Министерство образования и науки РФ
Федеральное государственное автономное образовательное учреждение
высшего профессионального образования
Санкт-Петербургский государственный университет аэрокосмического
приборостроения

Е.А.Крук, А.А.Овчинников

Основы теории кодирования

Учебное пособие

Санкт-Петербург
2013

УДК 621.391.251
ББК 32.811.4
К84

Рецензенты:
профессор кафедры вычислительной техники НИУ ИТМО,
доктор технических наук В.А.Богатырев;
доцент кафедры распределенных вычислений и компьютерных сетей
СПбГПУ,
кандидат технических наук П.В.Трифонов

Утверждено редакционно-издательским советом университета
в качестве учебного пособия

Крук Е.А., Овчинников А.А.

К84 Основы теории кодирования: учебное пособие / Е.А. Крук, А.А. Овчинников. СПб.: ГУАП, 2013. – 107 с.:ил.
ISBN 978-5-8088-0866-9

Настоящее пособие является конспектом лекций по теории помехоустойчивого кодирования. В пособии рассматриваются основы конечных полей, линейные блоковые коды, а также коды с алгебраической структурой, широко применяющиеся на практике — коды БЧХ и Рида-Соломона. Большое внимание уделено общим методам декодирования линейных кодов — декодированию по информационным совокупностям с использованием перестановок и покрывающих полиномов. Также рассматриваются классические алгоритмы декодирования циклических кодов.

Учебное пособие предназначено для студентов, обучающихся по направлениям 090900 «Информационная безопасность», 210700 «Инфокоммуникационные технологии и системы связи», может быть использовано для самостоятельной работы студентов и при выполнении заданий по НИР.

УДК 621.391.251
ББК 32.811.4

ISBN 978-5-8088-0866-9

Предисловие

Теория связи, как наука о «безошибочной» передаче информации, насчитывает, по меркам науки, не так уж много лет. В 1948 г. вышла в свет работа профессора Массачусетского технологического института К.Э.Шеннона «Математическая теория связи». Эта работа и стала точкой отсчета новой теории.

Инженеры-связисты вкладывали (и вкладывают) огромные усилия в создание технических средств – линий связи, устройств передачи и приема информации, способных повысить надежность связи. Надежность любых технических средств, однако, конечна. Поэтому в «дошенноновские» времена казалось очевидным, что поскольку искажения – явление природное, то и избавиться от ошибок при передаче информации полностью невозможно.

Основным результатом названной выше работы Шеннона было доказательство теорем, получивших название теорем кодирования теории информации. Эти теоремы утверждают, что существует метод передачи информации по каналу связи с ошибками, обеспечивающий сколь угодно малую вероятность ошибочного приема информации при условии, что скорость передачи не превышает некоторой константы, зависящей от свойств канала и называемой его (канала) пропускной способностью.

Теоремы Шеннона были, однако, теоремами существования, т.е. они доказывали, что решение задачи надежной связи существует, но не указывали пути достижения этого решения. Поиском эффективных методов передачи (кодирования передаваемой информации) занимается теория помехоустойчивого кодирования, введением в которую и может служить настоящее пособие.

Пособие представляет собой курс лекций, в разное время читавшихся в Санкт-Петербургском государственном университете аэрокосмического приборостроения. В нем рассматриваются классические методы кодирования и дается регулярное решение задачи построения кодов, исправляющих заданное число ошибок. Для этого в первом разделе излагаются основы теории конечных полей, которая используется в последующих разделах для построения кодов и методов их декодирования. Второй раздел описывает общие способы задания линейных кодов, их характеристики и границы. Третий посвящен заданию специального класса линейных кодов – циклических кодов Боуза–Чоудхури–Хоквингема (БЧХ) и кодов Рида–Соломона (РС). Эти коды являются наиболее широко применимыми на практике. Четвертый и пятый раздел посвящены задачам декодирования линейных кодов в общем случае

и алгебраическому декодированию циклических кодов, соответственно. Разделы снабжены задачами, которые могут лечь в основу практических и лабораторных занятий.

Авторы хотели бы выразить свою глубокую признательность аспиранту ГУАП Дарье Ильиной за помощь при подготовке этого пособия.

1. Конечные поля

Делая арифметические операции с действительными числами, мы не задаемся вопросом, можно ли их выполнить. Сложение, вычитание, умножение и деление (конечно, не на ноль) всегда определены. Это связано с тем, что действительные числа образуют поле, алгебраическую структуру, которая определяется таким образом, чтобы над ее элементами были возможны арифметические действия. Правила этих действий таковы, что позволяют находить решение уравнения $ax+b=c$.

Поле действительных чисел не является единственным. Большое значение для практики, в частности кодирования сообщений, имеют поля с конечным (в отличие от поля действительных чисел) числом элементов. Такие поля носят название *конечных полей* или *полей Галуа*. В этом разделе будут описаны основные свойства полей Галуа.

1.1. Определение поля

Полем F называется множество элементов, для каждой пары которых определены две операции: сложение ($\oplus$) и умножение ($\otimes$), обладающие следующими свойствами [10, 14]:

1. **Замкнутость**. Для любых $\alpha, \beta \in F$:

$$\alpha \oplus \beta = \gamma; \quad \gamma \in F;$$
$$\alpha \otimes \beta = \theta; \quad \theta \in F.$$

2. **Ассоциативность**. Для любых $\alpha, \beta, \gamma \in F$:

$$\alpha \oplus (\beta \oplus \gamma) = (\alpha \oplus \beta) \oplus \gamma;$$
$$\alpha \otimes (\beta \otimes \gamma) = (\alpha \otimes \beta) \otimes \gamma.$$

3. **Коммутативность**. Для любых $\alpha, \beta \in F$:

$$\alpha \oplus \beta = \beta \oplus \alpha;$$
$$\alpha \otimes \beta = \beta \otimes \alpha.$$

4. **Существование нейтральных элементов**. В поле F существует нейтральный элемент по сложению, называемый *нулем поля* (0), такой, что $\alpha \oplus 0 = 0 \oplus \alpha = \alpha$; $\forall \alpha \in F$. Нуль поля в каждом поле единственный.

В поле F существует нейтральный элемент по умножению, называемый *единицей поля* (1), такой, что $\alpha \otimes 1 = 1 \otimes \alpha = \alpha$; $\forall \alpha \in F$. Единица поля в каждом поле единственна.

5. **Существование обратных элементов**. Для любого $\alpha \in F$ в поле F существует единственный аддитивный обратный элемент β (обратный по сложению), такой, что $\alpha \oplus \beta = \beta \oplus \alpha = 0$. Аддитивный обратный элемент к α принято обозначать $-\alpha$.

Для любого ненулевого элемента $\alpha \in F$ в поле F существует единственный мультипликативный обратный элемент γ, такой, что $\alpha \otimes \gamma = \gamma \otimes \alpha = 1$. Мультипликативный обратный элемент к α обозначают α^{-1}.

6. **Дистрибутивность**. Для любых $\alpha, \beta, \gamma \in F$:

$$\alpha \otimes (\beta \oplus \gamma) = (\alpha \otimes \beta) \oplus (\alpha \otimes \gamma).$$

Пример 1.1. Поле из двух элементов $F = \{0, 1\}$. Результаты операций определяются следующим образом:

$$\begin{aligned} 0 \oplus 0 &= 0; & 0 \otimes 0 &= 0; \\ 1 \oplus 1 &= 0; & 1 \otimes 1 &= 1; \\ 0 \oplus 1 = 1 \oplus 0 &= 1; & 0 \otimes 1 = 1 \otimes 0 &= 0. \end{aligned}$$

Умножение в этом поле аналогично умножению целых чисел, а сложение отличается тем, что результат вычисляется по модулю 2.

Поля Галуа, содержащие q элементов, обозначаются $GF(q)$, в частности, поле из рассмотренного примера – $GF(2)$. Если $q = p$ – простое число (делящееся нацело только на себя и на 1), то можно определить так называемое *простое* поле $GF(p)$ как множество целых чисел от 0 до $p-1$, $F = \{0, 1, \ldots, p-1\}$, где сложение и умножение определяются по модулю p (т. е. элементы поля складываются и умножаются как целые числа, а полученное в результате число делится на p. Остаток от деления и есть результат операции в конечном поле). Операция нахождения остатка от деления целого a на ненулевое целое b обозначается как $a \bmod b$ (читается: «a по модулю b»).

Пример 1.2. Поле $GF(5)$. $F = \{0, 1, 2, 3, 4\}$. Умножим 3 на 4 в поле $GF(5)$:

$$3 \times 4 = 12; \quad 12 \bmod 5 = 2. \quad \text{Итак, } 3 \otimes 4 = 2.$$

Сложим эти же два элемента:

$$3 + 4 = 7; \quad 7 \bmod 5 = 2. \quad \text{Итак, } 3 \oplus 4 = 2.$$

Сложение и умножение в $GF(5)$ можно задать и при помощи таблиц (рис. 1.1).

$\oplus$	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

$\otimes$	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

Рис. 1.1. Операции в поле $GF(5)$

Для задания операций вычитания «$\ominus$» и деления «$/$» используют наличие в каждом поле обратных элементов относительно сложения и умножения. Так, $\alpha \ominus \beta = \alpha \oplus (-\beta)$, где $-\beta$ является аддитивным обратным к элементу β. Аналогично $\alpha/\beta = \alpha \otimes \beta^{-1}$, где β^{-1} является мультипликативным обратным к элементу β.

В случае простого поля $GF(p)$ нахождение обратных элементов осуществляется также по правилам модульной арифметики. Так, для любого $\alpha \in GF(p)$ аддитивное обратное находится как [2, 9]:

$$-\alpha = p - \alpha.$$

Нахождение мультипликативного обратного является не столь простой задачей. Во-первых, заметим, что мультипликативного обратного для 0 не существует, т. е. в конечных полях, как и в поле вещественных чисел, делить на 0 нельзя. В остальных случаях требуется применение расширенного алгоритма Евклида. Сокращенная версия этого алгоритма, непосредственно используемая для нахождения мультипликативного обратного по модулю простого числа p, приведена на рис. 1.2. После выполнения инициализации основной шаг алгоритма повторяется для $i = 0, 1, \ldots$, вычисляя остаток R_{i+1} от деления R_{i-1} на R_i (операция $\lfloor \cdot \rfloor$ обозначает взятие целой части от деления) до тех пор, пока очередное R_{i+1} не станет равным 0, тогда y_i с предыдущего шага и определит искомое значение.

Пример 1.3. Вернемся к рассмотрению поля $GF(5)$. Вычислим $3 \ominus 4$:

$$3 - 4 = -1; \quad -1 \bmod 5 = 5 - 1 = 4. \quad \text{Итак, } 3 \ominus 4 = 4.$$

Заданная таким образом операция вычитания является обратной к операции сложения. Чтобы убедиться в этом, вычислим:

$$4 \oplus 4 = (4 + 4) \bmod 5 = 3.$$

Вход: p – простое, $\alpha \in \{1, 2, \ldots, p-1\}$.
Выход: Мультипликативное обратное $\alpha^{-1} \bmod p$.
Инициализация:

$$R_{-1} = p; \quad R_0 = \alpha;$$
$$y_{-1} = 0; \quad y_0 = 1.$$

Основной шаг алгоритма:
Для $i = 0, 1, 2, \ldots$ выполнять:

$$q_i = \lfloor R_{i-1}/R_i \rfloor;$$
$$R_{i+1} = R_{i-1} - R_i q_i;$$
$$y_{i+1} = y_{i-1} - y_i q_i.$$

Условие остановки:
Если $R_{i+1} = 0$, то $\alpha^{-1} \bmod p = y_i \bmod p$.

