**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Федеральное государственное бюджетное образовательное учреждение   
высшего образования

**«Сибирский государственный университет науки и технологий   
имени академика М.Ф. Решетнева»**

Институт информатики и телекоммуникаций

Кафедра информационно-управляющих систем

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ**

Теория информации

|  |
| --- |
| Энтропия сложной системы. Условная энтропия |

Руководитель А.Н. Бочаров

подпись, дата инициалы, фамилия

Обучающийся БПИ22-02, 221217039 Н.Ю. Сюткин

номер группы, зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2023 г.

# ЦЕЛЬ РАБОТЫ

Закрепить теоретические знания и получить практические навыки при определении энтропии.

# порядок выполнения работы

1. Ознакомиться с общей постановкой задачи.
2. Ознакомится с вариантом задания – соответствует вашему номеру в списке группы.
3. Выполнить задания согласно варианту.
4. Подготовить отчет по лабораторной работе. Отчет должен включать в себя:   
   •  титульный лист;   
   •  цель лабораторной работы;   
   •  постановку задачи;   
   •  ход работы;   
   •  краткие ответы на контрольные вопросы;   
   •  выводы по лабораторной работе.
5. Защитить лабораторную работу перед преподавателем.

# постановка задачи

Влияние помех в канале связи описывается канальной матрицей, с помощью условных вероятностей и , где – источник информации, – приемник информации.

1. Провести исследование канала информации со стороны источника информации и со стороны приемника информации.

По результатам исследования определить:

- потери информации , которые приходятся на каждый переданный сигнал и потери при передаче всех сигналов ;

- потери информации ,, которые приходятся на каждый принятый сигнал и потери при приеме всех сигналов .

Исходные данные:

и получить из матрицы совместных вероятностей, размером 10 × 10, которую задать самостоятельно.

2. Определить энтропию двухбуквенного сочетания и условную энтропию заданного текста, где и – 32-х буквенный алфавит. Использовать текстовый файл из первой лабораторной работы.

# ХОД РАБОТЫ

Для решения поставленных задач были разработаны два метода. Первый метод ShowMatrix выполняет задачу корректного вывода матрицы на экран. Второй метод ShowMatrixBottomSum является усовершенствованным методом ShowMatrix. Его задача в выводе матрицы с суммами строк и столбцов.

class ShowMatr

{

public static void ShowMatrix(double[,] matr)

{

for (int i = 0; i < matr.GetLength(0); i++)

{

Console.Write("|");

if (i == 0)

{

Console.Write(" Y/X\t|");

for (int h = 0; h < matr.GetLength(1); h++)

{

Console.Write(" X" + (h + 1) + "\t|");

}

Console.WriteLine();

Console.WriteLine("-----------------------------------------------------------------------------------------");

Console.Write("|");

}

Console.Write(" Y" + (i + 1) + "\t|");

for (int j = 0; j < matr.GetLength(1); j++)

{

Console.Write(" " + Math.Round(matr[i, j], 2) + "\t|");

}

Console.WriteLine();

Console.WriteLine("-----------------------------------------------------------------------------------------");

}

}

public static void ShowMatrixBottomSum(double[,] matr)

{

double[] array = new double[10];

for (int i = 0; i < matr.GetLength(0); i++)

{

double temp = 0;

Console.Write("|");

if (i == 0)

{

Console.Write(" Y/X\t|");

for (int h = 0; h < matr.GetLength(1) + 1; h++)

{

if(h == 10)

{

Console.Write(" P(Y)" + "\t");

}

else

Console.Write(" X" + (h + 1) + "\t|");

}

Console.WriteLine();

Console.WriteLine("-------------------------------------------------------------------------------------------------");

Console.Write("|");

}

Console.Write(" Y" + (i + 1) + "\t|");

for (int j = 0; j < matr.GetLength(1); j++)

{

temp += matr[i, j];

array[i] += matr[j, i];

Console.Write(" " + Math.Round(matr[i, j], 2) + "\t|");

if(j == 9)

{

Console.Write(" " + Math.Round(temp, 2) + "\t");

}

}

Console.WriteLine();

Console.WriteLine("-------------------------------------------------------------------------------------------------");

}

Console.Write("| P(X)" + "\t|");

for(int i = 0;i < array.Length; i++)

{

Console.Write(" " + Math.Round(array[i],2) + "\t|");

}

Console.WriteLine();

