МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

имени М.В. Ломоносова

Факультет вычислительной математики и кибернетики

Отчет по заданию по Прикладной Алгебре

Задание выполнил

студент 327 учебной группы факультета ВМК МГУ

Малоян Нарек Гагикович

Москва, 2017 г.

Постановка задачи

В задании выдаётся список всех примитивных многочленов степени q над полем F_2 для всех $q = 2,\ldots,16$. В этом списке каждый многочлен представлен десятичным числом, двоичная запись которого соответствует коэффициентам полинома над F_2, начиная со старшей степени. Для выполнения задания требуется:

1. Реализовать основные операции в поле F_2^q : сложение, умножение, деление, решение СЛАУ, поиск минимального многочлена из $F_2[x]$ для заданного набора корней из поля F_2^q
2. Реализовать основные операции для работы с многочленами из $F_2^q[x]$: произведение многочленов, деление многочленов с остатком, расширенный алгоритм Евклида для пары многочленов, вычисление значения многочлена для набора элементов из F_2^q;
3. Реализовать процедуру систематического кодирования для циклического кода, заданного своим порождающим многочленом;
4. Реализовать процедуру построения порождающего многочлена для БЧХ-кода при заданных n и t;
5. Построить графики зависимости скорости БЧХ-кода $r = k/n$ от количества исправляемых кодом ошибок t для различных значений n. Какие значения t следует выбирать на практике для заданного n?
6. Реализовать процедуру вычисления истинного минимального расстояния циклического кода d, заданного своим порождающим многочленом, путем полного перебора по всем $2k-1$ кодовым словам. Привести пример БЧХ-кода, для которого истинное минимальное расстояние больше, чем величина $2t+1$;
7. Реализовать процедуру декодирования БЧХ-кода с помощью метода PGZ и на основе расширенного алгоритма Евклида. Провести сравнение двух методов декодирования по времени работы;
8. С помощью метода стат. испытаний реализовать процедуру оценки доли правильно раскодированных сообщений, доли ошибочно раскодированных сообщений и доли отказов от декодирования для БЧХ-кода. С помощью этой процедуры убедиться в том, что БЧХ-код действительно позволяет гарантированно исправить до t ошибок. Может ли БЧХ-код исправить больше, чем t ошибок? Как ведут себя характеристики кода при числе ошибок, превышающем t?

Реализация

Все задания были выполнены в соответсвии с описаниями алгоритмов в документе с описанием задания. Также для упрощения анализа кода и процесса дебагинга были добавлены подробные комментарии перед каждым действием.

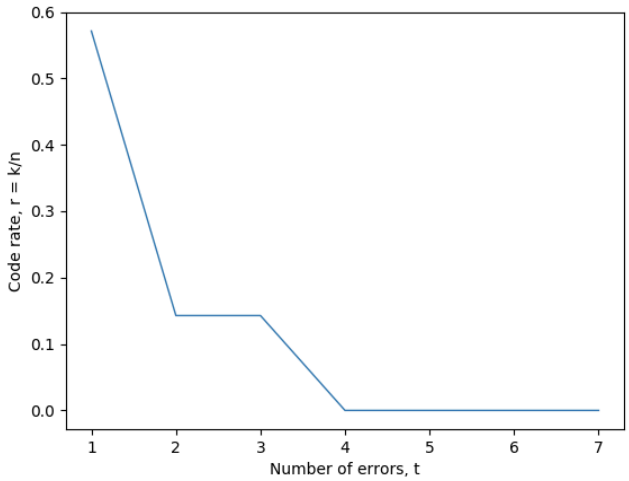
Пункты 1, 2 выполнены в файле gf.py с подробным описанием всех действий в комментариях.

