elements.

$$GF(q^m) = \{0, 1, \alpha^1, \alpha^2, ,, , \alpha^{q^m-2}\}$$

$GF(q^m)$上のt次の多項式（ゴッパ多項式）を以下のように定義する。
（生成多項式ではないことに注意せよ）

We define Goppa polynomial as the t-th order polynomial on GF(q^m) as follows. (Note that it is not a generator polynomial)

$$g(z) = \sum_{i=0}^{t} g_i z^i = g_t z^t + g_{t-1} z^{t-1} + g_{t-2} z^{t-2} + \cdots + g_1 z^1 + g_0 \quad (1)$$

$GF(q^m)$のn個の要素からなる部分集合Lを以下とする。

Let the n-element subset L of GF(q^m) be,

$$L = \{\alpha_1, \alpha_2, \alpha_3, ,, , \alpha_n\} \subseteq GF(q^m)$$

ただし、$g(\alpha_i) \neq 0$である。（すなわち、ゴッパ多項式の根ではない）
where g(α_i)≠0. (i.e. each element is not the root of the Goppa polynomial)

Let us consider a vector c={c_1,c_2,c_3,,,c_n} on GF(q) and polynomial R_C(z)=Σ_(i=1)^n c_i/(z-α_i).

The vector c={c_1,c_2,c_3,,,c_n} that satisfies the following equation is called a Goppa code.

$$R_C(z) = \sum_{i=1}^{n} \frac{c_i}{z - \alpha_i} = 0 \, mod \, g(z) \quad (2)$$

Let us consider a vector c={c_1,c_2,c_3,,,c_n} on GF(q) and polynomial R_C(z)=Σ_(i=1)^n c_i/(z-α_i).
The vector c={c_1,c_2,c_3,,,c_n} that satisfies the following equation is called a Goppa code.

$$R_C(z) = \sum_{i=1}^{n} \frac{c_i}{z - \alpha_i} = 0 \, mod \, g(z) \quad (2)$$

７．３．３　ゴッパ符号のパラメータ(n, k, d)
7.3.3 Goppa code parameters (n, k, d)

$g(z)$はt次であるから、$\frac{1}{z-\alpha_i} \; mod \, g(z)$は、$t-1$次以下の多項式で表現できる。すなわち、以下のように表現できる。

Since g(z) is of degree t, 1/(z-α_i) mod g(z) can be expressed by a polynomial of degree t-1 or less. That is, it can be expressed as follows.

$$\frac{1}{z-\alpha_i} = p_i(z) = p_{it}z^{t-1} + p_{i(t-1)}z^{t-2} + p_{i(t-2)}z^{t-2} + \cdots + p_{i2}z + p_{i1} \mod g(z)$$

従って、(2)式は、以下のように書き換えられる。
Therefore, equation (2) can be rewritten as follows.

$$R_C(z) = \sum_{i=1}^{n} \frac{c_i}{z-\alpha_i} = \sum_{i=1}^{n} c_i p_i(z) = \sum_{i=1}^{n} c_i(p_{it}z^{t-1} + p_{i(t-1)}z^{t-2} + p_{i(t-2)}z^{t-2} + \cdots + p_{i2}z + p_{i1})$$
$$= 0 \mod g(z) \quad (3)$$

z^jの項の係数はすべて０である。すなわち、
The coefficients of the terms of z^j are all zero. i.e.

$$\sum_{i=1}^{n} c_i p_{ij} = 0 \ for\ 1 \le j \le t \quad (4)$$

$g(z)$で定義されるゴッパ符号は、$GF(q^m)$上のt個の線形式で定義される。これは、$GF(q)$上でのmt以下の方程式に縮退可能である。従って、kは$n-mt$以上でなければならない。従って、$k \ge n-mt$ である。

A Goppa code defined by g(z) is defined by t linear equations over GF(q^m).
It is degradable to at most mt equations on GF(q).
Therefore, k must be greater than or equal to n - mt. Therefore, k ≥ n-mt.

また、

$$\sum_{i=1}^{n} \frac{c_i}{z-\alpha_i} = \frac{\sum_{j=1}^{\omega} c_{ij} \prod_{1 \le k \le \omega, k \ne j}(z-\alpha_{i_k})}{\prod_{j=1}^{\omega}(z-\alpha_{i_j})}$$

であるため、分母は$g(z)$と共通の因子を持たないので、$g(z)$分子を割り切らねばならない。

分子の次元は、$\omega-1$以下であるので、$\omega-1 \ge t$となり、これにより最小距離は

$d \geq t+1$　　となる。

符号長n, 情報ビットk, 最小距離dのゴッパ符号を(n,k,d)ゴッパ符号と表現する。

From the above equation, the denominator has no factors in common with g(z), so that we must divide the numerator of g(z).
Since the dimension of the numerator is less than or equal to ω-1, ω-1≥t, so the minimum distance is d≥t+1.
A Goppa code with code length n, information bit k, and minimum distance d is expressed as Goppa (n, k, d).

７．３．４　検査行列と生成行列

7.3.4 Check matrix and generator matrix

式(4)を再掲する。

Let us see the equation (4) again.

$$\sum_{i=1}^{n} c_i p_{ij} = 0 \ \ for\ 1 \leq j \leq t$$

すべてのjに関してまとめて以下のように表現できる。

For all j, we can express as follows.

$$c = (c_1, c_2, c_3, , , c_n)$$

$$(c_1, c_2, c_3, , , c_n) \begin{pmatrix} p_{11} & p_{12} & p_{13} & & p_{1t} \\ p_{21} & p_{22} & p_{23} & \cdots & \\ p_{31} & & & & \\ & \vdots & & \ddots & \\ p_{n1} & & & & p_{nt} \end{pmatrix} = 0$$

係数　　z^0　z^1　z^2　　z^{t-1}

これは、検査ルールを表現している。したがって、検査行列Hを以下のように定義すれば、検査ルールを簡単に表現できる。

The equation means the checking rule.
The checking rule can be easily expressed if we define the check matrix H as follows.

$$H = \begin{pmatrix} p_{11} & p_{21} & p_{31} & & p_{n1} \\ p_{12} & p_{22} & p_{32} & \cdots & \\ p_{13} & & & & \\ & \vdots & & \ddots & \\ p_{1t} & & & & p_{nt} \end{pmatrix}$$

$$cH^T = 0$$

なお、Hは、$t \times n$行列である。
Note that H is a t×n matrix.

さらに、
Moreover, since

$$p_i(z) = \frac{1}{z-\alpha_i} = \frac{g(z)-g(\alpha_i)}{z-\alpha_i}\frac{-1}{g(\alpha_i)} \quad mod\ g(z)$$

（証明は※１）
(Proof is *1)

※１ ----------

$$(z-\alpha_i)\frac{g(z)-g(\alpha_i)}{z-\alpha_i}\frac{-1}{g(\alpha_i)} = \frac{-1}{g(\alpha_i)}\big(g(z)-g(\alpha_i)\big) = \frac{-1}{g(\alpha_i)}g(z) + \frac{g(\alpha_i)}{g(\alpha_i)} = \frac{-1}{g(\alpha_i)}g(z) + 1$$
$$= 1\ mod\ g(z)$$

よって、
Thus,

$$\frac{g(z)-g(\alpha_i)}{z-\alpha_i}\frac{-1}{g(\alpha_i)}=\frac{1}{z-\alpha_i}$$

$$p_i(z)=\frac{1}{z-\alpha_i}=-\frac{g_t z^t+g_{t-1}z^{t-1}+g_{t-2}z^{t-2}+\cdots+g_1 z^1+g_0-g(\alpha_i)}{z-\alpha_i}\frac{1}{g(\alpha_i)}$$

$$=-\frac{g_t(z^t-{\alpha_i}^t)+g_{t-1}(z^{t-1}-{\alpha_i}^{t-1})+g_{t-2}(z^{t-2}-{\alpha_i}^{t-2})+\cdots+g_1(z^1-{\alpha_i}^1)+g_0(z^0-{\alpha_i}^0)}{z-\alpha_i}\frac{1}{g(\alpha_i)}$$

$$=-\{(g_t(z^{t-1}+z^{t-2}{\alpha_i}^1+z^{t-3}{\alpha_i}^2+\cdots+{\alpha_i}^{t-1})+g_{t-1}(z^{t-2}+z^{t-3}{\alpha_i}^1+z^{t-4}{\alpha_i}^2+\cdots+{\alpha_i}^{t-2})+\cdots$$

$$+g_2(z^1+{\alpha_i}^1)+g_1\}\frac{1}{g(\alpha_i)}$$

と変形でき、この式と次式のz^jの係数を比較して以下を得る。
Then, by comparing this equation with the coefficiet of z^j of the following equations, we obtain,

$$p_i(z)=p_{it}z^{t-1}+p_{i(t-1)}z^{t-2}+p_{i(t-2)}z^{t-2}+\cdots+p_{i2}z+p_{i1}$$

z^0: $p_{i1}=-(g_t{\alpha_i}^{t-1}+g_{t-1}{\alpha_i}^{t-2}+g_{t-2}{\alpha_i}^{t-3}+\cdots+g_2{\alpha_i}^1+g_1)\frac{1}{g(\alpha_i)}$

z^1: $p_{i2}=-(g_t{\alpha_i}^{t-2}+g_{t-1}{\alpha_i}^{t-3}+g_{t-2}{\alpha_i}^{t-4}+\cdots+g_2)\frac{1}{g(\alpha_i)}$

z^2: $p_{i3}=-(g_t{\alpha_i}^{t-3}+g_{t-1}{\alpha_i}^{t-4}+g_{t-2}{\alpha_i}^{t-5}+\cdots+g_3)\frac{1}{g(\alpha_i)}$

z^{t-2}: $p_{i(t-1)}=-\frac{(g_t{\alpha_i}^1+g_{t-1})1}{g(\alpha_i)}$

z^{t-1}: $p_{it}=-g_t\frac{1}{g(\alpha_i)}$

したがって、以下のように検査行列Hを３つの行列の積で表現できる。

Therefore, the parity check matrix H can be expressed as a product of three matrices as follows.

$$H = \begin{pmatrix} p_{11} & p_{21} & p_{31} & & p_{n1} \\ p_{12} & p_{22} & p_{32} & \cdots & \\ p_{13} & & & & \\ & \vdots & & \ddots & \\ p_{1t} & & & & p_{nt} \end{pmatrix} = XYZ$$

$$X = \begin{pmatrix} -g_t & -g_{t-1} & -g_{t-2} & & -g_1 \\ 0 & -g_t & -g_{t-1} & \cdots & -g_2 \\ 0 & 0 & -g_t & & -g_3 \\ & \vdots & & \ddots & \\ 0 & 0 & 0 & & -g_t \end{pmatrix}$$

Xは$t \times t$行列

X is a t×t matrix

$$Y = \begin{pmatrix} {\alpha_1}^{t-1} & {\alpha_2}^{t-1} & {\alpha_3}^{t-1} & & {\alpha_n}^{t-1} \\ {\alpha_1}^{t-2} & {\alpha_2}^{t-2} & {\alpha_3}^{t-2} & \cdots & {\alpha_n}^{t-2} \\ {\alpha_1}^{t-3} & {\alpha_2}^{t-3} & {\alpha_3}^{t-3} & & {\alpha_n}^{t-3} \\ & \vdots & & \ddots & \\ {\alpha_1}^{1} & {\alpha_2}^{1} & {\alpha_3}^{1} & & {\alpha_n}^{1} \\ 1 & 1 & 1 & & 1 \end{pmatrix}$$

Yは$t \times n$行列

Y is a t×n matrix

$$Z = \begin{pmatrix} \frac{1}{g(\alpha_1)} & 0 & 0 & & 0 \\ 0 & \frac{1}{g(\alpha_2)} & 0 & \cdots & 0 \\ 0 & 0 & \frac{1}{g(\alpha_3)} & & 0 \\ & \vdots & & \ddots & \\ 0 & 0 & 0 & & \frac{1}{g(\alpha_n)} \end{pmatrix}$$

Zは$n \times n$行列

Z is a n×n matrix

生成行列Gは、$GH^T = 0$より得られる。

（より正確には、Hの零空間(null space)の基底を求め、その基底から G を決定する。）

The generator matrix G is obtained from GH^T=0.

(More precisely, find the basis of the null space of H, and determine G from that basis.)

７．３．５　Goppa 符号の例

7.3.5 Examples of Goppa codes

■例１

■ Example 1

$GF(2^4)$上の Goppa 符号を考察する。$q = 2,\ \ m = 4$

Consider a Goppa code over GF(2^4). q=2, m=4

１）

$$x^{15} + 1 = (x + 1)(x^2 + x + 1)(x^4 + x^3 + 1)(x^4 + x + 1)(x^4 + x^3 + x^2 + x + 1)$$

２）原始多項式を$ap(x) = x^4 + x + 1$　とする。

2) Let the primitive polynomial be ap(x)=x^4+x+1.

・$GF(2)$上の任意の多項式を原始多項式$ap(x) = x^4 + x + 1$ (ベクトル表現 10011)で除した余りの集合は、以下となる。

- The set of remainders obtained by dividing any polynomial on GF (2) by the primitive polynomial x ^ 4 + x + 1 (vector expression is 10011) is as follows.

Polynomial expression Vector expression α exponentiation expression

多項式	ベクトル表現	αべき表現
0	0000	0
1	0001	α^0
x	0010	α^1
$x + 1$	0011	α^4

x^2		0100	α^2
$x^2 + 1$	0101	α^8	
$x^2 + x$		0110	α^5
$x^2 + x + 1$	0111	α^{10}	
x^3		1000	α^3
$x^3 + 1$		1001	α^{14}
$x^3 + x$		1010	α^9
$x^3 + x + 1$	1011	α^7	
$x^3 + x^2$		1100	α^6
$x^3 + x^2 + 1$	1101	α^{13}	
$x^3 + x^2 + x$		1110	α^{11}
$x^3 + x^2 + x + 1$		1111	α^{12}

拡大体は、$GF(2^4) = \{0, 1, \alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6, \alpha^7, \alpha^8, \alpha^9, \alpha^{10}, \alpha^{11}, \alpha^{12}, \alpha^{13}, \alpha^{14}\}$である。

The extension field is

$GF(2^4) = \{0, 1, \alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6, \alpha^7, \alpha^8, \alpha^9, \alpha^{10}, \alpha^{11}, \alpha^{12}, \alpha^{13}, \alpha^{14}\}$

３）ゴッパ集合Lを以下とする。$L \subseteq GF(2^4)$　（Lは$GF(2^4)$の部分集合）

$$L = \{\alpha^i \; such \; that \; 2 \le i \le 13\} = \{\alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6, \alpha^7, \alpha^8, \alpha^9, \alpha^{10}, \alpha^{11}, \alpha^{12}, \alpha^{13}\}$$

ゴッパ集合Lの要素数により、符号長nが決定される。この場合は、$n = 12$である。

3) Let the Goppa set L be L⊆GF(2^4) (L is a subset of GF(2^4))

L={α^i such that 2≤i≤13}={α^2, α^3, α^4, α^5, α^6, α^7, α^8, α^9, α^10, α^11, α^12, α^13}

The code length n is determined by the number of elements in the Goppa set L. In this case n=12.

４）ゴッパ多項式 $g(z)$を決める。$g(z)$の根は、ゴッパ集合Lに含まれないようにする。

ここでは、以下とする。（separable; 重根を持たない場合）

4) Determine the Goppa polynomial g(z). The root of g(z) should not be included in the Goppa set L.