}

}

**Задание №1.**

Составление матрицы совместных вероятностей включает в себя генерацию случайных вещественных чисел. Дабы сумма вещественных чисел (вероятностей) в матрице была равна единице матрица нормируется. Каждый элемент делится на сумму элементов матрицы. Таким образом, мы получаем матрицу совместных вероятностей.

double[,] matr\_Pij = new double[10, 10];

Random rnd = new Random();

int cnt = 0;

for (int i = 0; i < matr\_Pij.GetLength(0); i++)

{

for (int j = 0; j < matr\_Pij.GetLength(1); j++)

{

if (cnt % 4 == 0)

matr\_Pij[i, j] = 0;

else

matr\_Pij[i, j] = rnd.NextDouble();

cnt++;

}

}

double sum = matr\_Pij.Cast<double>().Sum();

for (int i = 0; i < matr\_Pij.GetLength(0); i++)

{

for (int j = 0; j < matr\_Pij.GetLength(1); j++)

{

matr\_Pij[i, j] /= sum;

}

}

ShowMatr.ShowMatrixBottomSum(matr\_Pij);

Матрица совместных вероятностей представлена на рис. 1.

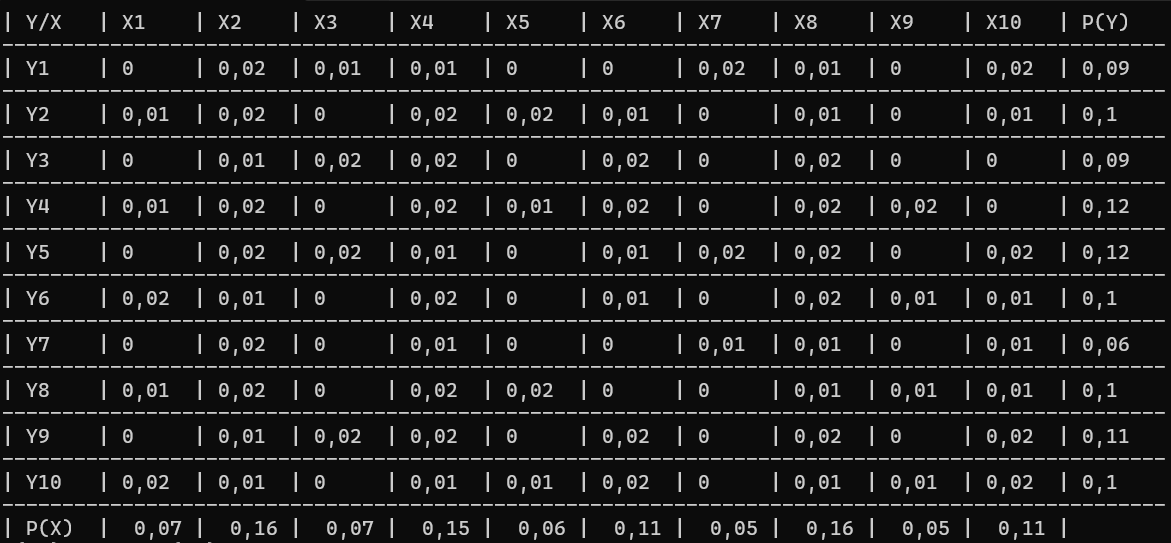


Рисунок 1 – Матрица совместных вероятностей

Сложив все вероятности матрицы вместе, в сумме получается 1. Проведем исследование канала информации со стороны источника и приёмника. Для этого составим матрицу для источника и приёмника циклом, который получает суммы строк/столбцов и . Далее поделим каждый элемент строки/столбца на его сумму.

double[] Pxi = new double[matr\_Pij.GetLength(0)];

double[] Pyj = new double[matr\_Pij.GetLength(1)];

for (int i = 0; i < matr\_Pij.GetLength(0); i++)

{

double sum\_row = 0;

double sum\_col = 0;

for (int j = 0; j < matr\_Pij.GetLength(1); j++)

{

sum\_row += matr\_Pij[i, j];

sum\_col += matr\_Pij[j, i];

}

Pxi[i] = sum\_col;

Pyj[i] = sum\_row;

}

for(int i = 0; i < 10; i++)

{

Console.WriteLine("P(Xi) = " + Math.Round(Pxi[i],2) + " P(Yj) = " + Math.Round(Pyj[i],2));

