

レポート提出票

科目名: 情報工学実験2

実験テーマ: 実験テーマ3 情報通信シミュレーション

実施日: 2020年 12月 7日

学籍番号: 4619055

氏名: 辰川力駆

共同実験者:

_____	_____
_____	_____
_____	_____
_____	_____

1 実験概要

(7,4) ハミング符号による符号化、復号を行うプログラムを作成して、ハミング符号の特徴を理解する。

2 実験手順

1. 4ビットの情報 w を生成
2. 生成多項式 $g(x)$ を用いて符号化
3. 1ビットの雑音を付与する
4. 受信語 y と検査行列 H からシンドローム s を計算
5. シンドロームから誤り位置を推定する
6. 推定語 \hat{x} を求め、もとの x と値を比較する

3 実験結果

ソースコードは付録に記述した。そのソースコードを実行した結果を下記に示す。

```
情報w : 0101
符号語x : 0101100
誤り位置(1から7) : 4
受信語y : 0100100
シンドロームs : 011
4ビット目を反転
推定語hat(x) : 0101100
復号後のビット誤り数 : 0
```

図 1: 情報 = 0101、誤り位置 = 4 の実行結果

```
情報w : 0101
符号語x : 0101100
誤り位置(1から7) : 1
受信語y : 1101100
シンドロームs : 101
1ビット目を反転
推定語hat(x) : 0101100
複号後のビット誤り数 : 0
```

図 2: 情報 = 0101、誤り位置 = 1 の実行結果

4 検討

4.1 課題 1

シミュレーションによるビット誤り率と理論値がほぼ同じ値になるにはどの程度のシミュレーション回数を実行する必要があるか。

4.2 課題 2

ϵ を非常に小さくした場合、これはどうなるか。

4.3 課題 3

rand() と MT の違いは何か。

A 付録

ソースコード 1: kadai3_2.cpp

```
1 //4619055 辰川力駆
2 #include <random> // 乱数生成
3 #include <stdio.h>
4 #include <iostream>
5 #include <iomanip>
6
7 using namespace std;
8
9 #define N 7 //符号化後のビット数
10 #define K 4 //デジタル情報の分けるブロックのビット数
11
12 int main()
13 {
14     int w[K] = {0, 1, 0, 1}; //4ビットの情報w
15     int x[N]; //7ビットの符号語x
16     int y[N]; //7ビットの受信語y
17     int ErrorPosition; //誤り位置 (1から 7)
18     int s[3]; //シンドローム s
19     int EstimationPosition; //誤り位置推定場所
20     int cnt = 0; //復号後のビット誤り数
21
22     cout << "情報 w: ";
23     for (int i = 0; i < K; i++)
24     {
25         cout << w[i];
26     }
27     cout << endl;
28
29     for (int i = 0; i < K; i++)
30     {
31         x[i] = w[i];
32     }
33     x[N - K + 1] = w[0] ^ w[1] ^ w[2];
34     x[N - K + 2] = w[1] ^ w[2] ^ w[3];
35     x[N - K + 3] = w[0] ^ w[1] ^ w[3];
36     cout << "符号語 x: ";
37     for (int i = 0; i < N; i++)
38     {
39         cout << x[i];
40     }
41     cout << endl;
42
43     cout << "誤り位置 (1から 7): ";
44     cin >> ErrorPosition;
45
46     cout << "受信語 y: ";
47     for (int i = 0; i < N; i++)
48     {
```

```

49     y[i] = x[i];
50     if (i == ErrorPosition - 1)
51     {
52         y[i] = x[i] ^ 1; //誤り位置は反転する
53     }
54     cout << y[i];
55 }
56 cout << endl;
57
58 s[0] = y[0] ^ y[1] ^ y[2] ^ y[4];
59 s[1] = y[1] ^ y[2] ^ y[3] ^ y[5];
60 s[2] = y[0] ^ y[1] ^ y[3] ^ y[6];
61 cout << "シンδροーム s□:□";
62 for (int i = 0; i < 3; i++)
63 {
64     cout << s[i];
65 }
66 cout << endl;
67
68 int point = 0;
69 for (int i = 0; i < 3; i++)
70 {
71     point += s[i] * pow(2, 2 - i);
72 }
73 switch (point)
74 {
75     case 5:
76         EstimationPosition = 1;
77         break;
78     case 7:
79         EstimationPosition = 2;
80         break;
81     case 6:
82         EstimationPosition = 3;
83         break;
84     case 3:
85         EstimationPosition = 4;
86         break;
87     case 4:
88         EstimationPosition = 5;
89         break;
90     case 2:
91         EstimationPosition = 6;
92         break;
93     case 1:
94         EstimationPosition = 7;
95         break;
96     default:
97         EstimationPosition = -1;
98         break;
99 }
100

```

```

101     if (EstimationPosition == -1)
102     {
103         cout << "誤りはなし" << endl;
104     }
105     else
106     {
107         cout << EstimationPosition << "ビット目を反転" << endl;
108     }
109
110     y[EstimationPosition - 1] = y[EstimationPosition - 1] ^ 1;
111     cout << "推定語  $\hat{x}$ : ";
112     for (int i = 0; i < N; i++)
113     {
114         cout << y[i];
115     }
116     cout << endl;
117
118     for (int i = 0; i < N; i++)
119     {
120         cnt += x[i] ^ y[i];
121     }
122     cout << "複号後のビット誤り数: " << cnt << endl;
123
124     return 0;
125 }

```