符号理論･暗号理論

- No.8 巡回符号 -

渡辺　裕

Coding Theory / Cryptography

- No.8 Cyclic Code -

Hiroshi Watanabe

通信路符号化

- 通信路において発生する伝送誤りに対処するための符号化
- 冗長性を持たせて誤り訂正能力を与える
- 代数学を基礎とする
- 代表的な誤り訂正符号
  - パリティ検査符号
  - ハミング符号
  - 巡回符号
  - BCH符号
  - RS符号

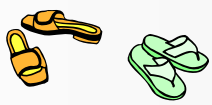
Channel Coding

- Coding to cope with transmission error at the channel
- Error correction ability given by additional redundancy
- Based on algebra
- Typical error correction code
  - Parity check code
  - Hamming code
  - Cyclic code
  - BCH code
  - RS code

パリティ検査符号

- 水平垂直パリティ符号
  - 12個のバイナリデータを3x4の配列に並べる
  - 水平，垂直の方向に “1” の個数が偶数になるように，パリティビットを付加する
  - 垂直パリティ符号に対する水平パリティ符号も加える
  - データ個数は 4x5=20 個に増大 (20/12=1.66…)

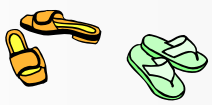
0	1	1	1	1
1	0	1	0	0
1	0	0	1	0
0	1	0	0	1



Parity Check Code

- Horizontal, Vertical Parity Code
  - Array 12 binary data to 3x4 matrix
  - Add parity bit to have even “1”s in row and column
  - Add horizontal parity bit to vertical parity bits
  - Increased number of data 4x5=20 (20/12=1.66…)

0	1	1	1	1
1	0	1	0	0
1	0	0	1	0
0	1	0	0	1



パリティ検査符号 (2)

- 誤り訂正能力
  - データおよびパリティ符号を受信して，“1”の和が偶数にならないとき，その行あるいは列のデータに誤りがあることがわかる
  - 1ビットの訂正能力
  - 誤りビットの断定は，水平および垂直のパリティ検査に異常がある行および列からわかる

0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
0	1	0	0	1

パリティ異常

Parity Check Code (2)

- Error Correction Ability
  - We know the existence of error when we do not have even numbers of “1” in row and column
  - 1 bit error correction
  - Determination of error bit can be turned out from horizontal and vertical parity check

0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
0	1	0	0	1

Parity error

ハミング距離

- バイナリ符号がどれくらい異なるかを定義
  - 例
    - 001100 が 001001 に誤った場合，符号が異なっているビットが2ヶ所あるので，距離を2と定義する
- 長さ n のバイナリ符号 X,Y のハミング距離

$$d(X,Y)=\sum_{i=1}^{n}(x_i \oplus y_i)$$
$$X = x_1 x_2 \cdots x_n$$
$$Y = y_1 y_2 \cdots y_n$$

Hamming Distance

- Define how binary codes are different
  - example
    - Distance equals to “2” when code “001100” is changed to “001001” since there are two locations where bits are different
- Hamming distance of binary code X,Y with length n

$$d(X,Y)=\sum_{i=1}^{n}(x_i \oplus y_i)$$
$$X = x_1 x_2 \cdots x_n$$
$$Y = y_1 y_2 \cdots y_n$$

ハミング距離 (2)

- 演算子は排他的論理和
  - 一致しているときには 0，不一致のときには 1

$$0 \oplus 0 = 0$$
$$0 \oplus 1 = 1$$
$$1 \oplus 0 = 1$$
$$1 \oplus 1 = 0$$

Hamming Distance (2)

- Operator is Exclusive or
  - Defined as 0 in accordance, otherwise 1

$$0 \oplus 0 = 0$$
$$0 \oplus 1 = 1$$
$$1 \oplus 0 = 1$$
$$1 \oplus 1 = 0$$

ハミング距離 (3)

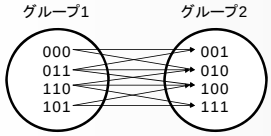
- ハミング距離を利用した誤り検査符号の構成1
  - 3ビット符号
    - 000, 001, 010, 011, 100, 101, 110, 111
  - お互いにハミング距離が2となるようにグループ化
    - グループ1: 000, 011, 101, 110
    - グループ2: 001, 010, 100, 111