}

double[,] canal\_matrPYjXi = new double[10, 10];

double[,] canal\_matrPXiYj = new double[10, 10];

for(int i = 0; i < 10; i++)

{

for(int j = 0; j < 10; j++)

{

if (Pxi[i] == 0)

canal\_matrPYjXi[j, i] = 0;

else

canal\_matrPYjXi[j, i] = matr\_Pij[j, i] / Pxi[i];

if (Pyj[i] == 0)

canal\_matrPXiYj[i, j] = 0;

else

canal\_matrPXiYj[i, j] = matr\_Pij[i, j] / Pyj[i];

}

}

Console.WriteLine("\nКанальная матрица P(Yj/Xi): ");

ShowMatr.ShowMatrixBottomSum(canal\_matrPYjXi);

Console.WriteLine("\nКанальная матрица P(Xi/Yj): ");

ShowMatr.ShowMatrixBottomSum(canal\_matrPXiYj);

Канальные матрицы  и представлен на рис. 2 и рис. 3 соответственно.

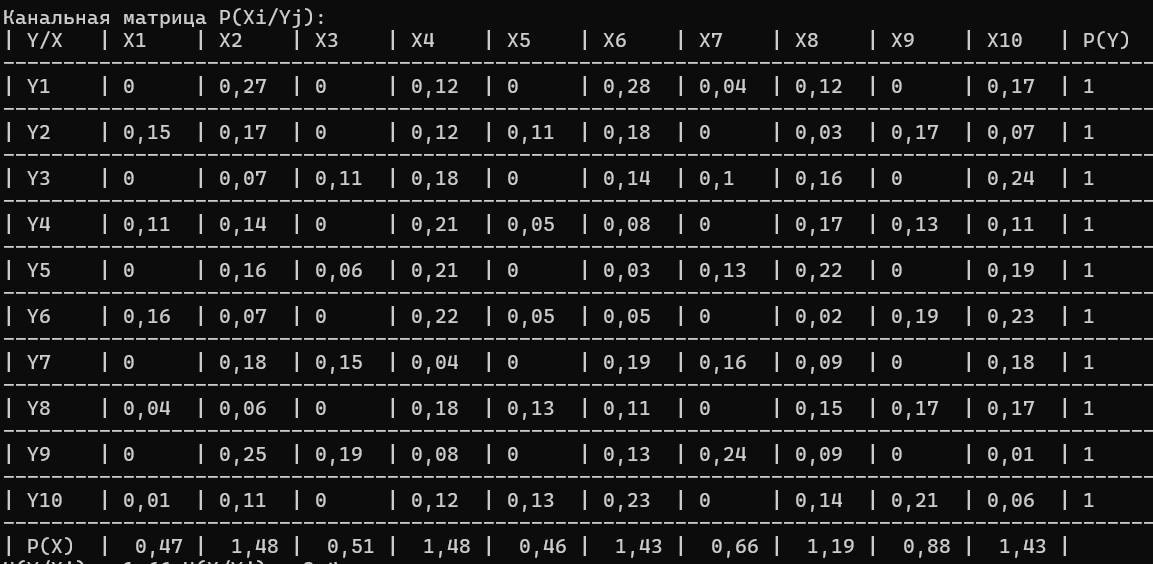
****

Рисунок 2 – Канальная матрица

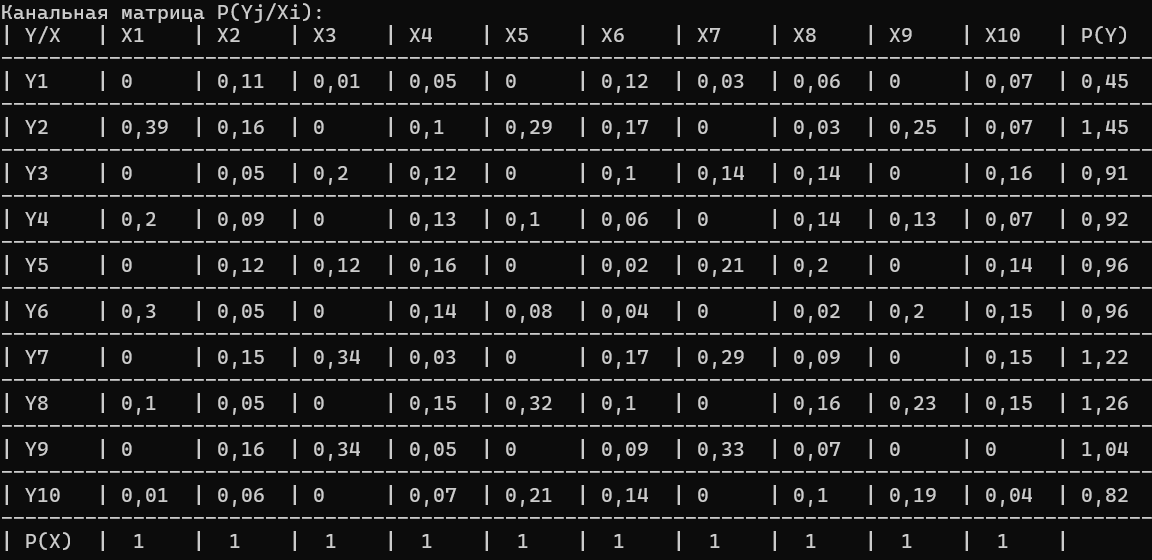


Рисунок 3 – Канальная матрица

По полученным данным определим потери информации и которые приходятся на каждый переданный и сигналы и потери и .

double[] H\_X\_yj = new double[10], H\_Y\_xi = new double[10];

for(int i = 0; i < matr\_Pij.GetLength(0); i++)

{

for(int j = 0; j < matr\_Pij.GetLength(1); j++)

{

if (canal\_matrPYjXi[i, j] == 0)

H\_Y\_xi[i] += 0;

else

H\_Y\_xi[i] += canal\_matrPYjXi[i, j] \* Log(canal\_matrPYjXi[i, j], 2);

if (canal\_matrPXiYj[i, j] == 0)

H\_X\_yj[i] += 0;

else

H\_X\_yj[i] += canal\_matrPXiYj[i, j] \* Log(canal\_matrPXiYj[i, j], 2);

