横浜自主公開講座 2005/2/9(水) 市吉 修
誤り訂正技術 16(〃) 日本電気株式会社

5.誤り訂正

5.1.Reed Slomon符号

Reed Solomon符号は二進代数体の拡大Galois体GF(2^m)上で定義される巡回符号である。以下にRS符号につき極く簡略に説明する。

[1]誤り訂正の基礎

誤り訂正は誤り率の改善や所要信号電力の低減等に有効である。誤り訂正は上述の変調の様にData単位で独立に送信するのではなく、一群のdataを一括して符号化して送信する。その際元の情報Dataよりも多くの符号を送信する事になる。すなわち符号化により必要な伝送速度と伝送路の帯域幅が増大する。この事は必然的に受信雑音の増加を持たらす。それでも猶送信電力に対する誤り率特性が改善されるのは次の事情による。すなわち符号化はより広い信号空間から送信信号を選択するため実際に送信される符号語は相互にある距離を以て離れているように設定できる事である。受信部における信号検出は受信信号と各符号語の間の距離(類似度)を測定しその受信信号に最も近い距離にある符号語が送信されたものと判定する。符号語の間の距離の内最も小さいものをその符号の最小距離という。距離の近い符号程誤り易いのは当然なので誤り率特性は殆ど最小距離で決まる。

[2]Block符号

(n,k)Block符号はk bitsの情報dataを一つの塊(Vector)としてn-k bitsのParity符号を付加してn bitsのBlockとして送信し、誤り訂正復号も各Block毎に独立に行なう。誤り率特性の上で支配的なのは符号語間の距離の内最小となるものであり、これを最小距離と呼び、誤り訂正符号の訂正能力の重要な目安となる。二進符号においては通常Hamming距離が用いられる。二つの二進信号vectorの間のHamming距離とは相互に相異なるvector内の成分の数である。

[3]巡回符号の多項式表現

よく用いられるBlock符号は巡回符号である。巡回符号とはn元vectorの符号語を左右に巡回したものも符号語となるような符号であり、その利点は多項式による表現と処理が可能な事である。即ち符号語Cはn-1次の多項式で表わされる。

$$C(x) = Cn-1 \cdot X^{(n-1)} + Cn-2 \cdot X^{(n-2)} + \cdots + C1 \cdot X + C0 \qquad (5.1\text{-}1)$$

但しXは

$$X^n = 1 \qquad (5.1\text{-}2)$$

二進伝送の場合は多項式の各係数は0か1の値をとる二進代数の変数である。Xは二進代数と式(5.1-2)の演算規則に従う変数である。

[4]生成多項式による符号化

(n,k)巡回符号はn-k次の生成多項式G(x)により生成される。

$$G(x) = X^{(n-k)} + G(n-k-1) \cdot X^{(n-k-1)} + \cdots + G(1) \cdot X + 1 \qquad (5.1\text{-}3)$$

k bitsの情報Dataも同じく多項式表現により、

$$D(x) = D(K-1) \cdot X^{(K-1)} + D(K-2) \cdot X^{(K-2)} + \cdots + D(1) \cdot X + D(0) \qquad (5.1\text{-}4)$$

この時符号化はつぎの方法で実行される。

'05-1/13.

$$C(x) = X^{n-k} \cdot D(X) + R(X) \qquad (5.1\text{-}5)$$

ここでR(X)は上の第一項を生成多項式G(X)で割った余りであり、高々n-k-1次の多項式となる。符号語は具体的には次の形をとる。

$$C(x) = D(k-1) \cdot X^{n-1} + D(k-2) \cdot X^{n-2} + \cdots + D(0) \cdot X^{n-k} + R(n-k-1) \cdot X^{n-k-1} + R(n-k-2) \cdot X^{n-k-2} + \cdots R(1) \cdot X + R(0) \qquad (5.1\text{-}6)$$

上の式より符号語C（x）は生成多項式G（x）で割り切れる。即ちC(x)をG(x)で割った余りは0となる。

$$\mathrm{Rem}\{C(x)/G(X)\} = R(x) + R(x) = 0 \qquad (5.1\text{-}7)$$

このように巡回符号は生成多項式の倍多項式として生成される。

巡回符号の符号化は情報系列D(x)から$X^{n-k} \cdot D(X)/G(X)$の剰余R(X)を計算し、それを$X^{n-k} \cdot D(X)$に加えればよい。次数の高い項が先に来ると決めると多項式は時系列で表現できる。従って上の巡回符号の符号化は図5.1-1のShift register回路で実行できる。図において[+]は二進加算を[M]は1 Sample遅延を表わす。

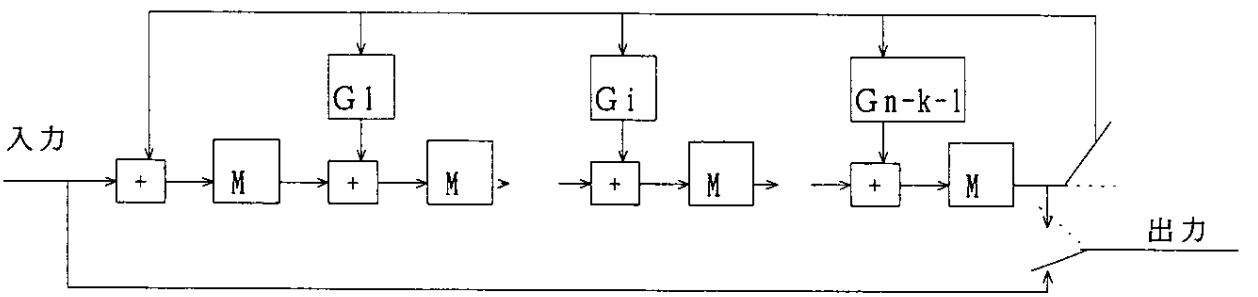


図2.2.1-1 巡回符号の符号化回路

[5]Syndromeの計算

受信信号は伝送路でBit誤りが加わり、次式で表わされる。

$$V(X) = C(X) + E(X) \qquad (5.1\text{-}8)$$

第一項は符号語であり生成多項式G(X)で割り切れる。受信Vectorを生成多項式G(X)で割って得られる剰余をSyndromeと言いS(X)で表わす。

$$S(X) = \mathrm{Rem}\{V(X)/G(X)\} = \mathrm{Rem}\{E(X)/G(X)\} \qquad (5.1\text{-}9)$$