Hamming Distance (3)

- Construction of error check code using Hamming distance
  - 3 bit code
    - 000, 001, 010, 011, 100, 101, 110, 111
  - Grouping so as to have Hamming distance=2 each other
    - Group 1: 000, 011, 101, 110
    - Group 2: 001, 010, 100, 111

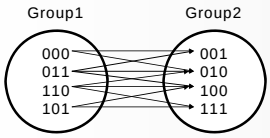
ハミング距離 (4)

- ハミング距離を利用した誤り検査符号の構成1(続き)
  - グループ1に1ビット誤りがあると，グループ2に移るため，誤りの検出が可能
    - グループ1は1の個数が偶数個(偶数パリティに相当)
    - グループ2は1の個数が奇数個(奇数パリティに相当)



Hamming Distance (4)

- Construction of error check code using Hamming distance(cntd.)
  - 1 bit error in Group 1 results to move to Group 2. Thus, error can be detected
    - Group 1 has even number of “1” (even parity)
    - Group 2 has odd number of “1” (odd parity)



ハミング距離 (5)

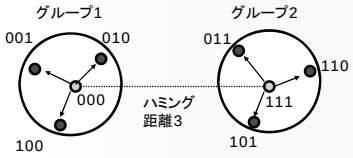
- ハミング距離を利用した誤り検査訂正の構成2
  - ハミング距離が3となる2つの3ビット符号
    - 000 と 111
  - 1ビット誤りが加わった場合のグループ分け
    - 000が正しく受信できた場合および，1ビット誤りが加わった場合の符号をグループ1とする
      - 000, 001, 010, 100
    - 111が正しく受信できた場合および，1ビット誤りが加わった場合の符号をグループ2とする
      - 111, 110, 101, 011

Hamming Distance (5)

- Construction of error correction code using Hamming distance
  - 3 bit code with Hamming distance 3
    - 000 and 111
  - Grouping for additional 1 bit error
    - 000 can be received correctly and 1 bit error is added (Group 1)
      - 000, 001, 010, 100
    - 111 can be received correctly and 1 bit error is added (Group 2)
      - 111, 110, 101, 011

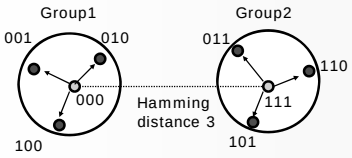
ハミング距離 (6)

– “1” の個数が1個であれば “000”, 2個であれば “111” に訂正できる



Hamming Distance (6)

– When the number of “1” is one, we can correct received code to “000”, if two, it should be “111”



誤り訂正能力

- ハミング距離の最小値

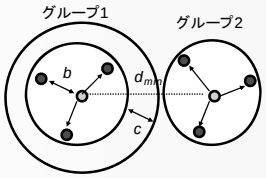
$$d_{min} = \min d(x_i, y_i)$$

- 誤りが b ビットの場合のグループ分離(誤り訂正)の条件

$$d_{min} > 2b$$

- 誤りが $b+1$ビット以上, $b+c$ビット以下の場合の誤り検出の条件

$$d_{min} > 2b + c$$



Error Correction Ability

- Minimum Distance

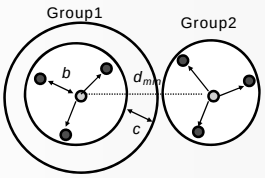
$$d_{min} = \min d(x_i, y_i)$$

- Condition for Grouping (error correction) when errors are b bits

$$d_{min} > 2b$$

- Condition for error detection when errors are more than $b+1$ and less than $b+c$ bits

$$d_{min} > 2b + c$$



(7,4)ハミング符号

- 4個の情報ビット x_1, x_2, x_3, x_4 に対して検査ビット c_1, c_2, c_3 を作成

$$\begin{aligned} c_1 &= x_1 + x_2 + x_3 \\ c_2 &= \quad x_2 + x_3 + x_4 \\ c_3 &= x_1 + x_2 \quad + x_4 \end{aligned}$$

