符号理論における代数的手法

Algebraic Methods in Coding Theory

松井 一 Hajime MATSUI

アブストラクト 本稿の目的は、代数的な誤り訂正符号とその手法を、分かりやすく解説することにある。仮定する予備知識としては、一部を除き主に高校数学のみとし、数値例に沿って解説を行う。代数的符号理論の習得のためには、数値例を理解し実行できるようになることが、一番の早道だからである。具体的には、ある種の漸化式と考えられる線形帰還シフトレジスタ (LFSR) の理解を出発点として解説を進めていく。扱うトピックは、リードソロモン符号、離散フーリエ変換、そしてグレブナー基底の初歩などである。

キーワード 誤り訂正符号、有限体、線形帰還シフトレジスタ、ユークリッドの互除法、リードソロモン符号、準巡回符号、アフィン多様体符号、Berlekamp-Massey-Sakata アルゴリズム

Abstract The objective of the study is to easily and briefly describe algebraic error-correcting codes and their techniques. The necessary knowledge to read the paper is assumed mainly within elementary mathematics except for a part and we discuss them along with various examples. This is because the easiest method of understanding the algebraic coding theory is considered to be the calculation of their numerical examples. We start our description of the theory of linear feedback shift registers (LFSR) which are regarded to be a class of recurrence relations. Then, we treat Reed-Solomon codes, discrete Fourier transforms, and the basics of Gröbner bases in this paper.

Key words Error-correcting codes, Finite fields, Linear feedback shift registers, Euclidean algorithm, Reed-Solomon codes, Quasi-cyclic codes, Affine variety codes, Berlekamp-Massey-Sakata algorithm

1. はじめに

本稿は、2013 年 11 月 26 日、伊東市で開催された情報理論研究会 (若手研究者のための講演会) における「線形帰還シフトレジスタ理論とその発展」⁽¹⁾に加筆・修正を行ったものである。

本稿の目的は、代数的な誤り訂正符号とその手法を、分かりやすく解説することである。代数的とは、通常は線形空間や群・環・体等の代数学を用いた方法のことを指す。これらはエレガントで強力な道具であるが、抽象的な感じを受けることもある。また最近では、シェアウェア・フリーウェアを問わずこれらの代数的手法を PC 上で実行するソフトウェアが充実しており、計算を直ちに実行できるようになってきた。しかしながら、これらの数式処理ソフトウェアを導入するのはそれほど容易とは限らず、またある程度の慣れの期間も必要である。

そこで本稿では、代数的な内容を解説しつつも、極力、初等的で平易に述べることに主眼を置いた。仮定する予備知識としては、一部を除き主に高校数学のみとした。もちろん誤り訂正

符号の性質上、線形符号の基礎を知っていればなお良いが、それでも特別な知識がなくても大部分は分かるように書いたつもりである。またほかに本稿を書く上で意識した点として、数値例に沿って解説することを心掛けた。代数的符号理論の習得のためには、まず簡単な数値例を理解し実行できるようになることが、一番の早道であると考えているからである。本稿は紙数の都合、また筆者の力量の限界もあり、代数的手法の一部分、また符号理論の一側面しか伝えることができない。しかしながら、こういった易しい解説文はこれまで少なかったと思われる。読者がこの分野を概観する一助になれば幸いである。

本稿の構成は以下のとおりである。2. は線形帰還シフトレジスタ (LFSR) の解説であり、3. で誤り訂正符号への応用、3.1 で有限体の解説、3.2 でリードソロモン符号の符号化、3.3 で復号の準備、3.4 で拡張ユークリッド法について、3.5 でユークリッド復号法について述べる。4. では LFSR と離散フーリエ変換との関係について解説し、4.1 で誤り訂正符号における復号について、4.2 で符号化について述べる。5. では LFSR の二次元への拡張とアフィン多様体符号への応用について述べる。6. では準巡回符号の基礎とその符号化について述べる。

2. LFSR とは？

LFSR の易しい解説を行う。LFSR とは、高校数学の用語で

松井 一 正員 豊田工業大学工学部
E-mail matsui@toyota-ti.ac.jp
Hajime MATSUI, Member (Undergraduate School of Engineering,
Toyota Technological Institute, Nagoya-shi 468-8511, Japan).
電子情報通信学会 基礎・境界サイエティ
Fundamentals Review Vol.8 No.3 pp.151-161 2015 年 1 月
©電子情報通信学会 2015

図1 段数 $L = 4$ の線形帰還シフトレジスタ (LFSR) .

言えば, ある種の漸化式である. 以下において, 四則演算のできる数体系を一つ固定する. これは実数でもよいし, また後の章の符号理論への応用の際は有限体とする. 本章では, L を正の整数とする.

[定義1] 線形帰還シフトレジスタ (LFSR: Linear Feedback Shift Register) とは, L 個の数の組 $(c_1, \dots, c_L)$ があって, L 個の項から成る任意の有限数列 $s_0, s_1, \dots, s_{L-1}$ から

$$s_j = - \sum_{i=1}^L c_i s_{j-i} \quad (j = L, L+1, L+2, \dots) \quad (1)$$

によって数列を定めることができるものである. このとき, $(c_1, \dots, c_L)$ をこの LFSR の係数, L を段数と呼ぶ. □

この定義, 特に式 (1) を装置化したものとしての LFSR を図1に示す. これは文献(2)にある図を $L = 4$ として描いたものである. 図1において, 長方形の素子はある単位時間の記憶, だ円形の素子は定数 $-c_i$ ($1 \leq i \leq L$) 倍の乗法, また $\oplus$ は加法素子をそれぞれ表す. このように LFSR とは, 係数 $(c_1, \dots, c_L)$ から定まり, 任意に与えられた初期値からある種の数列を生成するという仕組みを持つものである. 係数 $(c_1, \dots, c_L)$ のうちには, 0 のものがあるてもよい. LFSR のうちの '線形 (Linear)' とは, 式 (1) が定数項や二乗以上の項などを含まず, 一次式だけから成ることを意味する.

[例1] 数体系を実数または複素数とする. 漸化式で最も簡単なものは, 等差数列と等比数列だろう. 等差数列は $s_j = s_{j-1} + c$ (c は定数) となり, $c \neq 0$ ならば線形ではないことになる. 等比数列は $s_j = r s_{j-1}$ (r は定数) となり, これは段数 $L = 1$ の最も簡単な LFSR である. 恐らく最も有名な LFSR は, $L = 2$ として $c_1 = c_2 = -1$ とした漸化式 $s_j = s_{j-1} + s_{j-2}$ だろう. 初期値を $s_0 = 0, s_1 = 1$ としたときの数列に現れる数はフィボナッチ数と呼ばれ⁽³⁾, 数列は $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, \dots$ となる. また, 初期値を $s_0 = 2, s_1 = 1$ としたときの数列に現れる数はリュカ数と呼ばれ⁽³⁾, 数列は $2, 1, 3, 4, 7, 11, 18, 29, 47, \dots$ となる. こういった隣接3項間の漸化式による整数数列は, 素因数分解法の一つである $p+1$ 法において用いられる⁽⁴⁾. □

m を正の整数とするとき, m 個の項から成る有限数列 $s_0, s_1, \dots, s_{m-1}$ を, $\underline{s}$ と表す.

[定義2] 係数 $(c_1, \dots, c_L)$ によって定まる LFSR, 及び m 個の項 ($m = +\infty$ でもよい) から成るある数列 $\underline{s}$ について, この LFSR が $\underline{s}$ を生成するとは (i) $m \leq L$ のときは常にそうであると定め (ii) $m > L$ のときは, $(m-L)$ 本の式

$$s_j + \sum_{i=1}^L c_i s_{j-i} = 0 \quad (j = L, L+1, \dots, m-1) \quad (2)$$

が成立するときであると定める. □

さて, この定義のとおり, 一つ LFSR を定め, 更に又数列の初期値を与えれば, 数列全体が定まることになる. ここで素朴な疑問として, 次の二つがすぐに思いつく.

一つは, 数列全体が定まると言っても, 大きな j まで式 (1) を用いて s_j を求めていたのでは大変である. そこで, 数列の一般項, つまり j を与えたとき s_j を直接に計算する式は得られるであろうか? これについては, 特別なケースは 4.1 で取り扱う. 一般的なケースについては, 文献(1)にあるいは文献(3)の 7.3 を参照とする.

もう一つの素朴な疑問は, 逆問題と言うべきものである. すなわち, 与えられた有限数列を生成する LFSR を求めることができるだろうか? この LFSR を求めるアルゴリズムが Berlekamp-Massey アルゴリズム (BM 法)⁽²⁾⁽⁵⁾であり, 文献(1)で解説したので本稿では述べない. また, ユークリッド復号法を用いてもこの LFSR を求めることができ, 3. で解説する.