伝送路誤りが無ければSyndromeは0となる。逆は必ずしも真では無いが、いずれにしろ符号語を受信したのであり正しく受信した確率が高い。Syndromeの計算は送信部と同じ生成多項式G(X)によって受信信号V(X)を割って剰余を求める事であり、符号化回路と同様の回路構成で実現できる。

[6]BCH符号

(6.1)二進拡大体

巡回符号としてはBCH符号が広く用いられている。Block長nの巡回符号とは二進代数体上で次の条件を満足する。

$$X^n + 1 = 0 \qquad (5.1\text{-}10)$$

これは次のように因式分解される。

$$X^n + 1 = (X+1)\{X^{n-1} + X^{n-2} + \cdots + X + 1\} \qquad (5.1\text{-}11)$$

YOS-2

更に数体を拡大する事により次の様に因式分解が可能になる。

$$X^n + 1 = (X+1)(X+\alpha)(X+\alpha^2)(x+\alpha^3).....(X+\alpha^{(n-1)}) \quad (5.1\text{-}12)$$

ここでαはこの拡大体の原始根(Primitive root)と呼ばれる。0と原始根によってこの拡大体のすべての要素が生成される。即ち、

$$0, 1=\alpha^0, \alpha, \alpha^2, \alpha^3, ... \alpha^{(n-1)} \quad (5.1\text{-}13)$$

$\alpha^n=1$となるのでこの拡大体にてn次の多項式は丁度n個の根を有する。

ここで若干の用語を定義する。

既約多項式 ....二進代数の範囲でそれ以上因数多項式分解できない多項式をいう。

最小多項式.....ある原始根を根とする最小次数の多項式。

最小多項式(Minimal Polynomial)は当然既約多項式である。

二進拡大体においては一般に多項式m(X)は

$$m(X)^2 = m(X^2) \quad (5.1\text{-}14)$$

であるのでαがm(X)の根であれば、$\alpha^2, \alpha^4, \alpha^8 ... \alpha^{2^i}$もまたm(X)の根である。

従ってX^n+1の因子多項式はm[1](X), m[3](X), m[5](X)....となる。

例えば15次の多項式の場合には次表のようになる。

表5.1-1 $X^{15} + 1$の因子多項式

最小多項式	根の指数
m[0](X)=X+1	0
m[1](X)=X^4+X+1	1,2,4,8
m[3](X)=X^4+X^3+X^2+X+1	3,6,12,9
m[5](X)=X^2+X+1	5,10
m[7](X)=X^4+X^3+1	7,14,13,11

(6.2)BCH限界

BCH符号の構成は次のBCH boundに関する定理に基づいている。

多項式G(X)で生成される符号の最小距離はG(X)の連続する根の最大数よりも大きい。

但しα^(i+1), α^(i+2),.....α^(i+j)をj個の連続する根と呼ぶ。

(6.3)原始符号

Block長； $n = 2^m - 1$ (5.1-15)

の符号を原始符号という。この場合生成多項式を

$$G(X)=LCM\{m[1](X) \cdot m[3](X) \cdot \cdot \cdot \cdot m[2t-1](X)\} \quad (5.1\text{-}16)$$

に選べば少なくとも2t個の根が連続するのでG(X)で生成される符号の最小距離dは$2t+1$以上である。また最小多項式の次数はmを越えない事が知られているので、G(X)の次数は最大mtである。まとめると

「符号長が$n=2^m-1$、Parity bit数がm・t以下で

最小距離dが少なくとも$2t+1$のBCH符号が存在する。」

例

表5.1-2　n＝１５次の場合の生成多項式と最小距離d

n	k	d	G(X)	G(X)の根
15	14	2	m[0](X)	(0)
15	11	3	m[1](X)	(1,2,4,8)
15	10	4	m[0](X)・m[1](X)	(0),(1,2,4,8)
15	7	5	m[1](X)・m[3](X)	(1,2,4,8),(3,6,12,9)
15	6	6	m[0](X)・m[1](X)・m[3](X)	(0),(1,2,4,8)(3,6,12,9)
15	5	7	m[1](X)・m[3](X)・m[5](X)	(1,2,4,8)(3,6,12,19)(5,10)
15	4	8	m[0](X)・m[1](X)・m[3](X)・m[5](X)	(0)(1,2,4,8)(3,6,12,19)(5,10)

[7] Reed-Solomon符号

上のBCH符号の理論は二進数体に限らず、一般の多進数体においても同様に成り立つ。よく用いられるのはq=2^mなるGalois拡大体上の記号を用いた符号長n=2^m -1の符号である。Galois拡大体を得るための原始多項式としては次のものが知られている。

m	Primitive Polynomial
3	1 + X + X^3
4	1 + X + X^4
5	1 + X^2 + X^5
6	1 + X + X^6
7	1 + X^3 + X^7
8	1 + X^2 + X^3 + X^4 + X^8
9	1+ X^4 + X^9