Here, we have the following g(z) for example. (separable; without multiple roots)

$$g(z) = (z + \alpha^1)(z + \alpha^{14}) = z^2 + \alpha^7 z + 1 = g_2 z^2 + g_1 z + g_0$$

（　∵　$\alpha^1 + \alpha^{14} = (0\,0\,1\,0) + (1\,0\,0\,1) = (1\,0\,1\,1) = \alpha^7$　　）

５）以上より、このゴッパ符号のパラメータは以下となる。

5) From the above, the parameters of this Goppa code are as follows.

$$q = 2, \quad m = 4, \quad n = 12, \quad t = 2, \quad d \geq 2t + 1 = 4 + 1 = 5$$

$\alpha_1 = \alpha^2$ $\alpha_2 = \alpha^3$ $\alpha_3 = \alpha^4$ $\alpha_4 = \alpha^5$ $\alpha_5 = \alpha^6$ $\alpha_6 = \alpha^7$ $\alpha_7 = \alpha^8$ $\alpha_8 = \alpha^9$ $\alpha_9 = \alpha^{10}$ $\alpha_{10} = \alpha^{11}$ $\alpha_{11} = \alpha^{12}$ $\alpha_{12} = \alpha^{13}$

$g_0 = 1$　$g_1 = \alpha^7$　$g_2 = 1$

これにより、$(12, \geq 4, \geq 5)$Goppa 符号が構成される。

These parameters constructs a $(12, \geq 4, \geq 5)$ Goppa code.

６）検査行列 H を求める。

6) Find the parity check matrix H.

$H = XYZ$

$X: t \times t = 2 \times 2$、$Y: t \times n = 2 \times 12$、$Z: n \times n = 12 \times 12$、$H: t \times n = 2 \times 12$

まず、X行列を求める。

First, find the X matrix.

$$X = \begin{pmatrix} -g_t & -g_{t-1} & -g_{t-2} & \cdots & -g_1 \\ 0 & -g_t & -g_{t-1} & & -g_2 \\ 0 & 0 & -g_t & & -g_3 \\ & \vdots & & \ddots & \\ 0 & 0 & 0 & & -g_t \end{pmatrix} = \begin{pmatrix} -g_2 & -g_1 \\ 0 & -g_2 \end{pmatrix} = \begin{pmatrix} 1 & \alpha^7 \\ 0 & 1 \end{pmatrix}$$

次に、Y行列を求める。

Next, find the Y matrix.

$$Y = \begin{pmatrix} {\alpha_1}^{t-1} & {\alpha_2}^{t-1} & {\alpha_3}^{t-1} & \cdots & {\alpha_n}^{t-1} \\ {\alpha_1}^{t-2} & {\alpha_2}^{t-2} & {\alpha_3}^{t-2} & & {\alpha_n}^{t-2} \\ {\alpha_1}^{t-3} & {\alpha_2}^{t-3} & {\alpha_3}^{t-3} & & {\alpha_n}^{t-3} \\ & \vdots & & & \\ {\alpha_1}^{1} & {\alpha_2}^{1} & {\alpha_3}^{1} & \ddots & {\alpha_n}^{1} \\ 1 & 1 & 1 & & 1 \end{pmatrix}$$

$$= \begin{pmatrix} \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 & \alpha^7 & \alpha^8 & \alpha^9 & \alpha^{10} & \alpha^{11} & \alpha^{12} & \alpha^{13} \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

そして、Z行列を求める。

Then, find the Z matrix.

$$\frac{1}{g(\alpha_1)} = \frac{1}{(\alpha^2)^2+\alpha^7\alpha^2+1} = \frac{1}{\alpha^4+\alpha^9+1} = \{(0011)+(1010)+(0001)\}^{-1} = \{(1000)\}^{-1} = \{\alpha^3\}^{-1} = \alpha^{-3} = \alpha^{12}$$

$$\frac{1}{g(\alpha_2)} = \frac{1}{(\alpha^3)^2+\alpha^7\alpha^3+1} = \frac{1}{\alpha^6+\alpha^{10}+1} = \{(1100)+(0111)+(0001)\}^{-1} = \{(1010)\}^{-1} = \{\alpha^9\}^{-1} = \alpha^{-9} = \alpha^6$$

$$\frac{1}{g(\alpha_3)} = \frac{1}{(\alpha^4)^2+\alpha^7\alpha^4+1} = \frac{1}{\alpha^8+\alpha^{11}+1} = \{(0101)+(1110)+(0001)\}^{-1} = \{(1010)\}^{-1} = \{\alpha^9\}^{-1} = \alpha^{-9} = \alpha^6$$

$$\frac{1}{g(\alpha_4)} = \frac{1}{(\alpha^5)^2+\alpha^7\alpha^5+1} = \frac{1}{\alpha^{10}+\alpha^{12}+1} = \{(0111)+(1111)+(0001)\}^{-1} = \{(1001)\}^{-1} = \{\alpha^{14}\}^{-1} = \alpha^{-14} = \alpha^1$$

$$\frac{1}{g(\alpha_5)} = \frac{1}{(\alpha^6)^2+\alpha^7\alpha^6+1} = \frac{1}{\alpha^{12}+\alpha^{13}+1} = \{(1111)+(1101)+(0001)\}^{-1} = \{(0011)\}^{-1} = \{\alpha^4\}^{-1} = \alpha^{-4} = \alpha^{11}$$

$$\frac{1}{g(\alpha_6)} = \frac{1}{(\alpha^7)^2+\alpha^7\alpha^7+1} = \frac{1}{\alpha^{14}+\alpha^{14}+1} = \{1\}^{-1} = 1$$

$$\frac{1}{g(\alpha_7)} = \frac{1}{(\alpha^8)^2+\alpha^7\alpha^8+1} = \frac{1}{\alpha^{16}+\alpha^{15}+1} = \frac{1}{\alpha^1+1+1} = \alpha^{-1} = \alpha^{14}$$

$$\frac{1}{g(\alpha_8)} = \frac{1}{(\alpha^9)^2+\alpha^7\alpha^9+1} = \frac{1}{\alpha^{18}+\alpha^{16}+1} = \frac{1}{\alpha^3+\alpha^1+1} = \{(1000)+(0010)+(0001)\}^{-1} = \{(1011)\}^{-1} = \{\alpha^7\}^{-1} = \alpha^{-7} = \alpha^8$$

$$\frac{1}{g(\alpha_9)} = \frac{1}{(\alpha^{10})^2+\alpha^7\alpha^{10}+1} = \frac{1}{\alpha^{20}+\alpha^{17}+1} = \frac{1}{\alpha^5+\alpha^2+1} = \{(0110)+(0100)+(0001)\}^{-1} = \{(0011)\}^{-1} = \{\alpha^4\}^{-1} = \alpha^{-4} = \alpha^{11}$$

$$\frac{1}{g(\alpha_{10})} = \frac{1}{(\alpha^{11})^2+\alpha^7\alpha^{11}+1} = \frac{1}{\alpha^{22}+\alpha^{18}+1} = \frac{1}{\alpha^7+\alpha^3+1} = \{(1011)+(1000)+(0001)\}^{-1} = \{(0010)\}^{-1} = \{\alpha^1\}^{-1} = \alpha^{-1} = \alpha^{14}$$

$$\frac{1}{g(\alpha_{11})} = \frac{1}{(\alpha^{12})^2+\alpha^7\alpha^{12}+1} = \frac{1}{\alpha^{24}+\alpha^{19}+1} = \frac{1}{\alpha^9+\alpha^4+1} = \{(1010)+(0011)+(0001)\}^{-1} = \{(1000)\}^{-1} = \{\alpha^3\}^{-1} = \alpha^{-3} = \alpha^{12}$$

$\frac{1}{g(\alpha_{12})} = \frac{1}{(\alpha^{13})^2+\alpha^7\alpha^{13}+1} = \frac{1}{\alpha^{26}+\alpha^{20}+1} = \frac{1}{\alpha^{11}+\alpha^5+1} = \{(1110)+(0110)+(0001)\}^{-1} = \{(1001)\}^{-1} = \{\alpha^{14}\}^{-1} = \alpha^{-14} = \alpha^1$

$$Z = \begin{pmatrix} \frac{1}{g(\alpha_1)} & 0 & 0 & & 0 \\ 0 & \frac{1}{g(\alpha_2)} & 0 & \cdots & 0 \\ 0 & 0 & \frac{1}{g(\alpha_3)} & & 0 \\ & \vdots & & \ddots & \\ 0 & 0 & 0 & & \frac{1}{g(\alpha_n)} \end{pmatrix}$$

$$= \begin{pmatrix} \alpha^{12} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \alpha^6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \alpha^6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \alpha^1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \alpha^{11} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \alpha^{14} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha^8 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha^{11} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha^{14} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha^{12} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha^1 \end{pmatrix}$$

従って、H は以下のように求められる。

Therefore, H is calculated as follows.

$$H = XYZ = \begin{pmatrix} 1 & \alpha^7 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 & \alpha^7 & \alpha^8 & \alpha^9 & \alpha^{10} & \alpha^{11} & \alpha^{12} & \alpha^{13} \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} \alpha^{12} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \alpha^6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \alpha^6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \alpha^1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \alpha^{11} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \alpha^{14} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha^8 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha^{11} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha^{14} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha^{12} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha^1 \end{pmatrix}$$

$$= \begin{pmatrix} (\alpha^2+\alpha^7) & (\alpha^3+\alpha^7) & (\alpha^4+\alpha^7) & (\alpha^5+\alpha^7) & (\alpha^6+\alpha^7) & (\alpha^7+\alpha^7) & (\alpha^8+\alpha^7) & (\alpha^9+\alpha^7) & (\alpha^{10}+\alpha^7) & (\alpha^{11}+\alpha^7) & (\alpha^{12}+\alpha^7) & (\alpha^{13}+\alpha^7) \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$\begin{pmatrix} \alpha^{12} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \alpha^6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \alpha^6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \alpha^1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \alpha^{11} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \alpha^{14} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha^8 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha^{11} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha^{14} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha^{12} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha^1 \end{pmatrix} = \begin{pmatrix} \alpha^9 & \alpha^{10} & \alpha^9 & \alpha^{14} & \alpha^6 & 0 & \alpha^{10} & \alpha^8 & \alpha^2 & \alpha^7 & \alpha^{14} & \alpha^6 \\ \alpha^{12} & \alpha^6 & \alpha^6 & \alpha^1 & \alpha^{11} & 1 & \alpha^{14} & \alpha^8 & \alpha^{11} & \alpha^{14} & \alpha^{12} & \alpha^1 \end{pmatrix}$$

Hの最初の列に注目する。第1行目と第2行目にzをかけたものを加算した$\alpha^9+\alpha^{12}z$は、$\alpha^9+\alpha^{12}z =$

$\frac{1}{z-\alpha^2}$ である。なぜならば、

Focus on the first column of H. α^9+α^12 z, which is the sum of the first and second rows multiplied by z, is α^9+α^12 z=1/(z-α^2). Because,

$$(\alpha^9+\alpha^{12}z)(z-\alpha^2)=\alpha^{12}z^2+\alpha^9z-\alpha^{14}z-\alpha^{11}=\alpha^{12}z^2+\alpha^4z-\alpha^{11}=-\alpha^{11}-\alpha^{12}=1\ mod\ g(z)$$

以下の割り算を参考のこと。

See division below.

$$\begin{array}{lll|lll} 1 & \alpha^7 & 1 & \alpha^{12} & \alpha^4 & \alpha^{11} \\ & & & \alpha^{12} & \alpha^{19} & \alpha^{12} \\ & & & & & \alpha^{11}+\alpha^{12}=1 \end{array}$$

さて、$cH^T=0$であるから以下を得る。

Now, since cH^T=0, we get the following.

$$cH^T=(c_1,c_2,c_3,,,c_{12})\begin{pmatrix} \alpha^9 & \alpha^{10} & \alpha^9 & \alpha^{14} & \alpha^6 & 0 & \alpha^{10} & \alpha^8 & \alpha^2 & \alpha^7 & \alpha^{14} & \alpha^6 \\ \alpha^{12} & \alpha^6 & \alpha^6 & \alpha^1 & \alpha^{11} & 1 & \alpha^{14} & \alpha^8 & \alpha^{11} & \alpha^{14} & \alpha^{12} & \alpha^1 \end{pmatrix}^T=0$$

展開すると、

Expand this.

$$c_1\alpha^9+c_2\alpha^{10}+c_3\alpha^9+c_4\alpha^{14}+c_5\alpha^6+c_60+c_7\alpha^{10}+c_8\alpha^8+c_9\alpha^2+c_{10}\alpha^7+c_{11}\alpha^{14}+c_{12}\alpha^6=0$$
$$c_1\alpha^{12}+c_2\alpha^6+c_3\alpha^6+c_4\alpha^1+c_5\alpha^{11}+c_61+c_7\alpha^{14}+c_8\alpha^8+c_9\alpha^{11}+c_{10}\alpha^{14}+c_{11}\alpha^{12}+c_{12}\alpha^1=0$$

第二式にzをかけて両辺を加算する。

Multiply the second equation by z and add both sides.

$$\begin{aligned}&(c_1\alpha^9+c_2\alpha^{10}+c_3\alpha^9+c_4\alpha^{14}+c_5\alpha^6+c_60+c_7\alpha^{10}+c_8\alpha^8+c_9\alpha^2+c_{10}\alpha^7+c_{11}\alpha^{14}+c_{12}\alpha^6)\\&\quad+z(c_1\alpha^{12}+c_2\alpha^6+c_3\alpha^6+c_4\alpha^1+c_5\alpha^{11}+c_61+c_7\alpha^{14}+c_8\alpha^8+c_9\alpha^{11}+c_{10}\alpha^{14}+c_{11}\alpha^{12}+c_{12}\alpha^1)\\&\quad=c_1(\alpha^9+\alpha^{12}z)+c_2(\alpha^{10}+\alpha^6z)+c_3(\alpha^9+\alpha^6z)+c_4(\alpha^{14}+\alpha^1z)+c_5(\alpha^6+\alpha^{11}z)+c_6(0+z)+c_7(\alpha^{10}+\alpha^{14}z)\\&\quad+c_8(\alpha^8+\alpha^8z)+c_9(\alpha^2+\alpha^{11}z)+c_{10}(\alpha^7+\alpha^{14}z)+c_{11}(\alpha^{14}+\alpha^{12}z)+c_{12}(\alpha^6+\alpha^1z)\\&\quad=\frac{c_1}{z-\alpha^2}+\frac{c_2}{z-\alpha^3}+\frac{c_3}{z-\alpha^4}+\frac{c_4}{z-\alpha^5}+\frac{c_5}{z-\alpha^6}+\frac{c_6}{z-\alpha^7}+\frac{c_7}{z-\alpha^8}+\frac{c_8}{z-\alpha^9}+\frac{c_9}{z-\alpha^{10}}+\frac{c_{10}}{z-\alpha^{11}}+\frac{c_{11}}{z-\alpha^{12}}+\frac{c_{12}}{z-\alpha^{13}}\\&\quad=0\end{aligned}$$

この式は、Goppa 符号の定義式と一致している。
This equation agrees with the definition of the Goppa code.

Hをベクトルで表現すれば、以下となる。
Expressing H as a vector is as follows.

$$H=\begin{pmatrix}\alpha^{9} & \alpha^{10} & \alpha^{9} & \alpha^{14} & \alpha^{6} & 0 & \alpha^{10} & \alpha^{8} & \alpha^{2} & \alpha^{7} & \alpha^{14} & \alpha^{6}\\ \alpha^{12} & \alpha^{6} & \alpha^{6} & \alpha^{1} & \alpha^{11} & 1 & \alpha^{14} & \alpha^{8} & \alpha^{11} & \alpha^{14} & \alpha^{12} & \alpha^{1}\end{pmatrix}=\begin{pmatrix}1&0&1&1&1&0&0&0&0&1&1&1\\0&1&0&0&1&0&1&1&1&0&0&1\\1&1&1&0&0&0&1&0&0&1&0&0\\0&1&0&1&0&0&1&1&0&1&1&0\\1&1&1&0&1&1&1&0&1&1&1&0\\1&1&1&0&1&1&0&1&1&0&1&0\\1&0&0&1&1&1&0&0&1&0&1&1\\1&0&0&0&0&1&1&1&0&1&1&0\end{pmatrix}$$

生成行列Gは、$GH^T=0$より得られる。まず、Hの零空間(null space)の基底を求める。Hを以下のように基本変形して RREF(Reduced Row Echelon Form)を求める。なお、Hのランクは８である。

The generator matrix G is obtained from GH^T=0. To do this, first, find the basis of the null space of H. Then, obtain RREF (Reduced Row Echelon Form) by transforming H as follows. Note that the rank of H is 8.