}

H\_Y\_xi[i] = -H\_Y\_xi[i];

H\_X\_yj[i] = -H\_X\_yj[i];

}

for(int i = 0; i < 10; i++)

{

Console.WriteLine("H(Y/Xi) = " + Math.Round(H\_Y\_xi[i],2) + " H(X/Yj) = " + Math.Round(H\_X\_yj[i],2));

}

double H\_Y\_X = 0;

double H\_X\_Y = 0;

for (int i =0; i < matr\_Pij.GetLength(0); i++)

{

H\_X\_Y += Pyj[i] \* H\_X\_yj[i];

H\_Y\_X += Pxi[i] \* H\_Y\_xi[i];

}

Console.WriteLine("H(X/Y) = " + H\_X\_Y + " H(Y/X) = " + H\_Y\_X);

Результаты представлены на рис. 4.

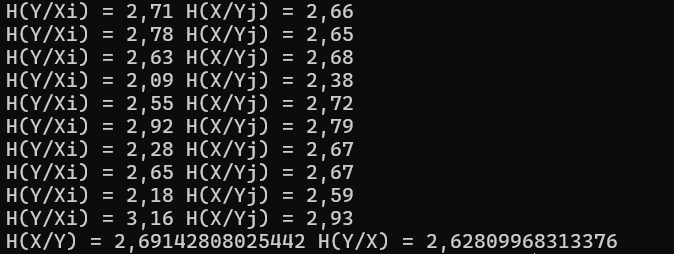


Рисунок 4 – Итоговые результаты

**Задание №2.**

Составим матрицу вероятности двухбуквенных сочетаний 32x32. Задаём все буквы алфавита в массиве и сравниваем вложенным циклом каждую букву с остальными. Полученные совпадения вносим в матрицу 32x32. Матрица сочетаний представлена на рис. 1.

string[] alphabet = {"а","б","в","г","д","е",

"ж","з","и","й","к","л","м","н","о","п","р"

,"с","т","у","ф","х","ц","ч","ш","щ","ь","ы"

," ","э","ю","я"};

string[] copy\_alphabet = new string[alphabet.Length];

Array.Copy(alphabet, copy\_alphabet, alphabet.Length);

int cnt;

//Перебираем все двухбуквенные сочетания для исходного текста

for (int i = 0; i < alphabet.Count(); i++)

{

for (int j = 0; j < alphabet.Count(); j++)

{

cnt = 0;

string line = alphabet[i] + copy\_alphabet[j];

for(int h = 0; h < text.Length-1; h++)