- 符号語の構成

$$w = (x_1, x_2, x_3, x_4, c_1, c_2, c_3)$$

(7,4)Hamming Code

- Create parity bits c_1, c_2, c_3 for 4 data bits x_1, x_2, x_3, x_4

$$\begin{aligned} c_1 &= x_1 + x_2 + x_3 \\ c_2 &= \quad x_2 + x_3 + x_4 \\ c_3 &= x_1 + x_2 \quad + x_4 \end{aligned}$$

- Structure of codeword

$$w = (x_1, x_2, x_3, x_4, c_1, c_2, c_3)$$

(7,4)ハミング符号 (2)

- 符号語

x_1	x_2	x_3	x_4	c_1	c_2	c_3
0	0	0	0	0	0	0
0	0	0	1	0	1	1
0	0	1	0	1	1	0
0	0	1	1	1	0	1
0	1	0	0	1	1	1
0	1	0	1	1	0	0
0	1	1	0	0	0	1
0	1	1	1	0	1	0

x_1	x_2	x_3	x_4	c_1	c_2	c_3
1	0	0	0	1	0	1
1	0	0	1	1	1	0
1	0	1	0	0	1	1
1	0	1	1	0	0	0
1	1	0	0	0	1	0
1	1	0	1	0	0	1
1	1	1	0	1	0	0
1	1	1	1	1	1	1

(7,4)Hamming Code (2)

- Codeword

x_1	x_2	x_3	x_4	c_1	c_2	c_3
0	0	0	0	0	0	0
0	0	0	1	0	1	1
0	0	1	0	1	1	0
0	0	1	1	1	0	1
0	1	0	0	1	1	1
0	1	0	1	1	0	0
0	1	1	0	0	0	1
0	1	1	1	0	1	0

x_1	x_2	x_3	x_4	c_1	c_2	c_3
1	0	0	0	1	0	1
1	0	0	1	1	1	0
1	0	1	0	0	1	1
1	0	1	1	0	0	0
1	1	0	0	0	1	0
1	1	0	1	0	0	1
1	1	1	0	1	0	0
1	1	1	1	1	1	1

(7,4)ハミング符号 (3)

- パリティ検査方程式

$$\begin{aligned} x_1 + x_2 + x_3 + c_1 &= 0 \\ x_2 + x_3 + x_4 + c_2 &= 0 \\ x_1 + x_2 + x_4 + c_3 &= 0 \end{aligned}$$

- 受信語 $y=(y_1, \ldots, y_7)$ に対するシンドローム

$$\begin{aligned} s_1 &= y_1 + y_2 + y_3 + y_5 \\ s_2 &= y_2 + y_3 + y_4 + y_6 \\ s_3 &= y_1 + y_2 + y_4 + y_7 \end{aligned}$$

シンドローム:パリティ検査方程式に受信語を代入した結果

(7,4)Hamming Code (3)

- Parity Check Equation

$$\begin{aligned} x_1 + x_2 + x_3 + c_1 &= 0 \\ x_2 + x_3 + x_4 + c_2 &= 0 \\ x_1 + x_2 + x_4 + c_3 &= 0 \end{aligned}$$

- Syndrome to received data $y=(y_1, \ldots, y_7)$

$$\begin{aligned} s_1 &= y_1 + y_2 + y_3 + y_5 \\ s_2 &= y_2 + y_3 + y_4 + y_6 \\ s_3 &= y_1 + y_2 + y_4 + y_7 \end{aligned}$$

Syndrome: result to substitute received data to parity check equation

(7,4)ハミング符号 (4)

- シンドロームは誤りパターン $e=(e_1, \ldots, e_7)$ で決まる

$$\begin{aligned} s_1 &= e_1 + e_2 + e_3 + e_5 \\ s_2 &= e_2 + e_3 + e_4 + e_6 \\ s_3 &= e_1 + e_2 + e_4 + e_7 \end{aligned}$$

(7,4)Hamming Code (4)

