畳込み符号

正会員 金子敏信†

1. ま え が き

誤り訂正符号には、ブロック符号と“畳込み符号”(Convolutional Code, または回帰符号 Recurrent Code)がある。今回は、畳込み符号を説明しよう。この符号も第1回の表1に示されるように、ブロック符号と同じく実用上有用な符号である。ここでは畳込み符号の表し方をまず紹介した後、符号を定義づける生成行列、検査行列を述べよう。畳込み符号は密接にその復号法と関わりを持っている。復号法としてはシンドロームパターン復号法、閾値復号法、最尤復号法としてビタビ復号法、逐次復号法を紹介しよう。

2. 畳込み符号の表し方

2.1 簡単な畳込み符号の例

図1に簡単な畳込み符号の符号器を示す。この符号は“ワイナー・アッシュ符号”(Wyner-Ash Code)と呼ばれる。情報ビット列 $i_0, i_1, i_2, \dots$ が1ビットずつ入り、右から2ビットずつ $(i_0, p_0), (i_1, p_1), (i_2, p_2), \dots$ の出力が出る。添え字は時刻を表す。時刻 t で情報 i_t が入り2ビットの出力 u_t, p_t が出る。なお、通信路には、 $i_0, p_0, i_1, p_1, i_2, p_2, \dots$ の順に送り出される。図の□は1単位時間の遅延素子、⊕はmod 2の加算回路である。図より

図1 ワイナー・アッシュ符号の符号器

$$p_t = i_t + i_{t-1} \quad (1)$$

である。 p_t は現在時刻の情報ビット i_t だけでなく、以前の情報ビット i_{t-1} も影響して定まる。

例えば、

$$\text{情報ビット列} = 1 \quad 1 \quad 0 \quad 1 \quad 0 \quad (2)$$

であれば

$$\text{符号器出力} = 11 \quad 10 \quad 01 \quad 11 \quad 01 \quad (3)$$

となる。

2.2 遅延演算子

(2)式の情報ビット列を D の多項式

$$\begin{aligned} I(D) &= 1 + 1D + 0D^2 + 1D^3 + 0D^4 + \dots \\ &= 1 + D + D^3 + \dots \end{aligned} \quad (4)$$

で表そう。 D^t の係数が時刻 t の情報ビットである。

(3)式の符号器出力も D を用いて

$$\begin{aligned} W(D) &= (11) + (10)D + (01)D^2 + (11)D^3 \\ &\quad + (01)D^4 + \dots \end{aligned} \quad (5)$$

と表せる。時刻 t の符号器出力

$$w_t = (i_t, p_t) \quad (6)$$

が D^t の係数である。

一般の畳込み符号の符号器では、各時刻に k ビットの情報が入り、 n ビットの出力が出る。時刻 $t (t = 0, 1, 2, \dots)$ において入力される k ビットの情報(情報ブロック)を

$$i_t = (i_{t1}, i_{t2}, \dots, i_{tk}) \quad (7)$$

とする。出力の n ビット(符号ブロック)を

$$w_t = (x_{t1}, x_{t2}, \dots, x_{tn}) \quad (8)$$

とする。この情報ブロックの系列 $i_0, i_1, i_2, \dots$ を情報系列と呼び、(4)式と同じく D の多項式として、

$$I(D) = i_0 + i_1 D + i_2 D^2 + \dots \quad (9)$$

と表そう。また、符号ブロックの系列 $w_0, w_1, w_2, \dots$ を符号系列と呼び

$$W(D) = w_0 + w_1 D + w_2 D^2 + \dots \quad (10)$$

と表す。 D^t の係数が時刻 t のブロックを表している。

この D を“遅延演算子”(delay operator)という。

ところで、情報系列 $I(D)$ に D を掛けると

† 東京理科大学 理工学部

“Introduction to Coding Theory (10). Convolutional Code” by Toshinobu Kaneko (Faculty of Engineering, Science University of Tokyo, Noda)

図 2(a) (n, k) ブロック符号

図 2(b) $R=k/n$ の畳込み符号

$$DI(D) = i_0D + i_1D^2 + i_2D^3 + \dots \quad (11)$$

D^t の係数は i_{t-1} である。 $I(D)$ の時刻 $t-1$ の情報ブロック i_{t-1} が (11) 式の $DI(D)$ で時刻 t のブロックである。情報系列 $I(D)$ を 1 単位時間遅らせた系列が $DI(D)$ となる。 D を掛けることで 1 単位時間の遅れを表わすので、 D を遅延演算子と呼ぶのである。

2.3 拘束長と符号化率

畳込み符号とブロック符号の違いを図 2 に示す。 (n, k) ブロック符号では k ビットの情報ブロックが n ビットの符号ブロックに符号化される。時刻 t の情報ブロック i_t のみで時刻 t の符号ブロック w_t が定まり、これを符号語と呼ぶ。畳込み符号でも、 k ビットの情報ブロック i_t が符号器に入力されると、 n ビットの符号ブロック w_t が出力される点は同じである。しかし、 w_t は現在の情報ブロック i_t のみで定まるのではなく、過去にわたる情報ブロック K 個が影響する。この K を“拘束長”(Constraint Length)という^(注)。この K 個の情報ブロックが時間と共に移動し、逐次的に符号系列が定まって行く。