{

if (line.ToLower() == (text[h].ToString() + text[h + 1].ToString()).ToLower() || line.ToUpper() == (text[h].ToString() + text[h + 1].ToString()).ToUpper())

cnt++;

}

bukva\_meets[i, j] = cnt;

}

}

ShowMatr.ShowMatrix(bukva\_meets);

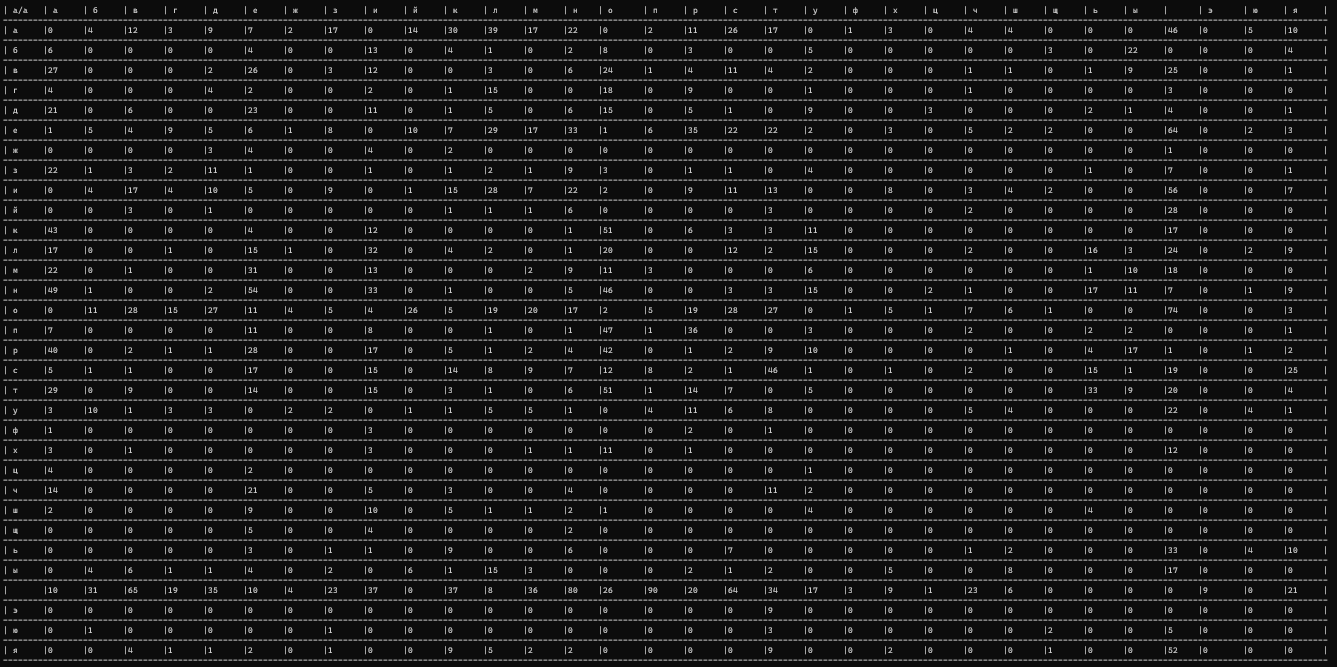


Рисунок 1 – Матрица двухбуквенных сочетаний

Следующим шагом преобразуем матрицу сочетаний в матрицу вероятностей путём деления каждого элемента матрицы на общую сумму всех элементов. Матрица вероятностей представлена на рис. 2.

//Выводим матрицу и задаем матрицу вероятностей P(i,j)

ShowMatr.ShowMatrix(bukva\_meets);

double sum = bukva\_meets.Cast<double>().Sum();

double[,] matr\_Pij = new double[alphabet.Length, alphabet.Length];

for (int i = 0; i < matr\_Pij.GetLength(0); i++)

{

for (int j = 0; j < matr\_Pij.GetLength(1); j++)

{

if (bukva\_meets[i,j] == 0)

matr\_Pij[i,j] = 0;

else

matr\_Pij[i, j] = bukva\_meets[i,j]/sum;

}

}

ShowMatr.GetSpace();

ShowMatr.ShowMatrix(matr\_Pij);

ShowMatr.GetSpace();

**

Рисунок 2 – Матрица вероятностей двухбуквенных сочетаний

Затем, на основе матрицы вероятностей, получим суммы строк/столбцов и создадим канальные матрицы. Суммы строк/столбцов представлены на рис.2. Канальные матрицы представлены на рисунке 3 и рисунке 4 соответственно.