上述のBCH定理からt-symbols以下の誤りを訂正可能な次の(n,k)RS符号を用いる事ができる。

符号長(Symbols)	n = 2^m-1 = q -1
Parity symbol数	n-k=2t

二進数体GF(2)の場合に比べてGalois拡大体GF(q) (q=2^m)上の符号を用いるRS符号は以下の優れた特長がある。

(1)Parity symbolが少なくてすむので符号化効率が高く誤り訂正能力が強力である。

(2)各Symbolはm bitsの二進dataなのでBurst誤り訂正能力がある。

但し演算回路は二進回路に比べて多少複雑になるが最近では回路技術の進歩により大きな問題ではなくなった。

YOS-4

Reed-Solomon符号

市吉　1996年7月24日

1.Reed-Solomon符号とは

Digital通信においてよく使用されるReed-Solomon符号とは

-Galois体GF(2^m)において定義され、

-t重誤り訂正のためには

Block長　：　n= 2^m -1(Symbols)

Parity数　：　2t

で実現できる(n, n-2t)なる巡回符号である。

2.Galois拡大体GF(2^m)

GF (2)　-----> GF (2^m)

原始多項式 p(x)

GF(2)={0,1}

0+0=0　　0・0=0

0+1=1+0=1　　1・0=0・1=0

1+1=0 (2=0)　1・1=1

例

GF(2^4) ={0,1,α,$\alpha^{2,}\alpha$^3,....,α^14}

原始多項式　p(x) = x^4 + x +1

原始根　α; p(α)= 0

0	(0,0,0,0)
1	(0,0,0,1)
α	(0,0,1,0)
α^2	(0,1,0,0)
α^3	(1,0,0,0)
α^4 =α+1	(0,0,1,1)
α^5 =α^2 + α	(0,1,1,0)
α^6 =α^3 + α^2	(1,1,0,0)
α^7 =α^4 + α^3 = α^3 + α +1	(1,0,1,1)
α^8 =α^4 + α^2 + α= α^2 + 1	(0,1,0,1)
α^9 =α^3 + α	(1,0,1,0)
α^10=α^4 + α^2 = α^2 + α + 1	(0,1,1,1)
α^11=α^3 + α^2 + α	(1,1,1,0)
α^12=α^4 + α^3 + α^2 = α^3 + α^2 + α + 1	(1,1,1,1)
α^13=α^4 + α^3 + α^2 + α = α^3 + α^2 + 1	(1,1,0,1)
α^14=α^4 + α^3 + α = α^3 + 1	(1,0,0,1)
α^15=α^4 + α = 1	

積は上の表による。

和は多項式の項別二進和をとる(4元vectorの和と考えてもよい。)

RS-1/3　YOS-5

3.R-S符号の生成多項式

t重誤り訂正R-S符号の生成多項式はGF(2^m)にて2t個の連続する根を有する多項式を採用すればよい。例えば

$$g(x) = (x+1)\cdot(x+\alpha)\cdot\cdot\cdot(x+\alpha^{(2t-1)})$$

例1 一誤り訂正符号(15,13)

$$g(x) = (x+1)\cdot(x+\alpha)$$
$$= x^2 + \alpha^4\cdot x + \alpha$$
$$= x^2 + (0,0,1,1)\cdot x + (0,0,1,0)$$

例2 二重誤り訂正符号(15,11)

$$g(x) = (x+1)\cdot(x+\alpha)\cdot(x+\alpha^2)\cdot(x+\alpha^3)$$
$$= x^4 + \alpha^{12}\cdot x^3 + \alpha^4\cdot x^2 + x + \alpha^6$$

-- 以下にR-S符号の理論的根拠を概略する。--

4.GF(q)上のFourier変換

$[v] = (v[n-1], v[n-2], .., v[1], v[0])$

に対するFourier変換

$[V] = (V[n-1], V[n-2], ..., V[1], V[0])$

が次式により定義される。

$$V[k] = [i;0,n-1\Sigma \quad v[i]\cdot\omega^{(k\cdot i)}$$

逆変換

$$v[i] = [k;0,n-1\Sigma \quad V[k]\cdot\omega^{(-k\cdot i)}$$

ここで

n ; q-1の約数

ω; $\omega^n = 1$)ωは原始根

$\omega^{n'} \neq 1$ (for $n'<n$)

上の[v], [V]はnを周期とする周期系列(Cyclic sequence)となる。

$v[i]=v[i+n]$,,

5.Cyclic complexity

系列[v]のcomplexityがtであるとは;

$$v[i] = -[j;1,t\Sigma \quad \Lambda[j]\cdot v((i-j)) \qquad (i=t,t+1,,,n-1)$$

と表しうる最小の項数がtである。

但し $v((i-j)) = v((i-j)(\mathrm{mod}\ n))$

$\Lambda[0] = 1$とすると

$$[j;0,t\Sigma \quad \Lambda[j]\cdot v((i-j)) = 0 \quad (i=0,1,2,...,n-1)$$

即ち

$$\Lambda(x)\cdot v(x) = 0 \quad (\mathrm{mod}\ x^n - 1)$$

但し

$$\Lambda(x) = [j;0,t\Sigma \quad \Lambda[j]\cdot x^j$$
$$v(x) = [i;0,n-1\Sigma \quad v[i]\cdot x^i$$

6.重みとComplexity

系列[v]の重み(0でない要素v[i]の数)はそのFourier変換[V]のcomplexityに等しい。

[V]のcomplexityをtとするとt次の多項式 Λ(x)により

$$\Lambda(x)\cdot V(x) = 0 \pmod{x^n - 1}$$

となるようにできる。即ち Λ(k)とV(m)の畳み込みが0となる。

$$\sum_{k=0}^{t} \Lambda[k]\cdot v((i-k)) = 0 \quad (i=0,1,2,\ldots,n-1)$$

そこで{Λ(k),V(k):k=0,1,..,n-1}のFourier逆変換により得られる{λ(i), v(i)}については

$$\lambda(i)\cdot v(i) = 0 \quad (i=0,1,2,\ldots,n-1)$$

となる。

[v]の重みをwとすると上の式から少なくともw個のλ(i)が零となる。

ところが

$$\lambda(i) = \sum_{k=0}^{t} \Lambda[k]\cdot \omega^{(-k\cdot i)} = \Lambda(\omega^{(-i)})$$

