数式処理 *J.JSSAC (2002)*
Vol. 9, No. 2, pp. 32 - 41

特集「2001 年度卒業論文・修士論文」

代数幾何符号に関する研究

上 原　剛*

愛媛大学大学院理工学研究科

Abstract

本稿では誤り訂正符号の一つである代数幾何符号について理論の概要を調べ、その実装を数式処理システム Risa/Asir 上で行った。実装においては代数曲線 x^3+y^3+1 を考え、代数幾何符号のアルゴリズムや、その可能性の知見を得た。結果として従来の符号の持つ問題点を克服できることを確認した。

1 はじめに

符号理論の研究は 1948 年に Shannon による情報理論の誕生と共に生まれた。1947 年には Hamming 符号が考案され、1960 年には現在も広く用いられている BCH 符号や RS 符号といった符号が発表されている [4]。これらは現在、宇宙通信分野や、コンパクトディスク、光ディスクなどの記憶装置などで広く用いられている符号である。しかし、BCH(Bose-Chaudhuri-Hocquenghem) 符号の場合は符号長を長くすると符号化率もしくは相対最小距離が悪くなることや、RS(Reed-Solomon) 符号の場合、符号長に制限があるといった問題点が存在する。1970 年に Goppa により Goppa 符号、さらに 1981 年に代数幾何符号として新しい符号が提唱された。符号長が大きいときに VG(Valshamov-Gilbert) 限界式を満たす符号の構成は容易ではないが、代数幾何符号においてはそのような符号を構成的に与えることが可能である [3]。現在も符号化、復号化に関して多くの研究が積極的に行われている。

本稿では代数幾何符号について理論の概要を調べ、代数曲線 x^3+y^3+1 をもとに実装を行った。以下に誤り訂正符号、代数幾何符号、SV(Skolobogatov-Vlădut) 復号アルゴリズムについての概略を示した後、Risa/Asir 上での実装に関して詳しく示す。主に文献 [5][6] を参考としている。

代数幾何符号の構成法や復号法において、グレブナ基底の概念を用いることの有効性については、多くの研究が行われているが [1][2] 本稿ではふれない。

*uehara@hpc.cs.ehime-u.ac.jp

2 誤り訂正符号

本節では簡単に誤り訂正符号について文献 [7] をもとに説明する。ある符号語において、その中の”0” でない元の総数をその語の重みという。符号語をベクトルと考えるときに、異なる 2 符号語間で各成分毎に比較し、異なる元の総数を符号語間の Hamming 距離という。また、その符号における存在する全ての符号語の Hamming 距離の最小値を符号の最小距離という。

一般に符号長 n, 情報長 k、最小距離 d のブロック符号を (n,k,d) ブロック符号という。(n,k,d) ブロック符号の場合、その符号化率 $r=\frac{k}{n}$ と、相対最小距離 $\delta=\frac{d}{n}$(厳密には $=\frac{d+1}{2n}$) がともに大きくなるものが良い符号と考えられる。符号化率、相対最小距離ともに等しい場合は、符号長が大きい符号の方が良い。相対最小距離が等しく符号長が長いなら、最小距離と符号長の比率が等しいということになるので、その最小距離も長くなる。つまりある部分でかたまって誤りが発生してもその障害に耐え得る可能性が高い。符号長の短い符号をいくつか並べてもかたまった誤りに対しては効果が期待できない。

3 代数幾何符号

代数幾何符号では有限体上の代数曲線をもとに構成される。代数曲線上のある任意の指定した点でしか極を持たないような関数体集合に、その他の有理点を代入してやることで符号を構成することができる。特に一点のみを選ぶとき一点代数曲線符号とよばれる。代数幾何符号としては Goppa 関数型符号、Goppa 留数型符号が代表的である。

符号を構成するには生成行列をもとめる必要がある。生成行列とは符号語ベクトル c, 情報語ベクトル i とするとき $c=iG$ となるような行列 G のことである。G は符号語を線形和の形で表現することと同一なので、G の行ベクトルは符号語空間の基底となる必要がある。