Рис. 1.2. Нахождение $\alpha^{-1} \bmod p$, p – простое

Поделим 3 на 4. Для этого нам надо найти $4^{-1} \bmod 5$. Выполним алгоритм, приведенный на рис. 1.2. При инициализации имеем $R_{-1} = 5$, $R_0 = 4$. Для шага $i = 0$

$$q_0 = \lfloor R_{-1}/R_0 \rfloor = \lfloor 5/4 \rfloor = 1;$$
$$R_1 = R_{-1} - R_0 q_0 = 5 - 4 \cdot 1 = 1;$$
$$y_1 = y_{-1} - y_0 q_0 = 0 - 1 \cdot 1 = -1.$$

Так как $R_1 \neq 0$, выполним еще один шаг для $i = 1$:

$$q_1 = \lfloor R_0/R_1 \rfloor = \lfloor 4/1 \rfloor = 4;$$
$$R_2 = R_0 - R_1 q_1 = 4 - 1 \cdot 4 = 0.$$

Так как $R_2 = 0$, искомым обратным является $y_1 \bmod p = -1 \bmod 5 = 4$.

Итак, $4^{-1} \bmod 5 = 4$, т. е. в поле $GF(5)$ число 4 обратно самому себе. Тогда

$$3/4 = 3 \cdot 4^{-1} \bmod 5 = 3 \cdot 4 \bmod 5 = 2.$$

Заданное таким образом деление является операцией, обратной к умножению, так как

$$2 \otimes 4 = (2 \cdot 4) \bmod 5 = 3.$$

Как видно из примеров, элементы конечных полей, как и вещественные числа, можно складывать, вычитать, умножать и делить, используя заданные операции.

В соответствии с определением задать поле значит задать множество его элементов F и указать правила сложения и умножения. При этом сами элементы поля могут быть произвольными. Это могут быть числа, но могут быть и векторы, матрицы или полиномы и т.д. Однако конечные поля обладают важным свойством *изоморфизма*, согласно которому все поля с одинаковым числом элементов определенным образом эквивалентны (изоморфны), т. е. каждому элементу $(\cdot)$ одного поля F можно взаимнооднозначно поставить в соответствие элемент $(\cdot)'$ другого поля F', так что, если

$$\alpha \oplus \beta = \gamma; \quad \alpha \otimes \beta = \theta, \quad \text{то}$$

$$\alpha' \oplus \beta' = \gamma'; \quad \alpha' \otimes \beta' = \theta',$$

где α', β', γ', θ' – элементы из F', соответствующие элементам α, β, γ, θ из F, т. е. соответствие переводит сумму в сумму и, аналогично, произведение в произведение. Свойство изоморфизма позволяет рассматривать различные поля с одинаковым числом элементов как различные представления одного и того же поля и использовать для проведения операций те представления, в которых эти операции удобно выполнять. Так, для поля $GF(p)$, p – простое число, удобно представить элементы поля целыми числами от 0 до $p-1$, а действия – с помощью арифметики по модулю p.

1.2. Свойства полей Галуа

Определение 1.1. *Мультипликативным порядком* ненулевого элемента α поля $GF(q)$ называется наименьшее положительное число m, для которого $\alpha^m = 1$.

Теорема 1.1. *Порядок любого ненулевого элемента $GF(q)$ является делителем числа $q-1$.*

Доказательство. Пусть $\alpha \in GF(q)$ имеет порядок m. Тогда элементы

$$\alpha^0 = 1, \quad \alpha^1, \quad \alpha^2, \quad \ldots, \quad \alpha^{m-1} \tag{1.1}$$

представляют собой различные элементы поля. Пусть $\varepsilon_1 \in GF(q)$, $\varepsilon_1 \neq 0$, но не входит в последовательность (1.1). Тогда элементы

$$\alpha^0 \cdot \varepsilon_1, \quad \alpha^1 \cdot \varepsilon_1, \quad \ldots, \quad \alpha^{m-1} \cdot \varepsilon_1$$

различны между собой. Аналогично построим последовательности

$$\begin{array}{llll} \alpha^0 \cdot \varepsilon_2, & \alpha^1 \cdot \varepsilon_2, & \ldots, & \alpha^{m-1} \cdot \varepsilon_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots & & & \\ \alpha^0 \cdot \varepsilon_{i-1}, & \alpha^1 \cdot \varepsilon_{i-1}, & \ldots, & \alpha^{m-1} \cdot \varepsilon_{i-1}, \end{array} \tag{1.2}$$

используя в качестве порождающего элемента i-й последовательности ε_{i-1} – ненулевой элемент поля $GF(q)$, который не принадлежит ранее построенным последовательностям.

Процесс построения последовательностей (1.2) можно продолжать до тех пор, пока в них не будут перечислены все ненулевые элементы поля. Пусть N – максимальное число последовательностей типа (1.2). Для любых i и j и любых различных порождающих элементов ε_l и ε_s

$$\varepsilon_l \cdot \alpha^i \neq \varepsilon_s \cdot \alpha^j,$$

так как противное означало бы, что $\varepsilon_l = \varepsilon_s \cdot \alpha^{j-i}$, т. е. что ε_l и ε_s порождают одну и ту же последовательность. Следовательно, все последовательности типа (1.2) содержат одинаковое число элементов m и ни один элемент не входит в две такие последовательности. Следовательно, произведение числа последовательностей N на число элементов m в них равно числу ненулевых элементов поля

$$Nm = q - 1. \tag{1.3}$$

Соотношение (1.3) доказывает теорему.

Из теоремы 1.1 следует, что для любого элемента $\alpha \in GF(q)$, $\alpha \neq 0$

$$\alpha^{q-1} = \alpha^{mN} = (\alpha^m)^N = 1^N = 1,$$

где m – порядок элемента α. Следовательно, ненулевые элементы поля $GF(q)$ представляют собой все корни многочлена $x^{q-1} - 1$ и

$$x^{q-1} - 1 = \prod_{i=1}^{q-1} (x - \alpha_i), \tag{1.4}$$

где $\alpha_1, \ldots, \alpha_{q-1}$ – различные элементы поля $GF(q)$.

Определение 1.2. *Примитивным элементом* поля из q элементов называется элемент, у которого мультипликативный порядок в точности равен $q - 1$.

Если α – примитивный элемент, то элементы $\alpha, \alpha^2, \ldots, \alpha^{q-1}$ различны и, следовательно, содержат все ненулевые элементы поля.

Теорема 1.2. *Любое поле Галуа содержит примитивный элемент.*

Доказательство. Пусть $\alpha_1, \alpha_2, \ldots, \alpha_{q-1}$ – ненулевые элементы поля Галуа $GF(q)$, а $m_1, \ldots, m_{q-1}$ – мультипликативные порядки этих элементов соответственно.

Обозначим через m наименьшее общее кратное чисел m_1, ..., m_{q-1}:

$$m = \text{НОК}\{m_1, \ldots, m_{q-1}\}.$$

Тогда для любого элемента α_i выполняется $\alpha_i^m = 1$, и все $q-1$ ненулевых элементов поля являются корнями многочлена $x^m - 1$. Но тогда $m \geq q-1$, поскольку $x^m - 1$ должен иметь не менее, чем $q-1$ корней. Это в сочетании с теоремой 1.1 доказывает теорему.

Доказанная теорема показывает, что элементы конечного поля могут быть заданы степенями его примитивного элемента.

Определение 1.3. *Подполем* некоторого поля называется поле, элементы которого представляют собой подмножество элементов первоначального поля с операциями сложения и умножения, совпадающими с операциями первоначального поля. Если поле F имеет подполе F_1, то говорят, что F является *расширением* F_1.

Теорема 1.3. *Любое поле Галуа содержит простое подполе Галуа, причем это простое подполе единственно.*

Доказательство. Любое поле F содержит элементы 0 и 1. Будем рассматривать элементы $\sum_{i=1}^{2} 1 = 1+1$; $\sum_{i=1}^{3} 1 = 1+1+1$ и т.д. Если эти числа, которые мы будем называть *числами поля*, не все различны, то существуют такие целые числа m и n, что $\sum_{i=1}^{m} 1 = \sum_{i=1}^{n} 1$, или

$$\sum_{i=1}^{n-m} 1 = 0. \tag{1.5}$$

Тогда сложение чисел поля является сложением по модулю $(n-m)$. Множество чисел замкнуто относительно умножения, т.к.

$$\left(\sum_{i=1}^{n} 1\right) \times \left(\sum_{i=1}^{m} 1\right) = \sum_{i=1}^{nm} 1.$$

Тогда с учетом равенства (1.5) умножение чисел поля также производится по модулю $(n-m)$. Если $(n-m)$ есть произведение целых чисел j_1 и j_2, $n-m = j_1 j_2$, то

$$\left(\sum_{i=1}^{j_1}\right) \times \left(\sum_{i=1}^{j_2} 1\right) = \sum_{i=1}^{j_1 j_2} 1 = \sum_{i=1}^{n-m} 1 = 0,$$

что невозможно, поскольку $(\sum_{i=1}^{j_1} 1)$ и $(\sum_{i=1}^{j_2} 1)$ являются ненулевыми элементами поля F. Следовательно, $n - m$ – простое число. Но множество чисел по модулю простого p является полем. Таким образом, числа поля представляют собой простое подполе поля F. Любое другое подполе F содержит числа поля F в качестве подполя. Поэтому для окончательного доказательства теоремы достаточно, действуя так же, как при доказательстве теоремы 1.1, показать, что число элементов поля делится на число элементов подполя.

Определение 1.4. Число p элементов в простом подполе поля F называется *характеристикой* этого поля.

Многочлен $f(x) = \sum_{i=0}^{m} f_i x^i$ степени m называется *многочленом над полем* F, если его коэффициенты $f_i \in F$, $i = \overline{0, m}$.

Многочлен $f(x)$ называется *неприводимым* над полем F, если его нельзя представить в виде произведения многочленов ненулевой степени над этим полем.

Элемент $\alpha \in F$ является корнем многочлена $f(x)$ над полем F, если

$$f(\alpha) = \sum_{i=0}^{m} f_i \alpha^i = 0. \tag{1.6}$$

Действия в выражении (1.6) производятся в поле F.

Определение 1.5. Пусть E является подполем поля F. Тогда *минимальный многочлен* $f_\alpha(x)$ элемента $\alpha \in F$ над полем E определяется как многочлен над E наименьшей степени, у которого α является корнем.