であるのでΛ(x)は少なくともw個の零を持つ。->Λ(x)の次数はは少なくともwである。

7.R-S符号の最小距離

生成多項式 g(x): 2t個の連続した根を有する。

R-S符号c(x)は生成多項式の倍多項式である;

$$c(x) = c[n-1]\cdot x^{(n-1)} + \ldots + c[1]\cdot x + c[0] = Q(x)\cdot g(x)$$

ここで

$$C[k] = \sum_{i=0}^{n-1} c[i]\cdot \omega^{(k\cdot i)} = c(x) \quad (x=\omega^k) = 0 \text{ for } k= n-1,\ n-2,\ \ldots,\ n-2t$$

← 連続する2t個の零,

とすると

$$C(y) = \sum_{k=0}^{n-1} C[k]\cdot y^k = \sum_{k=0}^{n-2t-1} C[k]\cdot y^k$$

は次数が高々n-2t-1の多項式である。即ちC(y)の零の個数は高々n-2t-1である。

他方

$$c(i) = C(\omega^{(-i)})$$

であるので{c(i):i=0,1,2,..,n-1}のうち少なくとも2t+1個は0でない。

即ちR-S符号の最小距離は2t+1でありt重誤り訂正可能である。

Richard E. Blahut
Algebraic Methods for Signal Processing and Communication Coding,
Springer-Verlag

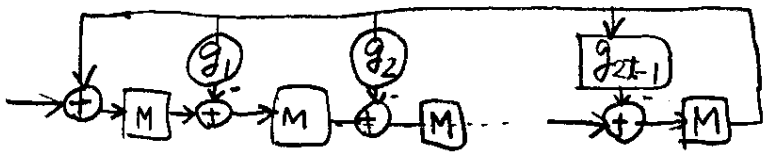
巡回符号の復号法

受信符号 $V(x)=\sum_{i=0}^{n-1} v_i x^i$, $v_i=c_i+e_i$ (有)

から

Syndrome を計算する。

〃 〃 計算回路は符号器と同じである



これにブロックデータを入力し終った時に M に残っているのが syndrome である。

誤りが無ければ all 0 となる。

$$S(x)=\sum_{k=0}^{2t-1} S_k x^k$$

この $S_k = E_k$ $(k=0,1,2,\cdots 2t-1)$

求めるには

$$S(x)=\mathrm{Rem}\{V(x)/g(x)\}$$

生成多項式 $g(x)=(x-1)(x-\omega)\cdots(x-\omega^{2t-1})$

$V(x)$ を $(x-\omega^k)$ で割った余りは

$$S_k=V(\omega^k)=C_k+E_k=E_k$$

⟹

8

Decoding of Cyclic Codes

Cyclic codes have been defined from a spectral point of view based on the Fourier transform. Toeplitz matrices are directly related to spectral analysis, so it will not be surprising that the algorithms for inverting Toeplitz systems of equations can be applied to the decoding of cyclic codes. This chapter will make this connection explicit by developing a number of decoding algorithms for Reed–Solomon codes.

8.1 Decoding of Reed–Solomon Codes

The spectral estimation problem that arises in the decoding of Reed–Solomon codes is the following: Find the vector $\mathbf{e}$ with the smallest number of nonzero components such that the Fourier transform $\mathbf{E}$ has $2t$ components E_k for $k = 0, \ldots, 2t-1$ that agree with the known values of these $2t$ components. This is the most likely transmitted codeword if the channel has a probability of symbol error smaller than one half. The procedure that we shall derive is predicated on the assumption that there is a solution $\mathbf{e}$ with at most t nonzero components because this is the error-correcting capability of the code.

The codeword $\mathbf{c}$ is transmitted and the channel makes errors described by the vector $\mathbf{e}$, which is nonzero in not more than t places. The received word $\mathbf{v}$ is written componentwise as

$$v_i = c_i + e_i, \qquad i = 0, \ldots, n-1.$$

The decoder must process the received word $\mathbf{v}$ so as to remove the error word $\mathbf{e}$; the data is then recovered from the corrected codeword $\mathbf{c}$. The received noisy codeword $\mathbf{v}$ has a Fourier transform

$$V_k = \sum_{i=0}^{n-1} \omega^{ik} v_i, \qquad k = 0, \ldots, n-1$$

with components $V_k = C_k + E_k$ for $k = 0, \ldots, n-1$. But, by construction of a Reed–Solomon code,

$$C_k = 0, \qquad k = 0, \ldots, 2t-1.$$

Hence

$$V_k = E_k, \qquad k = 0, \ldots, 2t-1.$$

The block of $2t$ components of $\mathbf{v}$ gives us a window through which we can look at $2t$ consecutive components of the n components of the transform of the error pattern. These $2t$ components of $\mathbf{E}$ are referred to as the *syndromes* of $\mathbf{E}$. The decoder must find all n components of $\mathbf{E}$ given a segment consisting of $2t$ consecutive components of $\mathbf{E}$ and the additional condition that at most t components of the time domain error pattern $\mathbf{e}$ are nonzero. Once $\mathbf{E}$ is known, the computation is trivial because $C_k = V_k - E_k$. From $\mathbf{C}$ one can compute $\mathbf{c}$. The data symbols are recovered easily in various ways depending on the method of encoding.