式 (2) は, $c_0 = 1$ とすれば $\sum_{i=0}^L c_i s_{j-i} = 0$ となる. これの左辺は数列の畳込みと呼ばれ, 以下に見るように数列の母関数の積に現れる. 式 (2) が $m = +\infty$ で成立するとき, 数列の母関数 $C(D)$ と $S(D)$ を,

$$C(D) = \sum_{i=0}^L c_i D^i, \quad S(D) = \sum_{l=0}^{+\infty} s_l D^l$$

と定める. このとき, 積 $C(D)S(D)$ について,

$$C(D)S(D) = \sum_{j=0}^{+\infty} \left(\sum_{i=0}^{\min\{j, L\}} c_i s_{j-i} \right) D^j$$

となるが, $j \geq L$ ならば式 (2) から内側の和は 0 だから, 右辺は次数 $(L-1)$ 以下の多項式となり, $P(D)$ と置く. よって $C(D)S(D) = P(D)$ となる. 言い換えると,

$$\frac{P(D)}{C(D)} = S(D)$$

となり, 数列 $\underline{s}$ の母関数 $S(D)$ は有理式による表示を持つことが分かった. この論法は後の 3.5 で用いられる.

3. 誤り訂正符号との関係: RS 符号

本章では, リードソロモン (RS: Reed-Solomon) 符号の復号をユークリッド復号法⁽⁶⁾を用いて実演する. RS 符号の位置付け・重要性については文献(7)を, ユークリッド復号法の発見の様子については文献(8)を, アルゴリズムの詳細については文献(9)(10)を参照とする. また最近の和書では文献(11)(12)がある.

3.1 有限体の準備

有限体についての詳細は文献(13)~(15)参照とし, 概略のみを述べる. 有限体の一つである 11 元体 $\mathbb{F}_{11}$ とは, 次の 11 個の元から成る集合である.

これを誤り訂正符号で用いる理由は，四則演算（ $+$ ， $-$ ， $\times$ ， $\div$ ）ができるからである．以下において演算を定義していく．まず $+$ ， $-$ ， $\times$ について，これらは通常と同様であるが，演算の結果が定義式 (3) の $0 \sim 10$ の範囲外，つまり 0 より小や 10 より大となった場合は 11 で割った余りを取ると定める．例えば

である．最後に， $\div$ （除算）は $\times$ の逆演算であり，分数でもそうであったように逆数を掛けることによって求める．ここで逆数とは $1 \div x$ の答えであり， x^{-1} と書く．つまり $x \times x^{-1} = 1$ を満たす x^{-1} であり，この式は通常の整数の演算の形で書けば $xx^{-1} + 11y = 1$ であり，11 は素数で $x = 1 \sim 10$ のいずれとも互いに素だから，試行錯誤（あるいは後で解説する拡張ユークリッド法）により， x^{-1} を求めることができる．表にすると次のようになる．

この表が正しいことは簡単にチェックできる（例えば $2 \times 6 = 1$ 等々）．表を用いて $7 \div 8$ をやってみると， $8^{-1} = 7$ だから， $7 \div 8 = 7 \times 7 = 5$ と分かる．こうして有限体 $\mathbb{F}_{11}$ の演算が全て定まった．これらの演算を大量に行う場合，実際には数式処理ソフトウェア等を用いることになるが，本稿で扱う程度の量であれば手計算が可能である．

ただし，表で“ mod 11 ”と書いてあるのは，演算の定義でもやった 11 で割った余りを取ることを示す．このように確かに $\mathbb{F}_{11}$ の 0 以外の全ての元が出てきている．ちなみに 2^{10} を計算してみると， $6 \times 2 = 1 = 2^0$ と先頭に戻る．このように 2 のべきによる表示は巡回構造を持つ．この事実は離散フーリエ変換との関係において後で用いられる．

以上で有限体の準備が終わり，11 元 (10,6) RS 符号を構成する．この ‘11 元 (10,6)’ の意味は，有限体 $\mathbb{F}_{11}$ の要素について，10 個を用いて 6 個を送信できる，ということである．余分な 4 個の要素を付け加えることにより，2 個までの誤り訂正ができるようになる．今送りたい情報が

という 6 個の 0 ~ 10 までの数字の並びであるとする．まず，こ

図 2 11 元体係数を持つ多項式の除法．情報多項式の生成多項式による商と余りを求める．ただし途中の計算を省略している．

を作る．送りたい情報から作った多項式なので，これを情報多項式という．次に，2 の 0 乗から 3 乗までの 4 個の根を持つ多項式

の展開をあらかじめ計算しておく． $G(x)$ を生成多項式という．このとき，情報多項式を生成多項式で割った余りを求め，その余りを情報多項式から引く手続きを符号化という．図 2 のとおり，余り $[9\ 2\ 0\ 7]$ が求まるから，この余りを情報多項式から引くと次のとおりとなる．

このように，生成多項式で割り切れるようにしたもの（同じことだが $2^0, 2^1, 2^2, 2^3$ を根に持つようにしたもの）は符号語と呼ばれる．この符号語を通信路へ送信する．

上の生成多項式は $2^0 \sim 2^3$ の 4 個の根を持つとしたが、これは次で実演するように、根の個数の半分 $4/2 = 2$ 個までの誤りを訂正できるようにするためである．生成多項式の根を $2^0 \sim 2^5$ の 6 個とすると今度は 3 個までの誤りが訂正できる．ただしその分、送信できる情報が 4 個に減ってしまう．

このように RS 符号では，生成多項式の根の個数の半数以下の誤りを全て訂正することができる（線形代数で学ぶ Vandermonde 行列式か，あるいは離散フーリエ変換を用いて示される．文献（9）（10）参照．）

さてこの RS 符号は通信路で 2 個までの誤りが起こっても訂正することができる．送信された符号語は，通信路において誤りが加算されて受信される．つまり

誤り: $[e_0 \ e_1 \ e_2 \ e_3 \ e_4 \ e_5 \ e_6 \ e_7 \ e_8 \ e_9]$

受信語: $[J_0 \ J_1 \ J_2 \ J_3 \ J_4 \ J_5 \ J_6 \ J_7 \ J_8 \ J_9]$

とするとき、受信語の各成分は $J_i = c_i + e_i$ ($i = 0, 1, \dots, 9$) というように符号語と誤りの各成分の和である (より正確に言えば、通信路の誤りがこのようになっているときに訂正できる。) 以下においては 2 誤り、つまり i 個目と j 個目 ($i < j$ とする) の e_i と e_j のみが 0 でない、2 個の誤りが起こったケースについて解説する。

今通信路を通ってきた受信語が

$$[2 \ 9 \ 0 \ 2 \ 1 \ 7 \ 3 \ 2 \ 0 \ 10]$$

となったとする。もちろん送信した符号語と見比べればどの数字が間違っているか分かるが、通常は受信側ではそんなことはできないから、純粹にこの受信語だけから誤りを推定する必要がある。これを復号という。これは次のような流れから成る。

図3 復号アルゴリズムの概要。

まず「シンドローム」ブロックにおいては、受信語から得られる次の多項式

$$J(x) = 2 + 9x + 0x^2 + 2x^3 + 1x^4 + 7x^5 + 3x^6 + 2x^7 + 0x^8 + 10x^9$$

に生成多項式の根 $2^0 \sim 2^3$ を代入した値 (シンドロームと呼ばれる) を求める。以下の式で * と ' はそれぞれ行列の積と転置を表す。

$$\begin{aligned} J(1) &= [2 \ 9 \ 0 \ 2 \ 1 \ 7 \ 3 \ 2 \ 0 \ 10] * [1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1]' = 3 \\ J(2) &= [2 \ 9 \ 0 \ 2 \ 1 \ 7 \ 3 \ 2 \ 0 \ 10] * [1 \ 2 \ 4 \ 8 \ 5 \ 10 \ 9 \ 7 \ 3 \ 6]' = 3 \\ J(4) &= [2 \ 9 \ 0 \ 2 \ 1 \ 7 \ 3 \ 2 \ 0 \ 10] * [1 \ 4 \ 5 \ 9 \ 3 \ 1 \ 4 \ 5 \ 9 \ 3]' = 8 \\ J(8) &= [2 \ 9 \ 0 \ 2 \ 1 \ 7 \ 3 \ 2 \ 0 \ 10] * [1 \ 8 \ 9 \ 6 \ 4 \ 10 \ 3 \ 2 \ 5 \ 7]' = 1 \end{aligned} \quad (5)$$

これらの値は一体何かといえば、まずシンドロームは実は誤りだけから計算され、符号語には関係しない。なぜならば符号語 $[c_0 \ \dots \ c_9]$ から同様に作った多項式を $c(x)$ 、誤り $[e_0 \ \dots \ e_9]$ から同様に作った多項式を $e(x)$ とすると、 $J(x) = c(x) + e(x)$ であり、しかも $c(2^l) = 0$ ($l = 0, 1, 2, 3$) だから (こうなるように符号語を作った)、 $J(2^l) = e(2^l)$ ($l = 0, 1, 2, 3$) である。また、誤りの成分は e_i と e_j 以外 0 だから、結局

$$J(2^l) = e_i 2^{il} + e_j 2^{jl} \quad (l = 0, 1, 2, 3)$$

である。つまりシンドロームとは、二つの等比数列 (初項 $e_i \cdot$ 公比 2^i 、及び初項 $e_j \cdot$ 公比 2^j) の和になっている。こうして得られた 4 個のシンドロームの値から多項式 $S(x) = 1 + 8x + 3x^2 + 3x^3$ を作っておく ($S(x)$ を $[3 \ 3 \ 8 \ 1]$ でなく引つ繰り返した $[1 \ 8 \ 3 \ 3]$ とするのは後の都合である。)