- Syndrome is determined by error pattern $e=(e_1, \ldots, e_7)$

$$\begin{aligned} s_1 &= e_1 + e_2 + e_3 + e_5 \\ s_2 &= e_2 + e_3 + e_4 + e_6 \\ s_3 &= e_1 + e_2 + e_4 + e_7 \end{aligned}$$

(7,4)ハミング符号 (5)

- 単一誤りに対するシンドローム

誤りパターン							シンドローム		
e_1	e_2	e_3	e_4	e_5	e_6	e_7	s_1	s_2	s_3
1	0	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	1	1	1
0	0	1	0	0	0	0	1	1	0
0	0	0	1	0	0	0	0	1	1
0	0	0	0	1	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0
0	0	0	0	0	0	1	0	0	1
0	0	0	0	0	0	0	0	0	0

(7,4)Hamming Code (5)

- Syndrome to one bit error

Error pattern							Syndrome		
e_1	e_2	e_3	e_4	e_5	e_6	e_7	s_1	s_2	s_3
1	0	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	1	1	1
0	0	1	0	0	0	0	1	1	0
0	0	0	1	0	0	0	0	1	1
0	0	0	0	1	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0
0	0	0	0	0	0	1	0	0	1
0	0	0	0	0	0	0	0	0	0

一般化ハミング符号

- 符号の行列表現

$$\boldsymbol{w} = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \\ w_7 \end{bmatrix} = \begin{bmatrix} 1 & & & 0 \\ & 1 & & \\ & & 1 & \\ 0 & & & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \equiv \boldsymbol{Gx}$$

Generalized Hamming Code

- Matrix representation for code

$$\boldsymbol{w} = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \\ w_7 \end{bmatrix} = \begin{bmatrix} 1 & & & 0 \\ & 1 & & \\ & & 1 & \\ 0 & & & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \equiv \boldsymbol{Gx}$$

一般化ハミング符号 (2)

- パリティ検査方程式の行列表現

$$\boldsymbol{H} \equiv \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{Hw} = \boldsymbol{0}$$

- シンドロームの行列表現

$$\boldsymbol{s} = \boldsymbol{Hy} = \boldsymbol{H}(\boldsymbol{w} + \boldsymbol{e}) = \boldsymbol{Hw} + \boldsymbol{He} = \boldsymbol{He}$$

Generalized Hamming Code (2)

- Matrix representation for Parity check equation

$$\boldsymbol{H} \equiv \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{Hw} = \boldsymbol{0}$$

- Syndrome by the matrix

$$\boldsymbol{s} = \boldsymbol{Hy} = \boldsymbol{H}(\boldsymbol{w} + \boldsymbol{e}) = \boldsymbol{Hw} + \boldsymbol{He} = \boldsymbol{He}$$

一般化ハミング符号 (3)

- パリティ検査行列の一般化
  - 符号長: $n=2^m-1$
  - 検査ビット長: m
  - 情報ビット数: $k=n-m$

$$H=\begin{bmatrix} 1 & \cdots & 0 & 1 & 1 & & & \boldsymbol{0} \\ 0 & \cdots & 1 & 1 & & 1 & & \\ 1 & \cdots & 1 & 0 & & & 1 & \\ 1 & \cdots & 0 & 0 & \boldsymbol{0} & & & 1 \end{bmatrix}$$

Generalized Hamming Code (3)

- Parity check matrix for general case
  - Code length: $n=2^m-1$
  - Parity bits: m
  - Information bits: $k=n-m$

$$H=\begin{bmatrix} 1 & \cdots & 0 & 1 & 1 & & & \boldsymbol{0} \\ 0 & \cdots & 1 & 1 & & 1 & & \\ 1 & \cdots & 1 & 0 & & & 1 & \\ 1 & \cdots & 0 & 0 & \boldsymbol{0} & & & 1 \end{bmatrix}$$

巡回符号

- 巡回符号とは、符号を1ビットずつシフトして構成できる符号
- BCH符号や巡回ハミング符号など
- シフトレジスタと加算器により簡単に構成できる
- 誤り検査も簡単な回路で構成できる