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

ここから、null space の基底は、(0 0 1 0 1 0 1 1 1 0 0 0) (1 1 1 1 0 1 0 1 0 1 0 0) (0 1 0 0 1 1 1 1 0 0 1 0) (0 1 0 1 0 0 1 1 0 0 0 1) となる。
これにより、生成行列Gが求められる。
Thus, the basis of the null space is (0 0 1 0 1 0 1 1 1 0 0 0) (1 1 1 1 0 1 0 1 0 1 0 0) (0 1 0 0 1 1 1 1 0 0 1 0) (0 1 0 1 0 0 1 1 0 0 0 1).
As a result, the generator matrix G is obtained.

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

全符号語は、Gの線形結合によって求められる。

All codewords are obtained by linear combinations of rows of G.

Goppa(12, 4, 5)符号

Goppa (12, 4, 5) code

行の番号	線形結合	重み
Row number	Linear combination	weight
①	***0010 1011 1000***	5
②	***1111 0101 0100***	7
③	***0100 1111 0010***	5
④	***0101 0011 0001***	5
①＋②	***1101 1110 1100***	8
①＋③	***0110 0100 1010***	5
①＋④	***0111 1000 1001***	6
②＋③	***1011 1010 0110***	7
②＋④	***1010 0110 0101***	6
③＋④	***0001 1100 0011***	5
①＋②＋③	***1001 0001 1110***	6
②＋③＋④	***1110 1001 0111***	8
①＋②＋④	***1000 1101 1101***	7
①＋③＋④	***0011 0111 1011***	8
①　＋②＋③＋④	***1100 0010 1111***	7
①＋①	***0000 0000 0000***	0

・最初の４ビットに 0000 – 1111 の１６語が現れている。

・この符号は、符号語間の距離がすべて５以上であり（確認せよ）、２誤り訂正が可能である。

・符号効率：k/n=4/12=0.33

Cf. ２誤り生成可能な BCH(15, 7)符号 k/n=7/15=0.467

Note that

- 16 words from 0000 to 1111 appear in the first 4 bits.
- This code has a minimum distance between all codewords greater than or equal to 5. It enables 2-error correction.
- Code efficiency: k/n=4/12=0.33
  Cf. BCH(15, 7) code with 2 errors k/n=7/15=0.467

■例２（Goppa で非巡回ハミング（7,4）となる例）

■ Example 2 (Example of non-cyclic Hamming (7, 4) code)

$GF(2^3)$上の Goppa 符号を考える。$q=2,\ \ m=3$
原始多項式を$ap(x)=x^3+x+1$　とする。

Consider a Goppa code over GF(2^3). q=2, m=3
Let the primitive polynomial be ap(x)=x^3+x+1.

１）x^7+1の因数分解。省略

Fist, we factorize x^7+1.

２）原始多項式からの拡大体の構成。

2) We construct the extension field from the primitive polynomial.

Polynomial expression Vector expression α exponentiation expression

多項式	ベクトル表現	αべき表現	
0	000	0	
1	001	1	$=\alpha^0$
x	010	α	$=\alpha^1$
$x+1$	011	$\alpha+1$	$=\alpha^3$
x^2	100	α^2	$=\alpha^2$
x^2+1	101	α^2+1	$=\alpha^6$
x^2+x	110	$\alpha^2+\alpha$	$=\alpha^4$
x^2+x+1	111	$\alpha^2+\alpha+1$	$=\alpha^5$

拡大体は$GF(2^3)=\{0,1,\alpha,\alpha^2,\alpha^3,\alpha^4,\alpha^5,\alpha^6\}$

The extension field is
$GF(2^3)=\{0,1,\alpha,\alpha^2,\alpha^3,\alpha^4,\alpha^5,\alpha^6\}$

３）ゴッパ集合Lを以下とする。$L \subseteq GF(2^3)$　（Lは$GF(2^3)$の部分集合）

3) Let the Goppa set L be L⊆GF (2^3) (L is a subset of GF(2^3)).

$$L = \{0, \alpha^1, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6\}$$

ゴッパ集合Lの要素数により、符号長nが決定される。この場合は、$n = 7$である。

The number of elements in the Goppa set L determines the code length n. In this case n=7.

４）ゴッパ多項式 $g(z)$を決める。$g(z)$の根は、ゴッパ集合Lに含まれないようにする。

ここでは、以下とする。

4) Determine the Goppa polynomial g(z). The root of g(z) should not be included in the Goppa set L. Here, it is assumed as follows.

$$g(z) = z + 1 = g_1 z + g_0$$

５）以上より、このゴッパ符号のパラメータは以下となる。

5) From the above, the parameters of the Goppa code are as follows.

$$q = 2, \quad m = 3, \quad n = 7, \quad t = 1, \quad d \geq 2t + 1 = 2 + 1 = 3$$

$\alpha_1 = 0$ $\alpha_2 = \alpha^1$ $\alpha_3 = \alpha^2$ $\alpha_4 = \alpha^3$ $\alpha_5 = \alpha^4$ $\alpha_6 = \alpha^5$ $\alpha_7 = \alpha^6$　$g_0 = 1$　$g_1 = 1$

これにより、(7,4,3)Goppa 符号が構成される。

This constructs a (7,4,3) Goppa code.

６）検査行列 H を求める。

6) Find the parity check matrix H.

$H = XYZ$

$X: t \times t = 1 \times 1$、$Y: t \times n = 1 \times 7$、$Z: n \times n = 7 \times 7$、$H: t \times n = 1 \times 7$

まず、X行列を求める。

First, find the X matrix.

$$X = \begin{pmatrix} -g_t & -g_{t-1} & -g_{t-2} & & -g_1 \\ 0 & -g_t & -g_{t-1} & \cdots & -g_2 \\ 0 & 0 & -g_t & & -g_3 \\ & \vdots & & \ddots & \\ 0 & 0 & 0 & & -g_t \end{pmatrix} = (-g_1) = (1)$$

次に、Y行列を求める。

Next, find the Y matrix.

$$Y = \begin{pmatrix} {\alpha_1}^{t-1} & {\alpha_2}^{t-1} & {\alpha_3}^{t-1} & & {\alpha_n}^{t-1} \\ {\alpha_1}^{t-2} & {\alpha_2}^{t-2} & {\alpha_3}^{t-2} & \cdots & {\alpha_n}^{t-2} \\ {\alpha_1}^{t-3} & {\alpha_2}^{t-3} & {\alpha_3}^{t-3} & & {\alpha_n}^{t-3} \\ & \vdots & & & \\ {\alpha_1}^{1} & {\alpha_2}^{1} & {\alpha_3}^{1} & \ddots & {\alpha_n}^{1} \\ 1 & 1 & 1 & & 1 \end{pmatrix} = (1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1)$$

Z行列を求める。

Then, find the Z matrix.

$\frac{1}{g(\alpha_1)} = \frac{1}{g(0)} = \frac{1}{1} = 1$ $\frac{1}{g(\alpha_2)} = \frac{1}{g(\alpha^1)} = \frac{1}{\alpha^1+1} = \frac{1}{\alpha^3} = \alpha^{-3}$

$\frac{1}{g(\alpha_3)} = \frac{1}{g(\alpha^2)} = \frac{1}{\alpha^2+1} = \{(100) + (001)\}^{-1} = \{(101)\}^{-1} = \alpha^{-6}$

$\frac{1}{g(\alpha_4)} = \frac{1}{g(\alpha^3)} = \frac{1}{\alpha^3+1} = \{(011) + (001)\}^{-1} = \{(010)\}^{-1} = \alpha^{-1}$

$\frac{1}{g(\alpha_5)} = \frac{1}{g(\alpha^4)} = \frac{1}{\alpha^4+1} = \{(110) + (001)\}^{-1} = \{(111)\}^{-1} = \alpha^{-5}$

$\frac{1}{g(\alpha_6)} = \frac{1}{g(\alpha^5)} = \frac{1}{\alpha^5+1} = \{(111) + (001)\}^{-1} = \{(110)\}^{-1} = \alpha^{-4}$

$\frac{1}{g(\alpha_7)} = \frac{1}{g(\alpha^6)} = \frac{1}{\alpha^6+1} = \{(101) + (001)\}^{-1} = \{(100)\}^{-1} = \alpha^{-2}$

$$Z=\begin{pmatrix} \frac{1}{g(\alpha_1)} & 0 & 0 & & 0 \\ 0 & \frac{1}{g(\alpha_2)} & 0 & \cdots & 0 \\ 0 & 0 & \frac{1}{g(\alpha_3)} & & 0 \\ & \vdots & & \ddots & \\ 0 & 0 & 0 & & \frac{1}{g(\alpha_n)} \end{pmatrix}=\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \alpha^{-3} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \alpha^{-6} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \alpha^{-1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \alpha^{-5} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \alpha^{-4} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \alpha^{-2} \end{pmatrix}$$

従って、H は以下のように求められる。

Therefore, H is calculated as follows.
Portanto, H é calculado da seguinte forma.

$$H=XYZ=(1)\cdot(1\quad 1\quad 1\quad 1\quad 1\quad 1\quad 1)\cdot\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \alpha^{-3} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \alpha^{-6} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \alpha^{-1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \alpha^{-5} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \alpha^{-4} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \alpha^{-2} \end{pmatrix}$$

$$=(1\quad 1\quad 1\quad 1\quad 1\quad 1\quad 1)\cdot\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \alpha^{-3} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \alpha^{-6} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \alpha^{-1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \alpha^{-5} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \alpha^{-4} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \alpha^{-2} \end{pmatrix}$$

$$=(1\quad \alpha^{-3}\quad \alpha^{-6}\quad \alpha^{-1}\quad \alpha^{-5}\quad \alpha^{-4}\quad \alpha^{-2})=(1\quad \alpha^{4}\quad \alpha^{1}\quad \alpha^{6}\quad \alpha^{2}\quad \alpha^{3}\quad \alpha^{5})=\begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

生成行列Gは、$GH^T=0$より得られる。

The generator matrix G is obtained from GH^T=0.

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

$H=\begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$ の 1 列目と 7 列目を入れ替えて、$H_{17}=\begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$。次に、3 列目と 6

列目を入れ替えて、$H_{17/36} = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$

よって、$G_{17/36} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}$　これを３列目と６列目、１ 列目と ７ 列目を入れ替えて、$G =$

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

Swap the 1rd column and 7th column of $H = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$ to obtain $H_{17} =$

$\begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$.

Then, swap the 3rd and 6th to get $H_{17/36} = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$.

Then, we obtain $G_{17/36} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}$.

Finally by swapping 3rd and 6th , then 1st and 7th, we get $G = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \end{pmatrix}$

全符号語は、Gの線形結合によって求められる。

All codewords are obtained by linear combinations of rows of G.

行の番号	線形結合　巡回のパタン	重み
Row number	Linear combination	weight
①	**1010 101**	- 4
②	**0110 100**	- 3
③	**1010 010**	- 3
④	**1001 100**	- 3
①＋②	**1100 001**	- 3
①＋③	**0000 111**	- 3

①＋④	0011 001	-	3
②＋③	1100 110	-	4
②＋④	1111 000	-	4
③＋④	0011 110	-	4
①＋②＋③	0110 011	-	4
①＋②＋④	0101 101	-	4
①＋③＋④	1001 011	-	4
②＋③＋④	0101 010	-	3
①　＋②＋③＋④	1111 111	-	7
①＋①	0000 000	-	0

・この Goppa(7, 4)符号は、非巡回ハミング(7, 4)符号である。

This Goppa (7, 4) code is an acyclic Hamming (7, 4) code.

７．３．６　McEliece 暗号
7.3.6 McEliece cipher

公開鍵暗号の一種である。
$GF(q^m)$上のt次の separable 多項式$g(z)$とLによって定義される$(n, k \geq n - mt, d \geq 2t + 1)$Goppa 符号を考える。$k \times n$の生成行列$G$、random dense $k \times k$ nonsingular 行列S、$n \times n$ permutation 行列Pを求める。$G^* = SGP$とする。G^*とtを公開し、それ以外を秘密鍵とする。

It is a kind of public key cryptography.
Consider a (n,k≥n-mt,d≥2t+1) Goppa code defined by L and a separable polynomial g(z) of degree t on GF(q^m).
A k×n generator matrix G, a random dense k×k nonsingular matrix S, and an n×n permutation matrix P are obtained.
Let G^*=SGP. G^* and t are made public, and the others are used as private keys.

送信者：
長さkの二進文字列を送信するとする。m個の二進文字列ごとに、長さnのランダムなエラーパタンeを計算する。ただし、eの最大重みはtとする。mを$y = mG^* + e$ と暗号化して送信する。

Sender side:
Suppose you want to send a binary string of length k.
Computes a random error pattern e of length n for every m binary strings.
However, the maximum weight of e is t.
Encrypt m as y=mG^*+e and send y.

受信者：

$$y' = yP^{-1} = mG^*P^{-1} + eP^{-1} = mSGPP^{-1} + e' = (mS)G + e'$$

ここでe'の最大重みはtである。通常の（暗号化していない時の）Goppa 復号によりe'を求め、$m' = mS$よりm'を求める。その後、$m = m'S^{-1}$よりmを求める。

Receiver side:
y^'=yP^(-1)=mG^* P^(-1)+eP^(-1)=mSGPP^(-1)+e^'=(mS)G+e^'
where the maximum weight of e^' is t.

Obtain e^’ by normal (unencrypted) Goppa decryption, and obtain m^’ from m^’=mS. After that, m is obtained from m=m^’ S^(-1).

McEliece は、具体的な方法として、$m = 10$, $t = 50$次元の既約多項式$g(z), n = 2^{10} = 1024$, $k \geq 1024 - 10 * 50 = 525$の暗号化を議論している。

McEliece discusses encryption of the irreducible polynomial g(z) of dimension m=10, t=50, n=2^10=1024, k≥1024-10*50=525.