We will use the properties of the Fourier transform to find a procedure to compute $\mathbf{E}$. Because $\mathbf{e}$ has weight at most t, we can find a locator polynomial $\Lambda(x)$ of degree at most t and with $\Lambda_0 = 1$ such that

$$\Lambda(x)E(x) = 0 \qquad (\text{mod } x^n - 1),$$

where

$$E(x) = \sum_{k=0}^{n-1} E_k x^k.$$

We can rewrite the convolution as the equation of an autoregressive filter with t taps:

$$E_j = -\sum_{k=1}^{t} \Lambda_k E_{j-k}, \qquad j = 0, \ldots, n-1.$$

If the taps of the filter were known, then the remaining components of $\mathbf{E}$ can be found by shifting the autoregressive filter as described by

$$E_j = -\sum_{k=1}^{t} \Lambda_k E_{j-k}, \qquad j = 2t, \ldots, n-1.$$

But we know $2t$ components of $\mathbf{E}$. The t equations

$$E_j = -\sum_{k=1}^{t} \Lambda_k E_{j-k}, \qquad j = t, \ldots, 2t-1$$

involve only known components of $\mathbf{E}$. These t equations are linear in the unknown components Λ_k. Hence we can find $\boldsymbol{\Lambda}$ by solving this system of linear equations. This system of equations has come up repeatedly and we need only repeat some of the main ideas in the context of the decoding problem.

The locator polynomial $\Lambda(x)$ has the form

$$\Lambda(x) = \prod_{\ell=1}^{\nu} (1 - x\omega^{i_\ell}),$$

where i_ℓ for $\ell = 1, \ldots, \nu \le t$ are the locations of the nonzero e_i. The inverse Fourier transform

$$\lambda_i = \frac{1}{n} \sum_{k=0}^{n-1} \Lambda_k \omega^{-ik}$$

can be obtained from $\Lambda(x)$ by evaluating $\Lambda(x)$ at $x = \omega^{-i}$. That is,

$$\lambda_i = \frac{1}{n} \Lambda(\omega^{-i}).$$

Therefore

$$\lambda_i = \frac{1}{n} \prod_{\ell=1}^{\nu} (1 - \omega^{-i}\omega^{i_\ell}).$$

Hence $\lambda_i = 0$ if and only if $e_i \neq 0$. We conclude that, in the time domain, $\lambda_i e_i = 0$ for all i. Therefore, because a product in the time domain corresponds to a convolution in the frequency domain, we see that the convolution in the frequency domain is equal to zero.

$$\boldsymbol{\Lambda} * \mathbf{E} = \mathbf{0}.$$

Hence a polynomial $\Lambda(x)$ that solves the cyclic convolution $\Lambda(x)E(x)$ (mod $x^n - 1$) does exist; it is the error-locator polynomial.

To solve the t equations,

$$\sum_{k=1}^{t} \Lambda_k E_{j-k} = -E_j, \qquad j = t, \ldots, 2t-1$$

is the computational problem of inverting a Toeplitz system of equations and has been studied in Chapter 6. The Berlekamp–Massey algorithm discussed in Section 6.2 has priority within the literature of decoding Reed–Solomon codes and is regarded as the most efficient algorithm.

8.1.1 The Forney Algorithm

Generation of $\mathbf{E}$ by recursive extension using the $2t$ known components of $\mathbf{E}$ and the autoregressive filter with taps $\boldsymbol{\Lambda}$ requires computing $n - 2t$ outputs of the filter; each output requires up to t multiplications. An alternative procedure is the Forney algorithm.

The cyclic convolution of the error polynomial and the locator polynomial can be written in acyclic form as

$$\Lambda(x)E(x) = \Gamma(x) - x^n\Gamma(x),$$

where both terms on the right involve a polynomial $\Gamma(x)$ because the right side is zero modulo $x^n - 1$. Moreover, because of Theorem 3.1.1 we know that $\deg \Gamma(x) < t$.

手順

誤り数は t 以下であるとの前提の下に Syndrome より この連立方程式が立つ。未知数 $\{\Lambda_k\}$

↓

Λ_k を求める

↓

λ_i を求める

$\lambda_i = 0$ となる i 番目の $e_i \neq 0$ となる。

e_i を求めて 誤りを訂正する

The formal derivative of this equation is

$$\Lambda'(x)E(x) + \Lambda(x)E'(x) = \Gamma'(x) - nx^{n-1}\Gamma(x) - x^n\Gamma'(x).$$

Set $x = \omega^{-i}$, noting that $\omega^{-n} = 1$. Then

$$\Lambda'(\omega^{-i})E(\omega^{-i}) + \Lambda(\omega^{-i})E'(\omega^{-i}) = -n\omega^i\Gamma(\omega^{-i}).$$

But we know that $e_i = \frac{1}{n}E(\omega^{-i})$ is nonzero only if $\Lambda(\omega^{-i})$ is zero. Therefore the error pattern is given by

$$e_i = \begin{cases} 0, & \text{if } \Lambda(\omega^{-i}) = 0, \\ -\frac{\Gamma(\omega^{-i})}{\omega^{-i}\Lambda'(\omega^{-i})}, & \text{if } \Lambda(\omega^{-i}) \neq 0, \end{cases}$$

where $\Lambda'(x)$ is computed as the formal derivative of $\Lambda(x)$,

$$\Lambda'(x) = \sum_{k=0}^{t} k\Lambda_k x^{k-1},$$

and by Theorem 3.1.1, $\Gamma(x)$ is given by $\Lambda(x)E(x) \pmod{x^t}$. It is equivalent (and more convenient) to write this as

$$\Gamma(x) = \Lambda(x)E(x) \pmod{x^{2t}},$$

because the superfluous coefficients are all zero.