Cyclic Code

- Cyclic code can be constructed by shifting every 1 bit of the original code
- Typical codes are BCH code and Humming code
- Easy construction by shift register and adder
- error check can also be realized by simple circuit

巡回符号の例

- 情報ビットを3bit, 検査ビットを4bitに設定した巡回符号の例

通報	符号						
	情報ビット			検査ビット			
	x_1	x_2	x_3	c_1	c_2	c_3	c_4
u_1	0	0	0	0	0	0	0
u_2	0	0	1	0	1	1	1
u_3	0	1	0	1	1	1	0
u_4	0	1	1	1	0	0	1
u_5	1	0	0	1	0	1	1
u_6	1	0	1	1	1	0	0
u_7	1	1	0	0	1	0	1
u_8	1	1	1	0	0	1	0

Example of Cyclic Code

- Information bit: 3bit, check bit: 4bit

message	code						
	Information bit			check bit			
	x_1	x_2	x_3	c_1	c_2	c_3	c_4
u_1	0	0	0	0	0	0	0
u_2	0	0	1	0	1	1	1
u_3	0	1	0	1	1	1	0
u_4	0	1	1	1	0	0	1
u_5	1	0	0	1	0	1	1
u_6	1	0	1	1	1	0	0
u_7	1	1	0	0	1	0	1
u_8	1	1	1	0	0	1	0

巡回符号の例 (2)

- 自然2進3ビット情報 (x_1, x_2, x_3) に対するパリティ4ビット (c_1, c_2, c_3, c_4) の作成

$$
\begin{aligned}
c_1 &= x_1 + x_2 \\
c_2 &= \quad x_2 + x_3 \\
c_3 &= x_1 + x_2 + x_3 \\
c_4 &= x_1 \quad + x_3
\end{aligned}
$$

Example of Cyclic Code (2)

- Calculation of 4 bit Parity (c_1, c_2, c_3, c_4) for 3 bit natural binary information (x_1, x_2, x_3)

$$
\begin{aligned}
c_1 &= x_1 + x_2 \\
c_2 &= \quad x_2 + x_3 \\
c_3 &= x_1 + x_2 + x_3 \\
c_4 &= x_1 \quad + x_3
\end{aligned}
$$

巡回符号の例 (3)

- パリティ検査方程式

$$
\begin{aligned}
w_1 + w_2 \quad + w_4 \quad\quad &= 0 \\
\quad w_2 + w_3 \quad + w_5 \quad &= 0 \\
w_1 + w_2 + w_3 \quad + w_6 \quad &= 0 \\
w_1 \quad + w_3 \quad\quad + w_7 &= 0
\end{aligned}
$$

Example of Cyclic Code (3)

- Parity check equation

$$
\begin{aligned}
w_1 + w_2 \quad + w_4 \quad\quad &= 0 \\
\quad w_2 + w_3 \quad + w_5 \quad &= 0 \\
w_1 + w_2 + w_3 \quad + w_6 \quad &= 0 \\
w_1 \quad + w_3 \quad\quad + w_7 &= 0
\end{aligned}
$$

巡回符号の例 (4)

- シンドロームの計算

$$
\begin{aligned}
s_1 &= w_1 + w_2 \quad + w_4 \\
s_2 &= \quad w_2 + w_3 \quad + w_5 \\
s_3 &= w_1 + w_2 + w_3 \quad + w_6 \\
s_4 &= w_1 \quad + w_3 \quad + w_7
\end{aligned}
$$

Example of Cyclic Code (4)

- Calculation of Syndrome

$$
\begin{aligned}
s_1 &= w_1 + w_2 \quad + w_4 \\
s_2 &= \quad w_2 + w_3 \quad + w_5 \\
s_3 &= w_1 + w_2 + w_3 \quad + w_6 \\
s_4 &= w_1 \quad + w_3 \quad + w_7
\end{aligned}
$$

巡回符号の例 (5)