Теорема 1.4. *Для любого подполя E поля $GF(q)$ каждому ненулевому элементу $\alpha \in GF(q)$ соответствует, и при этом единственный, минимальный многочлен $f_\alpha(x)$ над полем E. Многочлен $f_\alpha(x)$ неприводим над E.*

Доказательство. Все элементы поля $GF(q)$ являются, в соответствии с (1.4), корнями многочлена $x^{q-1} - 1$, который является нормированным многочленом над любым подполем (его коэффициенты 0 и 1 имеются в любом подполе) поля $GF(q)$. Следовательно, $f_\alpha(x)$ существует. Этот многочлен неприводим, так как если бы он раскладывался в произведение $f_\alpha(x) = g(x)h(x)$ многочленов меньшей степени, то условие

$$f_\alpha(\alpha) = g(\alpha)h(\alpha) = 0$$

означало бы, что имеется многочлен меньшей, чем $f_\alpha(x)$, степени ($g(x)$ или $h(x)$), для которого α является корнем. Далее, любой многочлен

$\varphi(x)$ над E можно представить в виде

$$\varphi(x) = f_\alpha(x)m(x) + r(x),$$

где $r(x)$ – многочлен степени меньшей, чем $f_\alpha(x)$. Тогда

$$\varphi(\alpha) = f_\alpha(\alpha)m(\alpha) + r(\alpha) = r(\alpha),$$

поэтому $\varphi(\alpha) = 0$ тогда и только тогда, когда $r(\alpha) = 0$. Так как $f_\alpha(x)$ – минимальный многочлен, а $r(x)$ имеет меньшую степень, чем $f_\alpha(x)$, то из условия $r(\alpha) = 0$ следует, что $r(x) = 0$. Следовательно, α является корнем $\varphi(x)$ в том и только в том случае, когда $\varphi(x)$ делится на $f_\alpha(x)$. Отсюда следует, что если $\varphi(x)$ и $f_\alpha(x)$ имеют одинаковую степень и имеют α в качестве корня, то

$$\varphi(x) = m(x)f_\alpha(x) = \mu f_\alpha(x),$$

где $m(x) = \mu$ – многочлен нулевой степени. Откуда с учетом нормированности минимального многочлена следует, что $\varphi(x) = f_\alpha(x)$.

Из доказанной теоремы следует, что если $\alpha_1, \alpha_2, \ldots, \alpha_{q-1}$ – ненулевые элементы $GF(q)$, то

$$x^{q-1} - 1 = \text{НОК}\{f_{\alpha_1}(x), f_{\alpha_2}(x), \ldots, f_{\alpha_{q-1}}(x)\}. \tag{1.7}$$

Мы доказали, что каждому элементу поля соответствует единственный минимальный многочлен, однако у некоторых различных элементов поля их минимальные многочлены могут совпадать.

Определение 1.6. Элементы поля, минимальные многочлены которых равны, называются *сопряженными*.

Для того, чтобы определить, какие элементы являются сопряженными, используют понятие циклотомических классов.

Определение 1.7. Множество целых чисел по модулю $p^m - 1$ относительно операции умножения на p распадается на непересекающиеся подмножества, называемые *циклотомическими классами по модулю* $p^m - 1$.

Таким образом, два элемента α^i и α^j поля $GF(p^m)$ являются сопряженными, если их степени i и j принадлежат одному циклотомическому классу.

Пример 1.4. Пусть $p = 2$, $m = 4$. Тогда циклотомическими классами по модулю $2^4 - 1 = 15$ являются:

$$C_0 = \{0\};$$
$$C_1 = \{1, 2, 4, 8\};$$
$$C_3 = \{3, 6, 12, 9\};$$
$$C_5 = \{5, 10\};$$
$$C_7 = \{7, 14, 13, 11\}.$$

В качестве индексов C_i указаны так называемые *представители* классов, в данном случае в этом качестве выбраны наименьшие числа класса (это удобно, хотя представителем класса может быть любой его элемент).

Отсюда видно, что, например, в поле $GF(2^4)$ сопряженными являются элементы α и α^8, а также α^5 и α^{10}.

Пользуясь теоремой 1.4, теперь легко доказать теорему.

Теорема 1.5. *Число элементов в поле Галуа $GF(q)$ равно степени его характеристики p,*

$$q = p^m,$$

где m – некоторое натуральное число.

Доказательство. Пусть α – примитивный элемент поля, а $f_\alpha(x) = x^m + \sum_{i=0}^{m-1} f_i x^i$ – минимальный многочлен над полем $GF(p)$. Тогда

$$f_\alpha(\alpha) = \alpha^m + \sum_{i=0}^{m-1} f_i \alpha^i = 0,$$

или

$$\alpha^m = -\sum_{i=0}^{m-1} f_i \alpha^i.$$

Умножая обе части последнего равенства на α, получим

$$\alpha^{m+1} = -\sum_{i=1}^{m-1} f_{i-1}\alpha^i - f_{m-1}\alpha^m =$$
$$= -\sum_{i=1}^{m-1} f_{i-1}\alpha^i + f_{m-1}\left(\sum_{i=0}^{m-1} f_i \alpha^i\right) = \sum_{i=0}^{m-1} f'_i \alpha^i,$$

где f'_i – элементы поля $GF(p)$, полученные как коэффициенты при α^i при сложении $-\sum_{i=1}^{m-1} f_{i-1}\alpha^i$ и $\sum_{i=0}^{m-1} f_{m-1} f_i \alpha^i$. Аналогично, все элементы поля $0, \alpha^0, \ldots, \alpha^{q-2}$ можно представить в виде

$$\alpha^j = \sum_{i=0}^{m-1} \varphi_i \alpha^i. \tag{1.8}$$

Это представление единственно. Действительно, пусть для некоторого α^j существует два представления в виде (1.8):

$$\alpha^j = \sum_{i=0}^{m-1} \varphi'_i \alpha^i = \sum_{i=0}^{m-1} \varphi''_i \alpha^i,$$

тогда

$$\sum_{i=0}^{m-1} (\varphi'_i - \varphi''_i)\alpha^i = 0.$$

Последнее равенство означает, что α является корнем многочлена степени $m-1 < m$, что противоречит определению $f_\alpha(x)$. Таким образом, все элементы $(\sum_{i=0}^{m-1} \varphi_i \alpha^i)$ являются различными элементами поля, причем все элементы поля представимы в виде таких элементов. Число элементов (1.8) равно p^m, и теорема доказана.

Из доказанных теорем следует, что если α – примитивный элемент поля $GF(p^m)$ и $f_\alpha(x)$ – минимальный многочлен этого элемента, то минимальное i, при котором двучлен $x^i - 1$ делится на $f_\alpha(x)$, равно $p^m - 1$. Многочлены, обладающие этим свойством, называются *примитивными многочленами*. Такие многочлены некоторых степеней приведены в табл. 1.1.

1.3. Вычисления в конечных полях

Для организации вычислений в поле $GF(p^m)$ удобно использовать два представления этого поля: векторным пространством и степенями примитивного элемента.

Пусть $f(x) = \sum_{i=0}^{m-1} f_i x^i + x^m$ – примитивный многочлен степени m над полем $GF(p)$. Обозначим через α корень этого многочлена и будем выписывать степени α:

$$\begin{aligned} &\alpha^0 = 1; \\ &\alpha^1 = \alpha^1; \\ &\dots \\ &\alpha^{m-1} = \alpha^{m-1}. \end{aligned} \tag{1.9}$$

С учетом того, что α – корень $f(x)$, получим

$$\alpha^m = -(f_{m-1} \cdot \alpha^{m-1} + f_{m-2} \cdot \alpha^{m-2} + \ldots + f_1 \cdot \alpha + f_0);$$

и далее

Таблица 1.1

Примитивные многочлены для некоторых конечных полей

Характеристика поля	Степень многочлена	Примитивные многочлены
$p = 2$	2	$x^2 + x + 1$
	3	$x^3 + x + 1$
	4	$x^4 + x + 1$
	5	$x^5 + x^2 + 1$
	6	$x^6 + x + 1$
	7	$x^7 + x^3 + 1$
	8	$x^8 + x^4 + x^3 + x^2 + 1$
	9	$x^9 + x^4 + 1$
	10	$x^{10} + x^3 + 1$
	11	$x^{11} + x^2 + 1$
	12	$x^{12} + x^6 + x^4 + x + 1$
	13	$x^{13} + x^4 + x^3 + x + 1$
	14	$x^{14} + x^{10} + x^6 + x + 1$
	15	$x^{15} + x + 1$
	16	$x^{16} + x^{12} + x^3 + x + 1$
	17	$x^{17} + x^3 + 1$
	18	$x^{18} + x^7 + 1$
	19	$x^{19} + x^5 + x^2 + x + 1$
	20	$x^{20} + x^3 + 1$
	21	$x^{21} + x^2 + 1$
	22	$x^{22} + x + 1$
	23	$x^{23} + x^5 + 1$
	24	$x^{24} + x^7 + x^2 + x + 1$
$p = 3$	2	$x^2 + x + 2$
	3	$x^3 + 2x + 1$
	4	$x^4 + x + 2$
	5	$x^5 + 2x + 1$
	6	$x^6 + x + 2$
$p = 5$	2	$x^2 + x + 2$
	3	$x^3 + 3x + 2$
	4	$x^4 + x^2 + 2x + 2$

$$
\begin{aligned}
\alpha^{m+1} &= \alpha \cdot \alpha^m = -\alpha(f_{m-1} \cdot \alpha^{m-1} + f_{m-2} \cdot \alpha^{m-2} + \dots \\
&\quad + f_1 \cdot \alpha + f_0) = \\
&= -f_{m-1} \cdot \alpha^m - (f_{m-2} \cdot \alpha^{m-1} + \dots \\
&\quad + f_1 \cdot \alpha^2 + f_0 \cdot \alpha) = \\
&= f_{m-1} \cdot (f_{m-1} \cdot \alpha^{m-1} + f_{m-2} \cdot \alpha^{m-2} + \dots + f_0) - \\
&\quad - (f_{m-2} \cdot \alpha^{m-1} + \dots + f_0 \cdot \alpha) = \\
&= f'_{m-1} \cdot \alpha^{m-1} + f'_{m-2} \cdot \alpha^{m-2} + \dots + f'_1 \cdot \alpha + f'_0; \\
&\dots \\
\alpha^{i+1} &= \alpha \cdot \alpha^i; \quad i = m+1, m+2, \dots
\end{aligned}
\tag{1.10}
$$

Сложение и умножение коэффициентов при степенях α (1.10) проводятся в простом поле $GF(p)$, т. е. по модулю p.

