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 11月 30日

学籍番号: 4619055

氏名: 辰川力駆

共同実験者: ＿＿＿＿＿＿＿＿ ＿＿＿＿＿＿＿＿

＿＿＿＿＿＿＿＿ ＿＿＿＿＿＿＿＿

＿＿＿＿＿＿＿＿ ＿＿＿＿＿＿＿＿

＿＿＿＿＿＿＿＿ ＿＿＿＿＿＿＿＿

1　実験概要

rand() によって発生させた乱数と、MT によって発生させた乱数それぞれについて誤り率をプロットしたグラフを作成するなどをして、ディジタル通信システムと誤り訂正符号の理解を深める。

2　実験手順

- 系列長 $K = 4$ の情報系列 $w = (w_1, ..., w_K) \in \{0, 1\}$ を乱数によって発生させ、BSC の各 ϵ に対するビット誤り率の値をシミュレーションにより求める。乱数の発生には、C 言語コンパイラ上で rand() 関数を用いたものと Mersenne Twister (MT) を用いた 2 種類のプログラムを作成する。
- 横軸を BSC の誤り確率 ϵ、縦軸 (指数表示) をビット誤り率 P_e としてグラフにプロットする。
- 誤り確率の理論値 (ϵ) を実線 (プロットなし) で表示してシミュレーションで得られた値と比較する。

3　実験結果

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。

図 1: 全ての項目による対数尤度グラフ

図 3 の対数尤度関数で最大値を取っている θ の値は、$\theta = 1.7$ である。よって自分の能力値は 1.7 であると推定できる。また、情報量 $I(\hat{\theta})$ は各問題に関する情報量 $I_i(\theta)$ の合計である。したがって、8 問の合計を求めたら良く、

である。

また、標準誤差は

$$\begin{aligned} se(\hat{\theta}) &= I(\hat{\theta})^{-\frac{1}{2}} \\ &= \frac{1}{\sqrt{1.65559}} \\ &\approx 0.77718 \end{aligned}$$

である。

4.4 課題 3-4

課題 1-3 で解いた項目の a_i が全て「1」だった場合の能力値、情報量、標準誤差を求め、結果について考察する。

表 1: G9 メンバーの推定結果

学籍番号 (46190)	15	28	38	55	58	64	94
能力値 θ	0	-0.1	1	1.7	0.4	1	1.3
偏差値 S	45	45	60	70	61.7	70	70
解答した問題	5	1	3	3	8	1	3
	11	5	8	9	20	15	6
	36	20	16	21	28	28	20
	49	25	49	24	35	46	34
			52	30	42	52	45
				35	46		
				40			
				51			

A 付録

ソースコード 1: kadai3_rand.cpp

```cpp
//4619055 辰川力駆
#include <random> // 乱数生成
#include <stdio.h>
#include <iostream>
#include <iomanip>

using namespace std;

#define SIM 1000000

#define real_rand (double)rand() / RAND_MAX; //
    RAND_MAX で割ることで 0 から 1 を返すようにしている。
#define seed 55 //学籍番号下 2 桁
#define K 4 //系列長

int main()
{
    srand(seed);
    int w[4], e[4], y[4];

    cout << "# SIM:" << SIM << endl;
    cout << "# ep    # BER" << endl;

    double ep = 0;
    int count = 0;
    for (int i = 0; i < 12; i++) //12回で設定
    {
        count = 0;
        ep = 0.0001 * (i + 1);

        for (int s = 0; s < SIM; s++)
        {
            for (int j = 0; j < K; j++)
            {
                double rd = real_rand; //乱数発生
                w[j] = rd * 2; //2倍することで 0.5より大きいか小さいかを判定できる。
            }
            for (int j = 0; j < K; j++)
            {
                double rd = real_rand; //乱数発生
                if (rd <= ep)
                {
                    e[j] = 1;
                }
                else
                {
                    e[j] = 0;
                }
```

```
48                  }
49                  for (int j = 0; j < K; j++)
50                  {
51                      y[j] = w[j] ^ e[j];
52                      count += w[j] ^ y[j];
53                  }
54              }
55              cout << fixed << setprecision(4) << ep;
56              cout.unsetf(ios::fixed);
57              cout << fixed << setprecision(7) << "␣" << (double)count / (K * SIM) <<
                    endl;
58          }
59
60          return 0;
61      }
```

ソースコード 2: kadai3_mt.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 SIM 1000000
10
11  #define seed 55 //学籍番号下２桁
12  #define K 4 //系列長
13
14  mt19937 mt(seed); //メルセンヌ・ツイスタ
15
16  int main()
17  {
18      srand(seed);
19      int w[4], e[4], y[4];
20
21      uniform_real_distribution<double> rand_real(0, 1);
22      normal_distribution<double> rand_n(0, 0.3);
23
24      cout << "#␣SIM:" << SIM << endl;
25      cout << "#␣ep␣␣␣#␣BER" << endl;
26
27      double ep = 0;
28      int count = 0;
29      for (int i = 0; i < 12; i++) //12回で設定
30      {
31          count = 0;
32          ep = 0.0001 * (i + 1);
33
34          for (int s = 0; s < SIM; s++)
35          {
```

```
36                  for (int j = 0; j < K; j++)
37                  {
38                      w[j] = rand_real(mt) * 2; //2倍することで0.5より大きいか小さいかを判
                            定できる。
39                  }
40                  for (int j = 0; j < K; j++)
41                  {
42                      if (rand_real(mt) <= ep)
43                      {
44                          e[j] = 1;
45                      }
46                      else
47                      {
48                          e[j] = 0;
49                      }
50                  }
51                  for (int j = 0; j < K; j++)
52                  {
53                      y[j] = w[j] ^ e[j];
54                      count += w[j] ^ y[j];
55                  }
56              }
57              cout << fixed << setprecision(4) << ep;
58              cout.unsetf(ios::fixed);
59              cout << fixed << setprecision(7) << "␣" << (double)count / (K * SIM) <<
                    endl;
60          }
61
62          return 0;
63      }
```