- シンドロームと対応するエラー

誤りパターン							シンドローム			
e_1	e_2	e_3	e_4	e_5	e_6	e_7	s_1	s_2	s_3	s_4
1	0	0	0	0	0	0	1	0	1	1
0	1	0	0	0	0	0	1	1	1	0
0	0	1	0	0	0	0	0	1	1	1
0	0	0	1	0	0	0	1	0	0	0
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	1	1	0	0	1	1	0	0
1	0	1	0	0	0	0	1	1	0	0

1bit error

Example of Cyclic Code (5)

- Syndrome and error

error pattern							syndrome			
e_1	e_2	e_3	e_4	e_5	e_6	e_7	s_1	s_2	s_3	s_4
1	0	0	0	0	0	0	1	0	1	1
0	1	0	0	0	0	0	1	1	1	0
0	0	1	0	0	0	0	0	1	1	1
0	0	0	1	0	0	0	1	0	0	0
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	1	1	0	0	1	1	0	0
1	0	1	0	0	0	0	1	1	0	0

1bit error

巡回符号の例 (6)

- シンドロームと対応するエラー
  - 1bit 誤りを訂正可能
  - 2bit 誤りを検出可能

Example of Cyclic Code (6)

- Syndrome and corresponding error
  - we can correct 1bit error
  - we can detect 2bit error

誤り検出

- 受信例1
  - (0011100)の場合
    - シンドロームを計算
      - s1=1, s2=0, s3=1, s4=1
    - 1が3個出現した場合には，0の現れた出力値の計算式に含まれない情報ビットを求める．これはx1であるから，ここに誤りがあったことが検出できる
    - 送信信号は (1011100) と判定する

Error Detection

- Example 1
  - In case of receiving (0011100)
    - Calculate syndrome
      - s1=1, s2=0, s3=1, s4=1
    - When we detect three “1”s, mark position for “0”. Here, it is “x1”. Thus, we know there is an error at this position.
    - Transmitted data is (1011100)

誤り検出 (2)

- 受信例2
  - (1110100)の場合
    - シンドロームを計算
      - s1=0, s2=1, s3=1, s4=0
    - 1が2個出現しているから，2ビット誤りがあることがわかる
    - 誤り位置の特定は不可能

Error Detection (2)

- Example 2
  - In case of receiving (1110100)
    - Calculate syndrome
      - s1=0, s2=1, s3=1, s4=0
    - We know, there are 2 bits errors since “1” appears at two positions.
    - We cannot identify the location of error.

誤り検出 (3)

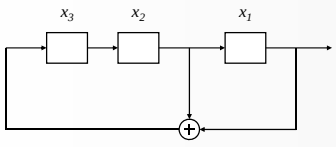
- 受信例3
  - (1011110)の場合
    - シンドロームを計算
      - s1=0, s2=0, s3=1, s4=0
    - 1の現れた3番目の検査ビットC3が誤りであることがわかる
    - 送信符号は (1011100) と判定する

Error Detection (3)

- Example 3
  - In case of receiving (1011110)
    - Calculate syndrome
      - s1=0, s2=0, s3=1, s4=0
    - We know, the third parity bit is an error since “1” appears.
    - Transmitted date is (1011100)

巡回符号の構成

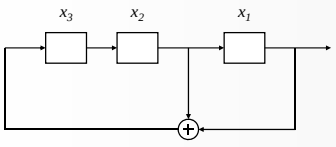
- シフトレジスタによる構成



□ : 遅延素子(1ビットシフトレジスタ)
⊕ : 排他的論理和

Structure of Cyclic Code

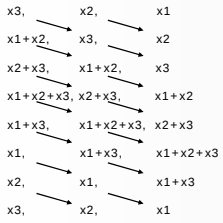
- Construction by shift register



□ : delay unit (1bit shift register)
⊕ : exclusive or

巡回符号の構成 (2)

- シフトと加算の処理



(x1+x2)+(x2+x3) = x1+x3

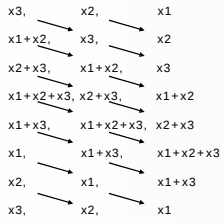
(x1+x2+x3)+(x2+x3) = x1

(x1+x3)+(x1+x2+x3) = x2

x3+(x1+x3) = x3

Structure of Cyclic Code (2)

- Processing of shift and addition



(x1+x2)+(x2+x3) = x1+x3

(x1+x2+x3)+(x2+x3) = x1

(x1+x3)+(x1+x2+x3) = x2

x3+(x1+x3) = x3

巡回符号の構成 (3)

- シフトレジスタへ3ビットの情報を代入
  - x1, x2, x3, x1+x2, x2+x3, x1+x2+x3, x1+x3, x1, x2 ...