Продолжая аналогично (1.10), мы получим последовательность степеней α,

$$
\alpha^0, \alpha, \dots, \alpha^{p^m-2}, \tag{1.11}
$$

различных между собой. Так как $f(x)$ является делителем $x^{p^m-1} - 1$, то элемент $\alpha^{p^m-1} = 1$. Добавим к элементам (1.10) элемент 0 и будем считать, что это все p^m элементов поля $GF(p^m)$:

$$
F = \{0, 1, \alpha, \dots, \alpha^{p^m-2}\}.
$$

Поставим в соответствие элементам из F векторы коэффициентов при $\alpha^0, \dots, \alpha^{m-1}$ в выражениях (1.9), (1.10) (нулевому элементу поставим в соответствие нулевой вектор):

$$
\begin{array}{lcccccccc}
0 & \longleftrightarrow & 0 & 0 & 0 & \dots & 0 & 0 \\
1 & \longleftrightarrow & 1 & 0 & 0 & \dots & 0 & 0 \\
\alpha & \longleftrightarrow & 0 & 1 & 0 & \dots & 0 & 0 \\
\alpha^2 & \longleftrightarrow & 0 & 0 & 1 & \dots & 0 & 0 \\
\dots\dots & & & & & & & & \dots\dots \\
\alpha^{m-1} & \longleftrightarrow & 0 & 0 & 0 & \dots & 0 & 1 \\
\alpha^m & \longleftrightarrow & -f_0 & -f_1 & -f_2 & \dots & -f_{m-2} & -f_{m-1} \\
\alpha^{m+1} & \longleftrightarrow & f'_0 & f'_1 & f'_2 & \dots & f'_{m-2} & f'_{m-1} \\
\dots\dots & & & & & & & & \dots\dots
\end{array}
\tag{1.12}
$$

Элемент α называется *примитивным элементом* поля $GF(p^m)$. Соотношение (1.12) задает два представления этого поля: с помощью степеней примитивного элемента и в виде m-векторов над $GF(p)$. Эти два представления поля $GF(p^m)$ позволяют определить его операции следующим образом:

1) результат сложения двух элементов в $GF(p^m)$ определяется как покомпонентная сумма по модулю p представлений этих элементов в виде векторов;

2) результат умножения двух элементов в $GF(p^m)$ определяется как произведение представлений этих элементов в виде степеней примитивного элемента поля (умножение проводится с учетом $\alpha^{p^m-1} = 1$).

Заметим, что многочлен, который в выражении (1.10) соответствует некоторому α^i, может быть получен как $x^i \bmod f(x)$ (с действиями в поле $GF(p)$). Если $f(x)$ – многочлен степени m над $GF(p)$, то всего существует как раз p^m остатков от деления на $f(x)$ (включая нулевой). Тогда поле $GF(p^m)$ может быть задано как множество таких остатков с операциями сложения и умножения (а также вычитания и деления) по модулю $f(x)$. Отметим, что определенное нами выше в п.1 сложение как раз соответствует правилам сложения многочленов.

Однако для операции умножения (и в дальнейшем деления) в п.2 используется представление в виде степеней примитивного элемента, так как для такого представления выполнять указанные операции гораздо проще. Вместе с тем, если используется поле, в котором количество $q = p^m$ элементов столь велико, что соответствие (1.12) не может храниться в памяти, для умножения и деления в поле используют умножение и деление многочленов по модулю $f(x)$.

Пример 1.5. Построим поле $GF(2^4)$. В качестве примитивного используем многочлен $f(x) = x^4 + x + 1$. Можно проверить, что его нельзя разложить на множители над $GF(2)$ и что двучлены $x^5 - 1$, …, $x^{14} - 1$ не делятся на него.

Пусть α – корень многочлена $f(x)$. Построение поля $GF(2^4)$ приведено на рис. 1.3.

Сложим элементы α^{10} и α^6. Найдем их представления в виде векторов:

$$\alpha^{10} \longrightarrow (1110), \quad \alpha^6 \longrightarrow (0011),$$

тогда

$$\alpha^{10} + \alpha^6 \longrightarrow (1110) + (0011) = (1101) \longrightarrow \alpha^7.$$

Таким образом,

$$\alpha^{10} + \alpha^6 = \alpha^7.$$

Так как вычитание по модулю 2 совпадает со сложением, в поле $GF(2^4)$ эти две операции совпадают.

Перемножим векторы (1101) и (1110). Найдем их представления в поле

$$(1101) \longrightarrow \alpha^7; \quad (1110) \longrightarrow \alpha^{10}.$$

		α^0	α^1	α^2	α^3
$0 = 0$	$\longleftrightarrow$	(0	0	0	0)
$\alpha^0 = 1$	$\longleftrightarrow$	(1	0	0	0)
$\alpha^1 = \alpha^1$	$\longleftrightarrow$	(0	1	0	0)
$\alpha^2 = \alpha^2$	$\longleftrightarrow$	(0	0	1	0)
$\alpha^3 = \alpha^3$	$\longleftrightarrow$	(0	0	0	1)
$\alpha^4 = \alpha + 1$	$\longleftrightarrow$	(1	1	0	0)
$\alpha^5 = \alpha^2 + \alpha$	$\longleftrightarrow$	(0	1	1	0)
$\alpha^6 = \alpha^3 + \alpha^2$	$\longleftrightarrow$	(0	0	1	1)
$\alpha^7 = \alpha^4 + \alpha^3 = \alpha^3 + \alpha + 1$	$\longleftrightarrow$	(1	1	0	1)
$\alpha^8 = \alpha^4 + \alpha^2 + \alpha = \alpha^2 + 1$	$\longleftrightarrow$	(1	0	1	0)
$\alpha^9 = \alpha^3 + \alpha$	$\longleftrightarrow$	(0	1	0	1)
$\alpha^{10} = \alpha^4 + \alpha^2 = \alpha^2 + \alpha + 1$	$\longleftrightarrow$	(1	1	1	0)
$\alpha^{11} = \alpha^3 + \alpha^2 + \alpha$	$\longleftrightarrow$	(0	1	1	1)
$\alpha^{12} = \alpha^4 + \alpha^3 + \alpha^2 = \alpha^3 + \alpha^2 + \alpha + 1$	$\longleftrightarrow$	(1	1	1	1)
$\alpha^{13} = \alpha^4 + \alpha^3 + \alpha^2 + \alpha = \alpha^3 + \alpha^2 + 1$	$\longleftrightarrow$	(1	0	1	1)
$\alpha^{14} = \alpha^4 + \alpha^3 + \alpha = \alpha^3 + 1$	$\longleftrightarrow$	(1	0	0	1)
$\alpha^{15} = \alpha^4 + \alpha = 1$	$\longleftrightarrow$	(1	0	0	0)

Рис. 1.3. Поле $GF(2^4)$

Тогда

$$(1101) \times (1110) \longrightarrow \alpha^7 \times \alpha^{10} = \alpha^{17} = \alpha^2 \longrightarrow (0010).$$

Таким образом,

$$(1101) \times (1110) = (0010).$$

Обратим внимание, что с учетом указанного выше замечания, произведение (1101) и (1110) можно было бы вычислить иначе, воспользовавшись умножением соответствующих многочленов по модулю $f(x)$. В самом деле,

$$(1101) \longrightarrow x^3 + x + 1; \quad (1110) \longrightarrow x^2 + x + 1$$

и

$$(x^3 + x + 1) \times (x^2 + x + 1) \bmod (x^4 + x + 1) = \\ = (x^5 + x^4 + 1) \bmod (x^4 + x + 1) = x^2 \longrightarrow (0010),$$

что совпадает с предыдущим результатом.

Уже зная, что $\alpha^7 \times \alpha^{10} = \alpha^2$, вычислим α^2/α^{10}:

$$\frac{\alpha^2}{\alpha^{10}} = \frac{1}{\alpha^8} = \frac{\alpha^{15}}{\alpha^8} = \alpha^7.$$

Выполнение этой операции с помощью многочленов потребовало бы нахождения мультипликативного обратного по модулю $f(x)$ с привлечением расширенного алгоритма Евклида. Мы не будем рассматривать такой пример здесь.

Пример 1.6. Построим поле $GF(3^2)$. В качестве примитивного многочлена возьмем многочлен $f(x) = x^2 + x + 2$. Пусть α – корень многочлена $f(x)$. Поле $GF(3^2)$ приведено на рис. 1.4.

Сложим α^2 и α^4:

$$\alpha^2 + \alpha^4 \longrightarrow (12) + (20) = (02) \longrightarrow \alpha^5.$$

Заметим, что действия над векторами в поле $GF(3^2)$ выполняются по модулю 3. Тогда

$$\alpha^6 - \alpha^4 \longrightarrow (21) - (20) = (01) \longrightarrow \alpha.$$

Операции умножения и деления с помощью степеней примитивного элемента полностью аналогичны приведенным в примере 1.5.

Пример 1.7. Построим поле $GF(4^2)$. Прежде всего составим таблицы сложения и умножения поля $GF(4)$. Для построения поля $GF(4)$ используется многочлен $x^2 + x + 1$. Пусть α – корень этого многочлена. Поле $GF(4)$ представлено на рис. 1.5. Таблицы сложения и умножения для этого поля приведены на рис. 1.6.

Теперь построим поле $GF(4^2)$ с помощью многочлена $x^2 + x + 2$, рис. 1.7.

При выполнении действий, вовлекающих в себя векторное представление этого поля, операции над элементами векторов осуществляются в поле $GF(4)$, т. е. в соответствии с рис. 1.6. Хотя поле $GF(4^2)$ является расширением поля $GF(4)$, его характеристика равна 2, поэтому в нем сложение и вычитание также совпадают, как и в других полях характеристики 2, например $GF(2^4)$.

Как видно из примеров 1.5 – 1.7, одним из возможных представлений поля $GF(p^m)$ является задание его элементов как последовательностей длиной m над $GF(p)$. Таких последовательностей всего существует p^m, и они образуют линейное векторное пространство. В рассмотренных нами примерах базис этого пространства соответствовал степеням примитивного элемента $\alpha^0 = 1$, α, α^2, ..., α^{m-1}. Такой базис стандартен, но конечно же, не единственен. Приведем еще один пример базиса конечного поля.

Определение 1.8. Элементы α, α^p, ..., $\alpha^{p^{m-1}}$ образуют *нормальный базис* поля $GF(p^m)$.

		α^0	α^1
$0 = 0$	$\longleftrightarrow$	(0	0)
$\alpha^0 = 1$	$\longleftrightarrow$	(1	0)
$\alpha^1 = \alpha^1$	$\longleftrightarrow$	(0	1)
$\alpha^2 = -\alpha - 2 = 2\alpha + 1$	$\longleftrightarrow$	(1	2)
$\alpha^3 = 2\alpha^2 + \alpha = 2(2\alpha + 1) + \alpha = 2\alpha + 2$	$\longleftrightarrow$	(2	2)
$\alpha^4 = 2\alpha^2 + 2\alpha = 2$	$\longleftrightarrow$	(2	0)
$\alpha^5 = 2\alpha$	$\longleftrightarrow$	(0	2)
$\alpha^6 = 2\alpha^2 = \alpha + 2$	$\longleftrightarrow$	(2	1)
$\alpha^7 = \alpha^2 + 2\alpha = \alpha + 1$	$\longleftrightarrow$	(1	1)
$\alpha^8 = \alpha^2 + \alpha = 1$			

Рис. 1.4. Поле $GF(3^2)$

		η^0	η^1		
$0 = 0$	$\longleftrightarrow$	(0	0)	$\longleftrightarrow$	0
$\eta^0 = 1$	$\longleftrightarrow$	(1	0)	$\longleftrightarrow$	1
$\eta^1 = \eta$	$\longleftrightarrow$	(0	1)	$\longleftrightarrow$	2
$\eta^2 = \eta + 1$	$\longleftrightarrow$	(1	1)	$\longleftrightarrow$	3

Рис. 1.5. Поле $GF(2^2)$

По определению базиса линейного пространства, любой элемент конечного поля может быть выражен с помощью линейной комбинации базисных элементов.

При реализации конечного поля и операций в нем элементы поля могут быть представлены по-разному. Элемент поля в векторном представлении может быть задан соответствующей последовательностью чисел (например, элемент (02) в поле $GF(3^2)$).

