QR コードの多色化

—誤り訂正能力が保証される条件—

早稲田大学大学院基幹理工学研究科
数学応用数理専攻修士 2 年 楫研究室所属
伊達 虎太郎 (5122A031)

指導教員：楫元

2024 年 1 月 29 日

目次

1　はじめに

1994 年にデンソーによって QR コードが開発されてからデータの読み取り方法として QR コードは多用されてきた。近年では決済方法や駅のホームドアの開閉、飛行機のチェックインなどその使用方法は多岐にわたる。

QR コードを読み取る際、QR コード自体が汚れていたり影がかかったりするなどして正しく読み取れない場合が起こりうる。そのような場合であっても誤りを検出、訂正し元の情報を正しく復元することを目的としたものが誤り訂正符号である。QR コードに用いられている誤り訂正符号はリード・ソロモン符号というものであり、リード・ソロモン符号は任意の有限体 $\mathbb{F}_q$ 上で定義することが可能である。しかし QR コードは白と黒の 2 色であるため QR コードでは標数 2 の体上でのみ用いられている。

よってリード・ソロモン符号を標数 3 や 5 の体上で考える、つまり QR コードの場合であれば 3 色や 5 色のものが存在した場合、どのような変化が生じ、多色化した QR コードには実用性があるのかを考察することがこの研究の目的である。

最終的な結論としては、ある条件を満たしていれば 3 色や 5 色にすることで今よりも小さい QR コードで同じだけの情報量を問題なく送ることができるようになる、というものになる。またこの研究では QR コードというリード・ソロモン符号を使っている 1 つの具体例について考えているが、本質的にはリード・ソロモン符号の標数を変えた場合の誤り訂正能力についての結果である。なのでこの結果を QR コードだけではなくリード・ソロモン符号が使われている他の情報伝達方法 (衛星通信や CD など) でも標数 3 や 5 の体上で考える際の 1 つの結果として捉えることができる。

また、先行研究として [12] によって QR コードの多色化が考察されているが、これは体の標数を変えることによって多色化しようという試みではなく、色の RGB 値ごとの値を変化させることで本来の QR コードを多層化しようという試みである。さらに SHIFT によって開発されたカメレオンコードという 2 次元バーコードも存在し、運用されている。しかし QR コードとは異なる点がいくつかあり、特にコード化できる情報量に関しては QR コードが大きく上回っている。一方で認識のスピードではカメレオンコードが上回っているため、カメレオンコードの仕組みは公開されていないが、おそらく QR コードとは異なる仕組みを使用していると思われる。

よってこの研究ではあくまでも QR コードの仕組みのまま、体の標数を 3 以上にすることによって多色化するとどのようになるのかを考えるものとする。

2　リード・ソロモン符号とその復号方法

この節では QR コードに使用されているリード・ソロモン符号に関連した定義、定理について書いていく。なおこの節の内容は [1] に記述されているものである。

前述のとおり誤り訂正符号とはデータを送信する際に起こりうる誤りを検出、訂正するための仕組みであり、具体的には長さ k の情報にあらかじめ $n-k>0$ の冗長を持たせ、長さ n にしておくことで誤りを訂正できるようにしておく。

定義 2.1.（符号化、復号）

$q=p^e$ $(e\geq 1)$ とする。長さ k の情報を $\mathbb{F}_q^k$ の元とし、$n>k$ としたとき $\mathbb{F}_q^k$ の元の符号化とは

$$E:\mathbb{F}_q^k\to\mathbb{F}_q^n$$

である。また、$C=E(\mathbb{F}_q^k)\subset\mathbb{F}_q^n$ を符号といい、C の元を符号語という。さらに $D:\mathbb{F}_q^n\to\mathbb{F}_q^k$ で

$$D\circ E=\mathrm{id}_{\mathbb{F}_q^k}$$

となるものを復号という。また、C が $\mathbb{F}_q^n$ の次元 k の部分空間となるとき C を線形符号という。

定義 2.2.（ハミング距離）

$x,y\in\mathbb{F}_q^n$ に対して

$$d(x,y)=\#\{i\mid x_i\neq y_i,1\leq i\leq n\}$$

で定義される $d(x,y)$ をハミング距離という。また明らかにハミング距離は距離関数である。

定義 2.3.（最小距離）

符号 C の最小距離 d を

$$d=\min\{d(x,y)\mid x\neq y\in C\}$$

で定義する。

つまり符号化とは語数を増やすことによって最小距離を大きくする操作である。最小距離が $d\geq 2t+1$ であるとき、C の符号語はそれぞれ少なくとも $2t+1$ 離れている。このとき $c\in C$ に対し、受信語 y が $y=c+e$ となっており誤りの個数 $\#\{i,1\leq i\leq n\mid e_i\neq 0\}\leq t$ であれば y とのハミング距離が t 以下である符号語は c ただ一つである。よって復号 D を y から最も近い符号語 c をとって $D(y)=D(c)$ とすれば t 以下の誤りについては正しく復号できることになる。

定義 2.4.（巡回符号）

任意の $c=(c_0,c_1,\ldots,c_{n-1})\in C$ に対して $(c_{n-1},c_0,c_1,\ldots,c_{n-2})\in C$ となる線形符号 C を巡回符号という。

命題 2.5. $\mathbb{F}_q^n$ の元を $\mathbb{F}_q[x]/\langle x^n-1\rangle$ の元に対応させる、つまり $(c_0, c_1, \ldots, c_{n-1})$ と $c_0+c_1\bar{x}+\cdots+c_{n-1}\bar{x}^{n-1}$ を対応させるとこれは線形同型である。さらに

$$\bar{x}(c_0 + c_1\bar{x} + \cdots + c_{n-1}\bar{x}^{n-1}) = c_{n-1} + c_0\bar{x} + c_1\bar{x}^2 \cdots + c_{n-2}\bar{x}^{n-1}$$

である。つまり $\mathbb{F}_q[x]/\langle x^n-1\rangle$ 上で $\bar{x}$ をかけることが、$\mathbb{F}_q^n$ 上で符号を右に巡回させることに対応している。

定理 2.6. $C \subset \mathbb{F}_q[x]/\langle x^n-1\rangle$ が巡回符号である $\Longleftrightarrow$ C が $\mathbb{F}_q[x]/\langle x^n-1\rangle$ のイデアル.