代数幾何符号では有限体上の代数曲線を用いて定義されるが、具体的には曲線の関数体に有理点を代入することで生成行列を構成する。空間の基底となるようなベクトルを見つける必要があるが、それには曲線の因子を用いる。

因子は代数曲線 C 上の点 $P_1,P_2,\cdots,P_k$ を任意に選んでその整数係数一次結合

$$D=m_1P_1+m_2P_2+\cdots+m_kP_k \tag{1}$$

で表される。

ある因子 $D=m_1P_1+m_2P_2+\cdots+m_sP_s-n_1Q_1-n_2Q_2-\cdots-n_tQ_t$ に対して体 K 上の代数曲線 C における関数体 $K(C)$ の部分集合 $L(D)$ を、

$$\begin{aligned} L(D)=\{\varphi\in K(C) \quad | \quad & \varphi=0、\text{または}\varphi\text{は}P_i(i\geq 0)\text{で高々}m_i\text{位の極をもち、} \\ & Q_j(j\geq 0)\text{で少なくとも}n_j\text{位の零点をもち、他の点では正則}\} \end{aligned} \tag{2}$$

と定義する。このとき $L(D)$ は K 上の有限次元ベクトル空間となる。$\varphi \in L(D)$ となる φ に D に含まれないような有理点を代入することで符号語を構成することができる。このことを利用し $L(D)$ の基底となるような関数体を選ぶことで、生成行列を構成することが可能となる。

ある体 K 上の代数曲線 $C : f(x, y)$ にたいし、$(P_1, P_2, \cdots, P_n)$ をある拡大体上の有理点とし、因子 $B = P_1 + P_2 + \cdots + P_n$ とする。また、B に含まれないような点 $Q_1, Q_2, \cdots, Q_m$ により、因子 $G = Q_1 + Q_2 + \cdots + Q_n$ をつくる。このとき次のように Goppa 関数型符号 C_L、Goppa 留数型符号 C_Ω が定義される。

$$
\begin{aligned}
C_L(C, B, G) &= \{[c_1, c_2, \cdots, c_n] \in K^n \mid \exists \varphi \in L(G), \\
&\qquad c_1 = \varphi(P_1), c_2 = \varphi(P_2), \cdots, c_n = \varphi(P_n)\} \qquad (3) \\
C_\Omega(C, B, G) &= \{[c_1, c_2, \cdots, c_n] \in K^n \mid \forall \varphi \in L(G), \\
&\qquad c_1\varphi(P_1) + c_2\varphi(P_2) + \cdots + c_n\varphi(P_n) = 0\} \qquad (4)
\end{aligned}
$$

Goppa 関数型符号と Goppa 留数型符号はお互いに双対な符号である。定義 (3), (4) よりどちらかの符号の生成行列がわかれば、それは他方の検査行列となる。

Goppa 関数型符号、Goppa 留数型符号の性能について考える。

- Plücker の公式
  $C : f(x, y) = 0$ を非特異な n 次の代数曲線とすると、
  種数 $g = \frac{1}{2}(n-1)(n-2)$ となる。

- Riemann の定理
  ある非特異曲線 C とその因子 D が与えられたとき、g を C の種数とすると、
  $\deg(D) > 2g - 2$ ならば $l(D) - \deg(D) = 1 - g$ となり、
  また全ての D で、$l(D) - \deg(D) \geq 1 - g$ となる。

- 因子 D,B に関して
  $\deg(D - B) < 0$ ならば $l(D - B) = 0$

Plücker の公式より代数曲線の種数 g が計算でき、Riemann の定理より、ある因子 D に関する $L(D)$ の次元 $l(D)$ が計算できる。これらを用いて代数曲線 C、因子 $G = aP_0$、$B = P_1 + P_2 + \cdots + P_n$ としたときの Goppa 関数型符号 $C_L(C, B, G)$、Goppa 留数型符号 $C_\Omega(C, B, G)$ のパラメータは、表 1 のように表される。