В случае представления поля степенями примитивного элемента элементы поля описываются множеством $\{-\infty, 0, 1, \ldots, q-2\}$, где $-\infty$ соответствует 0, а остальные числа представляют степени примитивного элемента.

Также удобным может быть представление целыми числами $\{0, 1, \ldots, q-1\}$, подобно тому, как в примере 1.7 были представлены элементы поля $GF(4)$. В этом случае 0 и 1 соответствуют элементам 0 и 1 в поле, в остальных случах числу i соответствует элемент поля α^{i-1}. Следует помнить, однако, что простое поле $GF(p)$ можно представить целыми числами $\{0, 1, \ldots, p-1\}$ и определить операции в нем как арифметические операции по модулю p, однако в общем случае, если поле

$\oplus$	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

$\otimes$	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	3	1
3	0	3	1	2

Рис. 1.6. Операции в поле $GF(4)$

		α^0	α^1
$0 = 0$	$\longleftrightarrow$	(0	0)
$1 = 1$	$\longleftrightarrow$	(1	0)
$\alpha = \alpha$	$\longleftrightarrow$	(0	1)
$\alpha^2 = -\alpha - 2 = \alpha + 2$	$\longleftrightarrow$	(2	1)
$\alpha^3 = \alpha^2 + 2\alpha = 3\alpha + 2$	$\longleftrightarrow$	(2	3)
$\alpha^4 = 3\alpha^2 + 2\alpha = \alpha + 1$	$\longleftrightarrow$	(1	1)
$\alpha^5 = \alpha^2 + \alpha = \alpha + 2 + \alpha = 2$	$\longleftrightarrow$	(2	0)
$\alpha^6 = 2\alpha$	$\longleftrightarrow$	(0	2)
$\alpha^7 = 2\alpha^2 = 2\alpha + 3$	$\longleftrightarrow$	(3	2)
$\alpha^8 = 2\alpha^2 + 3\alpha = \alpha + 3$	$\longleftrightarrow$	(3	1)
$\alpha^9 = \alpha^2 + 3\alpha = 2\alpha + 2$	$\longleftrightarrow$	(2	2)
$\alpha^{10} = 2\alpha^2 + 2\alpha = 3$	$\longleftrightarrow$	(3	0)
$\alpha^{11} = 3\alpha$	$\longleftrightarrow$	(0	3)
$\alpha^{12} = 3\alpha^2 = 3\alpha + 1$	$\longleftrightarrow$	(1	3)
$\alpha^{13} = 3\alpha^2 + \alpha = 2\alpha + 1$	$\longleftrightarrow$	(1	2)
$\alpha^{14} = 2\alpha^2 + \alpha = 3\alpha + 3$	$\longleftrightarrow$	(3	3)
$\alpha^{15} = 3\alpha^2 + 3\alpha = 1$			

Рис. 1.7. Поле $GF(4^2)$

$GF(q)$ представлено числами $\{0, 1, \ldots, q-1\}$, правила действий в этом поле определяются примитивным многочленом поля $f(x)$ и не совпадают с операциями с целыми числами по модулю q.

1.4. Существование полей Галуа

В предыдущих разделах мы показали, что если поле Галуа существует, то оно содержит степень простого числа p^m элементов. В настоящем разделе будет показано, что поле $GF(p^m)$ существует для любого p^m.

Теорема 1.6. *Для любых элементов* α *и* β *поля* $GF(q)$ *характеристики* p *и любого целого* i *справедливо*

$$(\alpha + \beta)^{p^i} = \alpha^{p^i} + \beta^{p^i}.$$

Доказательство. Докажем теорему для $i = 1$.

$$(\alpha + \beta)^p = \alpha^p + p\alpha^{p-1}\beta + C_p^2\alpha^{p-2}\beta^2 + \ldots + C_p^{p-1}\alpha\beta^{p-1} + \beta^p. \qquad (1.13)$$

Все слагаемые правой части соотношения (1.13), кроме α^p и β^p, кратны p и, следовательно, равны нулю в поле такой характеристики.

Отсюда

$$(\alpha + \beta)^p = \alpha^p + \beta^p. \qquad (1.14)$$

Теперь теорема доказывается i-кратным возведением в степень выражения (1.14):

$$(\alpha + \beta)^{p^2} = [(\alpha + \beta)^p]^p = (\alpha^p + \beta^p)^p = \alpha^{p^2} + \beta^{p^2}$$

и т.д.

Следствие 1.1. *Если $f(x) = \sum_{i=0}^{n-1} f_i x^i$ – многочлен над полем $GF(p)$, и α – элемент поля $GF(q)$ характеристики p, то при любом $j \geq 0$*

$$f(\alpha^{p^j}) = [f(\alpha)]^{p^j}.$$

Доказательство.

$$\begin{aligned} [f(\alpha)]^{p^j} &= \left[f_{n-1} \cdot \alpha^{n-1} + \left(\sum_{i=0}^{n-2} f_i \cdot \alpha^i \right) \right]^{p^j} = \\ &= f_{n-1}^{p^j} \cdot \left(\alpha^{n-1}\right)^{p^j} + \left(\sum_{i=0}^{n-2} f_i \alpha^i \right)^{p^j}. \end{aligned} \qquad (1.15)$$

Поскольку $f_i \in GF(p)$, то $f_i^{p-1} = 1$ и $f_i^{p^j} = f_i$. Тогда из (1.15) следует, что

$$[f(\alpha)]^{p^j} = f_{n-1}(\alpha^{p^j})^{n-1} + \left(\sum_{i=0}^{n-2} f_i \alpha^i \right)^{p^j}.$$

Производя аналогичные преобразования для $\left(\sum_{i=0}^{n-2} f_i \alpha^i\right)^{p^j}$, затем для $\left(\sum_{i=0}^{n-3} f_i \alpha^i\right)^{p^j}$ и т.д., получим

$$[f(\alpha)]^{p^j} = f_{n-1}(\alpha^{p^j})^{n-1} + f_{n-2}(\alpha^{p^j})^{n-2} + \ldots + f_0 = f(\alpha^{p^j}).$$

Определение 1.9. Определим *формальную производную* многочлена

$$f(x) = \sum_{i=0}^{n-1} f_i x^i$$

как многочлен

$$f'(x) = \sum_{i=1}^{n} i f_i x^{i-1}. \tag{1.16}$$

В определении 1.9 под if_i – операцией умножения целого числа i на элемент поля f_i (вообще говоря, не определенной), понимается i-кратное сложение элемента f_i с собой, по правилам поля, в котором он задан.

Если $C(x)$ – произведение многочленов $A(x)$ и $B(x)$ степеней n_A и n_B соответственно,

$$\begin{aligned} C(x) = A(x) \cdot B(x) &= \left(\sum_{i=0}^{n_A} a_i x^i\right) \cdot \left(\sum_{i=0}^{n_B} b_i x^i\right) = \\ &= \sum_{j=0}^{n_A+n_B} \left(\sum_{i=0}^{j} a_i b_{j-i}\right) x^j, \end{aligned}$$

то

$$\begin{aligned} C'(x) &= \sum_{j=0}^{n_A+n_B} j \left(\sum_{i=0}^{j} a_i b_{j-i}\right) x^{j-1} = \\ &= \sum_{j=0}^{n_A+n_B} \left(\sum_{i=0}^{j} i a_i b_{j-i} + \sum_{i=0}^{j} (j-i) a_i b_{j-i}\right) x^{j-1} = \\ &= \left(\sum_{i=1}^{n_A} i a_i x^{i-1}\right) \cdot \left(\sum_{i=0}^{n_B} b_i x^i\right) + \\ &\quad + \left(\sum_{i=0}^{n_A} a_i x^i\right) \cdot \left(\sum_{i=1}^{n_B} i b_i x^{i-1}\right) = \\ &= A'(x) \cdot B(x) + A(x) B'(x). \end{aligned}$$

Для формальной производной характерны те же свойства, что и для обычной. Однако, поскольку в конечных полях не определено понятие предела, то при анализе многочленов над конечным полем нельзя использовать понятие обычной производной.

Теорема 1.7. *Пусть $f(x)$ – многочлен над полем $GF(q)$, $f'(x)$ – его формальная производная, и $B(x) =$ НОД $\{f(x), f'(x)\}$. Тогда, чтобы элемент α был корнем кратности i многочлена $f(x)$, необходимо и достаточно, чтобы он был корнем кратности $i-1$ многочлена $B(x)$.*

Доказательство. Докажем необходимость. Если α – корень $f(x)$ кратности i, то $f(x)$ можно представить в виде

$$f(x) = (x-\alpha)^i A(x),$$

тогда формальная производная

$$f'(x) = i(x-\alpha)^{i-1}A(x) + (x-\alpha)^i A'(x)$$

делится на $(x-\alpha)^{i-1}$.

Для доказательства достаточности заметим, что, так как $A(x)$ не делится на $(x-\alpha)$, $f'(x)$ не делится на $(x-\alpha)^i$ и, следовательно, кратность корня в $f'(x)$ всегда на единицу меньше, чем его кратность в $f(x)$.

Используя теоремы 1.6 и 1.7, можно доказать теорему о существовании полей Галуа.

Теорема 1.8. *Для каждого простого p и целого положительного m существует поле Галуа с p^m элементами.*

Доказательство. Рассмотрим многочлен $g(x) = x^{p^m} - x$ как многочлен над полем $GF(p)$. Для доказательства теоремы достаточно показать, что множество корней многочлена $g(x)$ представляет собой поле с p^m элементами.

Прежде всего покажем, что множество корней многочлена $g(x)$ замкнуто относительно сложения и умножения и содержит обратные элементы по сложению и умножению для каждого ненулевого элемента.

Действительно, пусть α и β – корни $g(x)$, тогда, используя теорему 1.6, получим

$$\begin{aligned} g(\alpha+\beta) &= (\alpha+\beta)^{p^m} - (\alpha+\beta) = \\ &= (\alpha^{p^m} - \alpha) + (\beta^{p^m} - \beta) = 0 + 0 = 0; \\ g(\alpha\cdot\beta) &= (\alpha\beta)^{p^m} - \alpha\beta = \alpha^{p^m}\cdot\beta^{p^m} - \alpha\beta = \alpha\beta - \alpha\beta = 0, \end{aligned}$$

аналогично

$$g(-\alpha) = (-\alpha)^{p^m} - (-\alpha) = (-1)^{p^m}\cdot\alpha^{p^m} + \alpha = -\alpha + \alpha = 0$$

и

$$g(\alpha^{-1}) = \alpha^{-p^m} - \alpha^{-1} = (\alpha^{p^m})^{-1} - \alpha^{-1} = \alpha^{-1} - \alpha^{-1} = 0.$$

Теперь для доказательства теоремы достаточно показать, что все p^m корней многочлена $g(x)$ различны. Формальная производная

$$g'(x) = p^m x^{p^m-1} - 1 = -1,$$

так как $p^m = 0$ в поле $GF(p)$. Тогда НОД$\{g(x), g'(x)\}$ не имеет корней, и по теореме 1.7 $g(x)$ не имеет кратных корней.

1.5. Разложение $x^n - 1$ на множители

Как было показано в разделе 1.2, примитивный многочлен, с помощью которого строится поле $GF(p^m)$, является делителем многочлена $x^{p^m-1} - 1$. В этом разделе рассмотрим более общую задачу разложения многочлена $x^n - 1$ на множители над полем $GF(q)$. Помимо нахождения примитивного многочлена при $n = p^m - 1$, эта задача также используется при определении порождающих многочленов циклического кода.