Proof. ”C が線形符号である $\Longleftrightarrow$$C$ が和と定数倍で閉じている”であることと、命題 2.5. によって従う。

□

ここで、$\mathbb{F}_q[x]/\langle x^n-1\rangle$ の任意のイデアルは単項イデアルである。

定義 2.7.（生成元多項式）
巡回符号 $C = \langle g\rangle/\langle x^n-1\rangle$ であるとき g を C の生成元多項式という。

以降、簡単のために符号語 $c = c_0 + c_1\bar{x} + \cdots + c_{n-1}\bar{x}^{n-1} \in \mathbb{F}_q[x]/\langle x^n-1\rangle$ を $\mathbb{F}_q[x]$ の高々 $n-1$ 次の多項式 $c_0 + c_1x + \cdots + c_{n-1}x^{n-1}$ として扱う。同時に C を $\mathbb{F}_q[x]$ の $\langle x^n-1\rangle$ を含むイデアルとして扱う。

定義 2.8.（原始元）
$\mathbb{F}_q$ を有限体としたとき、$\mathbb{F}_q \setminus \{0\}$ の乗法群としての生成元を $\mathbb{F}_q$ の原始元という。

定義 2.9.（リード・ソロモン符号）
α を $\mathbb{F}_q$ の原始元、$k < q$ とする。また、$L_{k-1} = \{\sum_{i=0}^{k-1} a_i t^i \mid a_i \in \mathbb{F}_q\}$ とする。このとき、

$$C = \{(f(1), f(\alpha), \ldots, f(\alpha^{q-2})) \in \mathbb{F}_q^{q-1} \mid f \in L_{k-1}\}$$

により定義される $n = q-1$ の符号 C をリード・ソロモン符号という。

定理 2.10. リード・ソロモン符号は巡回符号である。

Proof. [1]p567 参照。

□

命題 2.11. リード・ソロモン符号のブロック長は $n = q-1$、次元は $k = \dim L_{k-1}$、最小距離は $d = q-k = n-k+1$ である。また、これを (n, k, d) リード・ソロモン符号と書く。

よって体 $\mathbb{F}_q$ を決めるとリード・ソロモン符号の語長 n は $q-1$ に固定されてしまう。実用上これでは不便なので短縮リード・ソロモン符号というものが扱われる。

定義 2.12. （短縮リード・ソロモン符号）
(n,k,d) リード・ソロモン符号 C と $\mu < k$ に対して、$i_1, i_2, \ldots, i_\mu$ を固定し、

$$\{c = (c_0, c_1, \ldots, c_{n-1}) \in C \mid c_{i_1} = c_{i_2} = \cdots = c_{i_\mu} = 0\} \subset C \subset \mathbb{F}_q^n$$

を $(n-\mu, k-\mu, d')$ 短縮リード・ソロモン符号という。ただし d' は短縮リード・ソロモン符号の最小距離である。また、今後は $i_1, i_2, \ldots, i_\mu$ を $n-\mu, n-\mu+1, \ldots, n-1$ として、短縮リード・ソロモン符号を

$$\begin{aligned}&\{c = (c_0, c_1, \ldots, c_{n-\mu-1}, 0, \ldots, 0) \in C\}\\ \simeq&\{c = (c_0, c_1, \ldots, c_{n-\mu-1}) \mid (c_0, c_1, \ldots, c_{n-\mu-1}, 0, \ldots, 0) \in C\} \subset \mathbb{F}_q^{n-\mu}\end{aligned}$$

として考える。つまり C を $\mathbb{F}_q[x]$ のイデアルとして見たとき短縮リード・ソロモン符号は、

$$\{c = c_0 + c_1 x + \cdots + c_{n-\mu-1} x^{n-\mu-1} \in C\} \subset \mathbb{F}_q[x]$$

となるので C の元のうち高々 $n-\mu-1$ 次のものを集めた集合が $(n-\mu, k-\mu, d')$ 短縮リード・ソロモン符号である。

定理 2.13. 短縮リード・ソロモン符号の最小距離 d' と短縮する前の最小距離 d の間には

$$d' \geq d$$

という関係が成り立つ。

Proof. 短縮リード・ソロモン符号はもとのリード・ソロモン符号の部分集合なので明らか。

□

短縮の方法によっては d' は真に d より大きくなることがあるが、実際に d' を導くのは難しい。なので d' を小さく見積もり、$d' = d = n-k+1$ として考えることとする。

定理 2.14. 次元が k で最小距離 $d = q-k$ の $\mathbb{F}_q$ 上のリード・ソロモン符号の生成元多項式は

$$g = (x-\alpha)\cdots(x-\alpha^{d-1})$$

である。

Proof. [1]p569 参照。

□

ここでリード・ソロモン符号における符号化 $E : \mathbb{F}_q^k \to \mathbb{F}_q^n$ について考える。符号化する前の情報 c' の次元は k であるから

$$c' = c_{n-k} x^{n-k} + c_{n-k+1} x^{n-k+1} + \cdots + c_{n-1} x^{n-1} \in \mathbb{F}_q[x]$$

とする。ここで生成元多項式を g として c' を g で割ると

$$c' = ag + r \ \ (\deg(r) < \deg(g))$$

となってさらに、

$$c = c' - r$$

とすれば c は $\langle g \rangle = C$ の元なので符号語である。ここで g は $d-1 = n-k$ 次式なので、余り r は高々 $n-k-1$ 次式である。よって、

$$r = -c_0 - c_1 x - \cdots - c_{n-k-1} x^{n-k-1}$$

と表せば、

$$\begin{aligned} c &= c' - r \\ &= c_0 + c_1 x + \cdots + c_{n-1} x^{n-1} \end{aligned}$$

となる。またこのことから、$c = c_0 + c_1 x + \cdots + c_{n-1} x^{n-1}$ の $c_{n-k} x^{n-k} + c_{n-k+1} x^{n-k+1} + \cdots + c_{n-1} x^{n-1}$ の部分を情報部分、$c_0 + c_1 x + \cdots + c_{n-k-1} x^{n-k-1}$ の部分をパリティチェック部分と呼ぶ。