畳込み符号でも、符号系列における情報ビットの割合 $R=k/n$ を符号化率と呼ぶ。 k/n は比としての意味だけでなく、慣例としてその分子と分母で情報ブロックと符号ブロックの長さを表す。ブロック符号では

(n, k) 符号というが、畳込み符号では、 $R=k/n$ の符号というのである。

図 1 の例は、 $R=1/2$ 、拘束長 2 の畳込み符号である。ここでは 2 元符号として畳込み符号を説明した。以下においても繁雑さを避けるため、説明は 2 元符号で行うが、一般には、 q 元符号であってよい。

3. 符号器と生成行列

3.1 生成行列

2.1 節の例を引き続き考えていこう。(5)式を整理すれば、次式となる。

$$W(D) = (1 + D + D^3 + \dots, 1 + D^2 + D^3 + D^4 + \dots) \quad (12)$$

この $W(D)$ は、各成分が D の多項式の 2 次元ベクトルである。(6)式より、この第 1 成分は i_t の系列 $I(D)$ 、第 2 成分は p_t の系列 $P(D)$

$$P(D) = p_0 + p_1D + p_2D^2 + \dots \quad (13)$$

である。(1)式と遅延演算子 D の意味を考えると

$$\begin{aligned} P(D) &= I(D) + DI(D) \\ &= (1 + D)I(D) \end{aligned} \quad (14)$$

となっている。したがって

$$\begin{aligned} W(D) &= (I(D), (1 + D)I(D)) \\ &= I(D)[1, 1 + D] \end{aligned} \quad (15)$$

となる。この式の D の多項式を要素とする 1 行 2 列の行列を $G(D)$

$$G(D) = [1, 1 + D] \quad (16)$$

とすれば、符号系列 $W(D)$ は次式である。

$$W(D) = I(D)G(D) \quad (17)$$

一般の $R=k/n$ の畳込み符号で説明しよう。(10)式の符号系列 $W(D)$ を整理すれば、次の n 次元ベクトルである。

$$W(D) = (W_1(D), W_2(D), \dots, W_n(D)) \quad (18)$$

ここで第 i 成分 $W_i(D)$ ($i=1, \dots, n$) は (8) 式の x_{0i} , x_{1i} , x_{2i} , ... を係数とする D の多項式

$$W_i(D) = x_{0i} + x_{1i}D + x_{2i}D^2 + \dots \quad (19)$$

である。情報系列 $I(D)$ は次の k 次元ベクトル

$$I(D) = (I_1(D), I_2(D), \dots, I_k(D)) \quad (20)$$

であり、その第 i 成分 $I_i(D)$ は (7) 式より

$$I_i(D) = i_{0i} + i_{1i}D + i_{2i}D^2 + \dots \quad (21)$$

である。この時、符号系列 $W(D)$ は次式となる。

$$W(D) = I(D)G(D) \quad (22)$$

この $G(D)$ は D の多項式を要素とする $k \times n$ 行列

$$G(D) = \begin{bmatrix} G_{11}(D) & G_{12}(D) & \dots & G_{1n}(D) \\ G_{21}(D) & G_{22}(D) & \dots & G_{2n}(D) \\ \vdots & \vdots & \ddots & \vdots \\ G_{k1}(D) & G_{k2}(D) & \dots & G_{kn}(D) \end{bmatrix} \quad (23)$$

(注) 拘束長には様々な定義があるので注意のこと。例えば、1 つの情報ブロックが影響を及ぼす符号系列の長さ Kn やその情報系列の長さ Kk を拘束長と呼ぶ場合もある。

であり、畳込み符号の“生成行列”(Generator Matrix)と呼ばれる。

3.2 基本生成行列

行列 $G(D)$ の要素 $G_{ij}(D)$ は任意ではなく、次の条件が満たされるように通常選ばれる。

基本生成行列の条件：

$G(D)$ の nC_k 個の $k \times k$ 小行列式 Δ_i で非零であるものの最大公約多項式が1である。

この条件を満足する $G(D)$ を“基本生成行列”(Basic Generator Matrix)という。証明は略すが、 $G(D)$ が基本生成行列でない場合、通信路における誤りが有限個であっても、いったん誤った訂正がおきると復号結果に波及し、無限個の誤りとして復号されてしまう時がある。このように誤りが波及する現象を“誤り伝搬”(Error Propagation)といい、無限長の誤り伝搬が生じ得る生成行列を“カタストロフィック(Catastrophic)な生成行列”という。

(例1) 次の生成行列はカタストロフィックな生成行列である。すなわち、基本生成行列でない。

$$\begin{aligned} G(D) &= [1+D, 1+D^2] \\ &= (1+D)[1, 1+D] \end{aligned} \quad (24)$$