Предположим, что n и q взаимно просты, т. е. НОД$(n, q) = 1$. Тогда существует наименьшее положительное целое m такое, что $q^m - 1$ делится на n. В этом случае многочлен $x^{q^m-1} - 1$ делится на $x^n - 1$, и $x^{q^s-1} - 1$ не делится на $x^n - 1$ ни для каких $s < m$. Тогда корни многочлена $x^n - 1$ лежат в расширении $GF(q^m)$ поля $GF(q)$ и не лежат ни в каком меньшем поле. Кроме того, так как НОД$(n, q) = 1$, то НОД$(x^n - 1, nx^{n-1}) = 1$ (где nx^{n-1} – формальная производная многочлена $x^n - 1$) и, следовательно, все n корней $x^n - 1$ различны.

Таким образом, в поле $GF(q^m)$ существует n различных элементов $\beta_0, \beta_1, \ldots, \beta_{n-1}$, являющихся корнями $x^n - 1$ (их называют корнями n-й степени из 1). Здесь β – *примитивный* корень степени n из единицы, его степенями можно получить все другие степени, т. е. $\beta_i = \beta^i$. Очевидно, в поле $GF(q^m)$ элемент β имеет порядок n (напомним, n – один из делителей $q^m - 1$).

Тогда разложение многочлена $x^n - 1$ над полем $GF(q^m)$, очевидно,

$$x^n - 1 = \prod_{i=0}^{n-1} (x - \beta_i). \tag{1.17}$$

Обобщим понятие циклотомических классов из определения 1.7.

Определение 1.10. *Циклотомический класс по модулю n над $GF(q)$* определяется как

$$C_s = \{s, sq, sq^2, \ldots, sq^{m_s-1}\},$$

где $sq^{m_s} = s \bmod n$. Назовем s *представителем класса.*

Тогда множество целых чисел по модулю n разбивается на циклотомические классы:

$$\bigcup_s C_s = \{0, 1, \ldots, n-1\},$$

где s пробегает множество представителей классов по модулю n. Отметим, что введенное выше число m равно $m_1 = |C_1|$.

Минимальный многочлен элемента β^s поля $GF(q^m)$ является многочленом над $GF(q)$ и равен

$$f_{\beta^s}(x) = \prod_{i \in C_s} (x - \beta^i). \tag{1.18}$$

Наконец, из выражений (1.17) и (1.18) получаем

$$x^n - 1 = \prod_s f_{\beta^s}(x),$$

где s пробегает все множество представителей по модулю n.

Пример 1.8. Пусть $n = 9$, $q = 2$. Найдем разложение $x^9 - 1$ на множители над $GF(2)$. Выпишем циклотомические классы по модулю 9:

$$C_0 = \{0\},$$
$$C_1 = \{1, 2, 4, 8, 7, 5\},$$
$$C_3 = \{3, 6\}.$$

Так как $|C_1| = 6$, корни $x^9 - 1$ лежат в поле $GF(2^6)$. Отметим, что 63 делится на 9, и элемент $\beta = \alpha^7$, где α – примитивный элемент $GF(2^6)$, имеет порядок 9.

Приняв из табл. 1.1, что примитивный многочлен поля $GF(2^6)$ равен $f(x) = x^6 + x + 1$, построим это поле, см. рис. 1.8.

Из (1.18) и с учетом того, что как $GF(2^6)$, так и $GF(2)$ – поля характеристики 2, в которых вычитание и сложение совпадают, получаем, что

$$f_{\beta^0}(x) = \prod_{i \in C_0} (x - \beta^i) = x + 1.$$

Далее,

$$\begin{aligned} f_{\beta^3}(x) = \prod_{i \in C_3} (x - \beta^i) &= (x + (\alpha^7)^3)(x + (\alpha^7)^6) = \\ &= (x + \alpha^{21})(x + \alpha^{42}) = x^2 + \alpha^{21}x + \alpha^{42}x + \alpha^{63} = \\ &= x^2 + x + 1. \end{aligned}$$

Степень	Элемент	Степень	Элемент	Степень	Элемент
$-\infty$	000000	21	110111	42	010111
0	100000	22	101011	43	111011
1	010000	23	100101	44	101101
2	001000	24	100010	45	100110
3	000100	25	010001	46	010011
4	000010	26	111000	47	111001
5	000001	27	011100	48	101100
6	110000	28	001110	49	010110
7	011000	29	000111	50	001011
8	001100	30	110011	51	110101
9	000110	31	101001	52	101010
10	000011	32	100100	53	010101
11	110001	33	010010	54	111010
12	101000	34	001001	55	011101
13	010100	35	110100	56	111110
14	001010	36	011010	57	011111
15	000101	37	001101	58	111111
16	110010	38	110110	59	101111
17	011001	39	011011	60	100111
18	111100	40	111101	61	100011
19	011110	41	101110	62	100001
20	001111				

Рис. 1.8. Поле $GF(2^6)$

Опустив промежуточные вычисления, получим, что

$$\begin{aligned} f_{\beta^1}(x) &= \prod_{i \in C_1} (x - \beta^i) = \\ &= (x + (\alpha^7))(x + (\alpha^7)^2)(x + (\alpha^7)^4) \times \\ &\qquad \times (x + (\alpha^7)^8)(x + (\alpha^7)^7)(x + (\alpha^7)^5) = \\ &= x^6 + x^3 + 1. \end{aligned}$$

Наконец,

$$x^9 - 1 = f_{\beta^0}(x) \cdot f_{\beta^1}(x) \cdot f_{\beta^3}(x) = (x+1)(x^6 + x^3 + 1)(x^2 + x + 1).$$

Разложения некоторых двучленов вида $x^n - 1$ над $GF(2)$ приведены в табл. 1.2.

Таблица 1.2

Разложение $x^n - 1$ на множители

Многочлен	Разложение многочлена над $GF(2)$
$x^7 - 1$	$(x^3 + x + 1)(x^3 + x^2 + 1)(x + 1)$
$x^{15} - 1$	$(x^4 + x + 1)(x^4 + x^3 + 1)\times$ $\times(x^4 + x^3 + x^2 + x + 1)(x^2 + x + 1)(x + 1)$
$x^{31} - 1$	$(x^5 + x^2 + 1)(x^5 + x^4 + x^3 + x^2 + 1)\times$ $\times(x^5 + x^4 + x^2 + x + 1)(x^5 + x^3 + x^2 + x + 1)\times$ $\times(x^5 + x^4 + x^3 + x + 1)(x^5 + x^3 + 1)(x + 1)$
$x^{63} - 1$	$(x^6 + x + 1)(x^6 + x^4 + x^2 + x + 1)\times$ $\times(x^6 + x^5 + x^2 + x + 1)(x^6 + x^3 + 1)\times$ $\times(x^6 + x^5 + x^3 + x^2 + 1)(x^6 + x^4 + x^3 + x + 1)\times$ $\times(x^6 + x^5 + x^4 + x^2 + 1)(x^6 + x^5 + x^4 + x + 1)\times$ $\times(x^6 + x^5 + 1)(x^3 + x^2 + 1)(x^3 + x + 1)\times$ $\times(x^2 + x + 1)(x + 1)$
$x^{127} - 1$	$(x^7 + x^3 + 1)(x^7 + x^6 + x^5 + x^4 + x^2 + x + 1)\times$ $\times(x^7 + x^4 + x^3 + x^2 + 1)(x^7 + x^3 + x^2 + x + 1)\times$ $\times(x^7 + x^5 + x^3 + x + 1)(x^7 + x^6 + x^4 + x^2 + 1)\times$ $\times(x^7 + x^4 + 1)(x^7 + x^6 + x^3 + x + 1)(x^7 + x + 1)\times$ $\times(x^7 + x^6 + x^5 + x^2 + 1)(x^7 + x^6 + x^5 + x^4 + 1)\times$ $\times(x^7 + x^5 + x^4 + x^3 + 1)(x^7 + x^5 + x^2 + x + 1)\times$ $\times(x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + 1)(x^7 + x^6 + 1)\times$ $\times(x^7 + x^5 + x^4 + x^3 + x^2 + 1)(x^7 + x^6 + x^4 + x + 1)\times$ $\times(x^7 + x^6 + x^5 + x^3 + x^2 + x + 1)(x + 1)$

1.6. Задачи

1. Пусть α – примитивный элемент поля $GF(2^6)$. Найти мультипликативный порядок всех ненулевых элементов этого поля 1, α, ..., α^{62}.

2. Написать таблицы сложения и умножения для поля $GF(7)$.

3. Выписать все примитивные элементы поля $GF(2^4)$.

4. Написать таблицу сложения и умножения подполя чисел поля $GF(125)$.

5. Найти минимальные многочлены для элементов α^7, α^{14}, α^{11} поля $GF(2^6)$ над полем $GF(2)$ и $GF(2^3)$.

6. Доказать, что многочлен $f(x) = x^3 + x^2 + 2$ неприводим над полем $GF(3)$.

7. Проверить, является ли многочлен $f(x) = x^3 + x^2 + 2$ примитивным многочленом поля $GF(27)$.

8. Произвести умножение и сложение двух произвольных пар элементов поля $GF(27)$.

9. Разложить многочлен $x^4 + x^2 + x + 1$ на неприводимые над полем $GF(2)$ множители.

10. Построить все подполя поля $GF(2^6)$.

11. Вычислить $3^{10} \bmod 5$.

12. Построить поле $GF(2^5)$.

2. Линейные коды

2.1. Основные понятия

Будем рассматривать схему передачи, изображенную на рис. 2.1. Здесь источнику сообщений требуется передать информацию получателю, при этом передаваемая информация может быть искажена каналом связи (физической средой, различными устройствами, процедурами обработки информации и т.п.) В задачу кодера и декодера входит такое преобразование информации, которое позволило бы обнаружить произошедшие в канале ошибки и попытаться их исправить.

Определим *код* как произвольное отображение множества всех последовательностей из k символов в последовательности из n символов. Будем предполагать, что последовательности заданы над некоторым дискретным конечным алфавитом. Назовем n *длиной* кода, а отношение k/n – *скоростью* кода R.

Если $k < n$, то заданный таким образом код обладает *избыточностью*. Предполагается, что информация представлена последовательностями из k символов, каждой из которых в процессе кодирования ставится в соответствие кодовая последовательность из n символов. Если в символах кодовой последовательности произошли ошибки (вследствие передачи, хранения или обработки информации), то избыточность кода может быть использована для обнаружения и исправления этих ошибок. В случае, когда декодер не может восстановить правильное слово, говорят об *ошибке декодирования*.

Фундаментальными результатами в теории кодирования (и в более общей теории информации) являются теоремы Шеннона [13, 15], сформулированные для каналов связи, характеризуемых некоторой величиной C, называемой *пропускной способностью* и зависящей от свойств канала.

Приведем здесь теоремы кодирования Шеннона для каналов с шумами без доказательства и в несколько упрощенном виде. Более подробно с классическими результатами теории информации можно ознакомиться, прочитав книги [3, 6].