７．３．７　発展

7.3.7 Advanced topics

・復号アルゴリズム　ベールカンプ＝マッシー法

・McEliece 暗号　の具体例

・Decryption algorithm: Beerkamp-Massey method

・Specific examples of McEliece cipher

７．４　その他の話題
7.4 Other topics

７．４．１　実際に使われている符号
7.4.1 Codes actually used

１）QR (Quick Response)コード
1) QR (Quick Response) code

符号語：８ビット　リードソロモン符号
各ブロックはリードソロモン符号語　　フォーマット情報:BCH 符号

Data: 8-bit Reed-Solomon code
Format information: BCH code

エラー訂正レベル
Error correction level

L (Low)	7% Error correctable
M (Medium)	15% Error correctable
Q (Quartile)	25% Error correctable
H (High)	30% Error correctable

(n, k, t)RS 符号の生成多項式　$G(x) = (x-1)(x-\alpha)(x-\alpha^2)\cdots(x-\alpha^{n-k-1})$

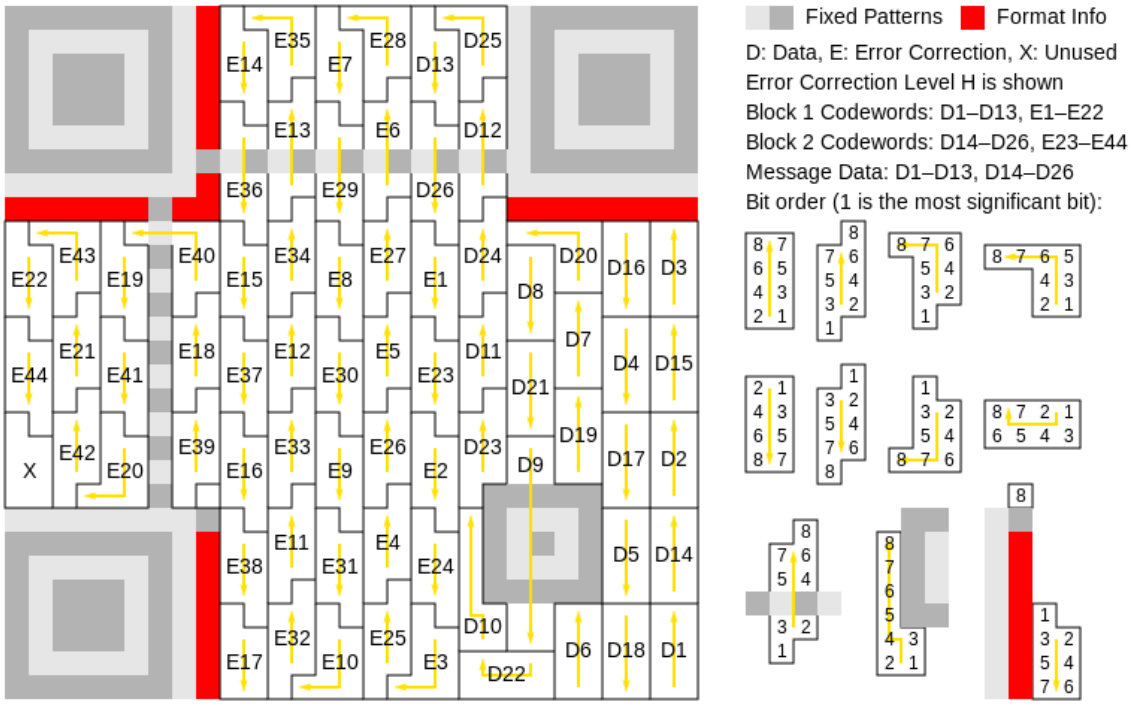
(n, k, t) RS code generator polynomial
G(x)=(x-1)(x-α)(x-α^2)…(x-α^(n-k-1))

型番　総コード数　誤り訂正レベル　　(n, k, t)　　RS ブロック数　　生成多項式次数

Model number	Total number of codes	Error correction level	(n, k, t)	Number of RS blocks	Generator polynomial order
1	26	L	26, 19, 2	1	7
1	36	M	26, 16, 4	1	10
1	26	Q	26, 13, 6	1	13
1	26	H	26, 9, 8	1	17
2	44	L	44, 34, 4	1	10
2	44	M	44, 28, 8	1	16
2	44	Q	44, 22, 11	1	22
2	44	H	44, 16, 14	1	28
3	70	L	70, 55, 7	1	15
3	70	M	70, 44, 13	1	26
3	70	Q	35, 17, 9	2	18
3	70	H	35, 13, 11	2	22

例：3-H の場合、(n, k, t)=(35, 13, 11)

Example: 3H (n, k, t)=(35, 13, 11)



(3H 型 QR コード　Wikipedia 英語版より)

(3H QR code see Wikipedia https://en.wikipedia.org/wiki/QR_code)

２）　CRC (Cyclic Redundancy Check)　巡回冗長検査

2) CRC (Cyclic Redundancy Check)

・誤り検出を目的とした符号化。多くの分野で用いられている。

- Codes for the purpose of error detection. They are used in many fields.

・よく利用される符号

- Practical codes

CRC-16-CCITT

$$p(x) = x^{16} + x^{12} + x^5 + 1 = (x+1)(x^{15} + x^{14} + x^{13} + x^{12} + x^4 + x^3 + x^2 + x + 1) \quad (0x1021)$$

この数字は 4bit ずつ区切った 0001 0000 0010 0001 を表現しており、

x^{12}, x^5, x^0が要素として存在することを意味している。（x^{16}は当然なので省略）

0x1021 represents 0001 0000 0010 0001 separated by 4 bits, meaning that x^12, x^5, and x^0 exist as elements. (x^16 is omitted because it is obvious.)

X.25, V.41, HDLC, Bluetooth, SD, etc.

CRC-16-ANSI(IBM)　$p(x) = x^{16} + x^{15} + x^2 + 1 = (x+1)(x^{15} + x + 1)$　(0x8005)

USB data

CRC-32　$p(x) = x^{32} + x^{26} + \cdots \cdot x^{15} + 1$ (0x04C11DB7)

V.42, Ethernet, IEEE802.3, Serial ATA, MPEG2, Zip, PNG etc.

32 ビット以下のバースト誤りを検出可能

Capable of detecting burst errors of 32 bits or less

バースト長 33 の誤りを検出できない確率=$\left(1/2\right)^{33}$ =20 億分の 1

Probability of not detecting an error with a burst length of 33 = (1/2)^33 = 1 in 2 billion.

その他、CRC-32C(Castagnoli), CRC-32K(Koopman)など

Other codes are CRC-32C (Castagnoli), CRC-32K (Koopman), etc.

・Koopman , Castagnoli at CMU らの議論(Koopman, 2002 など)

- Discussions by Koopman, Castagnoli at CMU, etc. (Koopman, 2002, etc.)

CRC-32 の（最小）ハミング距離=4
3bit 誤り検出可能、4bit 誤りでは 223059 の正しい符号語が存在。
フレーム長 1514bytes+CRC32bit=12144bit　で、4bit 誤る組み合わせは、

$$\binom{12144}{4} = 9.06 \times 10^{14}$$

よって、4bit 誤りを誤訂正する確率は、$223059/\binom{12144}{4} = 2.46 \times 10^{-10}$

The minimum Hamming distance of CRC-32 is 4.
So that 3-bit error can be detected, and 223059 correct codewords exist for 4-bit errors.
With a frame length of 12144 bits (1514 bytes + CRC32 bits), the number of all possible combination of a 4-bit error is C(12144, 4))=9.06×10^14.
Therefore, the probability of miscorrecting a 4-bit error is 223059/C(12144,4)=2.46×10^(-10).

３）　BCH, RS のその他の応用

3) Other applications of BCH and RS

・RS 符号　　ボイジャー Deep space communication

・QR コード　　RS+BCH

・地上デジタル放送

RS(255, 239)

符号化生成多項式　　$g(x) = (x+1)(x+\alpha^1)\cdot\cdot\cdot\cdot(x+\alpha^{15})$

体生成多項式　　$p(x) = x^8 + x^4 + x^3 + x^2 + 1$

を元にした短縮化 RS(204, 188)がトランスポートストリーム(TS)パケットの符号化に用いられている。

- RS code is used for Deep space communication (cf. Voyager)
- Terrestrial digital broadcasting uses RS(255, 239)

  Generator polynomial (Encode polynomial) g(x)=(x+1)(x+α^1)***(x+α^15)

  Primitive polynomial (Field generator polynomial) p(x)=x^8+x^4+x^3+x^2+1

  A shortened RS (204, 188) based on the above polynomials is used for transport stream (TS) packet encoding.

tentative

７．４．２　RS とフーリエ変換（RS 符号の別の表現）

7.4.2 RS and Fourier transform (another representation of RS code; Frequency Domain expression)

今井秀樹著「符号理論」電子情報通信学会　第７章　p.158 より抜粋

The following is from "Theory of Information, Code, and Cryptography", edited by Institute of Electronics, Information and Communication Engineers, Corona Publishing Co., Ltd. p.158 by Hideki Imai.

本書では、GF(q)（q は素数のべき乗）で議論を行っているが、ここでは$GF(2^4)$に限定する。

In this book, GF(q) (where q is a power of a prime number) is used for discussion, but here we limit it to GF(2^4).

定理７．２ $GF(2^4)$に限定した定理

Theorem 7.2 Theorem limited to GF(2^4)

$I_{k-1}, I_{k-2}, \cdots, I_1, I_0$ を$GF(2^4)$の任意の元とし、これらを係数とする多項式

$I(y) = I_{k-1}y^{k-1} + I_{k-2}y^{k-2} + \cdots + I_1y^1 + I_0$ を考える。

Let I_(k-1),I_(k-2),…,I_1,I_0 be any elements of GF(2^4) and consider polynomials with these as coefficients, that is, I(y)=I_(k-1) y^(k-1) + I_(k-2) y^(k-2) +… + I_1 y^1 + I_0.

このとき、α を$GF(2^4)$の原始根とすれば、

$$w = (I(\alpha^{14}), I(\alpha^{13}), \cdots, I(\alpha^1), I(\alpha^0))$$

という$GF(2^4)$上のすべてのベクトルを符号語とする符号長$2^4 - 1 = 15$の符号は、

$\alpha^1, \alpha^2, \cdots, \alpha^{16-1-k}$ を根とする$(2^4 - 1, k)$RS 符号となる。

Then, if α is the primitive root of GF(2^4), then w=(I(α^14),I(α^13),…,I(α^1),I(α^0)) over GF(2^4) whose code length 2^4-1=15 is a (2^4-1,k)RS code with roots of α^1,α^2,…,α^(16-1-k).

【証明】

【Proof】

この定理を証明するには、$\alpha^1, \alpha^2, \cdots, \alpha^{16-1-k}$ （k 個）が

$$W(y) = I(\alpha^{14})y^{14} + I(\alpha^{13})y^{13} + \cdots + I(\alpha^1)y^1 + I(1)$$

の根となっていればよい。

To prove this theorem, we show α^1, α^2,…,α^(16-1-k) (k pieces) are roots of W(y)=I(α^14) y^14 +I(α^13) y^13 + … + I(α^1) y^1 + I(1).

w は、$(0, 0, \cdots, 0, I_{k-1}, I_{k-2}, \cdots, I_1, I_0)$ という$2^4 - 1 = 15$次元ベクトルのフーリエ変換となっている。

w is the Fourier transform of the 2^4-1=15-dimensional vector (0,0,…,0,I_(k-1),I_(k-2),…,I_1,I_0).

一方、$(W(\alpha^{-14}), W(\alpha^{-13}), \cdots, W(\alpha^{-1}), W(1))$　は、wのフーリエ逆変換である。従って、

$$W(\alpha^{-14}) = W(\alpha^{-13}) = \cdots = W(\alpha^{-k}) = 0$$

が得られる。このことは、$W(y)$が、$\alpha^{-14} = \alpha^{1}$, $\alpha^{-13} = \alpha^{2}, \cdots$, $\alpha^{-k} = \alpha^{16-1-k}$を根として持つこと、従って 16 元$(2^4 - 1, k)$RS 符号の符号多項式であることを意味している。

On the other hand, (W(α^(-14)),W(α^(-13)),…,W(α^(-1)),W(1)) is the inverse Fourier transform of w.
Therefore,we can obtain W(α^(-14))=W(α^(-13))=…=W(α^(-k))=0.
This means that W(y) has roots of α^(-14)=α^1, α^(-13)=α^2,…, α^(-k)=α^(16-1-k).
Thus, it is the code polynomial of the 16-ary (2^4-1,k) RS code.

より一般の RS 符号については次の系が成立する。
The following corollary holds for more general RS code.

系 7.2.1
Corollary 7.2.1

定理７．２と同様の仮定のもとに、

$$\left(\alpha^{(l-1)*14} I(\alpha^{14}),\ \alpha^{(l-1)*13} I(\alpha^{13}),\ \cdots,\ \alpha^{(l-1)} I(\alpha^{1}),\ M(1)\right)$$

という$GF(2^4)$上のすべてのベクトルを符号語とする符号長$2^4 - 1 = 15$の符号は、

$$\alpha^{l}, \alpha^{(l+1)}, \cdots, \alpha^{(l+16-1-k)}$$

を根とする$(2^4 - 1, k)$ RS 符号である。

Under the same assumptions as in Theorem 7.2,
w=(α^((l-1)*14) I(α^14), α^((l-1)*13) I(α^13), …, α^((l-1)) I(α^1), M(1))　over GF(2^4) whose code length 2^4-1=15 is a (2^4-1,k) RS code with roots of α^l, α^((l+1)),…, α^(l+16-1-k).

Reference

Hideki Imai, "Theory of Information, Code, and Cryptography", edited by Institute of Electronics, Information and Communication Engineers, Corona Publishing Co., Ltd. p.158)

http://site.iugaza.edu.ps/ahdrouss/files/2010/02/Reed_Solmen_Code.pdf

７．４．３　BCH 符号、RS 符号の初期値 l

第 7 章の資料では、BCH 符号では l=1、RS 符号では l=0 を挙げて説明した。ここでは、１）BCH 符号で l=0 の場合、２）RS 符号で、l=1 の場合を考える。

１）BCH 符号で、l=0 の場合

BCH の生成多項式は以下で構成される。

$$g(x)=\left[\prod_{i=l}^{l+d_{min}-2}\left(x-\alpha^{i}\right)\right]A(x)=\left[\prod_{i=l}^{l+2t-1}\left(x-\alpha^{i}\right)\right]A(x)\quad\text{（式 6.4.3.1）}$$

（$d_{min}=2t+1$）ただし、$A(x)$は根α^{l}～α^{l+2t-1}の 2 乗、4 乗などを根として持つ多項式。

例えば、原始多項式をx^3+x+1とする。l=0、t=1 では l+2t-1=1 であるので、生成多項式は、

$$g(x)=\left[\prod_{i=0}^{1}\left(x-\alpha^{i}\right)\right]A(x)=(x-\alpha^{0})(x-\alpha^{1})A(x)$$

となる。0 乗, 1 乗の根が連続して入るため最小距離は 3 となりそうである。しかし、1 乗を根として持つと、２，４乗も根として持つ。よって実際には、

$$g(x)=(x-\alpha^{0})(x-\alpha^{1})A(x)=(x-\alpha^{0})(x-\alpha^{1})(x-\alpha^{2})(x-\alpha^{4})$$

であり、０乗、１乗、２乗が連続した根となっており、dmin=4 になる。これは、１誤り訂正可能、２誤り検出可能である。別の言い方をすると、l=1 の場合に対して、パリティが追加されている符号とも考えられる。従って、導入の例としてはあまり適切ではなく、ここでは l=1 を選択している。

２）RS 符号で、l=1 の場合

RS 符号の場合、l=0 でも l=1 でも特に問題なく同様な議論ができる。

例えば、資料で扱った RS(7,3)符号の方法を以下に示す。

■例１　(7,3)符号　　$m=3,n=7,k=3,t=2,$ 原始多項式が$p(x)=x^3+x+1$の場合

で、l=1 の場合を検討する。

①情報ビット系列を 3 ビット$(m=3)$ずつの 3 個$(k=3)$のブロック（バイト）に分割する。例えば

情報ビット　001 010 100

情報バイト　1　α^1　α^2

バイト番号　1　　2　　3

②情報バイトを多項式で表現する。

$$I(y)=I_{k-1}y^{k-1}+\cdots+I_1y+I_0=1y^2+\alpha^1y^1+\alpha^2y^0$$

③生成多項式を求める。

$$G(y)=\prod_{i=l}^{l+2t-1}\left(y-\alpha^{i}\right)=\prod_{i=1}^{4}\left(y-\alpha^{i}\right)=(y-\alpha^{1})(y-\alpha^{2})(y-\alpha^{3})(y-\alpha^{4})$$
$$=y^4+\alpha^3y^3+y^2+\alpha^1y^1+\alpha^3$$

④情報多項式$I(y)*y^{n-k}$を生成多項式$G(y)$　で除算した余りを求める。

$$I(y)=I_{k-1}y^{k-1}+\cdots I_1y+I_0=1y^2+\alpha^1y^1+\alpha^2y^0$$
$$r(y)=I(y)*y^{n-k}\ mod\ G(y)=(1y^2+\alpha^1y^1+\alpha^2y^0)*y^4\ mod\ y^4+\alpha^3y^3+y^2+\alpha^1y^1+\alpha^3$$
$$=\alpha^1y^3+\alpha^5y^2+\alpha^2y+\alpha^0$$

⑤以上より

情報ビット　001　010　100

情報バイト	1	α^1	α^2				
符号化後	1	α^1	α^2	α^1	α^5	α^2	α^0
RS 符号語	001	010	100	010	111	100	001
バイト番号	1	2	3	4	5	6	7

情報通信基礎Ⅰ　第８章　たたみ込み符号

情シス　渡辺

ポイント

・今まで議論してきた符号はブロック符号と呼ばれ、ランダム誤りに対応している。

・バースト誤りに強い符号として、たたみ込み符号がある。

・効率のよい復号法としてビタビ復号がある。

point

- The codes we have discussed so far are called block codes, and they deal with random errors.
- A convolutional code is a code that is resistant to burst errors. For decoding, Viterbi decoding is used as an efficient method.