3.3 組織符号

線形ブロック符号と同じように、生成行列が

$$G(D) = [I_k \quad Q(D)] \quad (25)$$

である畳込み符号を“組織符号”と呼び、符号ブロックの先頭 k ビットに情報ブロックが現れる。ここで、 I_k は k 次の単位行列、 $Q(D)$ は D の多項式を要素とする $k \times (n-k)$ 行列である。図1の例は(16)式のように組織符号となっている。

線形ブロック符号の場合、行列の基本行操作(第3回3.2節参照)によって常に組織符号とすることができた。畳込み符号においても、(23)式の生成行列に対し、行列の基本行操作を行っても、符号として同一である。しかし、結果として得られる生成行列の要素は D の多項式である必要があるため、常に組織符号に変形できるわけではない。

3.4 畳込み符号の符号器

次の生成行列 $G(D)$ を例に考えよう。

$$\begin{aligned} G(D) &= [G_{11}(D), G_{12}(D)] \\ &= [1+D^2, 1+D+D^2] \end{aligned} \quad (26)$$

符号系列 $W(D) = (W_1(D), W_2(D))$ は

$$\begin{aligned} W_1(D) &= I(D)G_{11}(D) \\ &= I(D)(1+D^2) \\ W_2(D) &= I(D)G_{12}(D) \\ &= I(D)(1+D+D^2) \end{aligned} \quad (27)$$

である。 D を掛けることは、1単位時間遅延した系

図3 畳込み符号の符号器例

図4 ボーゼンクラフト型符号器

列を考慮することであるから、(27)式の符号系列は情報系列 $I(D)$ を2単位時間まで遅延した系列の線形和となっている。 $G_{11}(D)$ の係数がその線形和回路の結線を表している。それゆえ(26)式の生成行列の畳込み符号の符号器は図3となる。

(23)式の生成行列 $G(D)$ に対して符号器を考よう。(22)式より符号系列 $W(D)$ の第 i 成分 $W_i(D)$ は

$$W_i(D) = \sum_{j=1}^k I_j(D)G_{ji}(D) \quad (28)$$

である。 $G_{ji}(D)$ の次数を m_{ji} とすれば、系列 $I_j(D)$ について m_{ji} 単位時間まで遅延したものについての線形和が $W_i(D)$ となっている。 $G_{ji}(D)$ が、その線形和回路の結線を表している。このような回路が $W_1(D)$ から $W_n(D)$ に対して作られる。もちろん系列 $I_j(D)$ を遅延する回路は共有し、線形和回路の結線のみ変えればよい。最大の遅延は多項式 $G_{ji}(D)$ の最大次数

$$m = \max_{j,i} [m_{ji}] \quad (29)$$

である。このように考えれば、(23)式の符号器は図4である。この図より拘束長 K は、

$$K = m + 1 \quad (30)$$

であることも明らかであろう。

図4のように遅延回路の出力を線形和回路に入れて符号系列を発生する符号器を“ボーゼンクラフト型符号器”(Wozencraft's Encoder)と呼ぶ。それに対し、 $R = (n-1)/n$ の組織符号では、情報系列の線形和を

図 5 $R=(n-1)/n$ の畳込み符号のマッシイ型符号器

遅延回路に入れる図5のような“マッシイ型符号器”(Massey's Encoder)で符号化することができる。

4. 検査行列とシンドローム系列

4.1 検査行列

線形ブロック符号と同じく、畳込み符号もパリティ検査行列を持つ。2.1節の符号の例で確認しよう。この符号の生成行列 $G(D)$ は(16)式である。次の行列

$$H(D)=[1+D, 1] \quad (31)$$

と $G(D)$ を掛ければ、mod 2 の計算で

$$G(D)H(D)^T=(1+D)+(1+D)=0 \quad (32)$$

となる。(31)式の $H(D)$ が、この符号の(パリティ)検査行列である。

(25)式の生成行列を持つ一般の畳込み符号であれば“検査行列” $H(D)$ は次の $(n-k) \times n$ 行列となる。

$$H(D)=[Q^T(D), I_{n-k}] \quad (33)$$

この $H(D)$ と符号系列 $W(D)$ の間には

$$W(D)H(D)^T=0 \quad (34)$$

が成り立つ。この式より2つの符号系列の和も同じ畳込み符号の符号系列となっていることがわかる。

4.2 シンドローム系列

符号系列 $w_0w_1w_2\cdots$ を通信路に送り出し、 $e_0e_1e_2\cdots$ の誤り系列が加わり、受信系列を $y_0y_1y_2\cdots$ とする。

$$y_t = w_t + e_t \quad (35)$$

これら系列を遅延演算子を使って表し、符号系列を(18)式の $W(D)$ 、誤り系列 $E(D)$ を

$$E(D)=(E_1(D), E_2(D), \cdots, E_n(D)) \quad (36)$$

受信系列 $Y(D)$ を

$$Y(D)=(Y_1(D), Y_2(D), \cdots, Y_n(D)) \quad (37)$$

としよう。 $Y(D)$ は

$$Y(D)=W(D)+E(D) \quad (38)$$

である。受信系列 $Y(D)$ を $H(D)$ に掛けて得られる $(n-k)$ 次元ベクトル $S(D)$

$$\begin{aligned} S(D) &= (S_1(D), S_2(D), \cdots, S_{n-k}(D)) \\ &= Y(D)H(D)^T \end{aligned} \quad (39)$$