また短縮リード・ソロモン符号でも符号化の操作は同様に考える。

ここからはリード・ソロモン符号の復号方法を記述する。その後にその復号方法が短縮リード・ソロモン符号にも適用できることを示す。復号方法はいくつか知られているがここでは [1] で紹介されているものを扱う。体 $\mathbb{F}_q$ とその原始元 α を固定する。このとき

$$g = (x-\alpha) \cdots (x-\alpha^{d-1}) = (x-\alpha) \cdots (x-\alpha^{n-k})$$

を生成元多項式とするリード・ソロモン符号 $C \subset \mathbb{F}_q / \langle x^{q-1} - 1 \rangle$ を考える。また $d = 2t+1$ と仮定して受信語を $y \in \mathbb{F}_q^n$ とすれば、$y - e \in C,\ \ d(e,0) \leq t$ となる e が見つかるならば t 個以下の誤りは必ず訂正できる。したがって d が奇数の場合のみを考えることとする。

$c = \sum_{j=0}^{q-2} c_j x^j$ をリード・ソロモン符号 C の符号語とする。c を送信したとき、誤りがいくつか加えられ受信語は $y = c + e \ \ (e = \sum_{i \in I} e_i x^i \ \ (e_i \neq 0))$ となる。このとき I を誤り添字集合と呼び、e_i を誤り値と呼ぶ。復号するということはこの I と各 e_i を特定することである。

定理 2.15. 受信語を y とし、y の誤りは t 個以下としたとき、

$$S(x) = \sum_{j=1}^{d-1} y(\alpha^j) x^{j-1}$$

と定義する。このとき、

$$\begin{cases} \bar{\Omega} \equiv \bar{\Lambda} S \pmod{x^{2t}} \\ \deg(\bar{\Lambda}) \leq t \\ \deg(\bar{\Omega}) < \deg(\bar{\Lambda}) \\ \bar{\Omega} \text{と} \bar{\Lambda} \text{が互いに素である} \end{cases} \tag{$*$}$$

を満たす $(\bar{\Omega}, \bar{\Lambda})$ は定数倍を除いてただ一つ存在する。

Proof. [1]p583 参照。

□

命題 2.16. 誤りが t 個以下と仮定したとき、次のように Λ と Ω を定義する。

$$\Lambda = \prod_{i \in I}(1 - \alpha^i x), \quad \Omega = \sum_{i \in I} e_i \alpha^i \prod_{\substack{j \neq i \\ j \in I}}(1 - \alpha^j x)$$

このとき、(Ω, Λ) は $(*)$ を満たす解のうち Λ の定数項が 1 のものである。

このような Λ を見つけられたとすると、$\Lambda = 0$ の解を見つけることで I が求まる。さらに I が求まっていれば $i \in I$ について

$$\Omega(\alpha^{-i}) = \alpha^i e_i \prod_{\substack{j \neq i \\ j \in I}}(1 - \alpha^j \alpha^{-i})$$
$$\therefore \quad e_i = \frac{\Omega(\alpha^{-i})}{\alpha^i \prod_{\substack{j \neq i \\ j \in I}}(1 - \alpha^j \alpha^{-i})}$$

によって誤り値を求めることが可能である。

定義 2.17. （$>_r$ 順序）
$r \in \mathbb{Z}$ を固定したとき、$\mathbb{F}_q[x]^2$ の単項式順序 $>_r$ を

$$x^m \mathbf{e}_i >_r x^n \mathbf{e}_j \Longleftrightarrow \begin{cases} m > n & i = j \text{ のとき} \\ m + r \geq n & i = 2, j = 1 \text{ のとき} \end{cases}$$

と定義する。

定理 2.18. $(*)$ の合同式の条件を満たす $(\bar{\Omega}, \bar{\Lambda}) \in \mathbb{F}_q[x]^2$ 全体の集合

$$K = \{(\bar{\Omega}, \bar{\Lambda}) \mid \bar{\Omega} \equiv \bar{\Lambda} S \pmod{x^{2t}}\}$$

を考える。このとき

$$\{(x^{2t}, 0), (S, 1)\}$$

は $>_{\deg(S)}$ 順序に関する K のグレブナー基底であり、さらに $(*)$ の条件をすべて満たす $(\bar{\Omega}, \bar{\Lambda})$ は K の $>_{-1}$ 順序に関する極小元である。よって $(x^{2t}, 0), (S, 1)$ から K を計算し、その $>_{-1}$ 順序に関するグレブナー基底を求めればその元のうちの一つが上で定義した (Ω, Λ) の定数倍である。

よって求まった (Ω, Λ) から Λ の定数項が 1 となるように定数倍をすれば、$\Lambda = \prod_{i \in I}(1 - \alpha^i x), \quad \Omega = \sum_{i \in I} e_i \alpha^i \prod_{\substack{j \neq i \\ j \in I}}(1 - \alpha^j x)$ となるので復号が可能となる。また、この復号方法は y からのハミング距離が t 以下の符号語 $c \in C$ を求めるという方法なので、リード・ソロモン符号 C から作られる短縮リード・ソロモン符号 C' についても $c \in C'$ に t 個以下の誤り e が加えられ受信語が $y = c + e$ となったとき、y に最も近い C

の元を探せばそれは c ただ一つであるため、短縮リード・ソロモン符号であっても同様の方法で復号が可能である。

例 2.19. $\mathbb{F}_{16}$ における $(10, 6, 5)$ 短縮リード・ソロモン符号を考える。生成元多項式 g は原始元を α として

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

である。ただし α は $\alpha^4 + \alpha + 1 = 0$ を満たす。送りたい情報が

$$c' = \alpha^8 x^9 + \alpha^{14} x^7 + x^6 + \alpha^3 x^4$$

とする。これを符号化するために c' を g で割った余り r を求めると

$$r = \alpha^5 x^3 + x + \alpha^3$$

となるから送信する符号語 c は

$$\begin{aligned} c &= c' - r \\ &= \alpha^8 x^9 + \alpha^{14} x^7 + x^6 + \alpha^3 x^4 - \alpha^5 x^3 - x - \alpha^3 \\ &= \alpha^8 x^9 + \alpha^{14} x^7 + x^6 + \alpha^3 x^4 + \alpha^5 x^3 + x + \alpha^3 \end{aligned}$$