$x_1x_2x_1$	$x_1x_2x_3$ $x_4x_5x_6x_7$
000 →	000 0000 (u0)
100 →	001 0111 (u1)
010 →	010 1110 (u2)
110 →	011 1001 (u3)
001 →	100 1011 (u4)
101 →	101 1100 (u5)
011 →	110 0101 (u6)
111 →	111 0010 (u7)

u7: 1110010
u6: 1100101
u4: 1001011
u1: 0010111
u2: 0101110
u5: 1011100
u3: 0111001

Structure of Cyclic Code (3)

- Input 3bit data to shift register
  - x1, x2, x3, x1+x2, x2+x3, x1+x2+x3, x1+x3, x1, x2 ...

$x_1x_2x_1$	$x_1x_2x_3$ $x_4x_5x_6x_7$
000 →	000 0000 (u0)
100 →	001 0111 (u1)
010 →	010 1110 (u2)
110 →	011 1001 (u3)
001 →	100 1011 (u4)
101 →	101 1100 (u5)
011 →	110 0101 (u6)
111 →	111 0010 (u7)

u7: 1110010
u6: 1100101
u4: 1001011
u1: 0010111
u2: 0101110
u5: 1011100
u3: 0111001

多項式表現

- 巡回符号を符号多項式F(x)で表す
  $F(x)=a_0+a_1x+a_2x^2+...+a_{n-1}x^{n-1}$
  ここに, $(a_0, a_1, a_2, ... a_{n-1})$ は符号語の各ビット
- 巡回符号の符号語の多項式表現の例
  - u4:1001011 → $u_4(x)=1+x^3+x^5+x^6$
  - u5:1011100 → $u_5(x)=1+x^2+x^3+x^4$
- x^7-1の因数かどうか?
  - $x^7-1=(x+1)\ u_4(x)$ 余り $x^5+x^4+x^3+x$
  - $x^7-1=(x^3+x^2+1)\ u_5(x)$ 余りなし

Polynomial Representation

- Cyclic code is represented by code polynomial F(x)
  $F(x)=a_0+a_1x+a_2x^2+...+a_{n-1}x^{n-1}$
  where, $(a_0, a_1, a_2, ... a_{n-1})$ is each bit for codeword
- Examples for polynomial representation
  - u4:1001011 → $u_4(x)=1+x^3+x^5+x^6$
  - u5:1011100 → $u_5(x)=1+x^2+x^3+x^4$
- Whether polynomial is a factor of x^7-1 or not?
  - $x^7-1=(x+1)\ u_4(x)$ residual $x^5+x^4+x^3+x$
  - $x^7-1=(x^3+x^2+1)\ u_5(x)$ no-residual

多項式表現 (2)

- $u_5(x)$はx^7-1の既約多項式(生成多項式G(x)と呼ぶ)
- 他の符号はすべて生成多項式で割り切れる
  - 例 $x^4 G(x) = x^4 (1+x^2+x^3+x^4) = x^4+x^6+x^7+x^8$
    $= 1+x+x^4+x^6$ (1100101:u_6)
- 符号多項式F(x)と生成多項式G(x)および商Q(x)の関係
  - F(x)=G(x)Q(x)
    情報ビット数k次, 検査ビット数m次
    F(x):最高次数n-1次
    G(x):最高次数m次
    Q(x):最高次数k-1次

Polynomial Representation (2)

- $u_5(x)$ is an irreducible polynomial (called Generator polynomial G(x)) of x^7-1
- Other codes are all reducible by Generator polynomial
  - example $x^4 G(x) = x^4 (1+x^2+x^3+x^4) = x^4+x^6+x^7+x^8$
    $= 1+x+x^4+x^6$ (1100101:u_6)
- Relation between Code polynomial F(x) and Generator polynomial G(x) and quotient Q(x)
  - F(x)=G(x)Q(x)
    information bit: k, check bit: m
    F(x): largest order n-1
    G(x): largest order m
    Q(x): largest order k-1