を畳込み符号の“シンドローム系列”または“シンドローム”という。シンドローム系列を計算する回路を“シンドローム形成器”(Syndrome Former)と呼ぶ。

(34), (38)式を(39)式に代入すれば

$$S(D)=E(D)H(D)^T \quad (40)$$

となり、シンドロームは誤り系列のみで定まる。そこで線形ブロック符号と同じくシンドロームを用いて復号を行う方法が考えられる。

5. シンドロームパターン復号法

シンドロームを使った基礎的な復号法が“シンドロームパターン復号法”(Syndrome Pattern Decoding)である。これはシンドローム系列内の特定パターンを検出し、誤りを訂正する復号法である。

5.1 簡単な畳込み符号の復号法の例

図1のワイナー・アッシュ符号の復号器を図6に示す。シンドローム形成器、パターン検出回路、誤り訂正回路、シンドローム修正回路の4つの部分から成っている。受信系列 $Y(D)$

$$Y(D)=(Y_1(D), Y_2(D)) \quad (41)$$

の第1成分 $Y_1(D)$ が情報系列に対応しているから、これに生じた誤りを訂正する回路のみ示されている。

シンドローム形成器は(31)式の $H(D)$ と受信系列を掛け、シンドローム系列 $S(D)$ を計算する回路である。 $S(D)$ は図6のS点から出力される。パターン検出回路では系列 $S(D)$ 内の $\cdots 11 \cdots$ なるパターンを検出するとT点に1を出力する。

この符号は拘束長内の1ビットの誤りを訂正することができる。確かめてみよう。シンドロームは(39)式のように誤り系列 $E(D)$ のみで定まる。そこで符号系列として全0を送り $Y_1(D)$ に1ビットの誤りが生じているとする。これ以外のビットは充分な区間すべて誤りなしとする。復号器のすべての遅延素子には0が蓄えられている。誤りビットが復号器に入るとシン

図 6 ウィナー・アッシュ符号の復号器

ドローーム形成器内の遅延素子に入ると共に S 点に出力される。次の時点も誤りなしであるから、先程の遅延素子内の誤りビットが S 点に出力され、シンドロームとして…0110…なる系列が出力される。この 11 パターンでパターン検出回路の出力 T に 1 が発生し、誤り訂正回路にちょうど到着した $Y_1(D)$ の当該ビットが訂正される。T 点の 1 は同時に、シンドローム修正回路に供給され、この誤りビットのシンドロームへの影響が除去される。 $Y_2(D)$ の 1 ビットのみに誤りが生じた場合は T 点に 1 は出ずに $Y_1(D)$ がそのまま出力される。

誤りブロックの後ろ(拘束長-1)=1 ブロック内に誤りがなければ復号器内の遅延素子には 0 が蓄えられ、次の誤りに対処できることは、この回路の動作から明らかであろう。このようにして、誤りブロックから数えて拘束長 2 ブロック=4 ビット内に 1 ビットの誤りしかない系列であれば、すべて正しく訂正できる。誤りブロックを正しく復号するのに必要なその誤りに引き続く誤りのない区間を“ガードスペース”(Guard Space)という。

5.2 シンドロームパターン復号法を用いる符号

シンドロームパターンを検出して復号する符号として、ワイナー・アッシュ符号、“岩垂符号”(Iwadare Code)、“ハーゲルバーガー符号”(Hagelbarger Code)等がある。ワイナー・アッシュ符号は、拘束長内の単一誤りを訂正できるのに対し、後二者はバースト誤り訂正符号である。いずれも、訂正しようとする誤り系列がシンドローム系列において、非零で異なるパターンになるように構成されている。

〈ワイナー・アッシュ符号〉

この符号は次の $1 \times n$ の検査行列を持つ。

$$H(D)=[H_1(D), H_2(D), \dots, H_n(D)] \quad (42)$$

ここで m は次式を満たす整数とし、

$$2^m \geq n \quad (43)$$

$H_1(D), \dots, H_n(D)$ は定数項が 1 の m 次以下の D の多項式であり、すべて異なるものを選ぶ。組織符号とするためには $H_n(D)=1$ である。この符号は拘束長 $K=m+1$ で符号化率 $R=(n-1)/n$ の畳込み符号であり、拘束長内の 1 ビットの誤りを訂正できる。