となる。ここに 2 個の誤り

$$e = \alpha^5 x^6 + \alpha^{13} x^2$$

が加えられ、

$$\begin{aligned} y &= c + e \\ &= \alpha^8 x^9 + \alpha^{14} x^7 + (1 + \alpha^5) x^6 + \alpha^3 x^4 + \alpha^5 x^3 + \alpha^{13} x^2 + x + \alpha^3 \end{aligned}$$

が受信されたとする。

$$\begin{aligned} S(x) &= \sum_{j=1}^{d-1} y(\alpha^j) x^{j-1} \\ &= \alpha^8 x^3 + \alpha^5 x^2 + \alpha^{12} \end{aligned}$$

であり、$\mathbb{F}_{16}[x]^2$ の加群 K は

$$K = \langle (x^{2t}, 0), (S, 1) \rangle$$

である。K の $>_{-1}$ 順序によるグレブナー基底 G を計算すると

$$G = \{(x^2 + \alpha^{10} x + \alpha^7, \alpha^{13} x + \alpha^{10}), (\alpha^7 x + \alpha^4, x^2 + \alpha^{10} x + \alpha^7)\}$$

となる。G の元のうち $(*)$ を満たすのは $(\alpha^7 x + \alpha^4, x^2 + \alpha^{10} x + \alpha^7)$ である。ここから Ω, Λ を導くために $x^2 + \alpha^{10} x + \alpha^7$ の定数項 α^7 で割ると $(\Omega, \Lambda) = (x + \alpha^{12}, \alpha^8 x^2 + \alpha^3 x + 1)$ となる。$\alpha^8 x^2 + \alpha^3 x + 1 = 0$ の解は $x = \alpha^9, \alpha^{13}$ であるので、$\Lambda = \prod_{i \in I} (1 - \alpha^i x)$ であることに注意すると α^9, α^{13} の逆元は α^6, α^2 なので誤り添字集合は

$$I = \{2, 6\}$$

である。よって

$$\begin{aligned} e_2 &= \frac{\Omega(\alpha^{-2})}{\alpha^2 \prod_{\substack{j\neq 2 \\ j\in I}}(1-\alpha^j\alpha^{-2})} \\ &= \frac{\Omega(\alpha^{13})}{\alpha^2(1-\alpha^6\alpha^{13})} \\ &= \frac{\alpha^{13}+\alpha^{12}}{\alpha^2(1-\alpha^4)} \\ &= \alpha^{13} \end{aligned}$$

$$\begin{aligned} e_6 &= \frac{\Omega(\alpha^{-6})}{\alpha^6 \prod_{\substack{j\neq 6 \\ j\in I}}(1-\alpha^j\alpha^{-6})} \\ &= \frac{\Omega(\alpha^{9})}{\alpha^6(1-\alpha^2\alpha^{9})} \\ &= \frac{\alpha^{9}+\alpha^{12}}{\alpha^6(1-\alpha^{11})} \\ &= \alpha^{5} \end{aligned}$$

となるので誤り e は $e = \alpha^5 x^6 + \alpha^{13} x^2$ とわかり送られた符号語は

$$\begin{aligned} c &= y - e \\ &= \alpha^8 x^9 + \alpha^{14} x^7 + x^6 + \alpha^3 x^4 + \alpha^5 x^3 + x + \alpha^3 \end{aligned}$$

とわかる。情報は c の 4 次以上の項を見ればよいので

$$c' = \alpha^8 x^9 + \alpha^{14} x^7 + x^6 + \alpha^3 x^4$$

が送りたい情報だったとわかり復号が成功した。

3　QR コードの規格 (JIS-X-0510)

現在使われている QR コードは QR コードモデル 2 というものであり、大きさが 1 型から 40 型まで存在し、送信するデータの量や誤り訂正レベルによって自動的に適切なものが選ばれる。この節の内容は [2]、[4] を参考にしている。

図 1　1 型

図 2　2 型

図 3　3 型

以降、QR コードの 1 つのマス目をセルと呼ぶこととする。例えば 1 型であれば 1 辺 21 セルの正方形なので全体で 441 セルである。そのうち QR コードが正しく認識されるために使われるセルがいくつかあるので実際にデータを入れられるセル数は 208 セルとなっている。

リード・ソロモン符号の符号語は $c = \sum_{j=0}^{q-2} c_j x^j$ と表され、通常の QR コードでは 8 セルで一つの c_j を表す。つまり 1 型の場合では語長 n は $208/8 = 26$ となる。

また、同じ型でも誤り訂正レベルがレベル L、M、Q、H の 4 段階設定されている。レベル L だと全体の約 7 %が読み取れなくても正しく復号できる。それぞれの誤り訂正レベルごとの復元可能割合は以下のとおりである。

誤り訂正レベル	L	M	Q	H
復元可能割合	7 %	15 %	25 %	30 %

1 型の誤り訂正レベル L の QR コードを 1-L と表すこととする。1 型のそれぞれの誤り訂正レベルごとの語長 n、次元 k は以下のとおりである。

	1-L	1-M	1-Q	1-H
n(語長)	26	26	26	26
k(情報量)	19	16	13	9

つまり型番の数字ごとに n は固定されており、誤り訂正レベルごとに k が変化する。パリティチェック部分の長さが $n - k$ なので k を小さくすることでパリティチェック部分を長くできるので復元可能割合が大きくなるのである。逆に情報部分の長さ k は小さくなるため、誤り訂正レベルを上げると送ることのできる情報は少なくなってしまう。

