Московский Государственный Университет им. М.В.Ломоносова
Факультет Вычислительной Математики и Кибернетики
Кафедра Суперкомпьютеров и Квантовой Информатики

Курс: Прикладная алгебра
Практическое задание. Конечные поля и коды БЧХ

Работу выполнил
Мокров К.С.
323 группа

Москва 2018

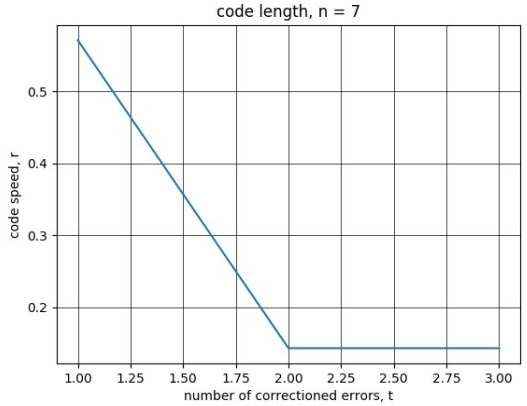
Содержание

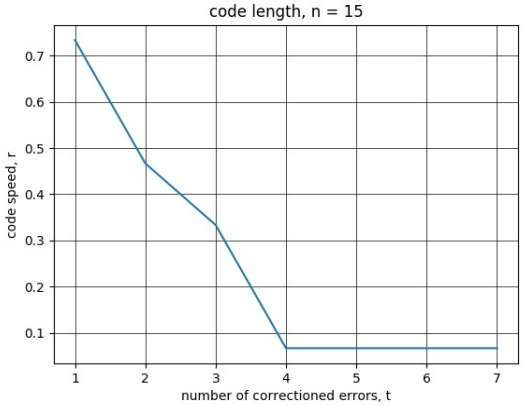
Постановка задачи:

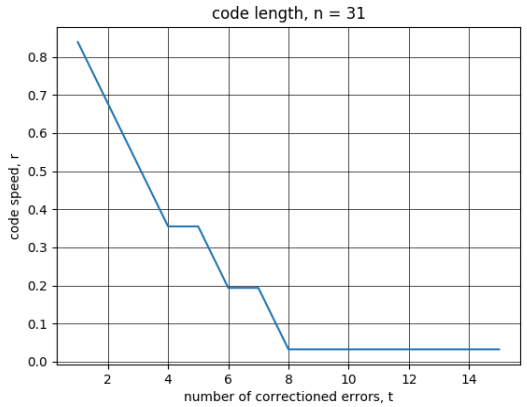
1. Реализовать основные операции в поле F^q_2: сложение, умножение, деление, решение СЛАУ, поиск минимального многочлена из $F_2[x]$ для заданного набора корней из поля F^q_2;
2. Реализовать основные операции для работы с многочленами из $F^q_2[x]$: произведение многочленов, деление многочленов с остатком, расширенный алгоритм Евклида для пары многочленов, вычисление значения многочлена для набора элементов из F^q_2;
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, путём полного перебора по всем 2^k - 1 кодовым словам. Привести пример БЧХ-кода для которого истинное минимальное расстояние больше, чем величина 2 * t + 1;
7. Реализовать процедуру декодирования БЧХ-кода с помощью метода PGZ и на основе расширенного алгоритма Евклида. Провести сравнения 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?

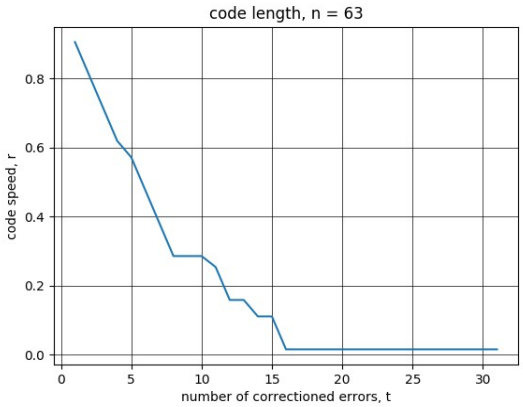
Скорость БЧХ-кода

Для n = 7, 15, 31, 63 построены функции зависимости скорости кода, от количества исправляемых ошибок r(t) = k / n, где n — длина кодового слова, k — количество «полезных» символов. Для фиксированного n, t изменяется от 1 до (n - 1) / 2.