Рис. 2.1. Система связи

Теорема 2.1 (прямая теорема кодирования)**.** *Пусть C – пропускная способность канала связи. Тогда для любого $R < C$ и любого $\delta > 0$ существует код G с длиной n и скоростью R, вероятность ошибки декодирования которого $P_e \leq \delta$.*

Теорема 2.2 (обратная теорема кодирования)**.** *Пусть C – пропускная способность канала связи и $R = C + \varepsilon$, где $\varepsilon > 0$ – произвольное число. Тогда существует $\delta > 0$ такое, что для всякого кода G с длиной n и скоростью R вероятность ошибки декодирования $P_e \geq \delta$.*

Смысл приведенных теорем в том, что если скорость передачи не превышает пропускной способности канала C (а фактически, подразумевается, что она может быть сколь угодно близка к ней), то существует способ кодирования со сколь угодно малой вероятностью ошибки, т. е., попросту говоря, возможна сколь угодно надежная связь.

С другой стороны, если скорость передачи превышает пропускную способность, вероятность ошибки ограничена снизу положительной величиной, другими словами, сколь угодно надежная связь невозможна.

К сожалению, теоремы Шеннона являются теоремами существования: они не предлагают практических способов построения сколь угодно надежных кодов, скорость которых достигает пропускной способности канала. Эта задача не решена и по сей день, более того – зачастую используемые сегодня коды заведомо плохи с точки зрения достижения пропускной способности, и могут лишь обеспечить некоторую вероятность ошибки декодирования, которую можно счесть приемлемой для конкретных практических параметров системы связи. Поиску таких способов кодирования и посвящена теория, основы которой излагаются в данном пособии.

Рассмотрим стандартную модель двоично-симметричного канала (рис. 2.2). Пусть мы хотим передавать по каналу символы 0 и 1. С некоторой вероятностью p мы будем получать на выходе не тот символ, который передавался. Естественный путь повышения надежности при передаче – повторение одного и того же символа заданное число раз (*код с повторением*) [11,12]. Очевидно, что, повторив символ 0 (или 1) n раз, мы обеспечим возможность обнаружения $n-1$ ошибок, и исправления $\lfloor n - 1/2 \rfloor$ ошибок. Вероятность ошибки стремится к нулю с ростом n, однако к нулю при этом стремится и скорость передачи $R = 1/n$. Вместе с тем, теоремы Шеннона утверждают, что сколь угодно надежная передача с использованием кодирования возможна при скоростях, не стремящихся к нулю. Хотя эти теоремы не дают конструктивных способов построения кодов, рассмотренная ранее идея внесения избы-

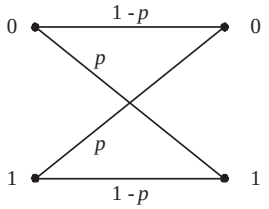

Рис. 2.2. Двоично-симметричный канал

точности может быть использована для получения хороших кодов. Для этого достаточно только заметить, что, если мы обрабатываем не один бит, а блок из k бит, ставя ему в соответствие блок из n бит, $n > k$, полученный код имеет скорость

$$R = \frac{k}{n},$$

которая не стремится к нулю, если величина в числителе стремится к бесконечности с такой же скоростью, как величина в знаменателе.

Тогда процедура кодирования может выглядеть примерно следующим образом. Пусть требуется передать N сообщений $\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_N$. Вместо этой последовательности сообщений будем передавать кодовые слова $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_N$:

$$\begin{aligned} &\mathbf{m}_1 \xrightarrow{\varphi} \mathbf{a}_1; \\ &\quad \ldots \\ &\mathbf{m}_N \xrightarrow{\varphi} \mathbf{a}_N. \end{aligned} \tag{2.1}$$

Пусть длина передаваемых сообщений равна k (символов в некотором алфавите), а длина кодовых слов равна n, $k < n$, т. е. внесена избыточность $r = n - k$. Множество слов $A = \{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_N\}$, каждое из которых представляет собой последовательность из n символов, $\mathbf{a}_i = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$, $i \in \overline{1, n}$, где α_i – символ некоторого конечного алфавита из q элементов, назовем *кодом*. В дальнейшем ограничимся рассмотрением только двоичных последовательностей, то есть $q = 2$. Отображение φ (2.1) безызбыточных сообщений на кодовые слова будем называть процедурой *кодирования*.

Параметрами кода являются кодовая длина n и число слов в коде $N = 2^k$. Однако, оба эти параметра не характеризуют вероятность

ошибки декодирования $\mathrm{P}_{\text{ош}}$. Попробуем определить параметр кода, влияющий на $\mathrm{P}_{\text{ош}}$. Снова рассмотрим код с повторением с $n = 3$, т. е. каждый ноль мы кодируем тремя нулями, а каждую единицу – тремя единицами. Тогда, очевидно, мы можем исправить одну ошибку: последовательности 100, 010 или 001 больше «похожи» на последовательность из всех нулей, чем из всех единиц, и можно принять решение, что передавался 0. Аналогично, чтобы исправить любые две ошибки, понадобится $n = 5$ кодовых бит.

Таким образом, процедура декодирования, то есть восстановления переданного слова по принятому, заключается в определении кодового слова, наиболее «похожего» на принятое. Такая похожесть может быть выражена с помощью функции *расстояния* между двумя векторами $\mathbf{x}$ и $\mathbf{y}$, то есть функции $d(\mathbf{x}, \mathbf{y})$, обладающей следующими свойствами:

$$\begin{array}{l} 1.\ d(\mathbf{x},\mathbf{y}) \geq 0; \quad d(\mathbf{x},\mathbf{y}) = 0 \Leftrightarrow \mathbf{x} = \mathbf{y}; \\ 2.\ d(\mathbf{x},\mathbf{y}) = d(\mathbf{y},\mathbf{x}); \\ 3.\ d(\mathbf{x},\mathbf{y}) \leq d(\mathbf{x},\mathbf{z}) + d(\mathbf{z},\mathbf{y}). \end{array} \tag{2.2}$$

Определение 2.1. *Расстояние Хэмминга* $d_H(\mathbf{x}, \mathbf{y})$ между двумя векторами $\mathbf{x}$ и $\mathbf{y}$ определяется как число позиций, в которых эти векторы различаются.

Расстояние Хэмминга – это функция, которая удовлетворяет всем свойствам (2.2). Тогда расстояние Хэмминга между векторами (11111) и (00000) равно 5. Ошибка в одном слове приближает нас к другому слову, поэтому может быть исправлена, только если она меньше половины расстояния между этими словами. Так, если расстояние между векторами d, то можно исправить любые

$$t = \left\lfloor \frac{d-1}{2} \right\rfloor$$

ошибок. Однако в общем случае код содержит не два слова, а гораздо больше, и эти слова могут находиться на различном расстоянии друг от друга. За меру, характеризующую код в целом, принимают *минимальное кодовое расстояние*:

$$d_0 = \min_{i \neq j} d(\mathbf{a}_i, \mathbf{a}_j).$$

Код с минимальным расстоянием d_0 может исправить любую комбинацию из $\lfloor (d_0 - 1)/2 \rfloor$ ошибок. Таким образом, минимальное кодовое расстояние является еще одним параметром кода. Величина d_0 связана с вероятностью ошибки декодирования $\mathrm{P}_{\text{ош}}$, однако, параметр d_0 не

характеризует *все* ошибки, которые может исправлять код. Подробнее вычисление $\mathrm{P}_{\text{ош}}$ рассматривается в разделе 4.1.

Любая система передачи информации характеризуется следующими параметрами:

1. $\mathrm{P}_{\text{ош}}$ – вероятность ошибки;
2. V – скорость передачи (бит/с);
3. ϑ – сложность устройств кодера/декодера.

При построении системы связи требуется обеспечить высокую скорость передачи при малой вероятности ошибки и малой сложности, что можно представить как

$$\mathrm{P}_{\text{ош}} \downarrow, \quad V \uparrow, \quad \vartheta \downarrow,$$

т. е. требуется минимизировать вероятность ошибки при как можно большей скорости передачи и как можно меньшей сложности. Эти параметры связаны с параметрами кода: сложность ϑ прямо зависит от длины кода n, скорость передачи прямо зависит от числа слов в коде N (так как она зависит от k, а $N = 2^k$), вероятность ошибки $\mathrm{P}_{\text{ош}}$ уменьшается с ростом расстояния d_0.

Таким образом, требуется обеспечить:

$$d_0 \uparrow, \quad N \uparrow, \quad n \downarrow .$$

Однако данные требования противоречивы. Длина кода n определяет размер пространства сообщений длины n. Если зафиксировать этот параметр, то увеличение N повлечет за собой уменьшение d_0, и наоборот. Таким образом, задача построения хорошего кода может формулироваться как оптимизация одного из параметров d_0, N, n при фиксированных остальных параметрах. Например, можно рассматривать задачу максимизации $N(n, d_0)$ при фиксированных значениях n и d_0.

Чтобы оценивать качество кода, нужно определить предельно достижимые границы для N при фиксированных n и d_0.

2.2. Нижняя и верхняя границы для числа слов в коде

Введем понятия нижней и верхней границы для числа слов в коде [4]. *Верхняя граница* $\bar{N}$ определяется утверждением: не существует кода, число слов в котором $N > \bar{N}$. *Нижняя граница* $\underline{N}$ определяется утверждением: существует код, число слов в котором $N \leq \underline{N}$. В связи

с этими определениями нижнюю границу иногда называют границей существования, верхнюю – границей несуществования.

Таким образом, граница $N = \bar{N}$ – граница, практически не достижимая (за исключением некоторых классов кодов, называемых *совершенными*, доказано, что неизвестных совершенных кодов не существует [11]), а $N = \underline{N}$ – граница, на которой лежат хорошие коды. Можно дать другую трактовку верхней и нижней границы: пусть N_0 – оптимальное значение числа слов в коде. Тогда $\underline{N} \leq N_0 \leq \bar{N}$.

2.2.1. Граница Хэмминга

Верхняя граница $\bar{N}$, или граница Хэмминга, строится следующим образом. Все пространство двоичных последовательностей длины n имеет размер 2^n. Для некоторого кода длины n с расстоянием d будем рассматривать сферы радиуса $t = \lfloor (d-1)/2 \rfloor$, центрами которых являются кодовые слова (рис. 2.3). Такая геометрическая трактовка обусловлена тем, что для d выполняются свойства расстояния, а следовательно, справедливы все геометрические понятия. Таким образом, можно определить сферы радиуса t – геометрическое место точек, находящихся на расстоянии Хэмминга t от центра. Обозначим через V_t объем полученной сферы, т. е. число точек внутри нее, и умножим на N – число слов в коде. Тогда

$$V_t \cdot N \leq 2^n \; -$$

объем пространства, занятый сферами. Мы получили верхнюю оценку

$$\bar{N} = \frac{2^n}{V_t}.$$

Найдем объем сферы V_t. Сфера содержит: само кодовое слово (центр сферы), векторы, которые отличаются от него в одной позиции (таких векторов $C_n^1 = n$), векторы, отличающиеся от него в двух позициях ($C_n^2 = n(n-1)/2$) и так далее, до векторов, отличающихся от данного в t позициях, т. е. находящихся на расстоянии t. Таким образом, $V_t = 1 + C_n^1 + C_n^2 + \ldots + C_n^t$. Тогда

$$\bar{N} = \frac{2^n}{\sum_{i=0}^{t} C_n^i} \; -$$

верхняя граница Хэмминга, которую иногда называют границей плотной упаковки.