次のことが言える。

- 種数 g を大きくすることで符号長 n はどこまでも大きくすることが可能である。

表 1: Goppa 関数型符号、留数型符号におけるパラメータ

	$C_L(C,B,G)$	$C_\Omega(C,B,G)$ $(2g-2<\deg(G))$
符号長 n	$n=\deg(B)$	$n=\deg(B)$
情報長 k	$k=l(G)-l(G-B)$	$k=n-a+g-1+l(G-B)$
最小距離 d	$d\geq n-a$	$d\geq a-(2g-2)$

- 情報長 k は符号長を限界に自由に設定できる。これは Riemann の定理より、$\deg(G)=a$ を変化させることで $L(G)$ を調整することで実現される。
- 最小距離 d は $n-k+1-g\leq d\leq n-k+1$ を満たす。

有限体を固定したときに、種数 g にたいして有理点を最も多く含む代数曲線を決定することにより、よい符号が構成される。

4 SV 復号アルゴリズム

代数幾何符号の復号アルゴリズムの一つとして SV(Skolobogatov-Vlădut) アルゴリズムがある。このアルゴリズムでは因子 B の各点 P に対して $A(P)=0$ となる補助因子 A をうまく利用する。しかし、このアルゴリズムが、t 誤りを訂正するための条件は、

$$\deg(A) < \deg(G)-(2g-2)-t \tag{5}$$

$$l(A) > t \tag{6}$$

である。この制約により設計距離 d で決まる能力より訂正能力が劣るものになる。また、アルゴリズム中で誤り位置をもとめる部分で一次従属な解をもとめる箇所が存在する。そのため解が一意に定まらない場合は誤り訂正に失敗する。現在はより優れたアルゴリズムが存在する。

受信語を f とする。f の補助因子は、関数型符号の場合は、留数型符号の符号語を構成する基底、留数型の場合は関数型符号の符号語を構成する基底となる。アルゴリズム中で因子 X,Y,A をとるが、一点代数曲線符号の場合同じ基底を用いることが可能となる。以下にそのアルゴリズムをのせる。

アルゴリズム 1 (SV 復号アルゴリズム)

- 因子 A の設定
  $l(A)>t$ かつ $d(A)<d(G)-(2g-2)-t$ を満たす A を設定
- 因子 X,Y の設定
  $X\leq A+Y\leq G$、$\deg(X)\geq\deg(A)+2g-1$、$deg(Y)\geq t+2g-1$ を満たす X、Y を設定 *(*例えば $X=G, Y=G-A$ となる。*)*

- 基底の選出
  $L(X):\{\varphi_1,\varphi_2,\cdots,\varphi_u\}$、$L(Y):\{\psi_1,\psi_2,\cdots,\psi_s\}$、$L(A):\{\chi_1,\chi_2,\cdots,\chi_r\}$ とする。
- シンドローム
  $S_{jk}(f)=\psi_j\chi_k f$(f は受信語) を計算。全て 0 となれば f は正しい。
- *error locator*
  $\sum_{j=1}^{s} S_{jk}(f)\alpha_j=0$ $(k\le r)$ となるような α をもとめて、$\theta=\sum\alpha_j\psi_j$ をもとめる。
- *error location*
  $\theta(P_l)=0$ となる点を探し、$P_l\in M$ とする。
- *error value*
  $\sum_{P_l\in M}\varphi_i(P_l)e_l=\varphi_i f_i$ $(i=1,\cdots,u)$ を解く。

5 Risa/Asir での実装

3、4 で述べた代数幾何符号、SV 復号アルゴリズムを Risa/Asir 上で実装した。具体的には、代数曲線 C として体 $K=GF(2^4)$ 上の x^3+y^3+1 を用いた。また、基底構成のためにある一点を選ぶ必要があるがそれには $(0,1)$ を用いた。