図3の残りのブロックのうち「誤り位置」が最も複雑である。ここで、最大公約数を求めるアルゴリズムである拡張ユークリッド法が応用される。

3.4 ユークリッドの互除法

ユークリッドの互除法あるいはユークリッド法とは、除法を繰り返し行い最大公約数を求めるアルゴリズムである。今二つの正の整数 a と b が $a \geq b$ であるとして除法

$$a = qb + r, \quad 0 \leq r < b \quad (6)$$

を行ったとき、 $a - qb = r$ だから、 a と b の最大公約数は b と r のそれと一致する、ということが本質的である。例えば 119 と 98 で実演してみると、

$$119 = 1 \times 98 + 21, \quad 98 = 4 \times 21 + 14, \quad 21 = 1 \times 14 + 7 \quad (7)$$

となり、 $14 = 2 \times 7$ であるから、最大公約数 7 が求まった。更に拡張ユークリッド法⁽³⁾とは、 a と b の最大公約数が d であるとき、 $am + bn = d$ を満たすような二つの整数 m と n を一組見つける方法である。これは式 (7) の計算を逆にたどって得られるが、復号にとっては式 (7) における途中の余りを表す方が重要なので、こちらを実演してみる。2 回目の余り 14 については次のようになる。

$$\begin{aligned} 14 &= 98 - 4 \times 21 = 98 - 4 \times (119 - 1 \times 98) \\ &= -4 \times 119 + 5 \times 98 \end{aligned}$$

3.5 RS 符号の復号 (II)

以上の計算を RS 符号の復号に応用する。この人類最初とも言われるアルゴリズムが、ほぼそのままの形で高速な復号を与えることは驚くべきことである。多項式に対する拡張ユークリッド法とは、二つの多項式 $a(x)$ と $b(x)$ が与えられたとき、 $a(x)\sigma(x) + b(x)\omega(x) = \phi(x)$ を満たす $\sigma(x)$ と $\omega(x)$ を求めるアルゴリズムである。RS 符号においては二つの多項式 $a(x) = x^{2t}$ 、 $b(x) = -S(x)$ に対し拡張ユークリッド法を用いることで復号を行う。このとき除法 (6) の条件 $0 \leq r < b$ は多項式の場合には $\deg r(x) < \deg b(x)$ になる (ただし $\deg$ は多項式の次数であり、例えば二次多項式なら 2。) 以下で解説するように、RS 符号の復号では、 $\deg r(x) < t$ となったときの計算結果が重要である。

図3において「誤り位置」のブロックでは、次の $\sigma(x)$ を求める。

$$\sigma(x) = (x - 2^i)(x - 2^j) \quad (8)$$

ただし i 個目と j 個目 ($0 \leq i < j \leq 9$) の数字が間違っているとしていた。この $\sigma(x)$ の根 $2^i, 2^j$ は誤り位置と呼ばれ、 $\sigma(x)$ は誤り位置多項式と呼ばれる。どこの数字が誤っているかという意味では、むしろ i と j を誤り位置と呼び計算した方がよい気もするが、そうはせずに、 2^i と 2^j を誤り位置と呼び計算するところが賢いところである。

更に図3における次のブロックである「誤り値」では、誤りの値 e_i, e_j を決定するが、それにはアルゴリズムにより $\sigma(x)$

と同時に得られる次の多項式

$$\omega(x) = \sigma(x) \left(\frac{e_i}{x-2^i} + \frac{e_j}{x-2^j} \right) = e_i(x-2^j) + e_j(x-2^i) \quad (9)$$

を用いる．これによって誤りの値は次のように表すことができる．

$$e_i = \frac{\omega(2^i)}{\sigma'(2^i)}, \quad e_j = \frac{\omega(2^j)}{\sigma'(2^j)} \quad (10)$$

ただし $\sigma'(x)$ は $\sigma(x)$ を形式的に微分したものである：

$$\sigma(x) = 2^{i+j} - (2^i + 2^j)x + x^2 \implies \sigma'(x) = -(2^i + 2^j) + 2x$$

したがって $\sigma'(2^i) = 2^i - 2^j$, $\sigma'(2^j) = 2^j - 2^i$ だから，式 (9) から誤り値の公式 (10) が直ちに得られる．さてシンドロームを一般的に $u_l = e_i 2^{il} + e_j 2^{jl}$ ($l = 0, 1, \dots$) と表す． $l = 0, 1, 2, 3$ については u_l は受信語から求めることができ， $u_l = J(2^l)$ だった．ここで $\frac{e_i}{x-2^i} = \frac{e_i/x}{1-2^i/x} = \frac{e_i}{x} + \frac{e_i 2^i}{x^2} + \dots$ (等比級数の公式) を用いると，式 (9) 及び $S(x) = u_0 x^3 + u_1 x^2 + u_2 x + u_3$ により，

$$\frac{\omega(x)}{\sigma(x)} = \frac{u_0}{x} + \frac{u_1}{x^2} + \dots = S(x)x^{-4} + u_4 x^{-5} + \dots$$

とシンドロームや $S(x)$ が出てくる．つまり $\frac{\omega(x)}{\sigma(x)}$ は数列 $\{u_l\}$ の母関数になっている．この式の両辺に $\sigma(x)x^4$ を掛けると $\omega(x)x^4 = \sigma(x)S(x) + \sigma(x)(u_4 x^{-1} + \dots)$ となるが，再び等比級数の公式から

$$\phi(x) = \sigma(x)(u_4 x^{-1} + \dots) = \sigma(x) \left(\frac{e_i 2^{4i}}{x-2^i} + \frac{e_j 2^{4j}}{x-2^j} \right)$$

となり， $\phi(x)$ は一次以下の多項式となる．こうして関係式 $\omega(x)x^4 = \sigma(x)S(x) + \phi(x)$ が得られた．以上の説明は， $\sigma(x)$ と $\omega(x)$ が，関係式 $\omega(x)x^4 = \sigma(x)S(x) + \phi(x)$ を満たすことの説明である．厳密には，拡張ユークリッド法によって求める多項式が $\sigma(x)$ と $\omega(x)$ に一致すること，つまりこの関係式を満たす $\sigma(x)$ と $\omega(x)$ の一意性を示す必要がある．証明は難しくはないが，文献 (9) (10) 参照とする．

こうして， $a(x) = x^4$, $b(x) = -S(x)$ として拡張ユークリッド法により $\omega(x)$ と $\sigma(x)$ を求めることができる．また $\phi(x)$ の次数が (誤りの個数) 2 より小さいので，除法を条件 $\deg r(x) < 2$ が満たされるまで繰り返す必要があることが分かった．

さてシンドロームから作った多項式は $S(x) = 1 + 8x + 3x^2 + 3x^3$ だった．除法を行って，

$$x^4 = (7 + 4x)S(x) + (4 + 6x + 2x^2),$$

$$S(x) = (8 + 7x)(4 + 6x + 2x^2) + (2 + 9x),$$

$\deg(2 + 9x) < 2$ となったので，除法を終了し，次が得られる．

$$\begin{aligned} 2 + 9x &= S(x) - (8 + 7x)(4 + 6x + 2x^2) \\ &= S(x) - (8 + 7x)\{x^4 - (7 + 4x)S(x)\} \\ &= (3 + 4x)x^4 + \{1 + (8 + 7x)(7 + 4x)\}S(x) \\ &= (3 + 4x)x^4 - (9 + 7x + 5x^2)S(x) \end{aligned}$$

したがって定数 $c \neq 0$ があって， $c\sigma(x) = 9 + 7x + 5x^2$, $c\omega(x) = 3 + 4x$ となることが分かった ($c = 5$ であるが誤り値の公式の分子分母でキャンセルするので放っておいてよい．以後書かないことにする．) この $\sigma(x)$ に $2^0, 2^1, \dots, 2^9 = 6$ を順次代入し， $\sigma(8) = \sigma(6) = 0$ であることが分かる．したがって誤り位置は 2^3 と $6 = 2^9$ ，つまり (左端が 0 個目として) 3 個目と 9 個目が誤っていることが分かった．また $\sigma'(x) = 7 + 10x$ であり， $e_3 = \frac{\omega(8)}{\sigma'(8)} = 9$, $e_9 = \frac{\omega(6)}{\sigma'(6)} = 5$ より誤りは

$$[0 \ 0 \ 0 \ 9 \ 0 \ 0 \ 0 \ 0 \ 0 \ 5]$$

であり，これを受信語

$$[2 \ 9 \ 0 \ 2 \ 1 \ 7 \ 3 \ 2 \ 0 \ 10]$$

から引いて正しい符号語

$$[2 \ 9 \ 0 \ 4 \ 1 \ 7 \ 3 \ 2 \ 0 \ 5]$$

が得られる．

4. 離散フーリエ変換との関係

誤り訂正符号，もっと広くは信号処理とは，DFT や IDFT による時間領域と周波数領域との間の変換の理論である，と言ったら言い過ぎであろうか．本章では 文献 (16)~(18) に従って，LFSR と DFT が ‘相性が良い’ ことを説明し，先の言明がながち言い過ぎでもないことを見る．