(例 2) $m=2$ で $n=4$ とする。拘束長 3、符号化率 3/4 のワイナー・アッシュ符号の検査行列は

$$H(D)=[1+D+D^2, 1+D^2, 1+D, 1] \quad (44)$$

である。

〈岩垂符号〉

この符号は、 $H_i(D)$ を

$$H_i(D)=\begin{cases} D^{d_i}(1+D^{n-i}), & i=1, \dots, n-1 \\ 1 & i=n \end{cases} \quad (45)$$

として、検査行列が

$$H(D)=[H_1(D), H_2(D), \dots, H_n(D)] \quad (46)$$

で与えられる $R=(n-1)/n$ の畳込み符号である。ここで

$$d_i=3n-2-2i+(n-1-i)(n-2-i)/2 \quad (47)$$

この符号で、長さ n までのバースト誤りが訂正できる。また、ガードスペースの長さは

$$g=3n^2-2n-1+n(n-1)(n-2)/2 \quad (48)$$

ビットである。

ハーゲルバーガー符号符号もバースト誤り訂正の畳込み符号であるが、岩垂符号の方が回路は簡単であり、ガードスペースの面でも優っている。

6. 閾値復号法

シンドローム系列を使う別の復号法として、“閾値復号法”(threshold decoding)がある。これはブロック符号の多数決論理復号法と同じ原理であり、畳込み符号の多数決論理復号法である。

6.1 閾値復号法の簡単な例

この復号法を簡単な例で示そう。検査行列が

$$H(D)=[1+D^2+D^5+D^6, 1] \quad (49)$$

で与えられる $R=1/2$ の畳込み符号を考えよう。この符号は組織符号であり、符号ブロックの第 1 成分が情報ビットである。誤り系列 $E(D)$

$$E(D)=(E_1(D), E_2(D)) \quad (50)$$

に対してシンドローム系列は、

$$\begin{aligned} S(D) &= Y(D)H(D)^T = E(D)H(D)^T \\ &= (1+D^2+D^5+D^6)E_1(D)+E_2(D) \end{aligned} \quad (51)$$

である。誤り系列の各成分を

$$\begin{aligned} E_i(D) &= e_{0i} + e_{1i}D + e_{2i}D^2 + \dots \\ i &= 1, 2 \end{aligned} \quad (52)$$

とする。これを(51)式に代入してシンドローム系列 $S(D)$ の 1, D^2, D^5, D^6 の係数 s_0, s_2, s_5, s_6 を求めると次のようになる。

$$\begin{aligned} s_0 &= e_{01} + e_{02} \\ s_2 &= e_{01} + e_{21} + e_{22} \\ s_5 &= e_{01} + e_{31} + e_{51} + e_{52} \\ s_6 &= e_{01} + e_{11} + e_{41} + e_{61} + e_{62} \end{aligned} \quad (53)$$

これらのシンドロームビットを見ると e_{01} はすべてに含まれているが、他は一度しか出てこない。すなわち、これらのシンドロームは e_{01} に対して直交するパリティ検査和である(第 8 回 6.1 節参照)。復号器では、この 4 個のシンドロームを調べ、過半数(3 個以上)が 1 となっている場合は第 0 ブロックの 1 ビット

目 y_{01} が誤っていると判断する。

この場合、(53)式に現れている誤りビットのうち $e_{ji}=1$ となっている(すなわち誤っている)ビットの数が2個以下ならば e_{01} は正しく推定できる。受信ブロックで考えれば $y_0=(y_{01}, y_{02})$ から $y_6=(y_{61}, y_{62})$ までの拘束長7ブロック=14ビット内の2個以下の誤りであれば y_{01} を正しく復号できることになる。このように y_{01} の復号を行ったあと、それが誤っていた場合には関係するシンδροームを修正し、 y_{11} 以降についても同様の手順で復号を行う。復号器として示せば、図7である。

誤りの推定があるビットで誤ったとき、それ以降の復号動作に悪影響を及ぼすことがある。この現象は誤り伝搬である。生成行列がカタストロフィックでなくても、復号法によっては誤りの影響が無限に続く場合もある。しかし、シンδροームの修正が常に1を減らす方向に行われるならば、誤りが伝搬する長さは有限であり、ここにあげた例ではそのようになっている。

6.2 閾値復号法を用いる符号

前節のように、直交するパリティ検査和を使って多数決論理復号を行うのが、閾値復号法である。直交するパリティ検査和が幾つ取れるかで訂正できる誤りの個数が決まる。復号器は簡単になるが符号の構造に制約ができるため、符号の効率は良くない。

閾値復号ができる符号として、“自己直交符号”(Self-orthogonal Code)、“拡散符号”(Defuse Code)、“一様符号”(Uniform Codes)等がある。前節の例は自己直交符号であり、これについて以下で述べる。拡散符号はランダム誤りとバースト誤りを閾値復号法で訂正する符号である。一様符号は $R=1/2^m$ と低い符号化率のものであるが、高い誤り訂正能力を持つ符号である。