4 実験用のプログラム

3 節の復号の方法に従い、ランダムに生成した符号語 c と誤り e に対して singular 上で復号を行うプログラムを作成した。以下がそのプログラムである。

RS.txt

```
int countsuccess = 0;
int countfailure = 0;
int countloop = 1;

if((n-k)%2 == 1){
    n = n-1;
}
ring A = (color^cell,a),x,(lp,c);
```

```
int t = (n-k) div 2;
int i = 1;
poly g = 1;
while(i<=n-k){
    g = g*(x-a^(i));
    i++;
}

while(countloop<=10000){
    poly c0 = 0;
    int i = n-1;
    while(i>=n-k){
        c0 = c0+a^(random(0,color^cell-2))*x^(i);
        i--;
    }
    poly c = c0-reduce(c0,g);
    int i = 0;
    while(i<=n-1){
        if(random(1,100)<=prob){
            e = e+a^(random(0,color^cell-2))*x^(i);
        }
        i++;
    }
    poly y = c+e;

    poly S = 0;
    int i = 1;
    while(i<=n-k){
        S = S+subst(y,x,a^(i))*x^(i-1);
        i++;
    }
    vector v1 = [S,1];
    vector v2 = [x^(2*t),0];
    module K = v1,v2;
    module M = groebner(K);
    int i = 1;
    while(1){
        if(M[i][1]<M[i][2]){
        int m = i;
        break;
```

```
        }
        i++;
    }
    poly r = subst(M[m][2],x,0);
    if(r==0){
        r = 1;
    }
    poly M1 = M[m][1]/r;
    poly M2 = M[m][2]/r;

    int i = 0;
    int j = 0;
    vector er = (0);
    while(i<=(color^cell)-2){
        if(subst(M2,x,a^(i))==0){
            j++;
            er = er+(1/a^(i))*gen(j);
        }
        i++;
    }
    int i = 1;
    while(i<=j){
        poly P(i) = 1;
        int j' = 1;
        while(j'<=j){
            if(j' != i){
                P(i) = P(i)*(1-er[j']/er[i]);
            }
        j'++;
        }
        poly e(i) = subst(M1,x,(1/er[i]))/(er[i]*P(i));
        i++;
    }
    int i = 1;
    while(i<=j){
        int j' = 0;
        while(j'<=(color^cell)-2){
            if(a^(j') == er[i]){
                int num(i) = j';
            }
```

```
            j'++;
        }
        i++;
    }
    poly E = 0;
    int i = 1;
    while(i<=j){
        E = E+e(i)*x^(num(i));
        i++;
    }
    if(y-E == c){
        countsuccess++;
    }
    else{
        countfailure++;
    }

    countloop++;
}

printf("success %p", countsuccess);
printf("failure %p", countfailure);
```

このプログラムは 3 節のアルゴリズムに以下の操作を付け加えている。

- 3 節の復号は $n-k+1=d=2t+1$ と仮定して行われていたため、入力される n と k について $n-k$ が奇数であれば n を $n-1$ に置き換える。
- 加群 K の $>_{-1}$ 順序に関するグレブナー基底から $(\bar{\Omega},\bar{\Lambda})$ を探す際、(*) の 3 つ目の条件 $\deg(\bar{\Omega}) < \deg(\bar{\Lambda})$ を満たすものを探している。
- $(\bar{\Omega},\bar{\Lambda})$ を $\bar{\Lambda}$ の定数項 r で割ることで (Ω,Λ) を求めているが、誤りが t 個以上ある場合では r が 0 になることがあるのでその場合は $r=1$ としている。

例 4.1. 2 色で 8 セルが 1 つのデータを表す 1-L 型の QR コードを考える。$n=26, k=19$ であるので、$(25,19,7)$ 短縮リード・ソロモン符号と各 e_j が 10 %の確率で生じる誤りをランダムに生成し、復号する作業を 1 万回行うと次のようになる。

```
> int n = 26;
> int k = 19;
> int color = 2;
```

```
> int cell = 8;
> int prob = 10;
> <"RS.txt";
        ⋮
success 7658
failure 2342
```

よってこの場合は 10000 回中、7658 回は復号に成功し 2342 回は失敗した。

5 実験

実験 1

(目的) まず色の数という条件のみを変えたら誤り訂正能力や計算時間がどのように変化するのかを調べる。

(実験内容) 2 色の場合と 3 色の場合で入力する値を color と prob 以外は統一して前述のプログラムを実行した。その他の値は 1-L 型、2-L 型の値を入力している。復号成功回数をまとめた結果が以下のようになっている。

(実験結果) 復号成功回数をまとめた結果が以下のようになっている。

n	k	color	cell	prob	1 回目	2 回目	3 回目
26	19	2	8	1	9998	10000	9997
26	19	3	8	1	9998	9999	9999
26	19	2	8	3	9940	9938	9950
26	19	3	8	3	9943	9937	9932
26	19	2	8	5	9662	9681	9672
26	19	3	8	5	9611	9650	9687
26	19	2	8	10	7655	7637	7613
26	19	3	8	10	7623	7591	7715

n	k	color	cell	prob	1 回目	2 回目	3 回目
44	34	2	8	1	10000	10000	10000
44	34	3	8	1	10000	10000	10000
44	34	2	8	3	9978	9981	9979
44	34	3	8	3	9981	9979	9979
44	34	2	8	5	9780	9776	9770
44	34	3	8	5	9790	9784	9780
44	34	2	8	10	7203	7263	7210
44	34	3	8	10	7181	7212	7277

一方で $n = 44$、$k = 34$、prob $= 5$ のときの実験にかかった時間は以下のとおりである。

color	2	3
時間 (s)	69	1323

実験 1 の結果の考察

この結果から色を増やすだけでは誤り訂正能力に変化はないことがわかる。これはなぜかというと、復号可能かどうかは誤りの数が $t = (d-1)/2 = (n-k)/2$ 以下であるかどうかにのみ依存するからである。つまり誤りが生起する確率 p に対して復号が失敗する確率 P_{fail} は

$$\begin{aligned} P_{\text{fail}} &= 1 - \sum_{j=0}^{t} \binom{n}{j} p^j (1-p)^{n-j} \\ &= \sum_{j=t+1}^{n} \binom{n}{j} p^j (1-p)^{n-j} \end{aligned}$$

である。これを計算するために以下のプログラムで計算を行う。

Pf.txt

```
LIB "general.lib";
if((n-k)%2 == 1){
n = n-1;
}
ring A = 0,x,dp;
int t = (n-k) div 2;
number pp = p;
number q = pp/100;
int j = t+1;
number Pf = 0;
while(j<=n){
Pf = binomial(n,j)*(q^j)*(1-q)^(n-j)+Pf;
```

```
j++;
}
printf("Pf=%p",Pf);
```