８．１　基礎

８．１．１　誤り

8.1 Fundamentals

8.1.1 Errors

ランダム誤り：各ビットが乱数で決定されるように誤る

バースト誤り：複数のビットがまとまって誤る

Random error: Each bit is flipped by a random number

Burst error: Multiple bits are changed continuously.

８．１．２　ブロック符号とたたみ込み符号

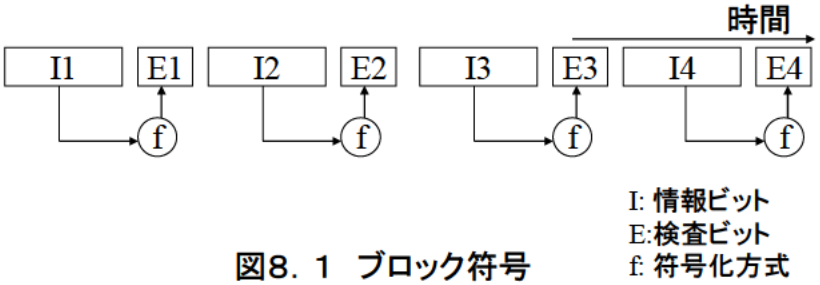


図８．１　ブロック符号

8.1.2 Block code and convolutional code

（１）ブロック符号

代表的な例としては、巡回符号、ＣＲＣ（Cyclic Redundancy Check）符号、ハミング符号、ＢＣＨ符号、リードソロモン符号などがある。ランダム誤り訂正に適している。

(1) Block code

Typical codes of block codes include cyclic codes, as known as CRC (Cyclic Redundancy Check) codes, Hamming codes, BCH codes, and Reed-Solomon codes. Block codes are suitable for random error correction.

（２）畳み込み符号

バースト誤りに適している符号化である。ただし、バースト誤り訂正を行えるブロック符号もある。例えば、リードソロモン符号、ファイア符号などがある。

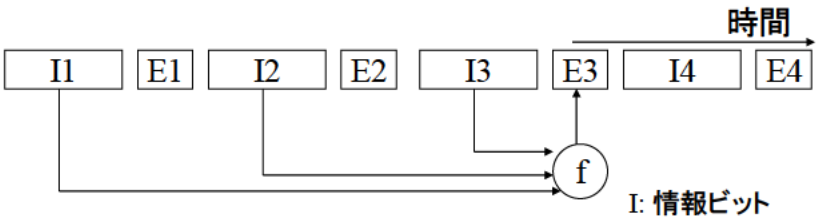


図８．２ たたみ込み符号

(2) Convolutional code

This type of codes is suitable for burst errors. There are also block codes that can perform burst error correction. Examples include Reed-Solomon codes and Fire codes.

（３）たたみ込み符号の重要なパラメータ

重要なパラメータとしては、以下の２つがある。

拘束長Ｋ：　情報１ビットが何ビット後まで影響するか。

符号化率Ｒ：　情報１ビットが何ビットになるか。

(3) Important parameters of convolutional codes

Two important parameters are:

Constraint length K: How many bits after 1 bit of information is affected?

Coding rate R: The number of bits that one bit of information creates.

拘束長 K は、レジスタ数に反映される
符号化率 R は、入力数と出力数に反映される

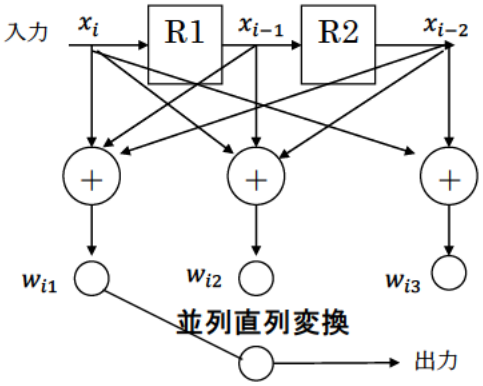
Constraint length K is reflected in the number of registers.
Coding rate R is reflected in the number of inputs and outputs.

８．１．３　拘束長３　符号化率１／３の例

8.1.3 Example of convolutional code with constraint length 3 and coding rate 1/3

K=3, R=1/3 の例を示す。

The example of convolutional code with K=3, R=1/3 is shown here.



$$w_{i1} = x_i + x_{i-1} + x_{i-2}$$
$$w_{i2} = x_i + x_{i-1} + x_{i-2}$$
$$w_{i3} = x_i + x_{i-2}$$

図8．３　拘束長3　符号化率1/3の例

レジスタの初期値を 0 として、入力が（101100）のときの出力 U は、以下となる。

U = (111, 110, 000, 001, 001, 111)

When the initial values of the registers are all 0 and the input is (101100), the output U is as follows.

U = (111, 110, 000, 001, 001, 111)

入力が(101100)の場合の状態遷移と出力

入力	出力
x_i	$x_i x_{i-1} x_{i-2} w_{i1} w_{i2} w_{i3}$
1	1 0 0 1 1 1
0	0 1 0 1 1 0
1	1 0 1 0 0 0
1	1 1 0 0 0 1
0	0 1 1 0 0 1
0	0 0 1 1 1 1

図8．４ (101100)に対する出力

８．２　ビタビ復号

8.2 Viterbi decoding

たたみ込み符号の効率のよい復号方式としてビタビ復号法がある。

Viterbi decoding is an efficient decoding method for convolutional codes.

８．２．１　ビタビ復号の基礎

8.2.1 Basics of Viterbi decoding

・受信側では、トレリスダイヤグラムを用意する。

・トレリスダイヤグラムは、０が送信された時と１が送信された時のレジスタ状態の遷移を表現している。

・初期状態は００、最終状態は００とする。最終状態が００となるように、送るべき情報ビットの最後にビット列（テールビット）を挿入する。

・受信したビット列と正しいと想定されるビット列との差をメトリックとして記録する。
（メトリック：正しい符号語列との距離；何ビット誤ったらこの状態にたどり着くか）

・同じ状態になる複数のパスがある場合は、メトリックの小さいパスを残存させる。

・同じ状態になる同一のメトリックの複数のパスがある場合は、いずれかを選択する。

- The sending side and receiving side share a trellis diagram of the code.

- The trellis diagram includes possible register states and state transitions when a 0 or 1 is sent.
- The initial state of the registers is set to 00 and the final state to 00. A bit string (tail bits) is inserted at the end of the information bits so that the final state is 00.
- Every time any bit is received, the receiver records as a metric the Hamming distance between the bit string received and the correct bit string to each possible state. So that, the metric shows how many bits are wrong to reach the state.
- If there are multiple paths to the same state, the path with the lowest metric is left and the others are eliminated.
- If there are multiple paths with the same metric that end up in the same state, one path randomly chosen is left.

８．２．２　拘束長３　符号化率１／３の例

8.2.2 An example of convolutional code with constraint length 3 and coding rate 1/3

図８．３の例に対して、次のようなトレリスダイヤグラムを作成できる。１ブロック（この例の場合３ビット）を受信する毎に１クロック時刻が進むとする。状態は、２つのレジスタの値で定義する。

For the example in Figure 8.3, the following trellis diagram can be created. Assume that the clock time elapsed by one each time one block (three bits in this example) is received. A state is defined by the values of two registers.

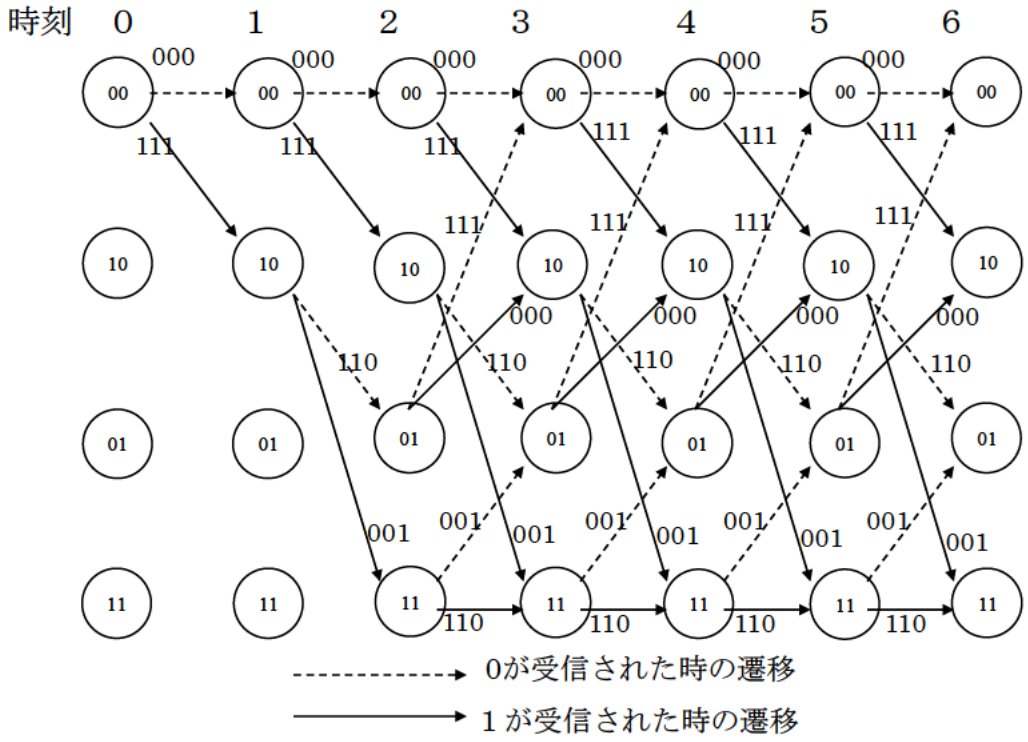


図8．5 トレリスダイヤグラム

今、情報 (1,0,1,1)に対してテールビットを加えた(1,0,1,1,0,0)を送ることを考える。符号化器で符号化し、Uを送ったところ、誤りが生じた符号語Vを受信したとする。

送信符号語列

U = (111, 110, 000, 001, 001, 111)

受信符号語列

V = (110, 010, 011, 101, 001, 111)

Now, let's consider the sender transmits bit sequence (1,0,1,1,0,0) with tail bits added to the information (1,0,1,1). Suppose that the receiver receives a codeword V with an error after sending U after encoding.

Transmitted codeword string

U = (111, 110, 000, 001, 001, 111)

Received codeword string

V = (110, 010, 011, 101, 001, 111)

符号語を受信する度に、トレリスダイヤグラムを辿り遷移可能な状態への正しいパスとの差を計算してメトリックを求める。

時刻３の時のトレリスダイヤグラムを図８．６に示す。

★では、状態 00 からのパスのメトリックが５であるのに対し、状態 01 からのメトリックが 3 である。つまり、状態 01 を経由する方が誤る個数が少ないことを意味している。ここでは、00 からのパスを削除し、01 からのパスを残す。

※では、00 からのパスのメトリックと、01 からのパスのメトリックが同一である。この場合、いずれかを選択する。

時刻６の最終状態を図８．７に示す。この残存パスを逆に辿って復号する。この場合、(111, 110, 000, 001, 001, 111)に正しく復号できる。

Each time a codeword is received, by the trellis diagram the Hamming distance between the current path and the possible paths is calculated to obtain metrics.

The trellis diagram at time 3 is shown in Figure 8.6.

At ★, the path metric from state 00 is 5, while the metric from state 01 is 3. It means that the number of errors is smaller when going through state 01. Here we remove the path from 00 and leave the path from 01.

At ∗, the metric of the path from 00 and the metric of the path from 01 are the same. In this case, choose either.

The final state at time 6 is shown in Figure 8.7. At this moment decoding can be done by tracing the remaining path in reverse. In this case, it can be decoded correctly to (111, 110, 000, 001, 001, 111).

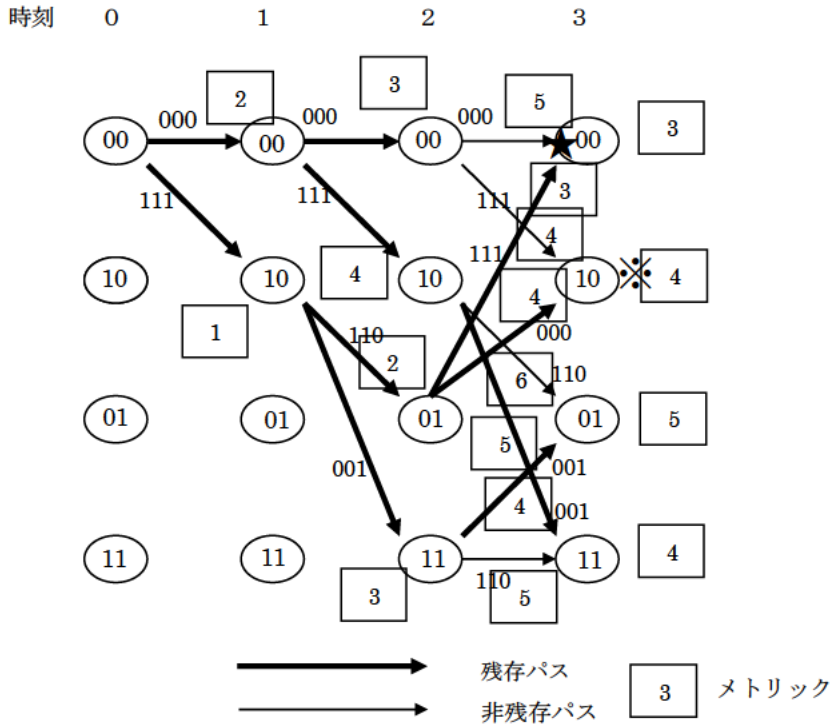


図8．6　時刻3での状態

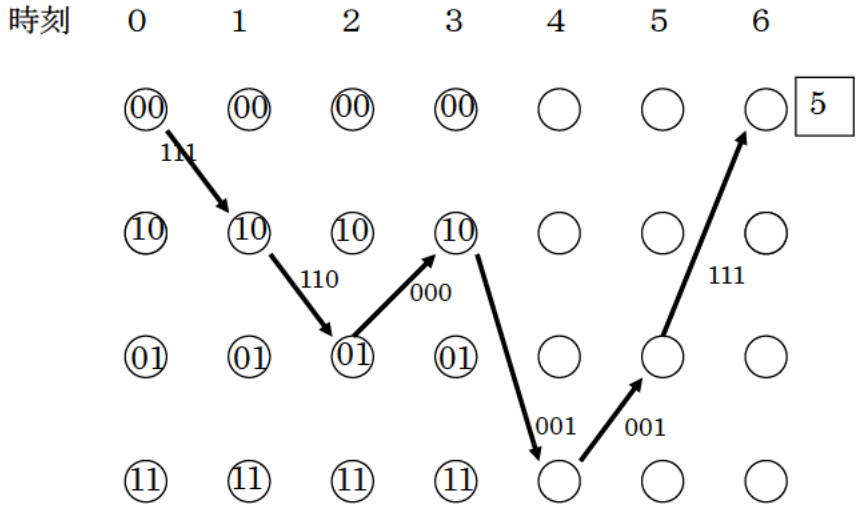


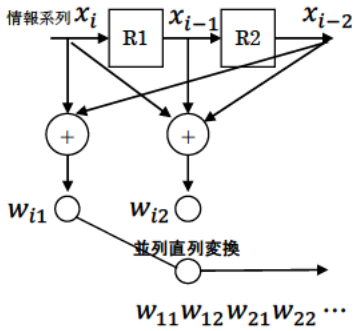
図8．7　時刻6での状態

８．２．３　拘束長３　符号化率１／２の例

8.2.3 An example of convolutional code with constraint length 3 and coding rate 1/2

K=3, R=1/2 の例の符号化器とトレリスダイヤグラムを図８．８、図８．９に示す。

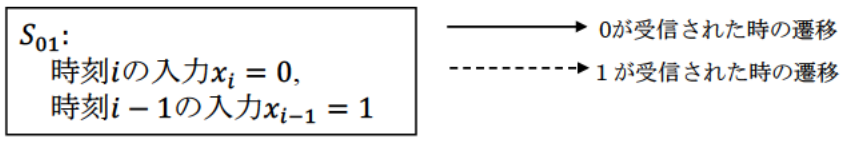
An example encoder and trellis diagram for K=3, R=1/2 are shown in Figs. 8.8 and 8.9.