＜自己直交符号＞

シンδροーム系列のいくつかのビットがそのまま直交パリティ検査和となっている符号を自己直交符号という。自己直交符号では、誤り伝搬が有限になる。

図7 $R=1/2$ の自己直交符号の符号器

$R=1/2$ の自己直交符号は次の検査行列をもつ。

$$H(D)=[H_1(D), 1] \quad (54)$$

ここで、

$$H_1(D)=D^{d_1}+D^{d_2}+\dots+D^{d_J} \quad (55)$$

ただし、 $d_1=0<d_2<\dots<d_J$ であり、整数の集合 $\{d_1, d_2, \dots, d_J\}$ について、 $J \geq j > i \geq 1$ なる i, j に対して $d_i - d_j$ がすべて異なるものとする。

このように $H_1(D)$ を選ぶと $s_{d_1}, s_{d_2}, \dots, s_{d_J}$ が e_{01} に対し直交する J 個のパリティ検査和となる。拘束長は d_J+1 である。この検査和を使って閾値復号を行えば、拘束長の区間の「 $J/2$ 」個以下の誤りは訂正ができる。拘束長に対し誤り訂正能力を高くするには、 J が大きく、 d_j が小さくなるような整数の集合 $\{d_1, d_2, \dots, d_J\}$ を選ぶ必要がある。なお、この集合を使って、一般に $R=(n-1)/n$ の自己直交符号が構成できる。

7. 最尤復号法

ブロック符号の場合、最尤復号は一般には難しいが(第2回5.1節)畳込み符号の場合、最尤復号法またはそれに近い復号が比較的簡単に行える。その方法として“ビタビ復号法”(Viterbi Decoding)がある。その復号器の複雑さは拘束長 K に関し指数関数的に増大するので、小さい K の符号に適用できる。 K が大きい場合は、最尤復号を近似する方法として“逐次復号法”(Sequential Decoding)がある。

ここでは、例を使ってこれら復号法の原理を説明しよう。まず準備として、畳込み符号の表現法を2つ紹介する。

7.1 樹枝状符号表現とトレリス表現

例として、生成行列が

$$G(D)=[1+D^2, 1+D+D^2] \quad (56)$$

で与えられる $R=1/2$ の畳込み符号を考える。この符号の符号器は図3である。この符号器には2段の遅延素子があり、左から情報系列が入ってくる。この遅延素子の状態(記憶内容)を復号器の状態という。遅延素子が2段であるから状態は00, 10, 01, 11の4通り

である。これらの状態をそれぞれ $S_{00}, S_{10}, S_{01}, S_{11}$ と呼ぼう。この符号器の状態が、入力情報ビットによりどのように変化するかを表したのが、図8の“樹枝状符号表現”(Tree Code Representation)である。なお、最初の状態は00としてある。図において、各枝(Branch)に書かれている00, ..., 11が各状態における出力(符号ブロック) $w=(x_1, x_2)$ を、()内が状態を表す。2つに枝別れ

図 8 樹枝状符号表現例

している点で上の枝は入力情報ビットが 0 であることを示し、下の枝は 1 であることを示している。畳込み符号の任意の符号系列はこの枝の連なりで表せる。これを“パス”(Path)という。例えば、情報系列(0110...)に対応する符号系列(00 11 10 10 ...)は図の太線で示したパスである。

この樹枝状表現で 3 番目の枝以降は同じ状態が重複して出ている。これを整理し、入力 0 によって生ずる枝を実線で、1 によって生ずる枝を点線で表し、その時点における入力に対応する符号器出力を枝に付せば図 9 となる。これを“トレリス表現”(Trellis Representation)という。なお図中、最上段は時刻を表わす。

任意の畳込み符号を、この樹枝状符号表現やトレリス表現で書くことができる。 $R=k/n$ の符号であれば、符号器には k ビットずつ情報が入力されるので、 2^k 本に枝別れる樹枝状符号表現やトレリス表現となる。また、遅延素子数が M ならば、符号器の状態数は 2^M 個である。

＜符号系列の終結＞

畳込み符号を実際に用いる場合、情報ビット列は有限であり、符号系列はどこかで終了する。最後の情報ブロックの符号化終了と同時に符号系列を打ち切ると、終わりに近い情報ブロックが影響をもつ符号ブロックを無視することになり、この情報ブロックは誤って復号され易くなる。そのため、通常、拘束長 K に対し最後の情報ブロックの後に $K-1$ ブロックの全 0 のダミーの情報系列を入力して系列を打ち切る。すな

図 9 トレリス表現の例

わち、符号器の内部状態を全部 0 として終了する。これを、符号系列を終結する (Terminate) という。樹枝符号表現やトレリス表現では、終結前の $K-1$ 時点は符号器入力がすべて 0 であり、各状態から出る枝の数は 1 本である。もちろん、いつ終結するかは、送受信側双方で打ち合わせしておく。ダミーの情報系列が入る分、実効的な符号化率は下がるので、終結を行う間隔は充分長くとる。