//Находим P(xi) и P(Yj)

double[] Pxi = new double[matr\_Pij.GetLength(0)];

double[] Pyj = new double[matr\_Pij.GetLength(1)];

for (int i = 0; i < matr\_Pij.GetLength(0); i++)

{

double sum\_row = 0;

double sum\_col = 0;

for (int j = 0; j < matr\_Pij.GetLength(1); j++)

{

sum\_row += matr\_Pij[i, j];

sum\_col += matr\_Pij[j, i];

}

Pxi[i] = sum\_col;

Pyj[i] = sum\_row;

}

for (int i = 0; i < alphabet.Count(); i++)

{

Console.WriteLine("P(Xi) = " + Math.Round(Pxi[i], 4) + " P(Yj) = " + Math.Round(Pyj[i], 4));

}

//Составляем канальные матрицы

double[,] canal\_matrPYjXi = new double[alphabet.Count(), alphabet.Count()];

double[,] canal\_matrPXiYj = new double[alphabet.Count(), alphabet.Count()];

for (int i = 0; i < alphabet.Count(); i++)

{

for (int j = 0; j < alphabet.Count(); j++)

{

if (Pxi[i] == 0)

canal\_matrPYjXi[j, i] = 0;

else

canal\_matrPYjXi[j, i] = matr\_Pij[j, i] / Pxi[i];

if (Pyj[i] == 0)

canal\_matrPXiYj[i, j] = 0;

else

canal\_matrPXiYj[i, j] = matr\_Pij[i, j] / Pyj[i];

}

}

Console.WriteLine("\nКанальная матрица P(Yj/Xi): ");

ShowMatr.ShowMatrixBottomSum(canal\_matrPYjXi);

Console.WriteLine("\nКанальная матрица P(Xi/Yj): ");

ShowMatr.ShowMatrixBottomSum(canal\_matrPXiYj);