$$w_{i1} = x_i + x_{i-2}$$
$$w_{i2} = x_i + x_{i-1} + x_{i-2}$$

図8．8　K=3, R=1/2の例

注意：
この図におけるS01とは、
R1=1, R2=0の状態



入力(010100)ー＞出力(00 11 10 00 10 11)

図8．9　K=3, R=1/2のトレリスダイヤグラム

８．３　その他のたたみ込み符号

8.3 Other convolutional codes

８．３．１　ハーゲルバーガー(14, 7)符号

8.3.1 Hagelberger (14, 7) code

島田良作、木内陽介、大松繁著「わかる情報理論」　日新出版　より抜粋

(mn_0, mk_0)の畳み込み符号で最も簡単な符号　であり、長さ６までのバースト誤りを訂正可能である。　$k_0 = 1, m_0 = 1, m = 7,\ n_0 = k_0 + m_0$

この符号の生成行列は以下となる。

It is the simplest convolutional code of (mn_0,mk_0) and can correct burst errors up to length 6. k_0=1,m_0=1,m=7,n_0=k_0+m_0
The generator matrix of this code is as follows.

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

符号化方法

情報ビットを$(a_1\ a_2\ a_3\ a_4\ a_5\ ,,,,)$とすると、検査記号　c_iを以下のように求める。

c_i: 検査記号　$c_i = a_{i-3} + c_{i-6}$

送信する符号語は、$(a_1 c_1\ a_2 c_2\ a_3 c_3\ a_4 c_4\ a_5 c_5 ,,,,)$ とする。

Encoding method
Assuming that the information bits are (a_1 a_2 a_3 a_4 a_5 ,,,,), the check symbol c_i is obtained as follows.
c_i: check symbol c_i=a_(i-3)+c_(i-6)
The codeword to be transmitted is (a_1 c_1 a_2 c_2 a_3 c_3 a_4 c_4 a_5 c_5 ,,,,).

情報通信基礎Ⅰ　第９章　復号方式

Chapter 9 Decoding

情シス　渡辺

ポイント

・この章では、受信側で復号する立場で考察する。

・最小距離は第５章は符号語をベクトルで扱った。ここでは多項式で扱う方法を学ぶ。

・復号方式としては、MAP 復号、MLD 復号、MDD 復号、BDD 復号がある。

・符号長、情報ビット長、誤り訂正能力の議論

Points

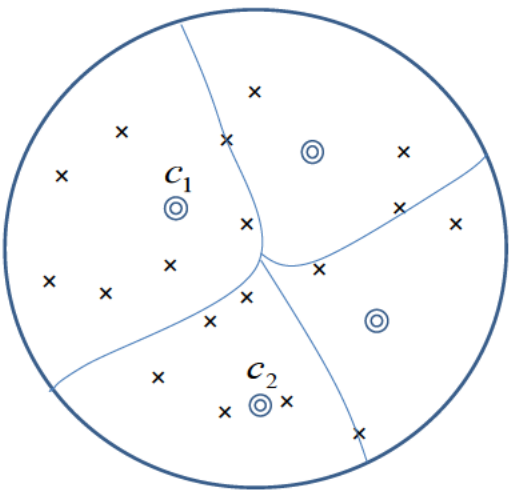
-In this chapter, we will consider coding from decoding view on the receiving side.

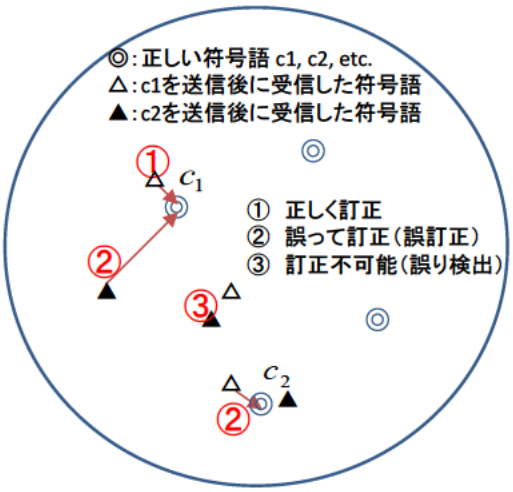
- Decoding methods include MAP decoding, MLD decoding, MDD decoding, and BDD decoding.

- We discuss code length, information bit length, and error correction capability.

９．１　復号方式

9.1 Decoding methods





９．１．１　復号の基礎

9.1.1 Basics of decoding

１）確率的復号

1) Probabilistic decoding

符号語の生起確率や通信路行列を利用する
MAP 復号、MLD 復号

This method uses probabilities of a codeword and channel matrices. It includes MAP decoding and MLD decoding.

２）代数的復号

2) Algebraic decoding

符号語と受信語の距離を利用する
シンドローム復号、MDD, BDD

This method uses Hamming distance between possible correct codeword and the received word. It includes Syndrome decoding, MDD, and BDD.

・MAP 復号　最大事後確率復号法 (maximum a posterior probability decoding)
・MLD 復号　最尤復号法 (maximum likelihood decoding)
・MDD 復号　最小距離復号法 (minimum distance decoding)
・BDD 復号　限界距離復号法 (bounded distance decoding)
・その他（繰り返し復号、ビタビ復号等）

- MAP decoding: Maximum a posterior probability decoding
- MLD decoding: Maximum likelihood decoding
- MDD decoding: minimum distance decoding
- BDD decoding: Bounded distance decoding
- Others (iterative decoding, Viterbi decoding, etc.)

９．１．２）MAP 復号　最大事後確率復号（maximum a posterior probability decoding）

9.1.2) MAP decoding: maximum a posterior probability decoding

受信語r に対して、すべての符号語c_i $(i = 1,2,\ M)$（M は符号語数）についての事後確率$P(c_i|r)$を求め、それを最大とする符号語を選ぶ。すなわち、

$\hat{m} = \arg\max_i P(c_i|r)$　　となる符号語$c_{\hat{m}}$を送信符号語と推定する復号法を MAP 復号という。最大事後確率復号法は、復号誤り確率を最小とする。

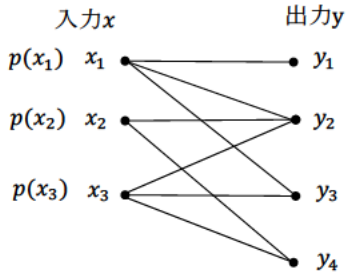
For the received word r, MAP obtains the posterior probability P(c_i|r) for all codewords c_i (i=1,2,M) (M is the number of codewords). Then, it selects the codeword with maximum value. That is, MAP assumes a codeword c_m^ satisfying m^=arg max i P(c_i | r) to be a transmission codeword. The maximum a posteriori probability decoding method minimizes the decoding error probability.

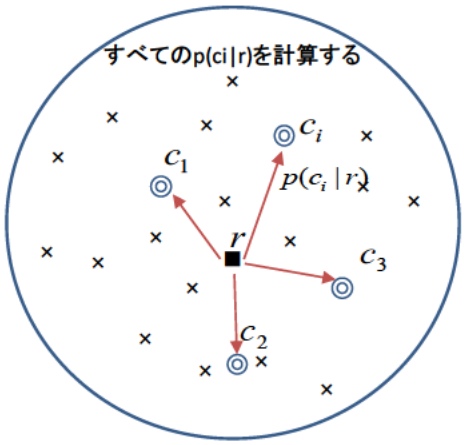
MAP 例：通信路行列および生起確率を以下とする。

MAP example: Assume the following channel matrix and occurrence probability.

$$\mathrm{P} = \begin{pmatrix} p(y_1|x_1) & p(y_2|x_1) & p(y_3|x_1) & p(y_4|x_1) \\ p(y_1|x_2) & p(y_2|x_2) & p(y_3|x_2) & p(y_4|x_2) \\ p(y_1|x_3) & p(y_2|x_3) & p(y_3|x_3) & p(y_4|x_3) \end{pmatrix} = \begin{pmatrix} 0.3 & 0.5 & 0.2 & 0 \\ 0 & 0.5 & 0 & 0.5 \\ 0 & 0.3 & 0.4 & 0.3 \end{pmatrix}$$

$p(x_1) = 0.5 \qquad p(x_2) = 0.3 \qquad p(x_3) = 0.2$





ベイズ則

We use Bayesian rule.

$$p(x_i|y_j) = \frac{p(x_i,y_j)}{p(y_j)} = \frac{p(x_i)p(y_j|x_i)}{\sum_{k=1}^{3} p(x_k)p(y_j|x_k)}$$

$$p(x_1|y_1) = \frac{p(x_1)p(y_1|x_1)}{p(x_1)p(y_1|x_1) + p(x_2)p(y_1|x_2) + p(x_3)p(y_1|x_3)} = \frac{0.5 * 0.3}{0.5 * 0.3 + 0.3 * 0 + 0.2 * 0} = 1$$

同様に、

Similarly,

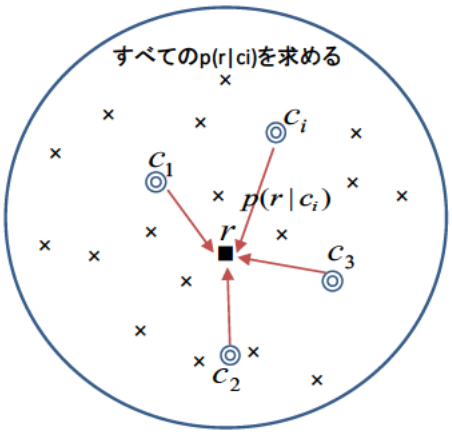
$$p(x_1|y_2)=\frac{0.5*0.5}{0.5*0.5+0.3*0.5+0.2*0.3}=\frac{25}{46} \qquad p(x_2|y_1)=0 \qquad p(x_3|y_1)=0$$
$$p(x_1|y_3)=\frac{5}{9} \qquad p(x_2|y_2)=\frac{15}{46} \qquad p(x_3|y_2)=\frac{6}{46}$$
$$p(x_1|y_4)=0 \qquad p(x_2|y_3)=0 \qquad p(x_3|y_3)=\frac{4}{9}$$
$$p(x_2|y_4)=\frac{5}{7} \qquad p(x_3|y_4)=\frac{2}{7}$$

以上より、
Then, we obtain,

$y_1 \to x_1$　　$y_2 \to x_1$　　$y_3 \to x_1$　　$y_4 \to x_2$

９．１．３　MLD 復号　最尤復号（maximum likelihood decoding）
9.1.3 MLD decoding: maximum likelihood decoding

通常、符号語c_i は等確率で生起すると仮定できる。（第一定理を考えよ。）このとき、事後確率$P(c_i|r)$を最大にする符号語を求めることは、ベイズ規則より尤度$P(r|c_i)$ を最大にする符号語を求めることと等しい。すなわち、$\hat{m}=arg\max_i P(r|c_i)$となる符号語$c_{\hat{m}}$を送信符号語とできる。この復号法を MLD 復号という。通信路行列から比較的容易に判断できる。最尤復号法は、符号語の生起確率が等確率のとき復号誤り確率を最小とする復号法である。



Usually, we can assume that the codeword c_i occurs with equal probability regardless of the index i. (Consider the Shannon first theorem.) At this time, finding a codeword that maximizes the posterior probability P(c_i |r) is equivalent to finding a code word that maximizes the likelihood P(r|c_i) according to the Bayesian rule. That is, a codeword c_m ^ that satisfies m ^=arg max I P(r|c_i) is supposed to be the transmitted codeword. This decoding method is called MLD decoding. It can be found easily from the channel matrix. The maximum likelihood decoding method is a decoding method that minimizes the probability of decoding errors when the probability of occurrence of codewords is equal.

MLD 例：通信路行列を以下とする。(MAP 例と同じ)

MLD example: Let’s assume the channel matrix is as follows. (the same to MAP example)

$$P = \begin{pmatrix} 0.3 & 0.5 & 0.2 & 0 \\ 0 & 0.5 & 0 & 0.5 \\ 0 & 0.3 & 0.4 & 0.3 \end{pmatrix}$$

列ベクトルの最大要素を選ぶ

Pick the largest element of a column vector

$$P = \begin{pmatrix} 0.3 & 0.5 & 0.2 & 0 \\ 0 & 0.5 & 0 & 0.5 \\ 0 & 0.3 & 0.4 & 0.3 \end{pmatrix}$$

以上より、

Then, we obtain,

$y_1 \rightarrow x_1$　　$y_2 \rightarrow x_1$ *or* x_2　　$y_3 \rightarrow x_3$　　$y_4 \rightarrow x_2$

９．１．４　MDD 復号　最小距離復号（minimum distance decoding）

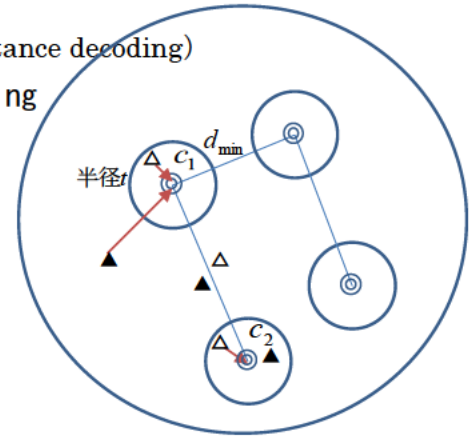
9.1.4 MDD decoding: minimum distance decoding

受信語rが受信されたとき、rから最小の距離に位置する符号語を送信符号語$c_{\hat{m}}$と推定する。二元対称通信路では、最小距離復号は最尤復号と等価である。

When the word r is received, the codeword located at the smallest distance from r is assumed to be the transmitted codeword c_m^. For binary symmetric channels, minimum distance decoding is equivalent to maximum likelihood decoding.