$n = 44$、$k = 34$、prob $= 10$ のときを計算すると以下のようになる。

```
> int n = 44;
> int k = 34;
> int p = 10;
> <"Pf.txt";
// ** loaded /usr/local/bin/../share/singular/LIB/general.lib (4.1.2.0,Feb_2019)
// ** loaded /usr/local/bin/../share/singular/LIB/ring.lib (4.3.1.3,Feb_2023)
// ** loaded /usr/local/bin/../share/singular/LIB/primdec.lib (4.3.2.3,Jun_2023)
// ** loaded /usr/local/bin/../share/singular/LIB/absfact.lib (4.1.2.0,Feb_2019)
// ** loaded /usr/local/bin/../share/singular/LIB/triang.lib (4.1.2.0,Feb_2019)
// ** loaded /usr/local/bin/../share/singular/LIB/random.lib (4.1.2.0,Feb_2019)
// ** loaded /usr/local/bin/../share/singular/LIB/elim.lib (4.3.2.5,Jul_2023)
// ** loaded /usr/local/bin/../share/singular/LIB/matrix.lib (4.3.1.3,Feb_2023)
// ** loaded /usr/local/bin/../share/singular/LIB/nctools.lib (4.1.2.0,Feb_2019)
// ** loaded /usr/local/bin/../share/singular/LIB/polylib.lib (4.2.0.0,Dec_2020)
// ** loaded /usr/local/bin/../share/singular/LIB/inout.lib (4.1.2.0,Feb_2019)
Pf=13721981563778887448540145433091091238618739
/500000000000000000000000000000000000000000000000
```

結果で出てきた分数は約 0.27444 である。これより $n = 44$、$k = 34$、prob $= 10$ のときは $t = 5$ なので

$$\begin{aligned} P_{\text{fail}} &= \sum_{j=6}^{44} \binom{44}{j} p^j (1-p)^{44-j} \\ &\fallingdotseq 0.27444 \end{aligned}$$

となるので成功確率はおよそ $1 - 0.27444 = 0.72556$ となる。これは実験結果の 2 色、3 色いずれのときとも一致している。

また色の数を増やすだけでは誤り訂正能力に変化はなかったが、計算時間には大きな差が生じた。$n = 44$、$k = 34$、prob $= 10$ で 10000 回繰り返すのにかかった時間は、2 色の場合は 1 分 10 秒ほどだったが 3 色の場合は 22 分以上かかっている。これは有限体の元の個数が 2 色のときは $2^8 = 256$、3 色のときは $3^8 = 6561$ であるため、プログラム中の体の元全体に対して行う while 文の部分が約 26 倍の時間がかかってしまうことに起因すると考えられる。

実験 2

2 色のとき 8 セルで 1 つの c_j を表していたが、これはつまり c_j は $0, 1, \alpha, \ldots, \alpha^{254}$ の 256 通りのデータになる。よって 3 色のときは $3^{\mathrm{cell}} \geq 256$ となれば cell の大きさとしては十分である。よって cell $= 6$ とすればよい。同様に 5 色なら cell $= 4$、7 色なら cell $= 3$ として考える。まとめると

色の数	2	3	5	7
セル数	8	6	4	3
体の元の個数	256	729	625	343

なお singular と RS.txt のプログラムの都合上、色の数は素数でないと計算できないため 4 色や 6 色の場合は考えない。

ここで 1-L 型の QR コードではデータを入れることのできるセル数は 208、そのうち情報部分を入れることのできるセル数は 152 である。なので 2 色の場合では $n = 208/8 = 26$、$k = 152/8 = 19$ となっていた。同様にして 3 色のときは $n = 208/6 = 34$、$k = 152/6 = 25$ とすればよい。

たとえば 1 型の QR コードでだと色の数と誤り訂正レベルに対する、送ることのできる k の最大値は下の表のようになる。

色の数	1-L	1-M	1-Q	1-H
2	19	16	13	9
3	25	21	17	12
5	38	32	26	18
7	50	42	34	24

よって n に対する k の割合を一定にしたときの同じ大きさの QR コードで送ることのできる情報量が色を増やすほど多くなる。なので今後は色の数によってセル数を適当に選んで実験を行う。

(目的) 誤り訂正レベルを固定して同じだけの情報を送る状況を考える。色の数を変えることで実験に使う型番も変わるのでその結果誤り訂正能力と計算時間がどのように変わるのかを確かめる。

(実験内容) 以下の 2 つのパターンで実験した。

1. 誤り訂正レベルを M(15 %訂正可能) とする。このとき色の数ごとの型番と送ることのできる k の最大値は下の表のようになる。

色の数	1 型	2 型	3 型	4 型	5 型
2	16	28	44	64	86
3	21	37	58	85	114
5	32	56	88	128	172
7	42	74	117	170	229

70 個のデータを送る場合を考える。上の表より 2 色のときは 5 型、3 色のときは 4 型、5 色のときは 3 型、7 色のときは 2 型を使うことになる。プログラムで実際に実験した結果が以下の表である。

n	k	color	cell	$\text{color}^{\text{cell}}$	prob = 1	3	5	10	時間 [s]	体の元の個数
134	86	2	8	256	10000	10000	10000	9982	144	256
134	85	3	6	729	10000	10000	10000	9987	331	729
141	88	5	4	625	10000	10000	10000	9994	322	625
119	74	7	3	343	10000	10000	10000	9981	161	343

なお、時間は 10000 回繰り返すのにかかった時間の平均である。

2. さらにもう一つ実験を行った。誤り訂正レベルを H(30 %訂正可能) とする。このとき色の数ごとの型番と送ることのできる k の最大値は下の表のようになる。

色の数	1 型	2 型	3 型	4 型	5 型
2	9	16	26	36	46
3	12	21	34	48	61
5	18	32	52	72	92
7	24	42	69	96	122

40 個のデータを送る場合を考える。上の表より 2 色のときは 5 型、3 色のときは 4 型、5 色のときは 3 型、7 色のときは 2 型を使うことになる。プログラムで実際に実験した結果が以下の表である。

n	k	color	cell	$\text{color}^{\text{cell}}$	prob = 3	5	10	20	時間 [s]	体の元の個数
134	46	2	8	256	10000	10000	10000	9998	158	256
134	48	3	6	729	10000	10000	10000	9995	342	729
141	52	5	4	625	10000	10000	10000	9996	309	625
119	42	7	3	343	10000	10000	10000	9992	164	343

実験 2 の結果の考察

以上 2 つの結果より色の数を変化させても、n と k が近い値で prob が等しいときほとんど同じだけの誤り訂正能力があることがわかる。ただし計算時間には差が出ており、特に体の元の個数に起因していると考えられる。