Как видно из графиков, функция r(t) получилась убывающей, ступенчатой. Тогда на практике для заданного значения n, нужно выбрать минимально допустимое для конкретной ситуации значение скорости кода r и взять t: r(t - 1) >= r(t) > r(t + 1). То есть наибольшее значение аргумента, на котором достигается это значение функции. Это позволит исправлять больше возможных ошибок при выбранной скорости кода.

Истинное минимальное кодовое расстояние

Был реализован метод dist(self) класса BCH, который вычисляет истинное минимальное расстояние циклического кода d, заданного своим порождающим многочленом, путём полного перебора по всем кодовым словам. Примеры БЧХ-кодов, для которых истинное минимальное расстояние больше, чем величина 2 * t + 1: (7, 2), (15, 4), (31, 8), (63, 16) — для этих пар истинное кодовое расстояние равно соответствующей n, то есть все они вырождаются в тождественные. Но неравенство реального кодового расстояния конструктивному, можно наблюдать не только с тождественными кодами: (31, 4) d(констр) = 2 * 4 + 1 = 9, однако реальное кодовое расстояние оказывается равным 11, то есть совпадает с реальным и кодовым (31, 5).

Так же можно заметить, что функции кодового расстояния и скорости кода имеют участки постоянства на одинаковых наборах t.

Сравнение двух методов декодирования по времени работы

Была реализована функция, которая конструирует класс БЧХ с параметрами (n, t), создаёт набор из 150 сообщений, кодирует их, в зависимости от аргумента функции, инвертирует в кодовых словах 1 или t бит и измеряет время декодирования с помощью процедур на основе расширенного алгоритма Евклида и метода PGZ(Peterson-Gorenstein-Zierler).

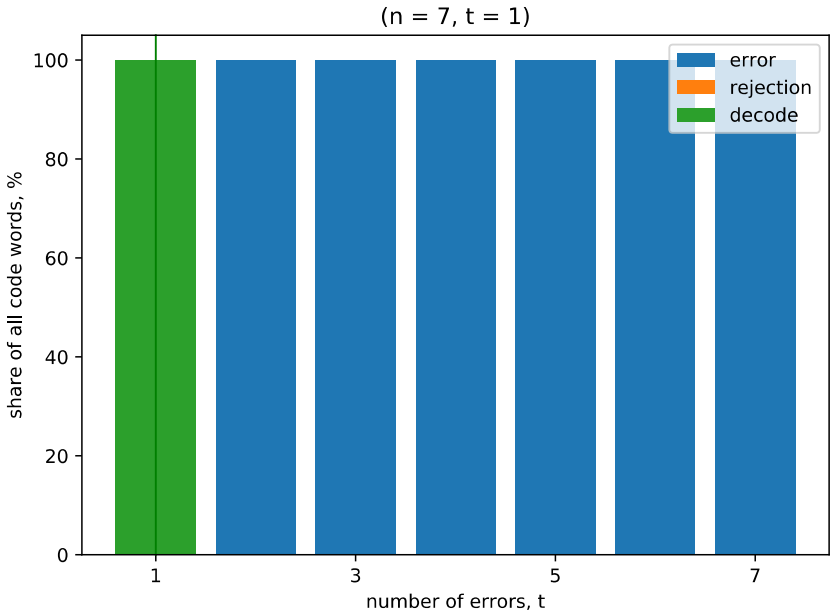
		Допущено ошибок			
		1		t	
n	t	Euclid	PGZ	Euclid	PGZ
7	1	0.051624	0.053723	0.051263	0.053622
15	3	0.128407	0.21542	0.191255	0.234463
31	5	0.314682	0.620921	0.529857	0.720749
61	4	0.500933	0.903766	0.733691	1.051602
63	11	1.13483	2.684522	2.122831	3.22281
127	12	2.379766	5.165709	4.111729	6.412052
127	21	3.965925	10.87388	7.717248	13.041571