4.1 復号における DFT

[定義 3] 一次元離散フーリエ変換 (One Dimensional Discrete Fourier Transform, **1D DFT**) とは， $\mathbb{F}_{11}$ の成分を持つ $\underline{c} = (c_0, c_1, \dots, c_9)$ に対し，

$$h_j = \sum_{i=0}^9 c_i 2^{ij} \quad (j = 0, 1, \dots, 9) \quad (11)$$

によって $\underline{h} = (h_0, h_1, \dots, h_9)$ を定める線形写像である．このとき $\underline{h} = \mathcal{F}(\underline{c})$ と表す．次に，一次元逆離散フーリエ変換 (1D Inverse DFT, **1D IDFT**) とは， $\mathbb{F}_{11}$ 成分を持つ $\underline{h} = (h_0, h_1, \dots, h_9)$ に対し，

$$c_i = - \sum_{j=0}^9 h_j 2^{-ij} \quad (i = 0, 1, \dots, 9) \quad (12)$$

によって $\underline{c} = (c_0, c_1, \dots, c_9)$ を定める線形写像である．このとき $\underline{c} = \mathcal{F}^{-1}(\underline{h})$ と表す．□

[定理 1] $\mathcal{F}$ と $\mathcal{F}^{-1}$ は互いに逆写像である．つまり， $\mathcal{F}^{-1}(\mathcal{F}(\underline{c})) = \underline{c}$ かつ $\mathcal{F}(\mathcal{F}^{-1}(\underline{h})) = \underline{h}$ が成り立つ．□

なぜならば， $\mathcal{F}^{-1}(\mathcal{F}(\underline{c})) = \underline{c}$ については， $\mathcal{F}^{-1}(\mathcal{F}(\underline{c}))$ の l 個目の成分 ($0 \leq l \leq 9$) について次が成り立つからである．

$$- \sum_{j=0}^9 \left(\sum_{i=0}^9 c_i 2^{ij} \right) 2^{-lj} = - \sum_{i=0}^9 c_i \left(\sum_{j=0}^9 2^{(i-l)j} \right) = c_l$$

ただし $\beta \neq 1, \beta^{10} = 1 \Rightarrow \sum_{j=0}^9 \beta^j = \frac{1-\beta^{10}}{1-\beta} = 0$ とここで 3.1 の最後に書いた巡回構造を用いた $\mathcal{F}(\mathcal{F}^{-1}(\underline{h})) = \underline{h}$ も同様に示される。

RS 符号の復号と DFT はどのように結び付くだろうか？ それには、 $\underline{c}$ が多項式 $c(x) = \sum_{i=0}^{n-1} c_i x^i$ と対応しているとき、式 (11) から

$$\mathcal{F}(\underline{c}) = (c(2^0), c(2), \dots, c(2^{n-1})) \quad (13)$$

であることに注意する。このとき、誤りを $\underline{e}$ とすると、シンδροームは DFT $\mathcal{F}(\underline{e})$ の一部分であることが分かる。もしこれが一部分ではなく全体であれば、 $\mathcal{F}^{-1}$ を施すと、 $\mathcal{F}^{-1}(\mathcal{F}(\underline{e})) = \underline{e}$ により $\underline{e}$ を求めることができ、受信語 $\underline{J}$ から引くことにより復号が完了する。ではどうやって全体にするかと言えば、式 (8) の誤り位置多項式 $\sigma(x)$ から得られる LFSR によって生成することができる。以下においては、拡張ユークリッド法によって $\sigma(x) = \sigma_0 + \sigma_1 x + x^2$ が求まっているとし、LFSR による生成及び IDFT により、誤り $\underline{e}$ を求めてみよう。

[例 2] 受信語 $\underline{J} = (2, 9, 0, 2, 1, 7, 3, 2, 0, 10)$ に対し、

シンδροーム $(u_0, u_1, u_2, u_3) = (3, 3, 8, 1)$

拡張ユークリッド法により $\sigma(x) = 4 + 8x + x^2$

LFSR により $\underline{u} = (u_0, \dots, u_9) = (3, 3, 8, 1, 4, 8, 3, 3, 10, 7)$

(例えば $u_4 = -4 \cdot 8 - 8 \cdot 1 = 4$)

IDFT $\underline{e} = \mathcal{F}^{-1}(\underline{u}) = (0, 0, 0, 9, 0, 0, 0, 0, 5)$

符号語 $\underline{c} = \underline{J} - \underline{e} = (2, 9, 0, 4, 1, 7, 3, 2, 0, 5)$ □

シンδροーム $(3, 3, 8, 1)$ を LFSR により伸ばした $\underline{u} = (3, 3, 8, 1, 4, 8, 3, 3, 10, 7)$ に対し IDFT $\mathcal{F}(\underline{u})$ を行うと、ほとんどの値が消えて誤りの値 9, 5 のみが現れるところが面白い。この現象の理由は、 $\underline{u}$ が $\underline{e}$ の DFT となるからであるが、それがなぜかを以下に簡単に述べる（詳細は文献 (1) 参照）。シンδροーム $u_l = e(2^l) = e_i 2^{li} + e_j 2^{lj}$ ($l = 0, 1, 2, 3$) に対し、式 (1) のとおり、

$$u_l = -\sigma_0 u_{l-2} - \sigma_1 u_{l-1} \quad (l = 4, 5, \dots, 9) \quad (14)$$

として、数列 $u_0, u_1, \dots, u_9$ を定める。このとき、 $0 \leq l \leq 3$ に対してだけでなく $4 \leq l \leq 9$ に対しても $u_l = e(2^l) = e_i 2^{li} + e_j 2^{lj}$ が成り立つ。なぜならば、

$$\begin{aligned} u_l &= -\sigma_0(e_i 2^{(l-2)i} + e_j 2^{(l-2)j}) - \sigma_1(e_i 2^{(l-1)i} + e_j 2^{(l-1)j}) \\ &= e_i 2^{li}(-\sigma_0 2^{-2i} - \sigma_1 2^{-i}) + e_j 2^{lj}(-\sigma_0 2^{-2j} - \sigma_1 2^{-j}) \\ &= e_i 2^{li} + e_j 2^{lj} \end{aligned}$$

だからである。ここで $\sigma(2^i) = 0$ だから $-\sigma_0 2^{-2i} - \sigma_1 2^{-i} = 1$ 等が成り立つことを用いた。こうして $l = 0, 1, \dots, 9$ について $u_l = e(\alpha^l)$ が得られた。これは $\underline{u}$ が $\underline{e}$ の DFT であることを示しているから、 $\underline{u}$ の IDFT を取ると $\underline{e}$ が得られる。

こうして、誤りを求めるために LFSR と IDFT を用いた。一般の RS 符号に対してこの方法は計算量が大いではないが、

と感じられるかもしれない。実際、定義どおりに評価すると、 t を訂正数、 n を符号長とすると、必要な有限体演算の回数として、計算量はそれぞれ tn 及び n^2 のオーダーとなる。一方、3. の方法の計算量は、 t^2 のオーダーである。これは、 t が小さいときは 3. の方法の計算量の方が小さいが、 t が大きくなってくると差は小さくなる。しかも IDFT には様々な高速フーリエ変換 (FFT) の手法を応用することができ⁽¹⁹⁾、計算量はほぼ $n \log n$ のオーダーとなる。このようなことから、 t が大きくなってくると、LFSR と IDFT を用いた方が、回路規模としては少ないことが文献 (21) に示されている。

4.2 符号化における DFT

4.1 の方法は、復号だけでなく符号化にも応用できる。

3.2 では、情報多項式から生成多項式による除法の余りを引くことにより符号化を行った。この方法は、組織的符号化と呼ばれる種類の、送りたい数字がそのまま符号語の成分となる符号化であり、実用の際に採用されることが多い。

一方、次のような組織的でない符号化もある。情報多項式 (4) に対応する $\underline{h}$ に IDFT を行い、 $\underline{c} = \mathcal{F}^{-1}(\underline{h})$ とすると、 $\underline{c}$ は符号語である。なぜならばフーリエ反転公式より $\mathcal{F}(\underline{c}) = \underline{h}$ であるが、式 (13) から $c(x)$ は $2^0, 2^1, 2^2, 2^3$ を根に持つので、 $c(x)$ は $G(x)$ で割り切れるからである。

ここで前節と同様の、LFSR を用いた拡張を行うことにより、この IDFT を用いた符号化を組織的符号化にできることを見よう。

[例 3] 情報 $\underline{h} = (0, 0, 0, 0, 1, 7, 3, 2, 0, 5)$ に対し、

生成多項式 $G(x) = 9 + x + 4x^2 + 7x^3 + x^4$

情報のシンδροーム $(h(1), h(2), h(2^2), h(2^3)) = (7, 3, 3, 1)$

$G(x)$ の LFSR により

$(r(1), r(2), \dots, r(2^9)) = (7, 3, 3, 1, 3, 0, 4, 4, 6, 4)$

(例えば $r(2^4) := -9 \cdot 7 - 1 \cdot 3 - 4 \cdot 3 - 7 \cdot 1 = 3$)

余り $\underline{r} = (9, 2, 0, 7, 0, 0, 0, 0, 0, 0)$

符号語 $\underline{c} = \underline{h} - \underline{r} = (2, 9, 0, 4, 1, 7, 3, 2, 0, 5)$ □