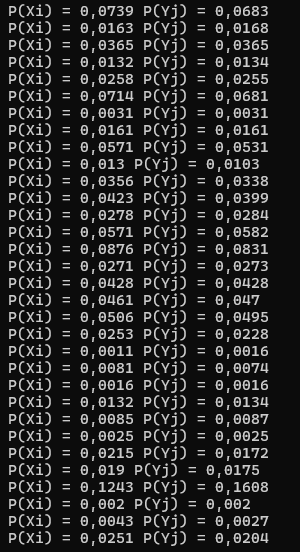


Рисунок 2 – и

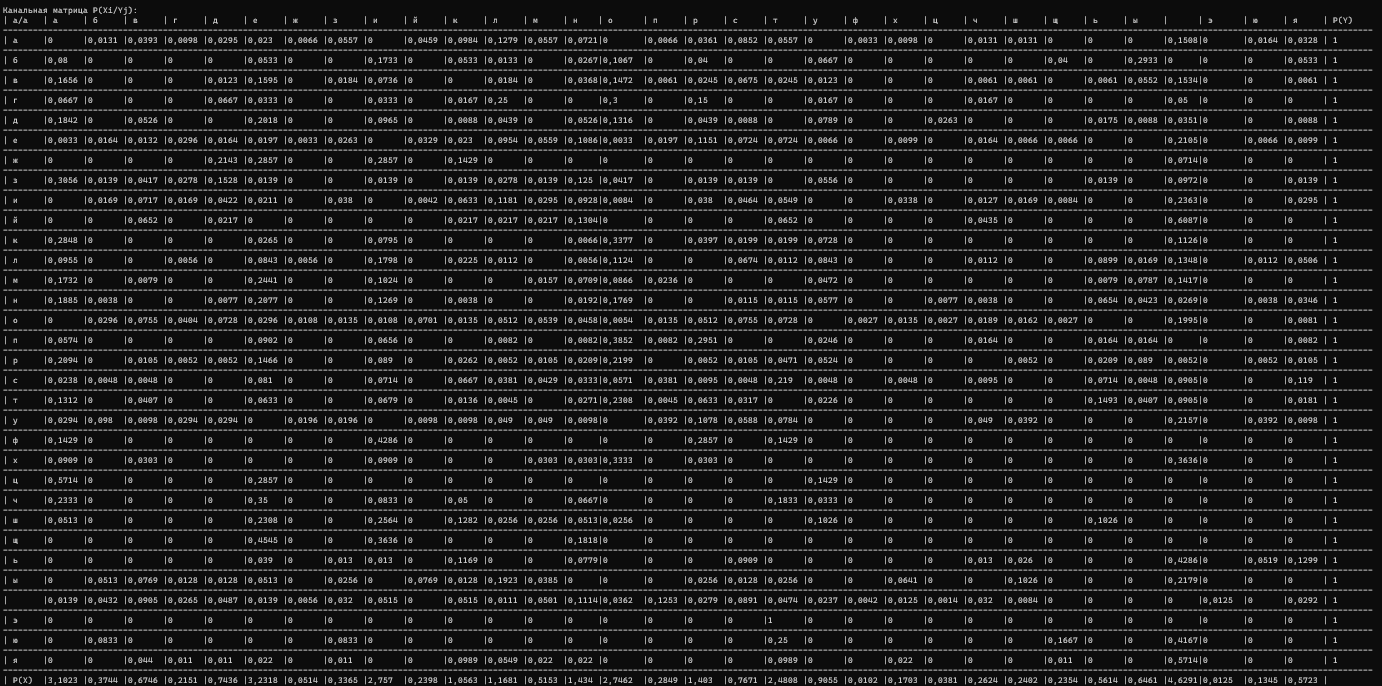


Рисунок 3 – Канальная матрица

На рис. 3 можно заметить, что суммы равны единицам.



Рисунок 4 – Канальная матрица

На рис. 4 можно заметить, что суммы равны единицам.

По полученным данным определим потери информации , которые приходятся на каждый переданный сигнал и потери при передаче всех сигналов . Далее найдем потери информации , которые приходятся на каждый принятый сигнал и потери при приеме всех сигналов . Результаты представлены на рис. 5.

double[] H\_X\_yj = new double[alphabet.Count()], H\_Y\_xi = new double[alphabet.Count()];

for (int i = 0; i < matr\_Pij.GetLength(0); i++)

{

for (int j = 0; j < matr\_Pij.GetLength(1); j++)

{

if (canal\_matrPYjXi[j, i] == 0)

H\_Y\_xi[i] += 0;

else

H\_Y\_xi[i] += canal\_matrPYjXi[j, i] \* Log(canal\_matrPYjXi[j, i], 2);

if (canal\_matrPXiYj[i, j] == 0)

H\_X\_yj[i] += 0;

else

H\_X\_yj[i] += canal\_matrPXiYj[i, j] \* Log(canal\_matrPXiYj[i, j], 2);

}

H\_Y\_xi[i] = -H\_Y\_xi[i];

H\_X\_yj[i] = -H\_X\_yj[i];

}

for (int i = 0; i < alphabet.Count(); i++)

{

Console.WriteLine("H(Y/Xi) = " + Math.Round(H\_Y\_xi[i], 2) + " H(X/Yj) = " + Math.Round(H\_X\_yj[i], 2));

}

double H\_Y\_X = 0;

double H\_X\_Y = 0;

//Находим потери H(X/Y) и H(Y/X)

for (int i = 0; i < matr\_Pij.GetLength(0); i++)

{

H\_X\_Y += Pyj[i] \* H\_X\_yj[i];

H\_Y\_X += Pxi[i] \* H\_Y\_xi[i];

}

Console.WriteLine("H(X/Y) = " + H\_X\_Y + " H(Y/X) = " + H\_Y\_X);

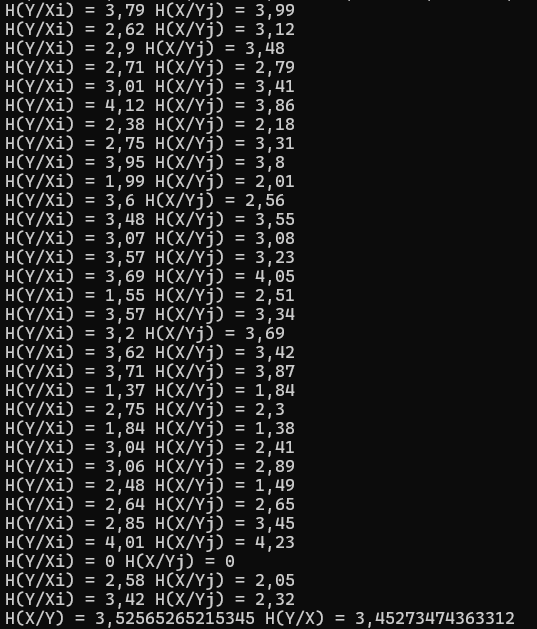


Рисунок 5 – Потери информации на каждый переданный сигнал, при передаче всех сигналов