8.1.2 The Berlekamp Algorithm

Rather than compute $\Gamma(x)$ after the computation of $\Lambda(x)$ is complete, an iteration for $\Gamma(x)$ can be executed in parallel with each iteration of the Berlekamp–Massey algorithm. The concurrent iteration of $\Lambda(x)$ and $\Gamma(x)$ as in Figure 8.1 is called the *Berlekamp algorithm.*

□ **Theorem 8.1.1 (Berlekamp algorithm)** *If*

$$\begin{bmatrix} \Gamma^{(0)}(x) \\ A^{(0)}(x) \end{bmatrix} = \begin{bmatrix} 0 \\ -x^{-1} \end{bmatrix}$$

and for $r = 1, \ldots, 2t$

$$\begin{bmatrix} \Gamma^{(r)}(x) \\ A^{(r)}(x) \end{bmatrix} = \begin{bmatrix} 1 & -\Delta_r x \\ \delta_r\Delta_r^{-1} & (1-\delta_r)x \end{bmatrix} \begin{bmatrix} \Gamma^{(r-1)}(x) \\ A^{(r-1)}(x) \end{bmatrix},$$

then $\Gamma^{(2t)}(x) = \Gamma(x)$.

Proof Define the iterates

$$\begin{aligned} \Gamma^{(r)}(x) &= E(x)\Lambda^{(r)}(x), && \pmod{x^r}, \\ A^{(r)}(x) &= E(x)B^{(r)}(x) - x^{r-1}, && \pmod{x^r}, \end{aligned}$$

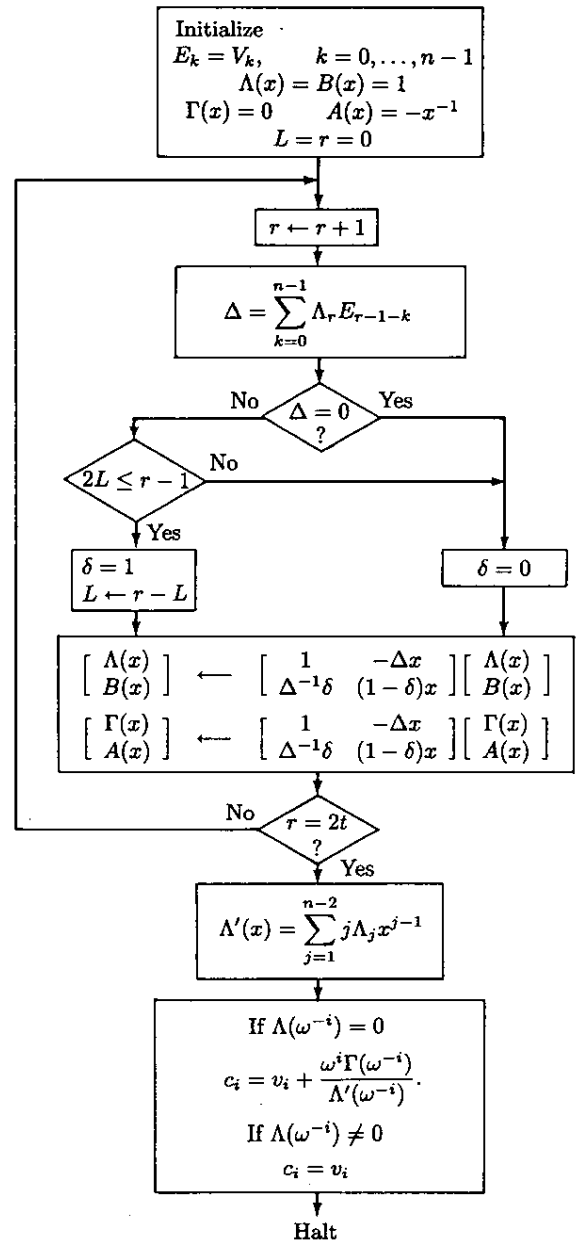


FIGURE 8.1. A decoder that uses the Berlekamp algorithm.

for $r = 1, \ldots, 2t$. Clearly, $\Gamma(x)$ is equal to $\Gamma^{(2t)}(x)$.

In particular, at iteration 1 these start out as

$$\begin{aligned} \Gamma^{(1)}(x) &= E(x)(1 - \Delta x), \qquad (\text{mod } x^1) \\ &= E_0 \end{aligned}$$

and

$$\begin{aligned} A^{(1)}(x) &= \begin{cases} E(x)x - x^0 & (\text{mod } x^1) \quad \text{if } E_0 = 0, \\ E(x)E_0^{-1} - x^0 & (\text{mod } x^1) \quad \text{if } E_0 \neq 0, \end{cases} \\ &= \begin{cases} -1 & \text{if } E_0 = 0, \\ 0 & \text{if } E_0 \neq 0. \end{cases} \end{aligned}$$

To verify the rth iteration, notice that because

$$\Gamma_k^{(r-1)} = \sum_{j=0}^{n-1} \Lambda_j^{(r-1)} E_{k-j}, \qquad k = 0, \ldots, r-2$$

and

$$\Delta_r = \sum_{j=0}^{n-1} \Lambda_j^{(r-1)} E_{r-1-j}$$

we have

$$\Gamma^{(r-1)}(x) = \Delta_r x^{r-1} = E(x)\Lambda^{(r-1)}(x) \qquad (\text{mod } x^r).$$