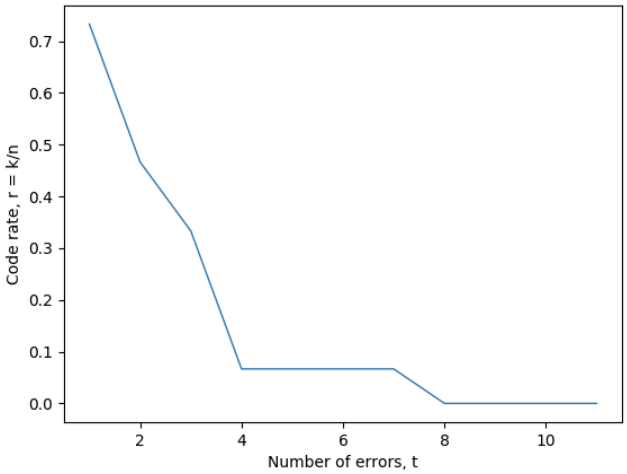
Пункты 3, 4, 6, 7 выполнены в файле bch.py

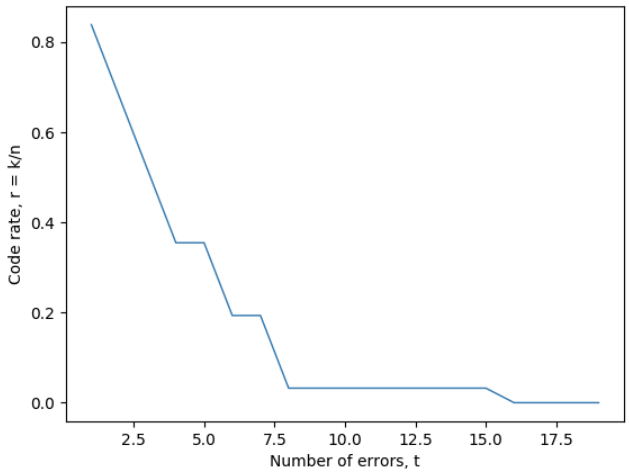
Выполнение 5, 7, 8 пунктов описаны ниже.

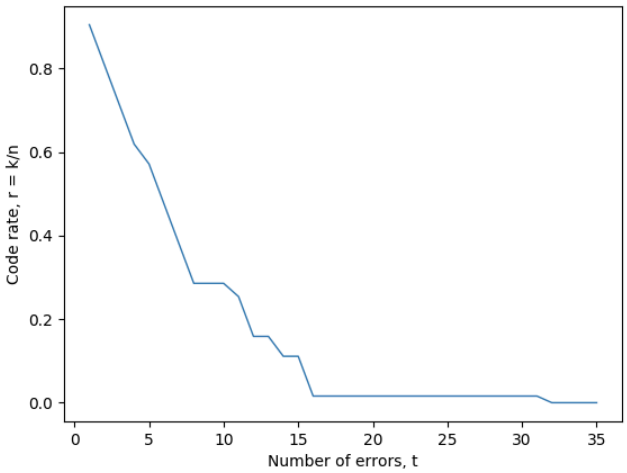
Эксперименты

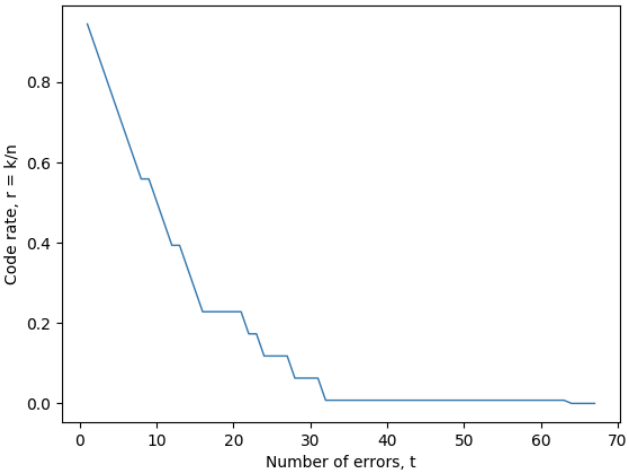
5) В ходе данного эксперимента были построены графики зависимости скорости БЧХ-кода $r = k/n$ от количества исправляемых кодом ошибок t для различных значений $n = 7,15,31,63,127,255,511$. Очевидно, что выбор оптимального t зависит от r, поэтому конкретную цифру дать нельзя, но можно сказать, что надо выбирать такие значения t, которые находятся правее всего на прямой, параллельной оси абсцисс. Например, для $n = 7$ (первый график) при $r = 0.15$ оптимально выбрать именно $t = 3$.

