**Министерство науки и высшего образования Российской Федерации**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

**Факультет безопасности информационных технологий**

**Дисциплина:**

«Теория информации»

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №6**

**Выполнил:**

Бехит Мохаммед Махмуд, студент группы N33511

(подпись)

**Проверил:**

Есипов Дмитрий Андреевич, ассистент ФБИТ

(отметка о выполнении)

(подпись)

Санкт-Петербург 2024 г.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_TOC_250007)

1. [ХОД РАБОТЫ 4](#_TOC_250006)
   1. [Определение параметров 4](#_TOC_250005)
   2. [Кодирование информационного вектора 10](#_TOC_250004)
   3. [Внесение ошибки 10](#_TOC_250003)
   4. [Сравнение 12](#_TOC_250002)

[ЗАКЛЮЧЕНИЕ 14](#_TOC_250001)

[ПРИЛОЖЕНИЕ А 15](#_TOC_250000)

# ВВЕДЕНИЕ

**Цель работы** – изучить БЧХ кодирование. Для достижения поставленной цели необходимо решить следующие задачи:

− для заданного систематического (n, k)-кода БЧХ определить порождающий полином, минимальные полиномы, циклотомические классы поля Галуа, проверочную матрицу и количество исправленных ошибок (гарантированных);

− закодировать информационный вектор;

− внести в полученный двоичный кодовый вектор двукратную ошибку и декодировать получившийся вектор;

− сравнить по методу Монте-Карло вероятностные характеристики двух кодов БЧХ;

− результаты выполнения работы оформить в виде отчета.

# ХОД РАБОТЫ

Начальные условия приведены в таблице 1 и относятся к варианту 2.

Таблица 1 – Параметры кода, инф. вектор, позиция ошибок, исправляющая способность

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **(n,k)** | **Информационный вектор** | **Err** | **(x,y)** |
| 2 | (63, 39) | [0…0 1 1 0 0 0 0] | 41; 16 | −2 −3  (2·10 , 2· 10 ,) |

Для выполнения лабораторной работы был использован GNU Octave.

## Определение параметров

Для выполнения лабораторной работы необходимо определить порождающий полином, минимальные полиномы, циклотомические классы поля Галуа, проверочную матрицу и количество исправленных ошибок (гарантированных).

Чтобы получить эти параметры, необходимо воспользоваться следующей функцией: [g , minpol , c, mat, t] = bchpoly (n , k). На вход функции задаются параметры кода n и k. На выходе выводятся образующий полином g (в виде двоичного вектора), матрица минимальных полиномов minpol, разложение поля на корни минимальных многочленов c, количество гарантированно исправленных ошибок t и проверочная матрица mat.

После выполнения функции [g , minpol , c, mat, t] = bchpoly (63 , 39) получены результаты, представленные в листинге 1.

Листинг 1 – Полученные данные после выполнения функции

g = 1 1 1 0 1 1 1 0 1 1 1 0 0 1 0 0 1 1 0 1 1 0 1 1 1

minpol =

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| c = |  |  |  |  |  |  |

{

[1,1] =

GF(2^6) array. Primitive Polynomial = D^6+D+1 (decimal 67)

Array elements = 1

[1,2] =

GF(2^6) array. Primitive Polynomial = D^6+D+1 (decimal 67) Array elements =

2 4 9 12 16 19

[1,3] =

GF(2^6) array. Primitive Polynomial = D^6+D+1 (decimal 67) Array elements =

3 5 8 13 17 18

[1,4] =

GF(2^6) array. Primitive Polynomial = D^6+D+1 (decimal 67) Array elements =

32 36 38 47 48 60

[1,5] =

GF(2^6) array. Primitive Polynomial = D^6+D+1 (decimal 67) Array elements =

6 11 20 26 28 31

[1,6] =

GF(2^6) array. Primitive Polynomial = D^6+D+1 (decimal 67) Array elements =

15 22 24

[1,7] =

GF(2^6) array. Primitive Polynomial = D^6+D+1 (decimal 67) Array elements =

34 35 44 45 52 53

[1,8] =

GF(2^6) array. Primitive Polynomial = D^6+D+1 (decimal 67) Array elements =

7 10 21 27 29 30

[1,9] =

GF(2^6) array. Primitive Polynomial = D^6+D+1 (decimal 67) Array elements =

40 43 51 54 57 62

[1,10] =

GF(2^6) array. Primitive Polynomial = D^6+D+1 (decimal 67) Array elements =

58 59

[1,11] =

GF(2^6) array. Primitive Polynomial = D^6+D+1 (decimal 67) Array elements =

41 42 50 55 56 63

[1,12] =

GF(2^6) array. Primitive Polynomial = D^6+D+1 (decimal 67) Array elements =

14 23 25

[1,13] =

GF(2^6) array. Primitive Polynomial = D^6+D+1 (decimal 67) Array elements =

33 37 39 46 49 61

}

mat =

Columns 1 through 27:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |

Columns 28 through 54:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |

Columns 55 through 63:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1  t = 4 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |