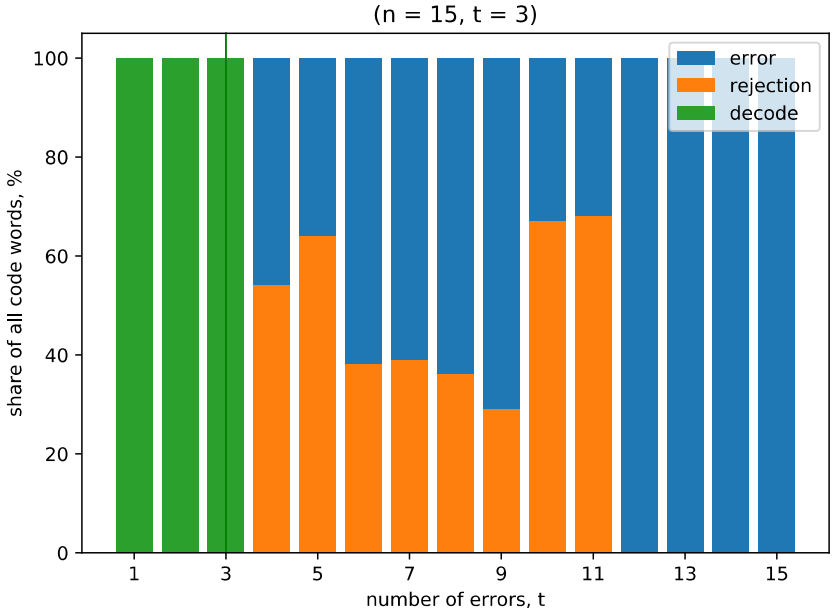
По таблице видно, что алгоритм Евклида работает быстрее при любых парах (n, t) и любом количестве ошибок.

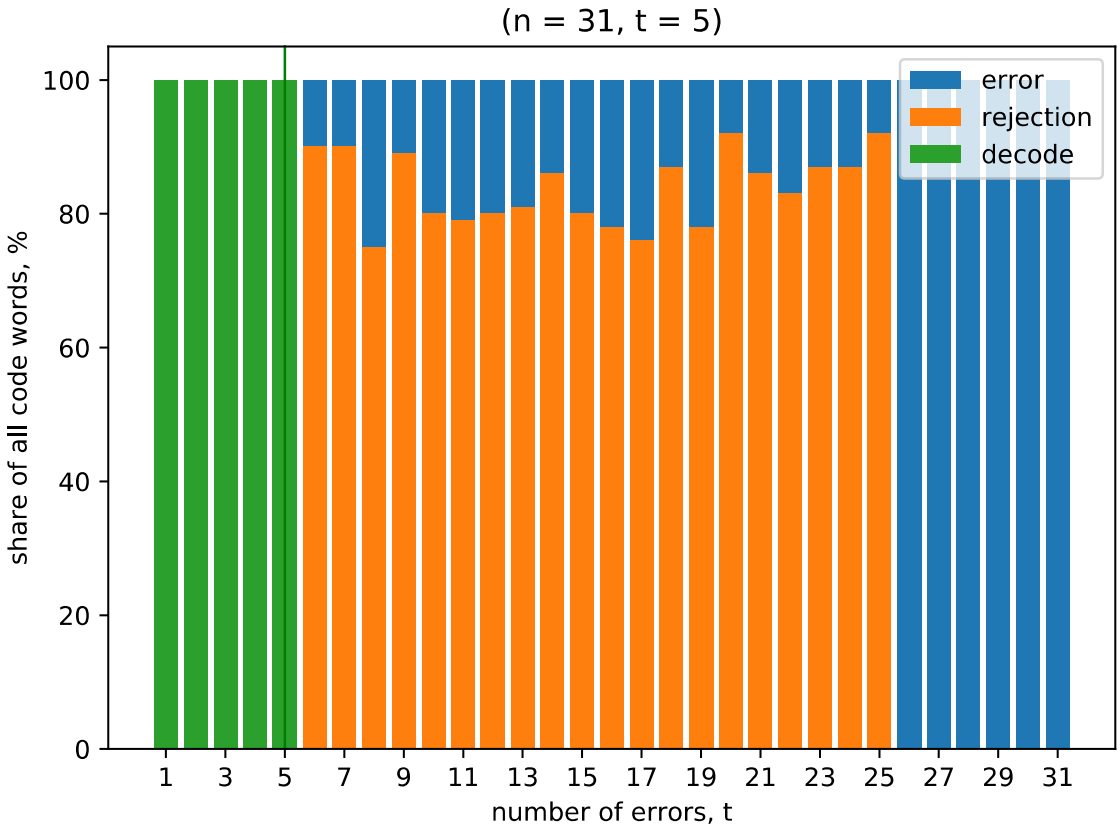
Обе процедуры работают быстрее для одной сделанной ошибки, чем для t, так как надо меньше исправлять. Помимо этого, метод PGZ имеет ещё одно преимущество, при одной допущенной ошибке, ведь не надо решать большую СЛАУ, метод t - 1 раз быстро выходит из функции решения системы с ответом numpy.nan, а на последней итерации решает СЛАУ маленького размера.