有理点は表 2 のようになる。ただし有理点を考慮する際に、射影平面も含めて考慮するので、斎次座標 $(x_0:x_1:x_2)$ 上で考え、$(x,y)=(\frac{x_0}{x_2},\frac{x_1}{x_2})$ とし, その他の平面での点を $(u,v)=(\frac{x_0}{x_1},\frac{x_2}{x_1})$,$(w,z)=(\frac{x_1}{x_0},\frac{x_2}{x_0})$ と対応させている。

表 2: $x^3+y^3+1(GF(2^4))$ における有理点　(α は x^4+x^3+1 の原始根)

$(x=0,y=1)$	$(x=1,y=0)$	$(u=1,v=0)$
$(x=0,y=\alpha^3+\alpha)$	$(x=0,y=\alpha^3+\alpha+1)$	$(x=\alpha^3+\alpha,y=0)$
$(x=\alpha^3+\alpha+1,y=0)$	$(u=\alpha^3+\alpha,v=0)$	$(u=\alpha^3+\alpha+1,v=0)$

表 2 で、点 P_0 を $(0,1)$ ととると、P_0 でのみ極となり、他の点では極とならないような関数体をとることができる。表 2 の P_0 以外の点を $P_1,P_2,\cdots,P_8$ とする。

このとき (7) のような関数体をとれば、今もとめたい関数体、つまり P_0 のみで極を持ち、$P_1,P_2,\cdots,P_8$ 以外で極を持たない関数体が得られる。

$$\varphi=\frac{x^iy^j}{(y+1)^{i+j}}\quad(i,j\ge 0)\tag{7}$$

因子 $B=P_1+P_2+\cdots+P_8$、$G=aP_0$ とし基底を構成できる。これらをもとに Goppa 関数型符号、Goppa 留数型符号の生成行列、検査行列を構成することができる。(7) より得られる関数体は表 3 のようになる。

表 3: $x^3+y^3+1, (0,1)$ における基底

関数体	位数	関数体	位数
1	0	$\frac{xy}{(y+1)^2}$	-5
$\frac{x}{y+1}$	-2	$\frac{x^3}{(y+1)^3}$	-6
$\frac{y}{y+1}$	-3	$\frac{x^2y}{(y+1)^3}$	-7
$\frac{x^2}{y+1}$	-4	$\vdots$	

具体的には入力値により因子 $G = aP_0$ の a を指定する。位数が a 以下の極となる関数体、つまり基底を表示し、基底に各点 $P_1, P_2, \cdots, P_8$ を代入し Goppa 関数型符号の生成行列を表示する。$a = 6$ のときの実行例を示す。例中の"@" は x^4+x^3+1 の原始根とする。

(例)Goppa 関数型符号 生成行列 / 留数型符号 検査行列 (位数 6)

```
[20] goppa_f(6);
Curve:x^3+y^3+1
Order:0
Select:[0,1]
XY:[[0,(@^3+@)],[0,(@^3+@+1)],[(1),0],[(@^3+@),0],[(@^3+@+1),0]]
UV:[[(1),0],[(@^3+@),0],[(@^3+@+1),0]]
WZ:[]
1
(x)/(y+1)
(y)/(y+1)
(x^2)/(y^2+2*y+1)
(y*x)/(y^2+2*y+1)
[ 1 1 1 1 1 1 1 1 ]
[ 0 0 (1) (@^3+@) (@^3+@+1) (1) (@^3+@+1) (@^3+@) ]
[ (@^3+@+1) (@^3+@) 0 0 0 (1) (1) (1) ]
[ 0 0 (1) (@^3+@+1) (@^3+@) (1) (@^3+@) (@^3+@+1) ]
[ 0 0 0 0 0 (1) (@^3+@+1) (@^3+@) ]
```

また、符号長 8、情報長 2、最小距離 6 の Goppa 留数型符号を対象に SV アルゴリズムを実装した。関数型符号の生成行列は留数型符号の検査行列であるので、線形符号の生成行列と検査行列の関係から、生成行列をつくりだす。