Using the iteration rule of the Berlekamp–Massey algorithm to expand the right side of

$$\begin{bmatrix} \Gamma^{(r)}(x) \\ A^{(r)}(x) \end{bmatrix} = \begin{bmatrix} E(x)\Lambda^{(r)}(x) \\ E(x)B^{(r)}(x) - x^{r-1} \end{bmatrix} \qquad (\text{mod } x^r)$$

leads to

$$\begin{aligned} \begin{bmatrix} \Gamma^{(r)}(x) \\ A^{(r)}(x) \end{bmatrix} &= \begin{bmatrix} 1 & -\Delta \\ \Delta^{-1}\delta & (1-\delta) \end{bmatrix} \begin{bmatrix} E(x)\Lambda^{(r-1)}(x) \\ xE(x)B^{(r-1)}(x) \end{bmatrix} \\ &\quad - \begin{bmatrix} 0 \\ x^{r-1} \end{bmatrix} \quad (\text{mod } x^r) \\ &= \begin{bmatrix} 1 & -\Delta \\ \Delta^{-1}\delta & (1-\delta) \end{bmatrix} \begin{bmatrix} \Gamma^{(r-1)}(x) + \Delta x^{r-1} \\ xA^{(r-1)}(x) + x^{r-1} \end{bmatrix} - \begin{bmatrix} 0 \\ x^{r-1} \end{bmatrix} \\ &= \begin{bmatrix} 1 & -\Delta \\ \Delta^{-1}\delta & (1-\delta) \end{bmatrix} \begin{bmatrix} \Gamma^{(r-1)}(x) \\ xA^{(r-1)}(x) \end{bmatrix}. \end{aligned}$$

This is the desired iteration. The first iteration gives

$$\begin{bmatrix} \Gamma^{(1)}(x) \\ A^{(1)}(x) \end{bmatrix} = \begin{bmatrix} 1 & -\Delta x \\ \delta\Delta^{-1} & (1-\delta)x \end{bmatrix} \begin{bmatrix} 0 \\ -x^{-1} \end{bmatrix},$$

11-5-91

which, because $E_0 = \Delta$, reduces to

$$\Gamma^{(0)}(x) = E_0$$

and

$$A^{(0)}(x) = \begin{cases} -1, & E_0 = 0, \\ 0, & E_0 \neq 0, \end{cases}$$

as is required for the first iteration. Thus the initialization provides the right start. □

8.2 Erasures and Errors Decoding

A received word on a channel that makes both errors and erasures consists of channel input symbols, some of which may be in error, and blanks denoting erasures. The decoder must correct the errors and fill the erasures. Any pattern of ν errors and ρ erasures can be decoded provided that

$$d_{\min} \geq 2\nu + 1 + \rho.$$

To decode a received word with ρ erasures, it is necessary to find a codeword that differs from the unerased portion of the received word in the fewest number of places provided the above inequality is satisfied. At most one such solution exists.

Decoding algorithms for Reed–Solomon codes can be extended to algorithms for decoding both erasures and errors. We treat only codes with the pattern of spectral zeros beginning at $k_0 = 0$.

Suppose that ρ erasures are made in locations $i_1, i_2, \ldots, i_\rho$. At these known locations, the received word v_i for $i = 0, \ldots, n-1$ has blanks, which we will initially fill with zeros. Define the erasure vector as that n-vector having component f_{i_ℓ} for $\ell = 1, \ldots, \rho$ equal to the erased symbol, and in other components $f_i = 0$. Then

$$v_i = c_i + e_i + f_i, \qquad i = 0, \ldots, n-1.$$

Define the modified received word as $v_i' = \psi_i v_i$, where ψ is any vector that is zero at every erased location and otherwise nonzero. Then

$$\begin{aligned} v_i' &= \psi_i(c_i + e_i + f_i) = \psi_i c_i + \psi_i e_i \\ &= c_i' + e_i', \end{aligned}$$

where the modified error vector is $e_i' = \psi_i e_i$ and the modified codeword is $c_i' = \psi_i c_i$. The modified error vector $\mathbf{e}'$ is nonzero in those components where $\mathbf{e}$ is nonzero.

The problem now is to decode $\mathbf{v}'$ to find $\mathbf{e}'$, which we know how to do if there are at least 2ν consecutive components where $C_k' = 0$.

5. まとめ

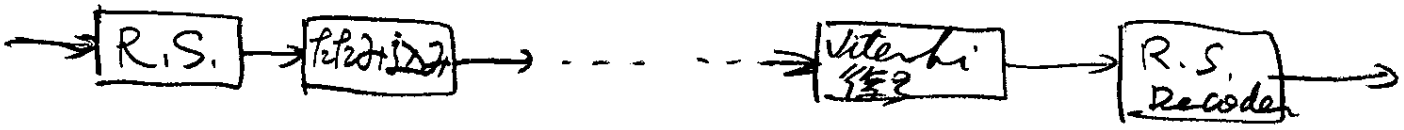
ＢＣＨ符号は二進演算だけで処理できるので１９８０年頃までの誤り訂正に広く用いられた。しかし符号化利得は高々３ｄＢであった。本講でも明らかなようにＢＣＨ符号は誤り訂正能力を上げると同時に符号化効率が低下してしまうので、符号化利得がなかなか上げられないのである。