## Кодирование информационного вектора

При помощи функции encode будет закодирован заданный по условию информационный вектор. Результат работы данной функции представлен в листинге 2.

Листинг 2 – Полученные данные после выполнения функции encode

>> msg\_enc = encode(msg, n, k, "bch")' msg\_enc =

Columns 1 through 27:

|  |  |  |
| --- | --- | --- |
| 0 1 0 0 0 1 1 0 | 0 0 0 0 0 | 1 1 1 0 1 0 1 1 1 0 1 0 0 0 |
| Columns 28 through 54: |  |  |
| 0 0 0 0 0 0 0 0 | 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| Columns 55 through 63: |  |  |
| 0 0 0 1 1 0 0 0 | 0 |  |

## Внесение ошибки

Для выполнения этой части работы необходимо внести ошибки в 41 и 16 битах (листинг 3), использовать XOR с уже закодированным сообщением msg\_enc (листинг 4) и декодировать сообщение (листинг 5).

Листинг 3 – Создание битов с ошибками

>> Err Err =

Columns 1 through 27:

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

Columns 28 through 54:

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

Columns 55 through 63:

0 0 0 0 0 0 0 0 0

Листинг 4 – Создание ошибочного сообщения

>> msg\_err = xor(msg\_enc, Err) msg\_err =

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Columns 1 through 36: |  | | | | | | |
| 0 1 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 1 0 1 | 1 1 0 1 | 0 | 0 0 0 0 | 0 | 0 0 0 0 | 0 | 0 |
| Columns 37 through 63: |  |  |  |  |  |  |  |
| 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0 1 1 0 | 0 | 0 0 |  |  |  |  |

Листинг 5 – Декодирование сообщения с ошибкой

>>msg\_dec = decode(msg\_err, n, k, "bch")' msg\_dec =

Columns 1 through 27:

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Columns 28 through 39:

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

## Сравнение

Используя метод Монте-Карло, необходимо сравнить вероятностные характеристики двух кодов БЧХ в соответствии с вариантом 2. В таблице 2 приведены коды БЧХ для сравнения.

Таблица 2 – Коды БЧХ для сравнения

|  |  |  |
| --- | --- | --- |
| **№** | **Код 1** | **Код 2** |
| 2 | (15, 7) | (31, 11) |

Листинг 6 приводит данные, полученные в ходе выполнения программы, проводящей сравнение. Полный код приведен в приложении А. На рисунке 1 изображен график сравнения.

Листинг 6 – Полученные значения

>> stat stat =

Columns 1 through 4:

4.485714285714286e-02 7.585714285714286e-02 1.105714285714286e-01

1.734285714285714e-01

1.900000000000000e-02 3.754545454545454e-02 7.172727272727274e-02

1.644545454545454e-01

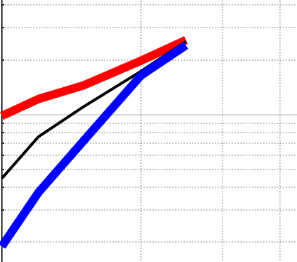
9.871428571428571e-02 1.221428571428571e-01 1.448571428571429e-01