Далее найдем и через и . Результаты представлены на рис. 6.

double H\_X = 0, H\_Y = 0;

for(int i = 0; i < alphabet.Count(); i++)

{

if (Pxi[i] == 0)

H\_X += 0;

else

H\_X += Pxi[i] \* Log(Pxi[i],2);

if (Pyj[i] == 0)

H\_Y += 0;

else

H\_Y += Pyj[i] \* Log(Pyj[i], 2);

}

H\_Y = -H\_Y;

H\_X = -H\_X;

Console.WriteLine("H(X) = " + H\_X + " ,H(Y) = " + H\_Y);



Рисунок 6 – и

Затем найдем

double H\_XX\_YY\_2 = 0;

for(int i = 0; i < matr\_Pij.GetLength(0); i++)

{

for(int j = 0; j < matr\_Pij.GetLength(1); j++)

{

if (matr\_Pij[i, j] == 0)

H\_XX\_YY\_2 += 0;

else

H\_XX\_YY\_2 += matr\_Pij[i,j] \* Log(matr\_Pij[i,j],2);

}

}

Далее сверим ответ, сложив и , получив .

double H\_XX\_YY = H\_X + H\_Y\_X;

Также можно проверить решение сложив и , полученное значение должно быть больше, чем найденное . Итоговые результаты представлены на рис. 7.

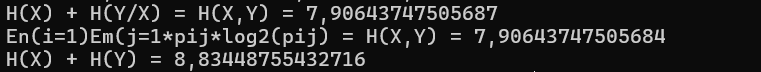


Рисунок 7 – Итоговые результаты

На рисунке 7 можно заметить, что .

# ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ

1.Что такое сложная система?   
 Сложная система — это система, состоящая из множества взаимосвязанных элементов или подсистем, обладающая сложной структурой и поведением.

2. Как описывается сложная система?   
 Сложная система описывается с использованием теории информации, которая позволяет изучать ее структуру, взаимосвязи между элементами и предсказывать ее поведение.

3. Как определяется энтропия сложной системы?   
 Энтропия сложной системы определяется как мера неопределенности или беспорядка в системе. Чем выше энтропия, тем больше неопределенности в системе.

4. В чем суть теоремы сложения энтропий для независимых систем?   
 Теорема сложения энтропий для независимых систем утверждает, что энтропия объединенной системы равна сумме энтропий каждой из независимых систем.

5. Что значит – системы зависимы?   
 Системы зависимы — это системы, в которых элементы или подсистемы взаимосвязаны друг с другом.

6. Что обозначает термин «условная энтропия»?   
 Условная энтропия — это мера неопределенности в одной случайной переменной при условии, что известно значение другой случайной переменной.

7. Как вычисляется частная условная энтропия?   
 Частная условная энтропия вычисляется как разность между полной энтропией и условной энтропией.

8. Как вычисляется полная условная энтропия?   
 Полная условная энтропия вычисляется как средняя условная энтропия по всем возможным значениям условия.

9. В чем суть теоремы сложения энтропий для зависимых систем?

Теорема сложения энтропий для зависимых систем утверждает, что энтропия объединенной системы равна сумме энтропий каждой из систем плюс взаимная информация между ними.

10. Что описывают канальные матрицы и в чем их особенности?

Канальные матрицы описывают вероятности переходов между состояниями системы и позволяют моделировать передачу информации через каналы связи. Их особенность заключается в том, что сумма элементов каждой строки равна 1, что отражает сохранение вероятности при передаче информации через канал.

# ВЫВОДЫ

В ходе лабораторной работы были закреплены теоретические знания и получены практические навыки при определении энтропии.

Были рассмотрены матрицы вероятностей событий системы, канальные матриц. Были подсчитаны потери информации, найдена энтропия сложной системы . Для проверки результатов энтропии сложной системы, энтропия была найдена альтернативным способом = , а также, полученное значение было подставлено в неравенство, которое дало верный исход.