具体的には、もとめた関数型符号の生成行列を組織的な形式の行列に変換し、単位行列な部分と単位行列でない部分とに分ける (実行例参照。$a = 7$ としている)。この単位行列でない部分を取りだして新たに行列を構成する。SV アルゴリズムの実装プログラムに受信語 (符

号語+誤り) と、先程得た行列を与えてやると正しい語を得ることができる。以下に実行例 ($a = 7$ とした Goppa 留数型符号。表 1 より、符号長:7 情報長:2 最小距離:6 となる) を載せる。

生成行列より単位行列でない部分の取りだし

```
[1119] goppa_f(7);
Curve:x^3+y^3+1
Order:0
Select:[0,1]
XY:[[0,(@^3+@)],[0,(@^3+@+1)],[(1),0],[(@^3+@),0],[(@^3+@+1),0]]
UV:[[(1),0],[(@^3+@),0],[(@^3+@+1),0]]
WZ:[]
1
(x)/(y+1)
(y)/(y+1)
(x^2)/(y^2+2*y+1)
(y*x)/(y^2+2*y+1)
(x^3)/(y^3+3*y^2+3*y+1)
[ 1 1 1 1 1 1 1 1 ]
[ 0 0 (1) (@^3+@) (@^3+@+1) (1) (@^3+@+1) (@^3+@) ]
[ (@^3+@+1) (@^3+@) 0 0 0 (1) (1) (1) ]
[ 0 0 (1) (@^3+@+1) (@^3+@) (1) (@^3+@) (@^3+@+1) ]
[ 0 0 0 0 0 (1) (@^3+@+1) (@^3+@) ]
[ 0 0 (1) (1) (1) (1) (1) (1) ]
```

行列を組織的な形に変換

```
[1120] systematic_mat(@@,x^4+x^3+1);
[ 1 0 0 0 0 0 (@^3+@) (@^3+@+1) ]
[ 0 (1) 0 0 0 0 (@^3+@) (@^3+@+1) ]
[ 0 0 (1) 0 0 0 (@^3+@+1) (@^3+@) ]
[ 0 0 0 (1) 0 0 0 (1) ]
[ 0 0 0 0 (1) 0 (1) 0 ]
[ 0 0 0 0 0 (1) (@^3+@+1) (@^3+@) ]
```

SV 復号アルゴリズム (上記の例でもとめた部分行列を G とする)

```
[710] A=@^3+@;
(@^3+@)
```

```
[711] B=@^3+@+1;
(@^3+@+1)
[712] G=newmat(2,8,[[A,A,B,0,1,B,1,0],[B,B,A,1,0,A,0,1]]);
[ (@^3+@) (@^3+@) (@^3+@+1) 0 1 (@^3+@+1) 1 0 ]
[ (@^3+@+1) (@^3+@+1) (@^3+@) 1 0 (@^3+@) 0 1 ]
```

受信語 F の生成 C:符号語 E:誤り (F=C+E)

```
[713] I=newvect(2,[A,1]);
[ (@^3+@) 1 ]
[714] C=I*G;
[ 0 0 (@^3+@+1) 1 (@^3+@) (@^3+@+1) (@^3+@) 1 ]
[715] E=newvect(8,[0,1,@^2,0,0,0,0,0]);
[ 0 1 (@^2) 0 0 0 0 0 ]
[716] F=E+C;
[ 0 1 (@^3+@^2+@+1) 1 (@^3+@) (@^3+@+1) (@^3+@) 1 ]
```