1.981428571428571e-01

Column 5: 2.525714285714286e-01

2.402727272727272e-01

2.571428571428572e-01



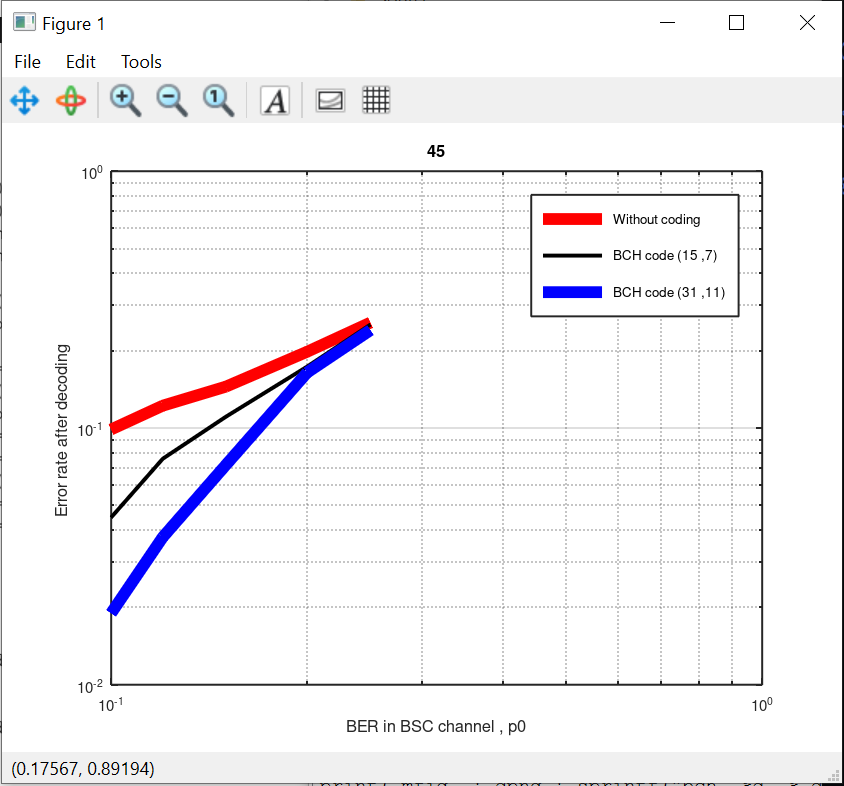


Рисунок 1 – Сравнительный график

import numpy as np

import matplotlib.pyplot as plt

from commpy.channelcoding.bch import BCH, bch\_encode, bch\_decode

from commpy.channels import bsc

from commpy.utilities import bitarray2dec, dec2bitarray

# Параметры BCH кодов

n1, k1 = 15, 7

n2, k2 = 31, 11

# Задание описания для графика

s1 = f"BCH code ({n1}, {k1})"

s2 = f"BCH code ({n2}, {k2})"

s3 = "Without coding"

# Вероятности ошибок в канале

p0 = [1e-1, 1.2e-1, 1.5e-1, 2e-1, 2.5e-1]

stat = np.zeros((3, len(p0)))

# Создание случайных сообщений

msg1 = np.random.randint(0, 2, (1000, k1))

msg2 = np.random.randint(0, 2, (1000, k2))

# Инициализация BCH кодеров

bch1 = BCH(n1, k1)

bch2 = BCH(n2, k2)

# Кодирование сообщений

menc1 = np.array([bch1.encode(msg) for msg in msg1])

menc2 = np.array([bch2.encode(msg) for msg in msg2])

# Обработка каждого значения вероятности ошибки в канале

for i, p in enumerate(p0):

# Пропускание через BSC канал

mrec1 = bsc(menc1, p)

mrec2 = bsc(menc2, p)

mrec3 = bsc(msg1, p)

# Декодирование полученных сообщений

mdec1 = np.array([bch1.decode(msg) for msg in mrec1])

mdec2 = np.array([bch2.decode(msg) for msg in mrec2])

# Вычисление битовых ошибок для каждого случая

stat[0, i] = np.mean([np.sum(m1 != m2) for m1, m2 in zip(msg1, mdec1)]) / k1

stat[1, i] = np.mean([np.sum(m1 != m2) for m1, m2 in zip(msg2, mdec2)]) / k2

stat[2, i] = np.mean([np.sum(m1 != m2) for m1, m2 in zip(msg1, mrec3)]) / k1

# Построение графиков

plt.figure()

# График без кодирования

L3, = plt.loglog(p0, stat[2, :], 'r', linewidth=3, label=s3)

# График для BCH (15,7)

L1, = plt.loglog(p0, stat[0, :], 'k', linewidth=1, label=s1)

# График для BCH (31,11)

L2, = plt.loglog(p0, stat[1, :], 'b', linewidth=3, label=s2)

# Настройка заголовка и меток осей

plt.title(f"BCH codes ({n1}, {k1}) and ({n2}, {k2}) in BSC channel")

plt.xlabel("BER in BSC channel, p0")

plt.ylabel("Error rate after decoding")

plt.legend()

plt.grid(True)

# Сохранение графика в файл

plt.savefig(f"bch-{n1}-{k1}\_cycl-{n2}-{k2}\_bsc\_err\_rate.png")

plt.show()