この計算は基本的には 4.1 の DFT による復号と同様である。やはり IDFT を施すと余りが求まりそれ以外は全て 0 になるところが面白い。

本節で述べた LFSR と IDFT による組織的符号化は、あらかじめ分かっている冗長位置の部分の符号語を求めているので、消失訂正⁽⁹⁾の特別な場合である。次に述べるアフィン多様体符号においても、この方法で組織的符号化ができるものがあり、その十分条件も得られる⁽²⁰⁾。

組織的符号化のみで考えると、シンδροームの拡張を行って IDFT を行うよりも、生成多項式で除法するだけの方がもちろんずっと計算量は少ない。しかし、4.1 の最後に述べたように、FFT の手法を用い、しかも組織的符号化と復号化を消失誤り訂正として統合することにより、 t がある程度大きくなってくると全体としては回路規模が小さいことが示される⁽²¹⁾。

5. アフィン多様体符号への拡張⁽²⁰⁾

4. までに見てきたとおり, LFSR とは一次元的な, 言わば一直線に並んだ数列を生成するものだった. この LFSR の一般化として, 二次元以上にすること, 及び複数の LFSR による数列の生成を考えること, が挙げられる. この意味で, 拡張された LFSR が用いられる最も広範な代数的符号のクラスは, アフィン多様体符号^{(22) (23)}であろう. 以下, アフィン多様体符号を定義し例を挙げたい. しかし, きちんと記号や写像を定義してから述べると, 長くなってしまい大変である. そこで, 正確な記述は文献 (17) (20) を参照とし, 本稿では例を用いながら直観的に説明する.

q をある素数の累乗とし, q 元から成る有限体を q 元体と呼び $\mathbb{F}_q$ と表す. 4. まででは $\mathbb{F}_{11}$ を用いていた. これを継続することは可能であるが, 自明でない数値例がかなり複雑となってしまうので, 本章では $q = 3^2$ とした 9 元体 $\mathbb{F}_9$ を用いる. 演算規則については文献 (9) (10) を参照とする.

$\mathbb{F}_q^\times = \mathbb{F}_q \setminus \{0\}$ (つまり 0 を除いた集合) と書く. また

$$\Omega = (\mathbb{F}_q^\times)^2 = \{(\omega_1, \omega_2) \mid \omega_1, \omega_2 \in \mathbb{F}_q^\times\}$$

と表す. (実は $\Omega = (\mathbb{F}_q^\times)^N$ でもできる (N は正の整数)⁽¹⁷⁾. $N = 1$ のときは RS 符号のケースに対応する (例 5). 本章では $N = 2$ とする. また, 0 を含んだ $\Omega = (\mathbb{F}_q)^N$ でもできる⁽²⁰⁾. しかし, IDFT を一般化する必要があるため本稿では考察しない.)

アフィン多様体符号は, まず Ω の部分集合 Ψ を一つ定めることから始まる. そして $n = |\Psi|$, つまり Ψ の要素数を符号長 n と置く. 実際, 後で Ψ は符号語の成分の添え字となる.

[例 4] $\mathbb{F}_9^\times$ の元を, $\alpha^2 + \alpha = 1$ を満たす原始元 α の累乗 $\alpha^0, \alpha^1, \dots, \alpha^7$ で表す. Ψ としては例えば $\Psi = \Omega$ でもよい. また $\Psi = \Psi_H = \{(\omega_1, \omega_2) \in \Omega \mid \omega_1^4 = \omega_2^3 + \omega_2\}$ とすると $|\Psi_H| = 24$ となることが確かめられる. もう一つの例として, $\Psi = \Phi = \Omega \setminus \Psi_H$ とする. 表 1 にこれらを示す. Ψ_H の元に対応するところは $\bigcirc$, Φ の元のところは $\times$ としている. $\square$

表 1 Ω, Ψ, Φ の例.

$\omega_1 \setminus \omega_2$	α^0	α^1	α^2	α^3	α^4	α^5	α^6	α^7
α^0	$\times$	$\times$	$\times$	$\times$	$\bigcirc$	$\bigcirc$	$\times$	$\bigcirc$
α^1	$\bigcirc$	$\bigcirc$	$\times$	$\bigcirc$	$\times$	$\times$	$\times$	$\times$
α^2	$\times$	$\times$	$\times$	$\times$	$\bigcirc$	$\bigcirc$	$\times$	$\bigcirc$
α^3	$\bigcirc$	$\bigcirc$	$\times$	$\bigcirc$	$\times$	$\times$	$\times$	$\times$
α^4	$\times$	$\times$	$\times$	$\times$	$\bigcirc$	$\bigcirc$	$\times$	$\bigcirc$
α^5	$\bigcirc$	$\bigcirc$	$\times$	$\bigcirc$	$\times$	$\times$	$\times$	$\times$
α^6	$\times$	$\times$	$\times$	$\times$	$\bigcirc$	$\bigcirc$	$\times$	$\bigcirc$
α^7	$\bigcirc$	$\bigcirc$	$\times$	$\bigcirc$	$\times$	$\times$	$\times$	$\times$

さて, アフィン多様体符号を定義するための一つの要素 Ψ について述べたが, もう一つだけ別の要素 L が必要である. これについては, 手短かに数学的に述べてしまおう. 2 変数多項式環 $\mathbb{F}_q[x, y]$ の商環 $V = \mathbb{F}_q[x, y]/(x^{q-1} - 1, y^{q-1} - 1)$ を考える. V は次のような有限個の多項式の集合と考えることができる.

$$V = \left\{ h(x, y) \in \mathbb{F}_q[x, y] \mid h(x, y) = \sum_{i=0}^{q-2} \sum_{j=0}^{q-2} h_{i,j} x^i y^j \right\} \quad (15)$$

V のイデアル $Z(\Psi) \subset V$ を

$$Z(\Psi) = \{f \in V \mid f(\omega_1, \omega_2) = 0 \text{ for all } (\omega_1, \omega_2) \in \Psi\}$$

として定める. このとき L は, 商環 $V/Z(\Psi)$ を $\mathbb{F}_q$ 上の線形空間と見たときのある部分空間と定める. 最後に, $\mathbb{F}_q^\Psi = \{\underline{c} = (c_P)_{P \in \Psi} \mid c_P \in \mathbb{F}_q\}$ は Ψ の元 P で添え字付けられたベクトル $\underline{c} = (c_P)_{P \in \Psi}$ の成す $\mathbb{F}_q$ 上の次元 n の線形空間を表す.

[定義 4] 部分集合 $\Psi \subset \Omega$, 部分空間 $L \subset V/Z(\Psi)$ に対し,

$$\begin{aligned} \mathcal{C}(L, \Psi) &= \left\{ (c_P)_{P \in \Psi} \in \mathbb{F}_q^\Psi \mid \begin{array}{l} (c_P)_{P \in \Psi} = (f(P))_{P \in \Psi} \\ \text{for some } f \in L \end{array} \right\}, \\ \mathcal{C}^\perp(L, \Psi) &= \left\{ (c_P)_{P \in \Psi} \in \mathbb{F}_q^\Psi \mid \begin{array}{l} \sum_{P \in \Psi} c_P f(P) = 0 \\ \text{for all } f \in L \end{array} \right\} \end{aligned} \quad (16)$$

と定める. $\mathcal{C}(L, \Psi)$ を双対アフィン多様体符号, $\mathcal{C}^\perp(L, \Psi)$ をアフィン多様体符号と呼ぶ⁽²²⁾. $\square$

定義から, これら $\mathcal{C}(L, \Psi)$ と $\mathcal{C}^\perp(L, \Psi)$ とは互いに双対符号^{(9) (10)}の関係にある. また次元は $k = \dim L$ と置くと $\dim \mathcal{C}(L, \Psi) = k$, $\dim \mathcal{C}^\perp(L, \Psi) = n - k$ となる⁽²³⁾. 通常は, $\mathcal{C}^\perp(L, \Psi)$ の方を考えることが多い. 理由は, RS 符号の復号においてもそうだったように, アフィン多様体符号の復号においても, 誤りベクトル $\underline{e} = (e_P)_{P \in \Psi}$ のシンドロームをまず求める必要があるからである. このときシンドロームは, L の基底の元 f に対する $\sum_{P \in \Psi} e_P f(P)$ である. 以後は $\mathcal{C} = \mathcal{C}^\perp(L, \Psi)$ とする.