受信語 F の誤り訂正

```
[719] sv_f(vtol(F),6);
Curve:x^3+y^3+1
Order:6
Select:  [0,1]
XY:  [[0,(@^3+@)],[0,(@^3+@+1)],[(1),0],[(@^3+@),0],[(@^3+@+1),0]]
UV:  [[(1),0],[(@^3+@),0],[(@^3+@+1),0]]
WZ:  []
FuncField:   [1,(x)/(y+1),(y)/(y+1),(x^2)/(y^2+2*y+1),(y*x)/(y^2+2*y+1)]
FuncOrder:   [0,-2,-3,-4,-5]
Syndrome
[ 1 (x)/(y+1) (y)/(y+1) ]
[ (x)/(y+1) (x^2)/(y^2+2*y+1) (y*x)/(y^2+2*y+1) ]
[ (y)/(y+1) (y*x)/(y^2+2*y+1) (y^2)/(y^2+2*y+1) ]
Syndrome
[ (@^2+1) (@^2) (@^3+@) ]
[ (@^2) (@^2) 0 ]
[ (@^3+@) 0 (@^3+@+1) ]
Alpha:    [ 1 1 (@^3+@+1) ]
ELXY: [[0,(@^3+@+1)],[(1),0]]
ELUV: [[(@^3+@+1),0]]
```

```
ELWZ: []
[ 1 0 0 (1) ]
[ 0 (1) 0 (@^2) ]
[ 0 0 (1) 0 ]
[ 0 0 0 0 ]
[ 0 0 0 0 ]
[0,1,(@^3+@^2+@+1),1,(@^3+@),(@^3+@+1),(@^3+@),1]
correct code
[ 0 0 (@^3+@+1) 1 (@^3+@) (@^3+@+1) (@^3+@) 1 ]
```

6 むすび

近年の誤り訂正符号研究の中心的位置を占めつつある、代数幾何符号について検討した。現在、広く用いられている符号である BCH 符号や RS 符号には、いくつかの問題点があることが指摘されている。前者では符号長を長くした場合に、符号化率、誤り訂正率の一方が極端に悪くなる。また、後者では符号長の制約といったことである。この点、代数幾何符号では、用いる代数曲線によって、符号長をどこまでも長くすることが可能であり、かつその性能も悪くならないような符号を構成することができる。このため、代数幾何符号は、従来の符号の持つ多くの問題点を克服することができる。今後は本稿を基礎として、他の曲線での代数幾何符号の実現や、他の符号化、復号化アルゴリズムとの比較、新しいアルゴリズムの提案などが課題となる。特に、本稿で触れることのできなかったグレブナ基底を用いた符号化、復号化のアルゴリズム等について研究を発展させていきたいと考えている。

なお、本稿は著者の愛媛大学工学部情報工学科における平成 13 年度卒業論文を整理したものである。卒業論文作成時に御指導頂いた愛媛大学工学部野田松太郎教授、甲斐博講師に感謝する。

参 考 文 献

[1] Cohen,A.M., Cuypers,H., Sterk,H., (Eds.), “Some Tapas of Computer Algebra”, Springer, 1998,
de Boer,M., Pellikaan,R., Chapter 10, “Gröbner Bases for Codes”;
de Boer,M., Pellikaan,R., Chapter 11, “Gröbner Bases for Decoding”.

[2] Cox,D., Little,J., O’Shea,D., “Using Algebraic Geometry”, Springer, 1998,
(邦訳; 大杉英史, 北村知徳, 日々孝之訳: “グレブナ基底 2”, シュプリンガー・フェアラーク東京, 2000.)

[3] 三浦晋示, “代数幾何符号の数理”, 電気情報通信学会論文誌 (A), vol.J82-A, no.8, pp.1223-1238, 1999.

[4] 水野弘文, “符号理論と代数幾何”, 数理科学, 1994 年 3 月号, pp.29-35.

[5] 岡本龍明, 太田和男, “暗号, 0 知識証明, 数論”;
岡本龍明, 櫻井幸一, 第 III 部, 第 5 章, “代数幾何的アルゴリズム”;
松本勉, 今井秀樹, 第 III 部, 第 6 章, “符号理論への応用”; 共立出版, 1995.

[6] Pretzel,O., “Codes and Algebraic Curves”, Oxford University Press, 1998.

[7] 内田興二, “有限体と符号理論”, サイエンス社, 2000.