7.2 ビタビ復号法

終結させた N ブロックの系列として畳込み符号を考え最尤復号を行うことは、符号長 Nn ビットの線形ブロック符号を最尤復号することと原理的に同一である。ここで、通信路が 2 元対称通信路としよう。この場合、最尤復号は受信系列 $y_0, y_1, y_2, \dots, y_{N-1}$ に最もハミング距離の近い符号系列 $w_0, w_1, w_2, \dots, w_{N-1}$ を推定することである(第 2 回 5.2 節)。ビタビ復号法では、この最も近い符号系列をトレリス線図を用いて効率的に探索する。具体的にはトレリス図上で次のように行う。

＜ビタビ復号の例＞

前節の符号について図 10 にビタビ復号法による復号例を示す。ここでは説明のため $N=7$ ブロックで終結するものとし、情報系列(0 1 1 0 1 0 0)を符号化している。系列の最後の 00 は終結するためのビット列である。符号系列は(00 11 10 10 00 01 11)である。これに通信路で誤り系列(00 01 10 00 00 10 00)が加わり受信系列(00 10 00 10 00 11 11)が受信されたとする。復号器は $N=7$ ブロックで終結することを知っているから 5、6 番目の情報ブロックが 0 0 で状態 S_{00} に終結するトレリス表現から受信系列に最も近いパスを探せばよい。

このトレリス表現では時刻 3 の各状態で 2 つのパスが合流している。まず、状態 S_{00} に合流する 2 つのパ

図 10 ビタビ復号法による復号例

スの符号系列(00 00 00)と(11 01 11)を受信系列(00 10 00)と比較する。前者との間のハミング距離は1, 後者との間は5である。そこで、距離の小さい方のみ生き残りパスとして残し、大きいパスは除去する。このようにしてもパス(00 00 00 … …)より(11 01 11 … …)が受信系列に近くなることはありえないから差し支えない。なお、時刻3の状態 S_{00} の○の中に生き残りパスのハミング距離1を書き込んでおく。時刻3で他の状態 S_{10} , S_{01} , S_{11} について合流する2つのパスについても同様の操作を行い、生き残りパスを残す。なお、距離が等しい場合はどちらを残してもよい。図中の△印は除去したパスを表す。

次に時刻4で状態 S_{00} に合流する2つの生き残りパス(00 00 00 00)と(00 11 01 11)について、受信系列(00 10 00 10)との間のハミング距離を比較し、距離の小さい方を生き残りとする。その際、時刻0にさかのぼって距離を求める必要はない。例えば、前者と受信系列の距離を求めるには、次のようにする。時刻3までの生き残りパス(00 00 00)と受信系列(00 10 00)の間の距離は、時刻3の状態 S_{00} の○内に求めている。この1と、時刻4で S_{00} に入るパスのブロック(00)と受信ブロック(10)のハミング距離1を加算し $1+1=2$ と計算することができる。

時刻4の状態すべてに対し上のように生き残りパスを1つずつ残す。以下同様に繰り返す。最後まで生き残り、時刻7の終結状態 S_{00} に達したパスが受信系列に最も近い符号系列である。図ではこれを太線で表しており、このパスに対応する情報系列(0 1 1 0 1 0 0)が復号系列である。

パスが長くなると、終結時刻まで待たないで、生き

残りパスの先頭の方は確定することが多い。この例では時刻3で生き残っているパスは(00 xx xx)であり(11 xx xx)は生き残っていない。この場合、時刻0の符号ブロックは00と確定できる。このように確定した部分(に対応する情報系列)は復号結果として出力してしまってもよい。通常、パスが拘束長の5～6倍程度の長さになるまでに確定することが多い。

ビタビ復号法は以上のような方法である。ここでは生き残りパスを選択するのにハミング距離を用いたが、選択の基準はこれに限るものではない。アナログの受信系列が利用できるならば、ユークリッド距離を基準とし、この距離の小さいパスを選択することにより畳込み符号の軟判定復号を行うことができ、復号誤り率を改善できる。

7.3 ビタビ復号法に適した符号

最尤復号法に用いる符号の良さは、正しく復号できる確率 P_c (または復号誤り率 $P_e=1-P_c$) で評価される。この確率 P_c は符号の最小距離が大きいほど高くなる。最小距離は異なる2つの符号系列の間のハミング距離の最小値である。畳込み符号の線形性から、これは非零の符号系列のハミング重みの最小値となる。この値を畳込み符号の“最小自由距離”(Minimum Free Distance) d_{free} という。

(例3)

図9のトレリス表現の畳込み符号を考える。状態 S_{00} から始まり S_{00} に戻ってくるパスの中で全0以外の系列のハミング重みの最小値が最小自由距離である。この場合、パス(11 01 11)が一番ハミング重みが小さい。この符号の $d_{free}=5$ である。

ビタビ復号法では、各時刻の各状態で合流するパス

を選択し、状態数 2^M 個だけの生き残りパスを残す。このパスを復号器内のバッファメモリに記憶しておかねばならない。符号器の遅延素子数 M は符号化率 R が同一であれば、ほぼ拘束長 K に比例する。したがって、復号器の複雑さは拘束長と共に指数関数的に増加するので、ビタビ復号法が適用できるのは K が 10 程度までである。