よって、同じ状況下で色を増やしても prob が同じなのであれば読み取りにかかる時間は増えてしまうが、誤り訂正能力を変えずに小さい QR コードを使って同じだけの情報を送ることが可能であるということである。実際にどのくらい小さくすることができるのかを考えていく。以下は [4]、[6] の表をもとにして、色の数ごとに k の値をまとめたものである。

2 色	1	2	3	4	5	6	7	8	9	10	11	12	13	14	···
L	19	34	55	80	108	136	156	194	232	274	324	370	428	461	···
M	16	28	44	64	86	108	124	154	182	216	254	290	334	365	···
Q	13	22	34	48	62	76	88	110	132	154	180	206	244	261	···
H	9	16	26	36	46	60	66	86	100	122	140	158	180	197	···

3 色	1	2	3	4	5	6	7	8	9	10	11	12	13	14	···
L	25	45	73	106	144	181	208	258	309	365	432	493	570	614	···
M	21	37	58	85	114	144	165	205	242	288	338	386	445	486	···
Q	17	29	45	64	82	101	117	146	176	205	240	274	325	348	···
H	12	21	34	48	61	80	88	114	133	162	186	210	240	262	···

5 色	1	2	3	4	5	6	7	8	9	10	11	12	13	14	···
L	38	68	110	160	216	272	312	388	464	548	648	740	856	922	···
M	32	56	88	128	172	216	248	308	364	432	508	580	668	730	···
Q	26	44	68	96	124	152	176	220	264	308	360	412	488	522	···
H	18	32	52	72	92	120	132	172	200	244	280	316	360	394	···

7 色	1	2	3	4	5	6	7	8	9	10	11	12	13	14	···
L	50	90	146	213	288	362	416	517	618	730	864	986	1141	1229	···
M	42	74	117	170	229	288	330	410	485	576	677	773	890	973	···
Q	34	58	90	128	165	202	234	293	352	410	480	549	650	696	···
H	24	42	69	96	122	160	176	229	266	325	373	421	480	525	···

たとえば 2 色で 14-M 型で送られている情報は $k \leq 365$ である。この情報は 3 色なら 12-M 型で送ることができる。また 5 色なら 10-M 型、もしくは 12-Q 型で送ることができる。

型を 2 つ以上下げることを考える。表より、

・2 色で L 型のもの
- 13 型以上であれば 3 色にすることで 2 つ小さい L 型にできる。
- 5 型以上であれば 5 色にすることで 2 つ小さい L 型にできる。
- 7 型以上であれば 5 色にすることで 2 つ小さい M 型にできる。

– 5 型以上であれば 7 色にすることで 2 つ小さい M 型にできる。
– 7 型以上であれば 7 色にすることで 2 つ小さい Q 型にできる。

・2 色で M 型のもの
– 13 型以上であれば 3 色にすることで 2 つ小さい M 型にできる。
– 5 型以上であれば 5 色にすることで 2 つ小さい M 型にできる。
– 10 型以上であれば 5 色にすることで 2 つ小さい Q 型にできる。
– 5 型以上であれば 7 色にすることで 2 つ小さい Q 型にできる。
– 10 型以上であれば 7 色にすることで 2 つ小さい H 型にできる。

・2 色で Q 型のもの
– 14 型以上であれば 3 色にすることで 2 つ小さい Q 型にできる。
– 5 型以上であれば 5 色にすることで 2 つ小さい Q 型にできる。
– 7 型以上であれば 5 色にすることで 2 つ小さい H 型にできる。
– 5 型以上であれば 7 色にすることで 2 つ小さい H 型にできる。

・2 色で H 型のもの
– 12 型以上であれば 3 色にすることで 2 つ小さい H 型にできる。
– 5 型以上であれば 5 色にすることで 2 つ小さい H 型にできる。

ということがわかる。

実験 3

実験 2 の結果だけで、色を増やせば誤り訂正能力を下げることなくより小さい型で情報を送ることができると結論づけることはできない。なぜならば色を増やすことによって同一の条件下で QR コードを読み取ったとしても必ず prob に差が生じるからである。具体的には 1 つのセルが誤って伝えられる確率を P_{cell} とし、1 つのデータが誤って伝えられる確率を P_{data} とすると、色を増やすと P_{cell} は大きくなり、逆に 1 つのデータを表すセル数を減らしているので P_{cell} に対する P_{data} は小さくなると考えられる。ではどういう条件の下では色を増やすことで誤り訂正能力を下げることなくより小さい型で情報を送ることができるのかを考える。

ここで $p \ll t/n = (n-k)/2n$ であり $t/n =: T$ は一定だと仮定する。また、誤りの個数を s とする。1 つのデータの誤り確率が p であるとき n を十分大きいとみなすと全体に対する誤りの割合 s/n の分布は期待値 p、分散 $\frac{p(1-p)}{n}$ の正規分布と近似できるので、標準化すると

$$Z \coloneqq \frac{\frac{s}{n} - p}{\sqrt{\frac{p(1-p)}{n}}} \sim N(0,1) \ \text{(正規標準分布)}$$

である。よって、

$$\begin{aligned}(T =)t/n < s/n &\Leftrightarrow \frac{T-p}{\sqrt{\frac{p(1-p)}{n}}} < Z \\ &\Leftrightarrow \frac{\sqrt{n}(T-p)}{\sqrt{p(1-p)}} < Z\end{aligned}$$

となる。よって n が大きいほど $s/n > t/n (\gg p)$ である確率、つまり $s > t$ である確率は小さくなる。

ここで t/n が一定であることと k/n が一定であることは同値である。また、型が異なっていても誤り訂正レベルが同じであれば k/n がほとんど同じである。つまり誤り訂正レベルが同じであれば型が大きいものの方が復号を成功する確率は高くなる。また同様に同じ型であっても色が多いものの方が復号を成功する確率は高くなる。

ここで次の実験を行う。

(目的) 各誤り訂正レベルが何%の prob まではほぼ確実に復号可能なのかを確かめる。

(実験内容) 色の数を 2 色に固定して 3 型、5 型、7 型で、各誤り訂正レベルの 10000 回中 10000 回復号に成功した最大の prob をまとめる。