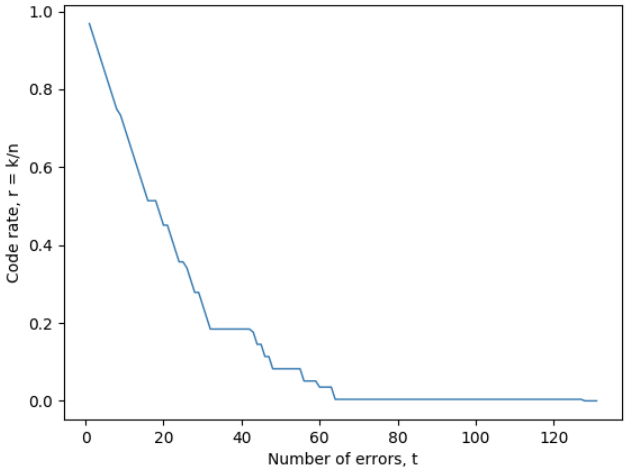


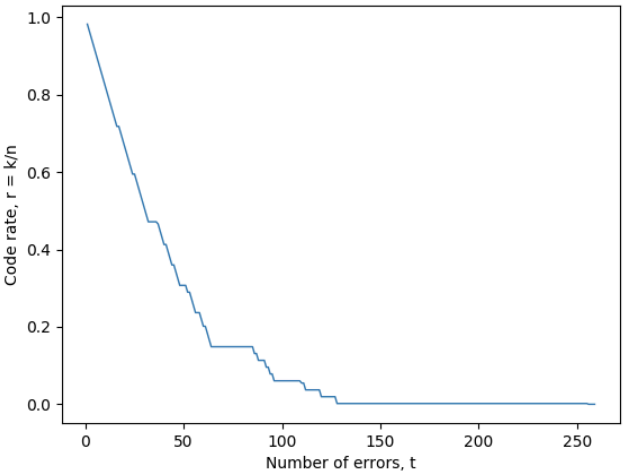







Code rate, r = k/n
0.0
0.2
0.4
0.6
0.8
0
10
20
30
40
50
60
70
Number of errors, t


Code rate, r = k/n
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80
100
120
Number of errors, t


Code rate, r = k/n
0.0
0.2
0.4
0.6
0.8
1.0
0
50
100
150
200
250
Number of errors, t

7) Так как для количества ошибок равную t метод PGZ будет иметь преимущество, т.к. ответ будет получаться на первой итерации, то было решено исследовать время работы при одной ошибке.

n	t	Euclid(sec)	PGZ(sec)
7	1	0,17	0,13
15	2	0,25	0,24
31	4	0,41	0,47
63	8	0,69	0,92
127	11	1,30	1,65

В принципе, при одной ошибке нельзя определенно дать ответ на вопрос какой метод работает быстрее, в отличие от ситуации, когда количество ошибок равно t - PGZ.

8) Были написаны функции: генерирующие сообщения, вносящие ошибки, проверяющие корректность работы декодера. Используя данные функции был проведен эксперимент, в ходе которого можно было убедиться, что при количестве ошибок r, не превышающих количество исправлений ошибок t, декодер отрабатывает со 100% точностью. При $r > t$ мы могли наблюдать, опять таки, предсказуемую картину. Точность срабатываний 0%. При этом возникали либо некорректные обработки, или вовсе отказ от декодирования.

n	t	$r \leq t$ (Корректно; Некорректно; Ошибка декодирования)	$r > t$ (Корректно; Некорректно; Ошибка декодирования)
7	1	(1,0;0,0;0,0)	(0,0;1,0;0,0)
15	2	(1,0;0,0;0,0)	(0,0;0.5;0.5)
31	4	(1,0;0,0;0,0)	(0,0;0,0;1,0)
63	8	(1,0;0,0;0,0)	(0,0;0,0;1,0)
127	11	(1,0;0,0;0,0)	(0,0;0,0;1,0)