7.4 逐次復号法

一般に符号化率 R が同一であれば拘束長 K が長いほど復号誤り率 P_e は小さくなる。ビタビ復号法は拘束長が長い場合、実際的ではない。そこで、最尤復号法を近似し、その装置化を簡単にする方法として、逐次復号法がある。

逐次復号法でも受信系列に近い符号系列を探していく。その際に、樹枝状符号表現を使用する。逐次復号法である“ファノ・アルゴリズム”(Fano Algorithm)では、すべてのパスを調べるのではなく、受信系列に近いと思われる 1 本のパスを取りあえず先に延ばして行き、間違ったパスを辿っていると判断されたら、パスを逆戻りして別のパスを調べ直す。受信系列に近いパスであるか、間違ったパスを辿っているかの判断を“ファノ評価量”(Fano Metric)で行う。このようにして逐次復号法ではパスを探す手間を少なくしている。

この復号法では、パスを先に延ばす・逆戻りする、の行きつ戻りつがあるので、復号に要する時間が変動する。そのため、一定の時間間隔で復号結果を出力するために充分大きなバッファ(拘束長の 100 倍以上)を用意する。

8. む す び

畳込み符号に関し、主要事項を列挙しておこう。

- 畳込み符号では情報系列を符号系列に符号化する。この対応は逐次的であり、ブロック符号の符号語に相当する区切りはない。
- $R=k/n$ の畳込み符号：情報系列を情報ブロック k ビットごとに符号化し、符号系列中の符号ブロック n ビットとなる。各符号ブロックは、過去にわたる K 個の情報ブロックから定まる。 K を拘束長という。
- 生成行列 $G(D)$ ：畳込み符号を特徴づける行列。情報系列 $I(D)$ と符号系列 $W(D)$ の関係を $W(D)=I(D)G(D)$ で与える $k \times n$ 行列。検査行列 $H(D)$ ： $G(D)$ とある関係にあり、符号系列

$W(D)$ は $W(D)H(D)^T=0$ を満たす。

- シンドロームパターン復号法：受信系列 $Y(D)$ からシンドローム系列 $S(D)=Y(D)H(D)^T$ を求め、 $S(D)$ 内の特定のパターンを検出して誤りを訂正する復号法。
- 閾値復号法：直交するパリティ検査和を用いて多数決論理により誤りを推定する。復号器は簡単となるが、符号の効率は良くない。
- ビタビ復号法：畳込み符号の最尤復号が効率良く行える方法。符号のトレリス表現を使用。拘束長 K の短い符号に適用される。
- 逐次復号法：樹枝状符号表現を使用した、近似的な最尤復号法。拘束長の長い符号にも適用できる。

第 2 回から今回までの講座に関し、その機会を与えて頂いた、横山克哉前編集長、江藤良純編集理事ならびに校閲し貴重など意見を頂いた編集幹事の方々に対し深謝します。さらに、紙面構成等でお世話頂いた学会事務局の沖宗恵子さんに感謝します。

〔参 考 文 献〕

- 第 2 回の代表的教科書・参考書中、畳込み符号を取り扱っているものは 1)～9) 11) 14) 15) である。
- 56) A. J. Viterbi & J. K. Omura: “Principles of Digital Communication and Coding”, McGraw-Hill (1979)
 - 57) J. M. Wozencraft & B. Reiffen: “Sequential Decoding”, MIT Press (1961)
 - 58) J. L. Massey: “Threshold Decoding”, MIT Press (1963)
 - 59) A. D. Wyner & R. B. Ash: “Analysis of Recurrent Codes”, IEEE Trans., **IT-9**, pp. 143-156 (1963)
 - 60) Y. Iwadare: “On Type B1 Burst Error Correcting Convolutional Codes”, IEEE Trans., **IT-14**, pp. 557-583 (1968)
 - 61) D. W. Hagelbarger: “Recurrent Codes: Easily Mechanized, Burst-Correcting, Binary Codes”, B. S. T. J., **38**, pp. 969-984 (1959)
 - 62) R. M. Fano: “A Heuristic Discussion of Probabilistic Decoding”, IEEE Trans., **IT-9**, pp. 64-74 (1963)
 - 63) P. Piret: “Convolutional Codes — An Algebraic Approach—”, MIT Press (1988)
 - 64) 安田, 平田: “畳込み符号の最尤復号とその特性”, 信学論, **J73-A**, pp. 218-224 (1990)
 - 65) 鈴木, 田島: “畳込み符号に対する最尤復号器の実現”, 信学論, **J73-A**, pp. 225-231 (1990)

かねこ しのぶ
金子 敏信 昭和 51 年、東京大学大学院博士課程修了。同年、東京理科大学講師、53 年、同助教授となり、現在、同大学理工学部電気工学科に勤務。符号理論、通信方式、情報セキュリティの研究に従事。工学博士。正会員。