(実験結果) 結果は次のようになった。

誤り訂正レベル	L	M	Q	H
最大の prob(3 型)	2	4	10	14
最大の prob(5 型)	3	8	14	18
最大の prob(7 型)	3	9	16	22

実験 3 の結果の考察

ここで 3 色のときの 1 つのデータが誤って伝えられる確率を P_3 とする。同様にして P_5、P_7 も定義する。この結果より例えば次のことがわかる。

例 5.1. 2 色で 14-M 型の QR コードを 3 色にして 2 段階小さな QR コードにできる条件を考える。

3 色のときは、k の値のみから考えると 13 型以上なので実験 2 の考察より 12-M 型にできる。またその結果誤り訂正能力が下がらないかについても考えなくてはならない。2 色、7-M 型のときの 10000 回中 10000 回復号に成功する最大の prob を $Q_{2,7,M}$ と表すことにすると、

$$9 = Q_{2,7,M} \leq Q_{2,12,M} \leq Q_{3,12,M}$$

である。よって 3 色で 12-M 型の QR コードは少なくとも $P_3 \leq 9$ であれば必ず復号ができる。

よって、$P_3 \leq 9$ である環境で使われている 2 色の 14-M 型の QR コードは 3 色にすることで 12-M 型にすることができ、誤り訂正能力も保証される。

例と同様にして他の場合も考えると、色を増やすことで QR コードの型を 2 つ小さくできる条件は以下のとおりである。

3 色にするとき				
2 色のときの誤り訂正レベル	L	M	Q	H
2 色のときの型番	13 以上	13 以上	14 以上	12 以上
P_3 の条件	$P_3 \leq 3$	$P_3 \leq 9$	$P_3 \leq 16$	$P_3 \leq 22$
3 色のときの誤り訂正レベル	L	M	Q	H

5 色にするとき							
2 色のときの誤り訂正レベル	L		M		Q		H
2 色のときの型番	5 以上	7 以上	5 以上	10 以上	5 以上	7 以上	5 以上
P_5 の条件	$P_5 \leq 2$	$P_5 \leq 8$	$P_5 \leq 4$	$P_5 \leq 16$	$P_5 \leq 10$	$P_5 \leq 18$	$P_5 \leq 14$
5 色のときの誤り訂正レベル	L	M	M	Q	Q	H	H

7 色にするとき					
2 色のときの誤り訂正レベル	L		M		Q
2 色のときの型番	5 以上	7 以上	5 以上	10 以上	5 以上
P_7 の条件	$P_7 \leq 4$	$P_7 \leq 14$	$P_7 \leq 10$	$P_7 \leq 22$	$P_7 \leq 14$
7 色のときの誤り訂正レベル	M	Q	Q	H	H

この結果よりたとえば現状 10-M 型で送られている情報は、その状況下で $P_5 \leq 16$ であれば 5 色の 8-Q 型にできるということがわかる。このように実験の結果によって色を増やすことで誤り訂正能力を下げることなく小さな QR コードにすることができる条件を導くことができた。

まとめ

- P_3 や P_5、P_7 の条件次第では QR コードを 3、5、7 色にすることで、誤り訂正能力を保証したまま小さな QR コードにすることができる。
- 一方で読み取りの時間は体の大きさに依存しており、3 色だと 2.3 倍、5 色だと 2.2 倍、7 色だと 1.2 倍ほどの時間がかかる。

6 今後の展望

この研究では P_3 や P_5 の値によっては 3 色や 5 色にすることで 2 段階以上小さい QR コードにすることができる、という結論に達することができた。しかし、実際どのような条件で P_3 や P_5 がどのような値を取るかに関しては全くわかっていない。その値は光の当たり方によって変化することが予測される。たとえば 3 色目として黄色を選んだとしたら、明るい環境では少なくとも肉眼では白と見分けにくいことがある。なので 3 色や 5 色にしたときに読み取る機械を実際に作り、様々な環境で P_3 や P_5 の値を実際に計測することで、ど

ういった環境では 3 色や 5 色にすることで QR コードを小さくすることができるのかという結論が得られるであろう。

また、QR コードに限らずリード・ソロモン符号を使用している他の通信路でも同様に P_3 などを定義すれば、それぞれの規格には依存するが同じような結果が得られるはずである。

謝辞

本研究において、毎週のセミナーで質問や実験の進め方について丁寧にアドバイスをくださった椙先生に感謝いたします。また、研究室の同期の福江氏や本研究に様々な意見をくださった皆様に感謝いたします。

参考文献

[1] D. コックス,J. リトル,D. オシー著: グレブナー基底 2. 大杉英史, 北村知徳, 日比孝之訳. 丸善出版.2012.

[2] 池田和興. 例題が語る符号理論. 共立出版.2007.

[3] イエルン・ユステセン, トム・ホーホルト著: 誤り訂正符号入門 [第 2 版]. 阪田省二郎, 栗原正純, 松井一, 藤沢匡哉訳. 森北出版.2019.

[4] tech-jp,QR コードの仕組み.https://www.tech-jp.com/QRCode/index.html (閲覧日: 2024 年 1 月 13 日)

[5] KEYENCE, バーコード講座.https://www.keyence.co.jp/ss/products/autoid/codereader/basic2d-qr-types.jsp (閲覧日: 2024 年 1 月 13 日)

[6] QRcode.com,https://www.qrcode.com/en/index.html (閲覧日: 2024 年 1 月 13 日)

[7] デンソー技術情報.https://www.denso-wave.com/ja/adcd/fundamental/2dcode/qrc/index.html (閲覧日: 2024 年 1 月 13 日)

[8] 濱屋進. 符号理論入門: 数学的な基礎知識から「QR コード」の作成まで. 工学社.2008.

[9] QR のススメ,QR コード作成サイト.https://qr.quel.jp/ (閲覧日: 2024 年 1 月 10 日)

[10] いろあと, https://www.iroato.com/cameleoncode/ (閲覧日: 2024 年 1 月 23 日)

[11] JIS-X-0510, https://kikakurui.com/x0/X0510-2018-01.html (閲覧日: 2024 年 1 月 25 日)

[12] 助川 修司, 伊藤 正都, 近藤 圭佑, 大囿 忠親, 新谷 虎松, QR コードの多色化による 2 次元コードの大容量化について, 全国大会講演論文集, 第 70 回 (コンピュータと人間社会)(2008),845-846