[例 5] (RS 符号のケース) この例では $N = 1$ 及び $\Omega = \mathbb{F}_{11}^\times$ とする. このとき $V = \mathbb{F}_{11}[x]/(x^{10} - 1)$ となる. また $\Psi = \Omega$ とすると, $Z(\Psi) = (x^{10} - 1) = \{0\}$ 及び $V/Z(\Psi) = V$ となる. 部分空間 $L \subset V/Z(\Psi)$ を, 次数が 3 以下の多項式全体 $L = \{a + bx + cx^2 + dx^3 \mid a, b, c, d \in \mathbb{F}_{11}\}$ と定める. この時 3.2 の RS 符号は $\mathcal{C}^\perp(L, \Psi)$ に一致する. 実際, RS 符号の符号語が定義式 (16) の条件を満たすことは, シンドロームの計算 (5) と同様に確かめることができる. $\square$

[例 6] ($N = 2$, 例 4 の続き) $\Psi = \Omega$ のときは $Z(\Omega) = (x^{q-1} - 1, y^{q-1} - 1) = \{0\}$ である. $\Psi = \Psi_H$ のときは, Ψ_H の定義から, $Z(\Psi_H)$ は $y^3 - x^4 + y$ で生成されるイデアルとなる. $y^3 - x^4 + y$ はエルミート曲線と呼ばれる代数曲線を定める多項式である⁽¹⁰⁾. $\Psi = \Phi$ のときは, 数式処理ソフトウェアで計算すると, $Z(\Phi)$ は $y^5 - y^3 + y + x^4 y^2 + x^4$ で生成されるイデアルとなる. $\square$

[補足 1] この例で述べたようなイデアル $Z(\Psi)$ の生成元を求めるためには, 連立方程式を立てて掃き出し法 (ガウス消去法) により解けばよい. その際の計算量は n^3 のオーダーである. 一方, Berlekamp-Massey-阪田 アルゴリズム (BMS 法)^{(24) (25)}を用いて求めることもできる. BMS 法の計算量は

n^3 のオーダーよりも小さい。BMS 法は、BM 法の一般化であり、与えられた多次元数列を生成する‘最小段数の多次元 LFSR’を求めるものである。ここで、最小段数の多次元 LFSR とは、後で述べる単項式順序を一つ定めたときに最小性を満たす多次元 LFSR のことである。これを多項式表示したものはグレブナー基底 (Gröbner basis) と呼ばれる^{(26) (27)}。つまり、次の対応関係が成り立つ。

BM 法 : 誤り位置多項式 = BMS 法 : グレブナー基底

BM 法やユークリッド復号法が RS 符号の復号に用いられたように、BMS 法はアフィン多様体符号の復号に用いられる。その際、BMS 法は、誤り位置を Ψ とするときのイデアル $Z(\Psi)$ のグレブナー基底を高速に求めるアルゴリズムである。BMS 法の解説文としては文献 (28) (29) が挙げられる。□

さて、本章の目的は、多次元の LFSR の例とアフィン多様体符号との関係を例示することにある。そのために、筆者が文献 (17) (20) で Main Lemma と呼んでいる性質について簡単に述べる。文献 (17) (20) における Main Lemma とは、多次元の LFSR と IDFT を用いて、 $V/Z(\Psi)$ から $\mathbb{F}_q^\Psi = C^\perp(\{0\}, \Psi)$ への標準的な符号化を与えるものである。この $C^\perp(\{0\}, \Psi)$ 自体は符号長も次元も n だから誤り訂正能力はないが、Main Lemma を入れ子構造で用いることによって、符号構成 (非組織的符号化)、復号、組織的符号化といった (4. で述べた) 代数的符号の多くの操作を統一的に述べることができる。

Main Lemma を具体的に解説するために、 Ψ を例 4 の Φ 、また $L = \{0\}$ としたアフィン多様体符号 $C = C^\perp(\{0\}, \Phi)$ を考える。このとき $V/Z(\Phi)$ は、 V の表示 (15) の部分集合として次のように表される。

$$V/Z(\Phi) = \left\{ h(x, y) \in V \mid h(x, y) = \sum_{i=0}^7 \sum_{j=0}^4 h_{i,j} x^i y^j \right\}$$

例として、 $h(x, y) \in V/Z(\Phi)$ を次の表 2 のように与える。

表 2 $h(x, y) \in V/Z(\Phi)$ の数値例。

$i \setminus j$	0	1	2	3	4
0	α^2	0	α^3	α^6	α^1
1	α^3	α^5	α^4	0	α^5
2	α^0	α^4	α^1	α^6	α^3
3	α^1	α^7	α^2	α^0	α^5
4	α^7	α^6	0	α^1	0
5	α^6	α^7	α^2	α^4	α^7
6	0	α^5	α^0	α^3	α^0
7	α^1	α^3	α^5	α^2	α^4

次に、二次元的な LFSR を、 $Z(\Phi)$ を生成する $y^5 - y^3 + y + x^4 y^2 + x^4$ より、 $0 \leq i \leq 7, 0 \leq j \leq 2$ に対し

$$h_{i,j+5} = h_{i,j+3} - h_{i,j+1} - h_{i+4,j+2} - h_{i+4,j} \quad (17)$$

と定める。(この作り方を天下りの説明すると、 $y^5 = y^3 - y - x^4 y^2 - x^4$ の両辺に $x^i y^j$ を掛けて $x^i y^{j+5} = x^i y^{j+3} - x^i y^{j+1} - x^{i+4} y^{j+2} - x^{i+4} y^j$ となり、累乗を添え字とすることにより得られる。) ただし、右辺の $h_{i+4,j+2}, h_{i+4,j}$

については、 $i+4 \geq 8$ となった場合には $i+4 = i+4 \pmod{8}$ として計算を行う (つまり係数を巡回させる)。式 (17) で定まる二次元的な LFSR により、表 (2) の $h(x, y)$ を二次元的な数列の初期値と見て V へ拡張する。そのためには、二次元的な数列の何らかの順序を定める必要がある。アフィン多様体符号では、単項式順序⁽²⁶⁾を用いることが多い。今は、次の辞書式順序と呼ばれる単項式順序の一つを用いる。 $0 \leq i, j, i', j' \leq 7$ である $(i, j), (i', j')$ について、

$$(i, j) \leq_{\text{lex}} (i', j') \iff j < j' \text{ or } (j = j' \text{ and } i \leq i')$$

と定める。これにより、 $(0, 0) \leq_{\text{lex}} (1, 0) \leq_{\text{lex}} \cdots \leq_{\text{lex}} (7, 0) \leq_{\text{lex}} (0, 1) \leq_{\text{lex}} \cdots \leq_{\text{lex}} (7, 7)$ という順序となる。

この順序で $h_{0,5}, h_{1,5}, \dots, h_{7,7}$ と帰納的に定め、再び $h(x, y) = \sum_{i,j=0}^7 h_{i,j} x^i y^j$ と表すと、 $h(x, y)$ は次の表 3 のようになる。

表 3 拡張された $h(x, y) = \sum_{i,j=0}^7 h_{i,j} x^i y^j \in V$ 。

$i \setminus j$	0	1	2	3	4	5	6	7
0	α^2	0	α^3	α^6	α^1	α^0	α^4	α^3
1	α^3	α^5	α^4	0	α^5	α^1	α^7	α^7
2	α^0	α^4	α^1	α^6	α^3	α^6	0	α^0
3	α^1	α^7	α^2	α^0	α^5	α^5	α^3	0
4	α^7	α^6	0	α^1	0	α^2	α^2	α^6
5	α^6	α^7	α^2	α^4	α^7	0	0	α^6
6	0	α^5	α^0	α^3	α^0	α^2	α^3	0
7	α^1	α^3	α^5	α^2	α^4	α^0	0	α^4

例えば $h_{0,5}, h_{7,7}$ については、次のように計算される。

$$\begin{aligned} h_{0,5} &= h_{0,3} - h_{0,1} - h_{4,2} - h_{4,0} \\ &= \alpha^6 - 0 - 0 - \alpha^7 = \alpha^0 \\ h_{7,7} &= h_{7,5} - h_{7,3} - h_{3,4} - h_{3,2} \\ &= \alpha^0 - \alpha^2 - \alpha^5 - \alpha^2 = \alpha^4 \end{aligned}$$

Main Lemma の最後のステップでは、拡張した $h(x, y)$ に対し二次元 IDFT を行う。これは、次の式 (18) で計算され、結果は表 4 のようになる。

$$c(x, y) = \sum_{a,b=0}^7 c_{a,b} x^a y^b, \quad c_{a,b} = -h(\alpha^{-a}, \alpha^{-b}) \quad (18)$$

表 4 拡張された $h(x, y)$ に対する IDFT $c(x, y)$ 。

$a \setminus b$	0	1	2	3	4	5	6	7
0	α^6	α^3	α^5	α^6	0	0	α^6	0
1	0	0	0	0	0	α^5	α^4	α^2
2	α^4	α^1	α^0	α^3	0	0	α^1	0
3	0	0	α^7	0	α^0	α^2	α^6	α^7
4	0	α^6	α^1	α^5	0	0	α^6	0
5	0	0	α^2	0	α^3	α^1	α^2	α^0
6	α^0	α^6	α^3	α^7	0	0	α^6	0
7	0	0	α^0	0	α^5	α^3	α^7	α^5

この表の注目すべき点は、 Φ 以外の元 (つまり Ψ_H の元) に

対応するところでは、値が 0 になっていることである (4.1 や 4.2 でもそうだった)。それは表 1 と比較して確認できる。こうして $\mathbb{F}_q^\Phi = \mathcal{C}^\perp(\{0\}, \Phi)$ のベクトルが得られた。