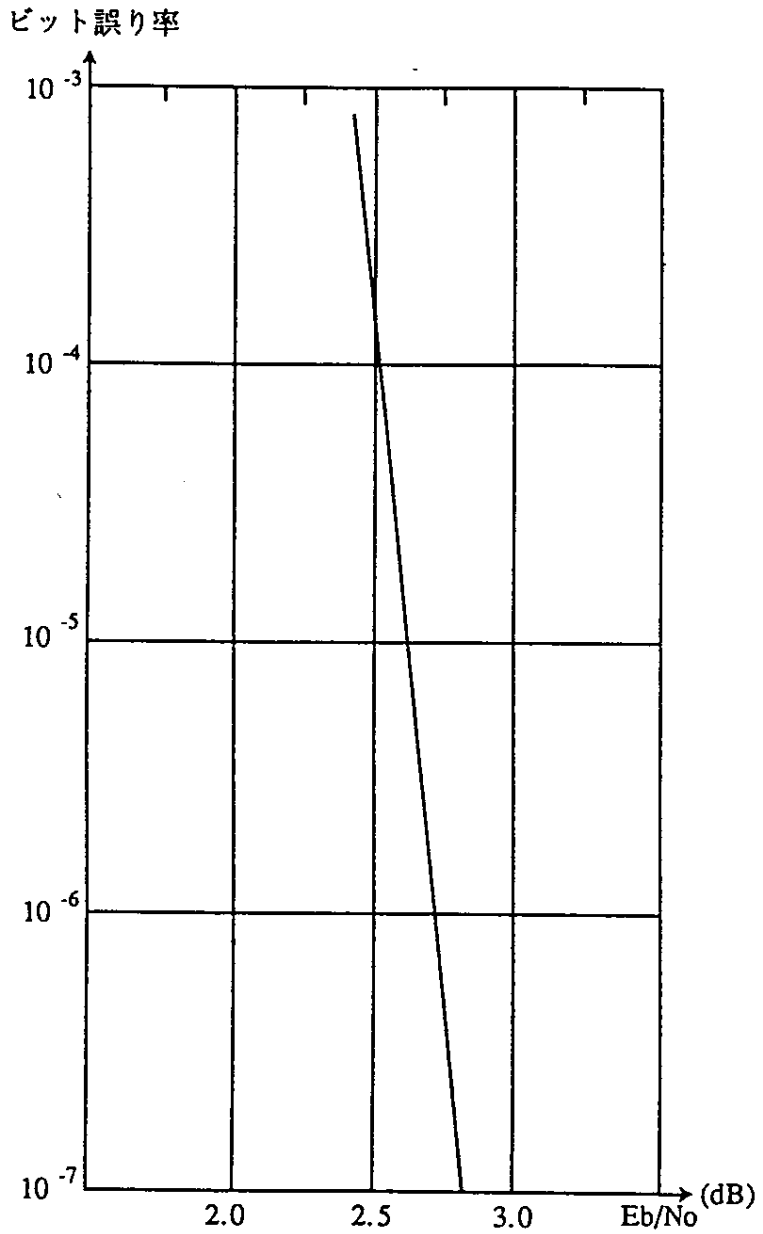
Reed-Solomon 符号はガロア拡大体 GF(2^m)上で(n,k,2t) (n=2^m-1)なるｔ重誤り訂正ブロック符号が構成できる。しかも GF(2^m)上の一語とはｍビットの二進符号でありブロック内で最大ｍｔビットのバースト誤り訂正能力がある。例えば衛星放送に広く用いられている(204,188)符号(m=8, n=2^m-1=255 であるが 51 ビットの情報は 0 であるとしてブロック長を短縮)においては t = (204-188)/2 = 8 重誤り訂正が可能、最大 mt = 64 ビット長のバースト誤りを訂正可能である。このようにＲＳ符号は高い符号化効率で強力な誤り訂正が可能であるがブロック長が大きく、また演算が GF(2^m)上で行われるためその実用化はメモリと演算素子の発展を待って始めて可能になった。

誤り訂正技術の別の流れは畳み込み符号であった。こちらも１９８０年頃までは二進演算に限定できる自己直交符号と閾値復号法が主として用いられた。例えばインテルサット SCPC 等電力制限のため極めて低いＣ／Ｎ条件で動作しなくてはならない衛星通信に広く採用されていた。Viterbi 復号法は１９６７年に発表されたが当初は二進演算の範囲（硬判定復調、符号間の距離は Hamming 距離）で処理がなされた。この信号処理方式では符号化利得の向上には拘束長Ｋを大きくする必要があるが、State 数が 2^K に比例して急激に大規模になるので自ずと限界があった。ディジタル信号処理（DSP）技術の発展により、１９８０年代に軟判定復調と組み合わせて Metric をユークリッド距離で処理することができるようになると共に Viterbi 復号法が急激に普及した。即ち３ビット程度の DSP でも一挙に 3dB もの符号化利得が追加で得られたのである。 発明者の A.J.Viterbi 本人が Linkabit(Qualcom の前身)という企業を起こして本格的に事業化を推進し、８０年代の後半には Viterbi 復号法が最尤度復号法（ＭＬＤ）として標準的な誤り訂正方式となり、通信だけでなく、Hard Disk 等情報機器にも応用分野が広がった。

Viterbi 復号法の復号後の残留誤りはバースト状になる。何故なら類似度(Metric)の計量動作において正しい経路から誤った経路に分岐（と合流）が起こると誤りが生じるが、分岐から合流までの区間は頻繁な誤りが続くからである。そこで前述のバースト誤り特性に優れたＲＳ符号と組み合わせれば有効である。これは二重符号、連接符号、Concatenated code と呼ばれ、直接衛星放送等に広く採用されている。特性例を次図に示す。

Viterbi 複合法は Feed forward 処理で遅延が小さい利点があるが、現在でも拘束長をＫ＞９にするのは回路規模が大きくなり、実現困難である。そこで大きなインターリーブで隔てられた拘束長の小さい畳み込み符号を二重に用いて Feesback 復号処理を行う Turbo 符号方式が繰り返し処理によってシャノン限界に迫る特性が得られ、３Ｇ移動通信や最近のディジタル映像放送（ＤＶＢ）等に採用されている。





畳み込み符号とビタビ復号の組合せとリード・ソロモン符号による2重符号化の特性例
(符号化率1/2，k＝7畳み込み符号：8値軟判定ビタビ復号(255, 223) 16誤り訂正リード・ソロモン符号)

図5.3 二重符号の誤り率特性例(文献[7])

Yos-13