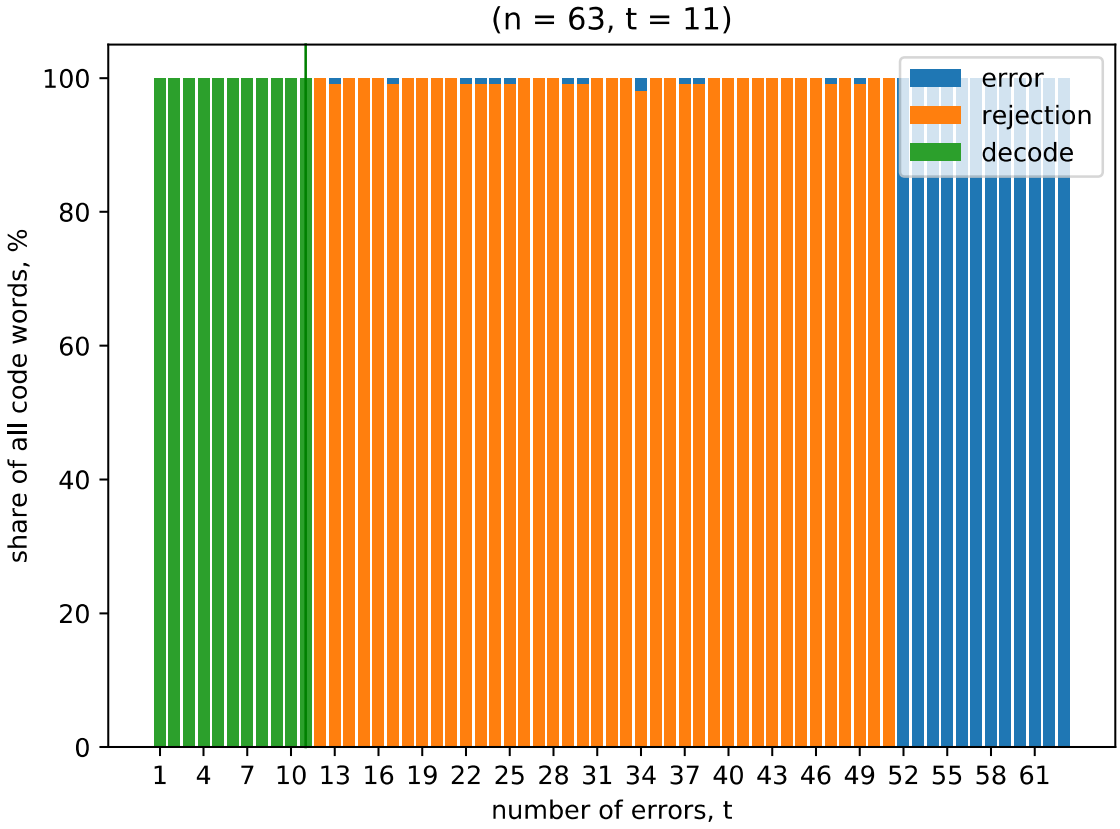
Процедура статистических испытаний

Оценим долю правильно раскодированных сообщений, доли ошибочно раскодированных сообщений и доли отказов от декодирования для БЧХ-кода. Для этой цели была написана функция, которая на вход получает параметры n и t, по ним конструирует класс БЧХ, создаёт 100 сообщений, кодирует их и собирает данные о результатах работы декодера, когда в кодовых словах допущена 1,2, …, n ошибок.






(n = 31, t = 5)
error
rejection
decode
share of all code words, %
0
20
40
60
80
100
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31
number of errors, t


(n = 63, t = 11)
error
rejection
decode
share of all code words, %
0
20
40
60
80
100
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61
number of errors, t

Как можно убедиться по графикам, БЧХ-код действительно позволяет гарантированно исправить до t ошибок. Но больше он исправить не способен. Как только количество реально сделанных ошибок превышает t, счётчик правильно декодированных слов становится равен нулю и резко возрастает количество отказов и ошибочно декодированных слов. Также по графикам видно, что когда происходит n - t ошибок, или больше, то декодер перестаёт выдавать отказы. Это происходит, потому что кодовые слова с ошибками попадают в шары радиуса t с центром в других кодовых словах, именно их и выдаёт декодер, но на этапе сравнения с исходными словами, становится понятно, что произошли ошибки.