巡回符号の性質

- 性質1
  - k次の生成多項式G(x)で割り切れる(n-1)次以下の符号多項式の集合は巡回符号を構成する
- 性質2
  - k次の生成多項式G(x)から構成される巡回符号は, 長さk以下の任意のバースト誤り(連続して発生する誤り)を訂正できる

Characteristics of Cyclic Code

- Characteristic 1
  - A set of code polynomial, which is divisible by k-order generator polynomial, with less than n-order can construct cyclic code
- Characteristic 2
  - Cyclic code, which can be made from k-order generator polynomial, can correct burst error less than or equals to k bit

巡回符号の性質 (2)

- 例 $G(x)=x^4+x^3+x^2+1$, 符号語:(1001011)の場合
  - 符号語の多項式表現Fc(x)は?
    - $Fc(x)=1+x^3+x^5+x^6$
  - Fc(x)はG(x)で割り切れる?
    - $Fc(x)=G(x)(x^2+1)$
  - 符号語を3ビット右に巡回置換して得られる符号はG(x)で割り切れる?
    - (1001011) → (0111001)
    - $x^3Fc(x)=x^3(1+x^3+x^5+x^6)=x+x^2+x^3+x^6$
    - 一致(割り切れる)

Characteristics of Cyclic Code (2)

- Example $G(x)=x^4+x^3+x^2+1$, Codeword:(1001011)
  - What is a polynomial representation for Fc(x) for the code?
    - $Fc(x)=1+x^3+x^5+x^6$
  - Is Fc(x) divisible by G(x) ?
    - $Fc(x)=G(x)(x^2+1)$
  - Is a code obtained by 3 bit shift for the codeword divisible by G(x) ?
    - (1001011) → (0111001)
    - $x^3Fc(x)=x^3(1+x^3+x^5+x^6)=x+x^2+x^3+x^6$
    - Accordance! (divisible)

巡回符号の作成

- 情報ビットの桁上げ
  - 情報ビット(d_0, d_1,...,d_{k-1})を要素とする多項式P(x)を，符号多項式F(x)の高次のkビットに対応させるためx^{n-k}を乗じる
    $x^{n-k}P(x)=d_0x^{n-k}+...+d_{k-1}x^{n-1}$
- 多項式$x^{n-k}P(x)$をG(x)で割り，商をQ(x)，余りをR(x)とする
  - $x^{n-k}P(x)= G(x)Q(x)+R(x)$
- 両辺にR(x)を加えると，G(x)で割り切れる符号を作成できたことになる
  - $x^{n-k}P(x)+R(x)= G(x)Q(x)$

Creation of Cyclic Code

- Carry up of information bit
  - Multiply x^{n-k} to polynomial P(x) having information bit (d_0, d_1,...,d_{k-1}) as elements in order to fit to higher k-th bit of code polynomial
    $x^{n-k}P(x)=d_0x^{n-k}+...+d_{k-1}x^{n-1}$
- Divide polynomial $x^{n-k}P(x)$ by G(x), Let quotient and residual be Q(x) and R(x)
  - $x^{n-k}P(x)= G(x)Q(x)+R(x)$
- By adding R(x) to both terms, code, which is divisible by G(x), can be obtained
  - $x^{n-k}P(x)+R(x)= G(x)Q(x)$

巡回符号の作成 (2)

- 符号多項式をF(x)とすると
  - $F(x)=G(x)Q(x)$
    $= x^{n-k}P(x)+R(x)$

  ここに，$x^{n-k}P(x)$:情報ビット
  R(x):検査ビット

Creation of Cyclic Code (2)

- Let code polynomial be F(x), thus we have
  - $F(x)=G(x)Q(x)$
    $= x^{n-k}P(x)+R(x)$

  where, $x^{n-k}P(x)$: information bits
  R(x): check bit