この例で $\Psi = \Phi$ とした理由は、二次元 LFSR (17) が簡単になるものを選んだからである。 Ψ は Ω の部分集合であれば任意であるが、一般には二次元 LFSR はもっと多数の和や定数倍、また複数の漸化式が必要である。 $\Psi = \Psi_H$ とし、また L とし、代数学曲線から定まるある部分空間を取ったものがエルミート曲線符号⁽¹⁰⁾である。このように、アフィン多様体符号は、代数曲線符号や代数幾何符号^{(9) (28)}の一般化であり、有効な符号構成が現在も調べられている。

[補足 2] 式 (18) の逆変換 (同じことだが、線形写像としての逆写像) は、次の二次元 DFT で与えられる。

$$h(x, y) = \sum_{i,j=0}^7 h_{i,j} x^i y^j, \quad h_{i,j} = c(\alpha^i, \alpha^j) \quad (19)$$

つまりフーリエ反転公式の二次元版が成立し、表 4 の数値にこの変換 (19) を施すと表 3 の数値に戻る。式 (19) の $h_{i,j}$ について

$$h_{i,j} = \sum_{a,b=0}^7 c_{a,b} \alpha^{ai+bj} = \sum_{b=0}^7 \left(\sum_{a=0}^7 c_{a,b} \alpha^{ai} \right) \alpha^{bj}$$

であり、内側の和は表 4 の各列ごとの一次元 DFT、外側の和はそうして得られた表の各行ごとの一次元 DFT である。これは二次元 DFT が、一次元 DFT を縦横に行ったものであることを示す。簡単な事実であるが、これが様々な FFT の手法⁽¹⁹⁾の第一歩である。実際、式 (19) の計算量は $(q-1)^4$ のオーダーであるが、一方、一次元 DFT の計算量を $(q-1)^2$ のオーダーとすると、縦横に行ったときの計算量は $2(q-1)^3$ のオーダーとなり、4 乗から 3 乗に改良される。二次元 IDFT についても同様である。□

6. 準巡回符号

本章では準巡回符号、特にその符号化について、主に文献 (30) に従って解説する。符号語の巡回シフトが再び符号語であるような性質を持つ線形符号のクラスは巡回符号と呼ばれ、5. までの RS 符号も巡回符号である。準巡回符号とは、巡回符号の一般化であり、更に広範な符号のクラスであり、多くの有用な LDPC 符号を含むことが知られている⁽¹⁶⁾。準巡回符号の厳密な定義は文献 (16) (30) 参照とし、本稿では具体的な例を用いて説明する。

本章で用いる有限体は二元体 $\mathbb{F}_2 = \{0, 1\}$ とする。二元体においては $1 + 1 = 0$ (つまり mod 2 を取る) であり、よって $-1 = +1$ であることに注意する。他の演算規則は通常の加減乗除と同様である。

[例 7] 次の生成行列 ($\tilde{G}_1$ と置く) で定義される符号長 7 の線形符号 C_1 を例として取り上げる。

$$\tilde{G}_1 = \left(\begin{array}{ccc|ccc|c} 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{array} \right) \quad (20)$$

このとき C_1 の符号語は、二元ベクトル $\underline{h} = (h_1, h_2, h_3, h_4)$ があって $\underline{h}\tilde{G}_1$ と表され、また逆も成り立つ。 $\tilde{G}_1$ について、1 行目 (101|100|0) の線で区切った各部分ごとの右巡回シフトは (110|010|0) となり、2 行目に一致する。他の行も全てこのような性質を持つ (4 行目の各部分ごとの右巡回シフトは 4 行目自身に一致する)。この性質を持つ生成行列が取れるとき、 C_1 は準巡回 (QC: Quasi-Cyclic) 符号と呼ばれる^(注1)。 C_1 の次元 (情報長) は $k = 4$ である。

一方、 C_1 の双対符号 $C_1^\perp$ の生成行列、すなわち C_1 の検査行列は次の行列で与えられる。

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

このように双対符号も QC 符号となり、これは一般的に成り立つ性質である。もし 4 列目の ${}^t(101)$ を右端に移動させると、検査行列は通常の二元 ($n = 7, k = 4$) ハミング符号^{(9) (10)}の検査行列に一致することが分かる。つまり C_1 は、通常の二元 (7, 4) ハミング符号の成分の順序を並び換えたものである。□

巡回符号は生成多項式を用いて定めることができるのと同様に、準巡回符号も多項式を用いた手法が有効である。

[例 8] 生成行列 (20) の各部分を多項式表示したものは、次の多項式行列となることが直ちに分かる。

$$\left(\begin{array}{ccc} 1+x^2 & 1 & 0 \\ 1+x & x & 0 \\ x+x^2 & x^2 & 0 \\ 1+x+x^2 & 0 & 1 \end{array} \right) \quad (21)$$

この多項式行列に、行基本変形を行うことにより、1 列目の最大公約多項式を求め、更に再帰的に計算を行うことが次の方針である。2 行目と 1 行目を交換し、1 行目のある多項式倍を他の行に加えることにより、次の多項式行列が得られる。

$$\left(\begin{array}{ccc} 1+x & x & 0 \\ 0 & 1+x+x^2 & 0 \\ 0 & 0 & 0 \\ 1 & x^2 & 1 \end{array} \right)$$

この多項式行列の 4 行目を $g_1 = (1, x^2, 1)$ 、2 行目を $g_2 = (0, 1+x+x^2, 0)$ と置くと、 $(1+x, x, 0) + (1+x)g_1 + xg_2 = (0, 0, 1+x)$ であることに注意する。このことを踏まえ、更に (000) を削除し並べ替えることにより、次の多項式行列が得られる。

(注1): 部分ごとの巡回シフト長が今の場合 3, 3, 1 と一定でない。このような場合は一般化準巡回 (GQC: Generalized QC) 符号と呼ばれるが⁽³⁰⁾、本稿ではこれも QC 符号と呼ぶ。

$$\begin{pmatrix} 1 & x^2 & 1 \\ 0 & 1+x+x^2 & 0 \\ 0 & 0 & 1+x \end{pmatrix}$$

このように多項式行列に対する行基本変形のみで、多項式行列 (21) を上半三角多項式行列に変形することができる。これを生成多項式行列と呼び、QC 符号にとって基本的な道具である。各巡回シフト長を $n(i)$ と置く ($i = 1, 2, 3$)。今の場合、 $n(1) = n(2) = 3, n(3) = 1$ である。このとき、生成多項式行列の (i, i) 成分は、必ず $x^{n(i)} - 1$ の約多項式となることが示される。

この生成多項式行列の第 1, 2, 3 行をそれぞれ g_1, g_2, g_3 とする。このとき、 C_1 の符号語は、多項式 f_1, f_2, f_3 があって $f_1g_1 + f_2g_2 + f_3g_3$ と表され、また逆も成り立つ (ただし各成分の次数が巡回シフト長を超える場合は mod $(x^{n(i)} - 1)$ を取ると定める。) この性質が、‘生成’ 多項式行列の名前の由来である。

各列の成分の次数について、対角成分以外の次数が対角成分の次数より小さいとき、生成多項式行列は被約であると呼ばれる (ただし零多項式の次数は任意の多項式の次数より小さいとする。) 生成多項式行列に更に行基本変形を行うことによって、被約にすることができる。 g_1 を $g_1 + g_2$ で置き換えることにより、次の被約な生成多項式行列が得られる (これを G_1 と置く)。

$$G_1 = \begin{pmatrix} 1 & 1+x & 1 \\ 0 & 1+x+x^2 & 0 \\ 0 & 0 & 1+x \end{pmatrix}$$

被約な生成多項式行列は、各々の QC 符号に対し一意に定まるという著しい特徴を持つ。また次の例のとおり、QC 符号の符号化の際、計算量が最小となるというメリットもある。□

最後に、生成多項式行列を用いる QC 符号の符号化について述べる。線形符号としての生成行列を用いる符号化の計算量が n^2 のオーダーになるのに対し、生成多項式行列を用いると計算量が ln のオーダー程度であり、削減される。ここで、 n は符号長、 l は巡回シフト部分の個数である。また生成多項式行列を用いる符号化は、本質的には LFSR (正確には多項式の乗法及び除法) であることから、ハードウェア化が容易であるという特長もある。

[例 9] G_1 の (i, i) 成分の次数を d_i とすると、 $(d_1, d_2, d_3) = (0, 2, 1)$ である。また $k_i = n(i) - d_i$ とすると、 $(k_1, k_2, k_3) = (3, 1, 0)$ となることから、計 4 ビットの情報を符号化できることが示される。例として $(u_{1,1}, u_{1,2}, u_{1,3}, u_{2,1}) = (0, 0, 1, 1)$ の符号化を行う。この情報を多項式 $u_1 = x^2, u_2 = x^2, u_3 = 0$ に変換する。次の多項式行列に対し、今までと類似の行基本変形を行う。

$$\left(\begin{array}{c} G_1 \\ u_1 \ u_2 \ u_3 \end{array} \right) = \left(\begin{array}{ccc|ccc} 1 & 1+x & 1 & & & \\ 0 & 1+x+x^2 & 0 & & & \\ 0 & 0 & 1+x & & & \\ \hline & x^2 & x^2 & 0 & & \end{array} \right) \quad (22)$$