９．１．５　BDD 復号　限界距離復号（bounded distance decoding）

9.1.5 BDD decoding: Bounded distance decoding



最小ハミング距離d_{min}に対して、

$d_{min} \geq 2t_1 + 1$

を満足するt_1を定め、受信語r が受信されたとき、

$d_H(c_i, r) \leq t_1$

なる符号語を送信符号語$c_{\hat{m}}$と推定する。

ただし、$d_H(c_i, c_j)$は、符号語c_iとc_jのハミング距離である。

この復号法では、t_1個以下の誤りを訂正する。

For the minimum Hamming distance d_min, define t_1 such that d_min≥ 2t_1+1. When a received word r is received, c_i satisfying

d_H (c_i,r) ≤ t_1

is assumed to be the transmitted codeword c_m ^,

where d_H (c_i, c_j) is the Hamming distance between codewords c_i and c_j. This decoding method corrects t_1 or less errors.

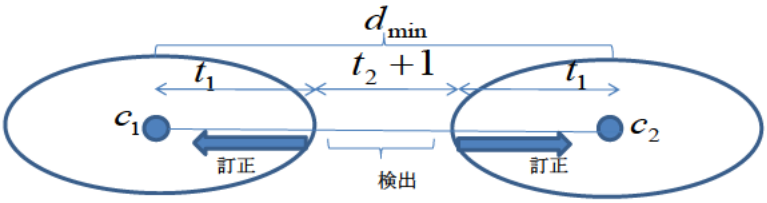
$t_0 = \left\lfloor \frac{d_{min}-1}{2} \right\rfloor$個までの誤りに対しては復号領域の重なりがないため，正しく訂正することが可能

である。すなわち、t_1の最大値は$t_0 = \left\lfloor \frac{d_{min}-1}{2} \right\rfloor$である。

Since there is no overlapping of decoding regions for errors up to t_0=⌊(d_min-1)/2⌋, errors can be corrected. That is, the maximum value of t_1 is t_0=⌊(d_min-1)/2⌋.

５－１）$d_{min} \geq 2t_1 + 1$となるようなt_1個以下の誤り訂正を行う

5-1) Error correction for d_min≥2t_1+1



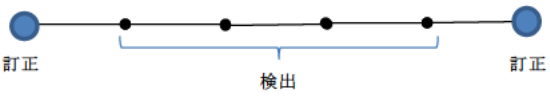
$t_2 + 1 = d_{min} - 2t_1$とおくと、上図では、t_1個以下の誤り訂正と、$t_1 + 1$個以上、$t_1 + t_2$個以下の誤り検出可能

t_1を大きくすると訂正能力は高くなるが誤って復号される確率も増大する。

As t_2+1=d_min-2t_1 in the above figure, it is possible to correct t_1 errors or less and detect t_1+1 errors or less, or t_1+t_2 errors or less.

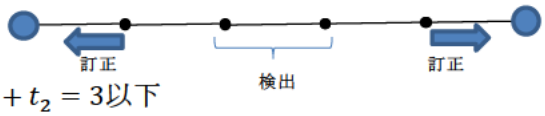
If we increase t_1, then the correction capability increase, however it increases the probability of erroneous decode.

(例)　$d_{min} = 5$の符号を考える。
訂正能力　$t_1 = 0$　の時　　$t_2 + 1 = 5$
従って検出能力は、$t_1 + 1 = 1$以上　$t_1 + t_2 = 4$以下



(Example1)
If d_min=5, when t_1=0 and t_2+1=5.
Therefore, the detection capability is t_1+1=1 or more, and t_1+t_2=4 or less.

訂正能力　$t_1 = 1$　の時　　$t_2 + 1 = 3$
従って検出能力は、$t_1 + 1 = 2$以上　$t_1 + t_2 = 3$以下



(Example2)
If d_min=5, t_1=1 and t_2+1=3.
Therefore, the detection capability is t_1+1=2 or more, and t_1+t_2=3 or less.

訂正能力　$t_1 = 2$　の時　　$t_2 + 1 = 1$
従って検出能力は、$t_1 + 1 = 3$以上　$t_1 + t_2 = 2$以下（不可）
　　訂正能力は増大するが、検出能力は０で、復号誤り確率が増大する。

(Example3)
If d_min=5, t_1=2 and t_2+1=1.
Therefore, the detection capability is t_1+1=3 or more, and t_1+t_2=2 or less (it means impossible).

Although the correction capability increases, the detection capability is 0 and the probability of erroneous decode increases.

訂正ができなくても検出ができれば、ARQ で再送する方法があり得る。従って、訂正能力をあまり大きくしない方が総合的にはよい場合がある。

If the detection is possible even if the correction is impossible, the detection enables retransmission by ARQ. Therefore, there are some cases where it is generally better not to increase the correction capability too much.

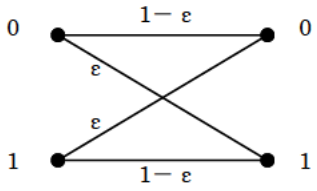
５－２）BSC(binary symmetric channel)における距離限界復号法の復号特性

5-2) Decoding of BDD method in BSC (binary symmetric channel)

符号長をn、ビット誤り率をεとすると、正しく復号される確率p_cは、

$$p_c = \sum_{i=0}^{t_1} \binom{n}{i} \varepsilon^i (1-\varepsilon)^{n-i}$$

となる。



If the code length is n and the bit error rate is ε, the probability of correct decoding, p_c is as follows.

一方$t_1 + t_2 + 1$個以上のビット誤りが生じると、復号誤りが発生するので、復号誤り率p_eは、以下が成立する。

On the other hand, if t_1+t_2+1 bits or more errors occur, a decoding error occurs, so that the following holds for the decoding error rate p_e.

$$p_e \leq \sum_{i=t_1+t_2+1}^{n} \binom{n}{i} \varepsilon^i (1-\varepsilon)^{n-i}$$

訂正不可能な誤りが検出される確率(誤り検出率): $p_d = 1 - p_c - p_e$

Therefore, the probability of detecting an uncorrectable error (error detection rate), p_d is p_d=1-p_c-p_e.

一般に、p_e、p_dは求めることが難しい。例えば、p_eには符号語の重み分布が必要であり、p_eは上界を示すことが多い。

Generally, it is difficult to obtain exact p_e and p_d. For example, p_e requires the codeword weight distribution, thus p_e typically indicates the upper bound of the error.

９．１．６　いくつかの議論

9.1.6 Some discussions

１）最尤復号 MLD、限界距離復号法 BDD の比較

1) Comparison between MLD and BDD

・BDD も MLD と同様に受信語に最も近い符号語を推定するが、すべての受信語に対して行うのではなく、各符号語を中心とする半径 t の球内に入る受信語に対してのみこの推定を行う。それ以外については推定を放棄する。

- BDD as well as MLD predicts the codeword closest to the received word. However, apart with MLD, BDD makes the prediction within a sphere of radius t centered on each code word. Otherwise, we abandon the prediction.

・正しく復号される確率p_cは、最尤復号の方が大きい　　pc（最尤復号）＞pc（限界距離復号）

- Probability p_c of correct decoding is greater than that in maximum likelihood decoding, that is, pc (MLD) > pc (MDD)

・BDD の方が比較的容易に実現できる。

- BDD can be implemented relatively easily.

・最小距離は「確実に」誤り訂正できる距離を示している。逆に言うと、誤り訂正の限界を示し

ているわけではない。

- The minimum distance indicates the distance within which distance any error can be corrected "certainly". Conversely, it does not indicate the actual limit of error correction.

例）符号語　A(0000000)　B(1100010)　C(0001011)　D(1110100)
　　最小距離＝３　従って、２ビット以上の誤りの訂正は保証されない。

Example) All codewords: A (0000000) B(1100010) C(0001011) D(1110100)
Since minimum distance of the code is 3, any errors of 2 bits or more can not be corrected.

・送信語 A(0000000)　受信語 V=(1100000)　の時
　最小距離から求めると、A, B, C, D との距離は、それぞれ 2, 1, 3, 2
　　B(1100010)に復号される－＞誤訂正

- When the sent codeword is A (0000000) and the received cordword V = (1100000), the distances from V to A, B, C, and D are 2, 1, 3, and 2, respectively. Thus, V is decoded to B(1100010). (Erroneous correction)

・送信語 A(0000000)　受信語 V=(0001100)　の時
　最小距離から求めると、A, B, C, D との距離は、それぞれ 2, 5, 3, 4
　　A(0000000)に復号できる。

- When the sent cordword is A (0000000) and the received cordword V = (0001100), the distances from V to A, B, C, and D are 2, 5, 3, and 4, respectively. Thus, V is decoded to A(0000000).

つまり同じ２ビット誤りでも正しく訂正される場合とそうでない場合がある。
In other words, the same 2-bit error may not be decoded correctly or incorrectly.

例えば、BSC（誤り率ε）で BCH(15, 7)符号を考える。BCH(15, 7)符号は最小距離$d_{min} = 5$。よって２ビット誤りまで訂正可能。

For example, let's consider a BCH(15,7) code in BSC (error rate is ε). BCH(15, 7) code has the minimum distance d_min=5. Therefore, up to 2-bit error can be corrected.

限界距離復号法で２ビット誤りまで訂正する（$t_1 = 2, t_2 = 0$）とすると、訂正できない誤りは３ビット以上であるので、

Assuming that up to 2-bit errors are corrected (t_1=2, t_2=0) by BDD, the number of uncorrectable errors is 3 or more bits, so for the error rate, p_e the following holds.

$$p_e = \sum_{i=t_1+t_2+1}^{n} \binom{n}{i} \varepsilon^i (1-\varepsilon)^{n-i} = \sum_{i=3}^{15} \binom{15}{i} \varepsilon^i (1-\varepsilon)^{15-i} = 1 - \sum_{i=0}^{15} \binom{15}{i} \varepsilon^i (1-\varepsilon)^{15-i}$$

$\varepsilon = 0.05$の場合、$p_e = 0.036$となる。一方、最尤復号法では（シミュレーションによって求めると）、$p_e = 0.0038$となる。この差は、最尤復号法では２ビット以上の誤りも訂正しているためである。

When ε=0.05, p_e=0.036. On the other hand, by the MLD, p_e=0.0038 by simulation. The difference is due to the fact that MLD corrects errors of 2 or more bits.

２）計算複雑度

2) Computational complexity

最尤復号法や最小距離復号法は、受信語とすべての符号語に対して、尤度やハミング距離の比較を行わなければならない。よって、符号長nの指数オーダの演算回数、例えば二元符号については$O(2^{nR})$を必要とする。R は符号化比率である。しかし、代数的復号法は、nの多項式オーダの演算回数（例えば二元 BCH 符号においては$O(n(\log n)^2)$）で実行する復号アルゴリズムが存在する。バーレカンプ-マッシーアルゴリズム、ユークリッド復号アルゴリズムなどはその例であり、広く実用に供されている。また、畳込み符号のように最尤復号を少ない計算量で実行する復号アルゴリズムも存在する。

MLD and MDD must compare likelihood and Hamming distance between the received cordword and all codewords. Therefore, the calculation requires the exponential order of the code length n. For example for a binary code, o(2^nR) is required, where R is the coding ratio. However, there are

algebraic decoding algorithms that perform in polynomial-order calculations of n (eg, o(n(logn)^2) for a binary BCH code). The examples include Berlekamp-Massey algorithm and Euclidean decoding algorithm, which are widely put to practical use. There is also a decoding algorithm that executes maximum likelihood decoding with a small amount of calculation, such as convolutional code.

参考書等
1．電子情報通信学会『知識の森』(http://www.ieice-hbkb.org/) 1 群（信号・システム）2 編（符号理論）1 章符号理論の基礎（鎌部浩、鴻巣敏之）
2．今井秀樹　情報理論　昭晃堂
3．村上孝三　情報通信基礎１講義資料
4．平澤茂一　符号理論

９．２　符号の限界式（発展）

9.2 Code bound

９．２．１　概要と準備

9.2.1 Preparations

概要

・線形符号の訂正能力と符号化率の間のトレードオフ

・符号長n、符号語数M、最小距離d_{min}の間の関係

・ハミング限界式、プロトキン限界式、シングルトン限界式

n、d_{min}に対して、符号語数Mの上界を与える

あるパラメータにおける符号の非存在性を与えている

・バルシャモフ-ギルバート限界式

n、M、d_{min}がある関係式を満たすならばその符号の存在を保証

Overview

- Trade-off between linear code correction capability and coding rate
- Relationship among code length n, number of code words M, minimum distance d_min
- Hamming bound, Plotkin bound, Singleton bound give upper bounds on the number of codewords M for n and d_min. They also show the non-existence of code in some parameters.
- Barshamov-Gilbert bound guarantees the existence of codes if n, M, d_min satisfy some relations.

・準備

q元(n, k, d_{min})線形符号Cを考える。

符号長n、符号語数M、最小距離d_{min}

nとMが与えられた時、d_{min}をできる限り大きくする

nとd_{min} が与えられた時、M をできる限り大きくする

情報シンボル数$k = log_q M$、検査シンボル数$m = n - k$、最大誤り訂正能力$t = \left\lfloor \frac{d_{min}-1}{2} \right\rfloor$

Consider a q-ary (n, k, d_min) linear code C, where code length n, number of code words M, and minimum distance d_min.
For given n and M, We would like to make d_min as large as possible, or, to make M as large as possible for given n and d_min.

Note that the number of information bits k is k=log_q M, number of check bits m is m=n-k, and maximum error correction capability t is t=⌊(d_min-1)/2⌋.

９．２．２　ハミング限界(Hamming bound)

9.2.2 Hamming bound

・符号長nと誤り訂正能力tに対する符号語数Mの上界は、以下で与えられる。

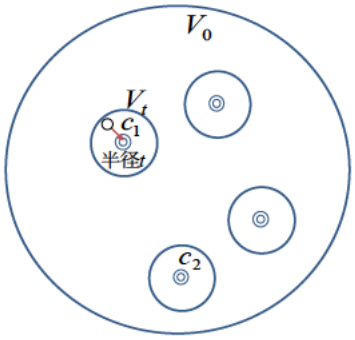
- The upper bound of the number of codewords M for the code length n and the error correction capability t is given below.

$$M \leq \frac{q^n}{\sum_{i=0}^{t} \binom{n}{i} (q-1)^i} \quad \cdots (1)$$

・等号が成立する符号は完全符号と呼ばれる

２元完全符号：ハミング符号、ゴーレイ符号

- A code for which the equal sign holds is called a complete code.
  Ex. Binary perfect codes include Hamming code, and Golay code



・証明概略：符号のq 次元ベクトル空間球体とその中の符号語c_iを中心とする球から、球体に存在しうる符号語数を求める。

- Outline of the proof

From the q-dimensional vector space sphere of code whose center is the codeword c_i, the number of codewords that can exist in the sphere is obtained.

c_iから距離 $0(i=0)$にある符号語の数 1

c_iから距離 $1(i=1)$ にある符号語の数 $\binom{n}{1}(q-1)^1$

c_iから距離 $2(i=2)$にある符号語の数 $\binom{n}{2}(q-1)^2$

従って、c_iから距離t以下の符号の総数（体積）V_tは、以下となる。

number of codewords at distance 0 (i=0) from c_i: 1
number of codewords at distance 1 (i=1) from c_i: (■(n@1)) 〖(q-1)〗^1
number of codewords at distance 2 (i=2) from c_i: (■(n@2)) 〖(q-1)〗^2

Therefore, the total number (volume) V_t of codes at a distance t or less from the center c_i is as follows.

$$V_t = \sum_{i=0}^{t} \binom{n}{i} (q-1)^i \quad \cdots (2)$$

全符号語数（空間体積）V_0は$V_0 = q^n$。球体の中に入る球の数から$M \leq \frac{V_0}{V_t}$となる。

Total number of symbol words (space volume) V_0 is V_0=q^n. From the number of inner spheres within a sphere, M≤V_0/V_t.

・（1）式は、$n-k=m \geq log_2 \sum_{i=0}^{t} \binom{n}{i}$ とも表現される。$(q=2)$

- Equation (1) can also be expressed as n-k=m≥log_2Σ_(i=0)^t(■(n@i)). (q=2)

９．２．３　プロトキン限界(Plotkin bound)

9.2.3 Plotkin bound

n、Mに対する最小距離d_{min}の上界は以下となる。

The upper bound of the minimum distance d_min for n and M is as follows.

$$d_{min} \leq \frac{nM(q-1)}{(M-1)q} \quad \cdots (3)$$

任意の２つの符号語間の距離の平均値は、最小距離より大きくなることから導出可能

It can be derived from the fact that the average distance between any two codewords is greater than the minimum distance.

・等号が成立する符号は等距離符号と呼ばれる。

　　等距離符号：シンプレックス符号

A code for which an equal sign holds is called an equidistant code.

　Ex. Simplex code

９．２．４　シングルトン限界(Singleton bound)

9.2.4 Singleton bound

$d_{min} \leq n - k + 1 \quad \cdots (4)$

・検査行列のランクが、高々$n-k$であることから導出可能。

It can be derived from the fact that the rank of a check matrix is at most n-k.

・等号を満たす符号は最大距離分離符号（maximum distance separable code; MDS 符号）と呼ばれる。

A code that satisfies the equal sign is called a maximum distance separable code (MDS code).

・二元の MDS 符号：(n,1,n)反復符号、$(n, n-1, 2)$パリティ検査符号（３項組は(n, k, d_{min})）。
q元の MDS 符号：リードソロモン符号
Binary MDS codes include (n,1,n) repetition code, (n,n-1,2) parity check code. A q-ary MDS code is Reed-Solomon code.

・（４）式は、$M = q^k \leq q^{n-d_{min}+1}$　とも表現できる。
- Formula (4) can also be expressed as M=q^k≤q^(n-d_min+1).

９．２．５　バルシャモフ-ギルバート限界（Varshamov-Gilbert bound）、VG 限界（線形符号）
9.2.5 Varshamov-Gilbert (VG) bound

以下の式が満たされる時、q元(n, k, d_{min})符号が存在する。（今井　p.143）
There exists a q-ary (n,k,d_min) code when the following equations are satisfied. (Imai p.143)

$$q^{n-k} > \sum_{i=0}^{d_{min}-2} \binom{n-1}{i} (q-1)^i \quad \cdots (5)$$

・ハミング限界、プロトキン限界、シングルトン限界は符号が存在するための必要条件であるのに対し、VG 限界は十分条件
- Hamming bound, Plotkin bound, and Singleton bound are necessary conditions for existence of a code, whereas VG bound is a sufficient condition.

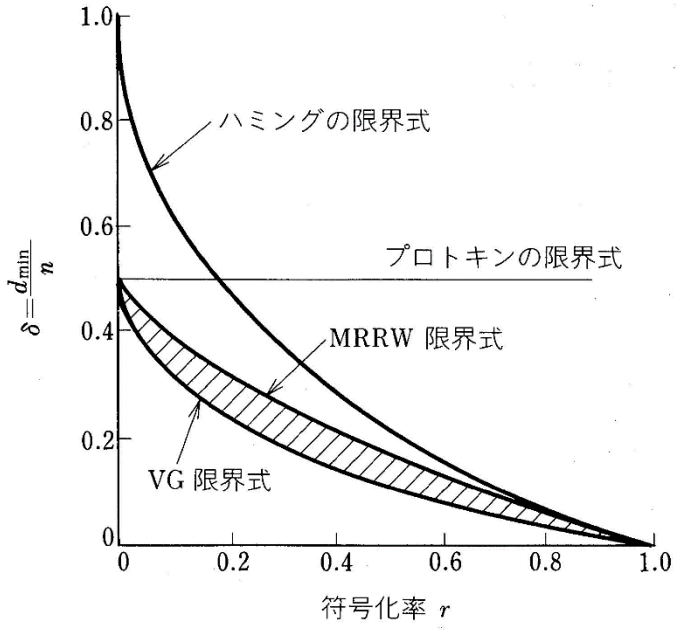
・VG 限界は符号長が短い場合は厳密な限界式にはならない。実際に符号長 1000 以下の BCH 符号は多くの場合この限界を超えている。一方、符号長が長い範囲ではこの限界式に達する符号を構成することは容易ではないが、代数幾何符号においていくつかの構成法が与えられている。

VG bound is not strict when code length is short. In practice, BCH codes whose code lengths are 1000 or less often exceed this bound. On the other

hand, it is not easy to construct a code that satisfies VG bound with long length, however some construction methods are given in algebraic-geometric codes.

・その他の上界の限界式として、マクエリース-ロディミッチ-ルンセィ-ウェルチ限界（McEliece-Rodemich-Rumsey-Welch bound; MRRW 限界）がある。

Another upper bound is McEliece-Rodemich-Rumsey-Welch bound (MRRW bound).



参考書等

1．今井秀樹　符号理論　電子情報通信学会

2．電子情報通信学会『知識の森』（http://www.ieice-hbkb.org/）1 群（信号・システム）2 編（符号理論）1 章符号理論の基礎（鴻巣敏之）、3 章符号の性能（和田山正、森井昌克）

プロトキン限界の証明

Proof of Plotkin bound

０）全体像

0) Overview

・以下では、$q = 3$、$n = 4$、$\{0,1,2\}$を例として挙げるが、容易にq元へ一般化可能

- We consider q=3, n=4, {0, 1, 2} as examples, but it can be generalized to more than q elements.

・符号語の全ペアの総距離を導出（１～８）、全ペア数で割って平均を導出（９～１１）

We firstly derive the total distance of all pairs of codewords (step 1 through 8), then divide it by the total number of pairs to obtain the average (step 9 through 11).

１）全符号語

1) All codewords

0000
0001
0002
0010
:
2222

$3^4 = q^n$

２）この中で、１番目の記号が０のものの数M_1は、以下のように求められる。

2) Among them, M_1, the number of codes whose first symbol is 0 is obtained as follows.

0000
0001
0002
0010
:
0222

$M_1 = 3^3 = q^{n-1}$

３）逆に１番目の記号が０ではないものの数は、$M - M_1$個

3) Conversely, the number of items whose first symbol is not 0 is M-M_1.

４）第１記号=0 の符号語と第１記号≠0 の符号語のペアの「第１記号だけに限った距離」（以下第１記号距離と表現）は、１。（例：0001 と 2012 の第１記号距離は１）
従って、第１記号=0 の符号語と第１記号≠0 の符号語の全ペアの第１記号距離の総和は、組み合わせの数と同数。つまり、２）と３）の組み合わせの数、$M_1(M-M_1)$。

4) The "distance limited to only the first symbol" (hereinafter referred to as First Symbol Distance) of a pair of a codeword with the first symbol=0 and a codeword with the first symbol is not 0 is 1. (Example: First Symbol Distance between 0001 and 2012 is 1).
Therefore, the total sum of the first symbol distances of all pairs of codewords with the first symbol is 0 and codewords with the first symbol is not 0 equals to the number of combinations. That is, the number of combinations of 2) and 3), M_1 (M-M_1).

５）次に第１記号=1 の符号語を考える。４）と同様に、第１記号=1 の符号語と第１記号≠1 の符号語のペアの第１記号距離は、１。従って、第１記号=1 の符号語と第１記号≠1 の符号語の全ペアの第１記号距離の総和は、組み合わせの数と同数。第１記号=1 の符号語の数をM_2とすれば、２）と３）と同様に考えて組み合わせの数、$M_2(M-M_2)$。
同様に、第１記号=$q-1$の符号語と第１記号≠ $q-1$の符号語の全ペアの第１記号距離の総和は、$M_q(M-M_q)$。

5) Then, let's consider a codeword with the first symbol=1. As in 4), the first symbol distance of a pair of codewords with first symbol is 1 and codewords with first symbol is not 1 is one. Therefore, the total sum of the first symbol distances of all pairs of codewords with the first symbol is 1 and codewords with the first symbol is not 1 equals to the number of combinations. If the number of codewords with the first symbol is 1 is M_2, the number of combinations is M_2 (M-M_2) in the same way as 2) and 3). Similarly, the sum of first symbol distances of all pairs of codewords with first symbol is q-1 and codewords with first symbol is not q-1 is M_q (M-M_q).

６）よって、第１記号距離の総和S_1は、以下で計算される。

6) Therefore, S_1, the sum of the first symbol distances is calculated as follows.

$$S_1 = M_1(M - M_1) + M_2(M - M_2) + \cdots + M_q(M - M_q) = M^2 - ({M_1}^2 + {M_1}^2 + \cdots + {M_q}^2)$$

７）$S_1 = M^2 - ({M_1}^2 + {M_1}^2 + \cdots + {M_q}^2)$　が最大となるのは、

$M_1 = M_2 = \cdots = M_q = \frac{M}{q}$　の時（証明は※）

7) S_1 is maximized when M_1=M_2=…=M_q=M/q (the proof is later ∗).

よって、

Therefore,

$$S_1 \leq M^2 - q\left(\frac{M}{q}\right)^2 = \frac{q-1}{q}M^2$$

８）１）～７）と同様に第２記号距離の総和S_2を求めると、$S_2 \leq \frac{q-1}{q}M^2$

8) Similarly to 1) to 7), S_2, the second symbol distances is satisfied the inequality S_2≤(q-1)/q M^2.

よって、全符号語ペアの距離の総和Sは、$S = S_1 + S_2 + \cdots + S_n \leq n\frac{q-1}{q}M^2$

Therefore, S, sum of the distances of all codeword pairs satisfy the inequality S=S_1+S_2+…+S_n≤n (q-1)/q M^2.

９）全符号ペア数は、５）で逆順（例えば、1-2 と 2-1）も重複してカウントしていることに注意すると、$_MP_2 = M(M-1)$　である。

9) The total number of code pairs is _M P_2=M(M-1), since in 5) the reverse order (e.g. 1-2 and 2-1) is duplicated.

１０）よって、１ペアの符号間の距離の平均、すなわち平均距離d_{ave}は、以下である。

10) Therefore, the average of the distances between one pair of codes, that is, the average distance d_ave is as follows.

$$d_{ave} = \frac{S}{{}_M P_2} = \frac{n\frac{M^2(q-1)}{q}}{M(M-1)} = \frac{nM(q-1)}{q(M-1)}$$

１１）d_{min}は、d_{ave}より小さいので、$d_{min} \leq d_{ave} = \frac{nM(q-1)}{q(M-1)}$　となる。

11) Since d_min is smaller than d_ave, we obtain d_min≤d_ave=(nM(q-1))/(q(M-1)).

※　$M = M_1 + M_2 + \cdots + M_q$ の条件下で、$f(\mathbb{M}) = \sum_{i=1}^{q} {M_i}^2$を最小化する。
Lagrange の未定定数法を用いる。

* Minimize f(M)=Σ_(i=1)^q 〖M_i〗^2 under the condition M=M_1+M_2+⋯+M_q.
We use Lagrange's method of undetermined constants as follows.

$$F(\mathbb{M}, \lambda) = \sum_{i=1}^{q} {M_i}^2 + \lambda g(\mathbb{M}) \quad \text{and } g(\mathbb{M}) = \sum_{i=1}^{q} M_i - M = 0$$

$\frac{\partial}{\partial M_i} F(\mathbb{M}, \lambda) = 2M_i + \lambda = 0$　　$\therefore M_i = -\frac{\lambda}{2}$ for all i

$g(\mathbb{M}) = \sum_{i=1}^{q} M_i - M = 0$

Thus, we finally obtain the following.

$M_i = \frac{M}{q}$

以下は没

以下は配布しない

BDD の詳細

$d_{\min} \geq 2t_1 + 1$　となるような t_1 個以下の誤り訂正を行う

t_1 の最大値は、$t_0 = \lfloor (d_{\min} - 1)/2 \rfloor$　　t_0 個以下の誤り訂正可能

$t_2 + 1 = d_{\min} - 2t_1$ とおくと

上図では、t_1 個以下の誤り訂正。$t_1 + 1$ 個以上、$t_1 + t_2$ 個以下の誤り検出可能

t_1 を大きくすると訂正能力は高いが同時に誤って復号される確率も増大する。

t_1 を大きくしないほうが良い場合もあるので t_1 の設計は重要。

(例題)　$d_{\min} = 5$　の符号を考える。

訂正能力　$t_1 = 0$　の時　　$t_2 + 1 = 5$。

従って検出能力、$t_1 + 1 = 1$　$t_1 + t_2 = 4$

訂正能力　$t_1 = 1$　の時　　$t_2 + 1 = 3$。

従って検出能力、$t_1 + 1 = 2$　$t_1 + t_2 = 3$

訂正能力　$t_1 = 2$　の時　　$t_2 + 1 = 1$。

従って検出能力、$t_1 + 1 = 3$　$t_1 + t_2 = 2$

訂正能力は増大したが、検出能力は０で、復号誤り確率が増大する。

符号 w_i の復号領域 Ω_i

$$\Omega_i = \{u; d_H(w_i, u) \leq t_1\}$$

Ω_i が重なり合わない場合:誤り個数が t_1 以下なら誤り訂正可能

$$d_{\min} \geq 2t_1 + 1$$

距離限界復号法:上式を満たす t_1 を定め復号を行なう

t_1 の最大値 t_0

$$t_0 = \lfloor (d_{\min} - 1)/2 \rfloor$$

ただし、$\lfloor x \rfloor$ は x 以下の最大の整数を表す

$t_2 + 1 = d_{\min} - 2t_1$ としたとき、$t_2 \geq 1$ ならば $t_1 + 1$ 個以上 $t_1 + t_2$ 個以下の誤りを検出可能

t_1 を大きくする　⇒　正しく復号される確率大、しかし誤って復号される確率も大

渡辺メモ

MLD　最尤
ビット誤り率が p の BSC を介して符号語 w を送ったとき、y が受信される確率は、
P(y|w)=p^t(1-p)^n-t
ただし、t は誤りの個数、
t=dH(w, y)
である。
最尤復号法は、y が受信された時、p(y|w)を最大とする符号語が送られたと推定する復号法である。
ビット誤り率 p は、通常 0<p<1/2 を満たすから、P(y|w)は単調減少関数である。従って、最尤復号を行うには、y に対し
t=dH(w, y)が最小となる符号語 w が送られたと推定すればよい。言い換えると、
受信語 y とハミング距離が一番近い w が送られたと推定するのが BSC における最尤復号である。

(n, k, d) 符号のすべての符号語 vm, m = 1, 2, ・・・ ,M, はすべて等確率で用いられるとする。vm が送信され w が受信されたとき、最尤復号(maximum likelihood decoding：MLD) 法は w から次式を満足する v ^m を見つけ出すことである。w → v ^m|[max P(w|vi) = P(w|v ^m)]
1≤i≤M
もし、 ^m =/= m なら復号誤りが生起している。誤り確率 ε , 0 ≤ ε < 0.5, の 2 元対称通信路を仮定すると、MLD 法は最小距離復号(minimum distancedecoding：MDD) 法と同一である。すなわち
DH (v ^m,w) = e
とすると
P(w|v ^m) = ε^e(1 − ε)^n−e
であるから、もし e が最小なら P(· |·) は最大である。したがって、ハミング距離が w に最も近い符号語 v^m を見出す方法が MLD 法といえる。

しかし、通常の代数的復号法は w から
w → v ^m |[DH (v ^m, w) ≤[[(d − 1)/2]]]
となる v ^m を見つけ出す。この方法は、限界距離復号(bounded distance decoding：BDD) 法と呼ばれる。

BDD
実際には限界距離復号法は、代数的符号の設計最小距離に基づき、符号の代数的構造を利用した代数的復号法により行われる。設計最小距離
dxxx
は

dxxx<=dmin

であり、設計最小距離を最小距離とみなし、復号の計算量が実用的な代数的復号法が行われる。