G_1 の第 1, 2, 3 行をそれぞれ g_1, g_2, g_3 とするとき、式 (22) の

右辺の第 4 行に $x^2g_1 + (1+x)g_2 + (1+x)g_3 = (x^2, 1+x^2, 1)$ を加えることにより、第 4 行は $(0, 1, 1)$ となる。これ以上 g_1, g_2, g_3 を第 4 行目に加えても次数を下げるができないため、ここで行基本変形は終了である。こうして、符号語 $c = (x^2, x^2, 0) + (0, 1, 1) = (x^2, 1+x^2, 1)$ が得られた。実際に、 $c = x^2g_1 + (1+x)g_2 + (1+x)g_3 \in C_1$ であることから、多項式ベクトル c は符号語であることが分かる。また c に対応する二元ベクトルは (0011011) だから、 $c \in C_1$ であることは次の式からも分かる。

$$(0011011) = (10111) \begin{pmatrix} 101 & 100 & 0 \\ 110 & 010 & 0 \\ 011 & 001 & 0 \\ \hline 111 & 000 & 1 \end{pmatrix} \quad \square$$

ここに書いたような QC 符号の扱いは、‘加群のグレブナー基底’の特別な場合となる^{(31) (32)}。また筆者らはこれを QC 符号であるような LDPC 符号に応用した⁽³³⁾。この論文⁽³³⁾の結果は、企業の何人かの方々から有用であったとおっしゃって頂いた。近年、筆者は整数符号⁽³⁴⁾や擬巡回符号⁽³⁵⁾、また QC 符号の数え上げ⁽³⁰⁾に対しても応用している。

7. む す び

幾つかの代数的な誤り訂正符号について、初等的な解説を試みた。この範ちゅうにある話題としては、他に二元 (7, 4) ハミング符号があり、これも易しく導入には最適であるが、他の文献^{(9) (10)}参照とする。なお、本稿では代数的なトピックばかりを集めた。その理由は、極力、本稿だけで self-contained となるようにしたかったからである。しかし数学的な道具立てについて言えば、本来は代数だけでなく、幾何や解析や確率が交差するところが面白いところである。また情報工学において、様々な分野が交錯するところが最も興味深いということは言うまでもない。本稿は自己完結するように心掛けた小さな世界であるが、同時に新たな展開への入り口でもある。

謝辞 原稿を読み有益なコメントをして頂いた読者の方々、八木編集委員、また白木 FR 誌編集長に感謝します。

文 献

- (1) 松井 一, “線形帰還シフトレジスタ理論とその発展,” 信学技報, IT2013-41, pp.1-9, Nov. 2013.
- (2) J.L. Massey, “Shift-register synthesis and BCH decoding,” *IEEE Trans. Inf. Theory*, vol.IT-15, no.1, pp.122-127, Jan. 1969.
- (3) R.L. Graham, D.E. Knuth, and O. Patashnik, *Concrete Mathematics: A Foundation for Computer Science*, Addison-Wesley Professional, 1994. (コンピュータの数学, 有沢 誠, 萩野達也, 安村通晃, 石畑 清 (共訳), 共立出版, 東京, 1993.)
- (4) 和田秀男, コンピュータと素因子分解, 遊星社, 東京, 1999.
- (5) E.R. Berlekamp, *Algebraic Coding Theory*, McGraw-Hill, 1968.
- (6) Y. Sugiyama, M. Kasahara, S. Hirasawa, and T. Namekawa, “A method for solving key equation for decoding Goppa codes,” *Inf. Control*, vol.27, no.1, pp.87-99, Jan. 1975.

- (7) 笠原正雄, “リードソロモン符号の半世紀—デジタル情報通信・記録技術発展の歴史の中で総合的に考える—,” IEICE Fundamentals Review, vol.5, no.1, pp.28–41, July 2011.
- (8) 平澤茂一, 笠原正雄, “ユークリッド復号法,” IEICE Fundamentals Review, vol.4, no.3, pp.183–191, Jan. 2011.
- (9) 今井秀樹, 符号理論, 電子情報通信学会, 東京, 1990.
- (10) J. Justesen and T. Høholdt, *A Course In Error-Correcting Codes*, European Mathematical Society, 2004. (ユステセン, ホーホルト (共著), 誤り訂正符号入門, 阪田省二郎, 栗原正純, 松井 一, 藤沢匡哉 (共訳), 森北出版, 東京, 2005.)
- (11) 萩原 学, 符号理論 ~ デジタルコミュニケーションにおける数学 ~, 日本評論社, 東京, 2012.
- (12) 坂庭好一, 渋谷智治, 代数系と符号理論入門, コロナ社, 東京, 2010.
- (13) R. Lidl and H. Niederreiter, *Finite Fields*, 2nd ed., Cambridge University Press, 2008.
- (14) 植松友彦, 代数系と符号理論 (TokyoTech Be-TEXT), オーム社, 東京, 2010.
- (15) 和田山正, 誤り訂正技術の基礎, 森北出版, 東京, 2010.
- (16) S. Lin and D.J. Costello, *Error Control Coding*, 2nd ed., Prentice Hall, June 2004.
- (17) H. Matsui, “Decoding a class of affine variety codes with fast DFT,” *International Symposium on Information Theory and Its Applications (ISITA2012)*, Honolulu, Hawaii, USA, pp.436–440, Oct. 2012.
- (18) 三田誠一, 西谷卓史, 澤口秀樹, 松井 一, 磁気ディスクの信号処理技術 PRML 方式の基礎と実際, 森北出版, 東京, 2010.
- (19) R.E. Blahut, *Theory and Practice of Error Control Codes*, Addison-Wesley, 1983.
- (20) H. Matsui, “Lemma for linear feedback shift registers and DFTs applied to affine variety codes,” *IEEE Trans. Inf. Theory*, vol.60, no.5, pp.2751–2769, May 2014.
- (21) H. Matsui and S. Mita, “A new encoding and decoding system of Reed–Solomon codes for HDD,” *IEEE Trans. Magn.*, vol.45, no.10, pp.3757–3760, Oct. 2009.
- (22) J. Fitzgerald and R.F. Lax, “Decoding affine variety codes using Gröbner bases,” *Des., Codes Cryptogr.*, vol.13, no.2, pp.147–158, Feb. 1998.
- (23) K. Saints and C. Heegard, “Algebraic-geometric codes and multidimensional cyclic codes: a unified theory and algorithms for decoding using Gröbner bases,” *IEEE Trans. Inf. Theory*, vol.41, no.6, pp.1733–1751, Nov. 1995.
- (24) D.A. Cox, J. Little, and D. O’Shea, “The Berlekamp–Massey–Sakata algorithm,” in *Using Algebraic Geometry*, 2nd ed., Chapter 10, pp.494–532, Springer, 2005.
- (25) S. Sakata, “Extension of the Berlekamp–Massey algorithm to N dimensions,” *Inf. Comput.*, vol.84, no. 2, pp.207–239, Feb. 1990.
- (26) D. Cox, J. Little, and D. O’Shea, *Ideals, Varieties, and Algorithms: An introduction to computational algebraic geometry and commutative algebra*, 2nd ed., Springer Publishers, New York, 1997.
- (27) グレブナー基底の現在, 日比孝之 (編), 数学書房, 東京, 2006.
- (28) 阪田省二郎, “代数幾何符号とその復号法について,” 数理科学, no.421, pp.33–40, July 1998. “代数幾何符号とその復号法について (続),” 数理科学, no.422, pp.58–65, Aug. 1998.
- (29) 阪田省二郎, “代数的符号理論,” IEICE Fundamentals Review, vol.1, no.3, pp.44–57, Jan. 2008.
- (30) H. Matsui, “On generator and parity-check polynomial matrices of generalized quasi-cyclic codes,” submitted.
- (31) C. Heegard, J. Little, and K. Saints, “Systematic encoding via Gröbner bases for a class of algebraic geometric Goppa codes,” *IEEE Trans. Inf. Theory*, vol.41, no.6, pp.1752–1761, Nov. 1995.
- (32) K. Lally and P. Fitzpatrick, “Algebraic structure of quasi-cyclic codes,” *Discrete Appl. Math.*, vol.111, no.1–2, pp.157–175, July 2001.
- (33) V.T. Van, H. Matsui, and S. Mita, “Computation of Gröbner basis for systematic encoding of generalized quasi-cyclic codes,” *IEICE Trans. Fundamentals*,

vol.E92-A, no.9, pp.2345–2359, Sept. 2009.

- (34) H. Matsui, “On generator matrices and parity check matrices of generalized integer codes,” *Des., Codes Cryptogr.* DOI 10.1007/s10623-013-9883-7
- (35) H. Matsui, “On generator polynomial matrices of generalized pseudo-cyclic codes,” *International Symposium on Information Theory and Its Applications (ISITA2014)*, Melbourne, Australia, Oct. 2014.

(IT 研究会提案, 平成 26 年 10 月 2 日受付)

11 月 12 日最終受付)

松井 一 (正員)

1994 静岡大・理・数学卒. 1996 新潟大大学院理学研究科数学専攻修士課程了. 1999 名大大学院多元数理科学研究科博士課程了. 博士 (数理学). 1999 豊田工大・工・ポスドクトラル研究員. 2002 同大学助手. 2006 同大学助教授 (2007 から准教授), 現在に至る. 情報工学, 符号理論の研究に従事.