Такое название связано с тем, что делается попытка «плотно упаковать» сферами все пространство векторов длины n, стараясь оценить

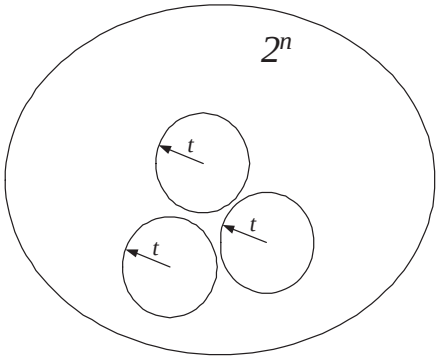

Рис. 2.3. Покрытие пространства из 2^n векторов сферами радиуса t

максимальное число сфер, которые при такой упаковке могут «уместиться» в пространство.

2.2.2. Граница Варшамова–Гилберта

Нижняя граница, или граница Варшамова–Гилберта, строится следующим образом. В отличие от границы Хэмминга, где мы пытались найти максимально возможное число слов в коде (при заданных ограничениях n и d), при этом получая границу несуществования, в данном случае указывается процедура построения кода с заданными n и d, при этом делается попытка максимизировать число $\underline{N}$ слов в этом коде, что соответствует границе существования – код с таким $\underline{N}$ точно существует.

Возьмем произвольную точку пространства всех векторов длины n и построим вокруг нее сферу радиуса $d-1$. Будем считать взятую точку кодовым словом, а все остальные точки внутри сферы радиуса $d-1$ запретим для использования в качестве кодовых слов, так как необходимо обеспечить минимальное расстояние кода d. Затем в качестве следующего кодового слова берется произвольная точка из незапрещенной области пространства. Вокруг нее также проводится сфера радиуса $d-1$ и запрещаются все точки внутри сферы (рис. 2.4). Так действовать можно до тех пор, пока в пространстве не останется незапрещенных слов. Так как сферы имеют радиус d, они могут пересекаться, и для того, чтобы максимизировать параметр N, нужно строить сферы таким образом, чтобы области их пересечений были как можно большими. Однако при

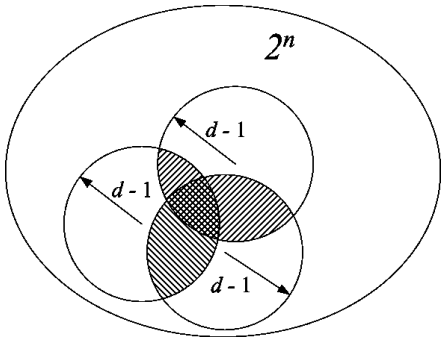

Рис. 2.4. Покрытие пространства из 2^n векторов сферами радиуса $d-1$

вычислении N сложно учесть объем пересечений, поэтому будем исходить из худшего случая, т. е. предполагать, что проведенные сферы не пересекаются. Тогда $NV_{d-1} \leq 2^n$. Отсюда получаем $\underline{N}V_{d-1} = 2^n$. Используя полученные в разделе 2.2.1 выражения для объема сферы заданного радиуса, получим

$$\underline{N} = \frac{2^n}{\sum_{i=0}^{d-1} C_n^i}.$$

Границу Варшамова–Гилберта иногда называют границей покрытий – с помощью как можно большего числа сфер делается попытка «покрыть» все пространство из 2^n векторов.

2.3. Задание линейных блоковых кодов

В предыдущих разделах мы рассматривали код как отображение (2.1) множества из 2^k информационных слов длины k в множество кодовых слов длины n. Очевидно, что уже при малых величинах k задание кода с помощью (2.1) становится нереализуемым. Необходим более простой метод задания кода (кодирования), такой метод может быть предложен с помощью математического аппарата линейных векторных пространств.

Множество V называется *векторным (линейным) пространством* над полем K, если:

1. V является абелевой группой по сложению.

2. Для любых $\mathbf{v} \in V$, $\lambda \in K$ определен $\lambda\mathbf{v} \in V$ (умножение вектора $\mathbf{v}$ на скаляр λ), при этом:

   а) $\lambda(\mathbf{v}_1 + \mathbf{v}_2) = \lambda\mathbf{v}_1 + \lambda\mathbf{v}_2$;

   б) $(\lambda_1 + \lambda_2)\mathbf{v} = \lambda_1\mathbf{v} + \lambda_2\mathbf{v}$;

   в) $(\lambda_1\lambda_2)\mathbf{v} = \lambda_1(\lambda_2\mathbf{v})$;

   г) $1_K \cdot \mathbf{v} = \mathbf{v}$.

Здесь 1_K – нейтральный элемент по умножению поля K.

Например, векторным пространством является множество последовательностей $\mathbf{a} = (\alpha_1, \ldots, \alpha_n)$, $\alpha_i \in K$, где операции сложения и умножения на скаляр определены как

$$\mathbf{a} + \mathbf{b} = (\alpha_1 + \beta_1, \ldots, \alpha_n + \beta_n);$$
$$\lambda\mathbf{a} = (\lambda\alpha_1, \ldots, \lambda\alpha_n).$$

Будем рассматривать векторные пространства в поле $GF(2)$.

В каждом линейном пространстве размерности k есть базис, т. е. набор из k линейно независимых векторов. Линейное пространство задается всеми 2^k линейными комбинациями векторов базиса. Это позволяет компактно задавать линейное пространство.

Определение 2.2. *Линейным двоичным* (n, k)*-кодом* будем называть k-мерное подпространство n-мерного пространства двоичных последовательностей.

Так как (n, k)-код является k-мерным пространством, он содержит базис из k линейно независимых векторов. Запишем этот базис в матрицу $\mathbf{G}$ размерности $(k \times n)$ и назовем ее *порождающей*, или *генераторной* матрицей кода G.

Тогда кодирование может быть представлено следующей процедурой. Пусть $\mathbf{m}$ – информационный вектор длины k. Тогда кодовое слово $\mathbf{a}$ получается умножением информационного вектора на порождающую матрицу $\mathbf{G}$:

$$\mathbf{m} \xrightarrow{\varphi} \mathbf{a} = \mathbf{m} \cdot \mathbf{G}.$$

Матрица $\mathbf{G}$ содержит информацию о параметрах кода: длине n, числе кодовых слов $N = 2^k$ и скорости $R = k/n$. Единственный параметр, не задаваемый в явном виде порождающей матрицей, – минимальное расстояние d_0.

Пусть порождающая матрица $\mathbf{G}$ имеет вид $\mathbf{G} = [\mathbf{C}_1|\mathbf{C}_2]$, где $\mathbf{C}_1$ – квадратная $(k \times k)$-матрица; $\mathbf{C}_2$ – некая матрица размерности $(k \times n-k)$.

Тогда, если $\mathbf{C}_1$ – невырожденная матрица, то, умножив матрицу $\mathbf{G}$ слева на $\mathbf{C}_1^{-1}$, получим другой базис $\mathbf{G}'$ этого же линейного пространства:

$$\mathbf{G}' = [\mathbf{I}_k|\mathbf{C}], \tag{2.3}$$

где $\mathbf{I}_k$ – единичная $(k \times k)$-матрица; $\mathbf{C} = \mathbf{C}_1^{-1}\mathbf{C}_2$. Тогда при кодировании с помощью матрицы $\mathbf{G}'$

$$\mathbf{m} \xrightarrow{\varphi} \mathbf{a} = \mathbf{m} \cdot \mathbf{G}' = (\underbrace{\mathbf{m}}_{k} | \underbrace{\mathbf{c}}_{n-k}),$$

т. е. первые k позиций кодового слова всегда содержат информационный вектор в явном виде. Код, заданный матрицей вида (2.3), называется *систематическим*.

Рассмотрим матрицу

$$\mathbf{H} = \left[-\mathbf{C}^T|\mathbf{I}_r\right] \tag{2.4}$$

размерности $(r \times n)$, где $r = n - k$ – число избыточных символов, $\mathbf{C}$ – $(k \times r)$-матрица из (2.3). Для двоичных кодов можно принять $-\mathbf{C} = \mathbf{C}$.

Заметим, что

$$\mathbf{G} \cdot \mathbf{H}^T = [\mathbf{I}_k|\mathbf{C}] \cdot \left[\frac{-\mathbf{C}}{\mathbf{I}_r}\right] = -\mathbf{C} + \mathbf{C} = \mathbf{0},$$

т. е., матрица (2.4) является базисом пространства, *ортогонального* коду G.

Необходимо отметить, что возможна ситуация, при которой первые k столбцов порождающей матрицы $\mathbf{G}$ образуют вырожденную матрицу $\mathbf{C}_1$. Однако, так как $\mathbf{G}$ – базис, ранг этой матрицы равен k, и, следовательно, в ней найдутся k линейно независимых столбцов. Переставив эти столбцы на первые k позиций внутри $\mathbf{G}$, получив матрицу $\mathbf{H}$, как указано в выражении (2.4), и осуществив в полученной матрице обратную перестановку столбцов, мы также получим базис пространства, ортогонального первоначальному коду G.

Для любого кодового слова будет выполняться равенство

$$\mathbf{a} \cdot \mathbf{H}^T = (\mathbf{m} \cdot \mathbf{G}) \cdot \mathbf{H}^T = \mathbf{m}(\mathbf{G} \cdot \mathbf{H}^T) = \mathbf{0}.$$

С другой стороны, ни для какого вектора $\mathbf{e}$, не являющегося кодовым словом, произведение $\mathbf{e} \cdot \mathbf{H}^T$ не может быть равно нулю (нулевому вектору), так как $\mathbf{e}$ не принадлежит пространству G. Таким образом, линейный код может быть задан и как множество слов $\{\mathbf{a}\}$ таких, что

$$\mathbf{a} \cdot \mathbf{H}^T = \mathbf{0}.$$

Матрица $\mathbf{H}$ также однозначно задает линейный код и называется *проверочной* матрицей кода. С помощью матрицы $\mathbf{H}$ легко проверить, является ли некоторый вектор кодовым словом. Таким образом, существует несложная процедура обнаружения ошибок. Пусть из канала связи принято слово $\mathbf{b}$. Вычислив *синдром*

$$\mathbf{S} = \mathbf{b} \cdot \mathbf{H}^T, \tag{2.5}$$

можно принять решение о том, что принято кодовое слово (если синдром равен нулю), или же обнаружена ошибка передачи (в противном случае).

Итак, линейные коды позволяют с помощью задания порождающей и проверочной матрицы кода легко решать задачи кодирования и обнаружения ошибок. С другой стороны, свойство линейности никак не использовалось для определения минимального расстояния кода d_0. Кроме того, существует возможность того, что, ограничив коды свойством линейности, были упущены из рассмотрения все хорошие коды, т. е. лежащие на границе Варшамова–Гилберта. Таким образом, необходимо показать, что существуют линейные коды, лежащие на границе Варшамова–Гилберта.

2.4. Теорема о минимальном расстоянии линейных кодов

В этом разделе мы рассмотрим задачу определения минимального расстояния линейных блоковых кодов.

Определение 2.3. Будем называть *весом Хэмминга* $W_H(\mathbf{x})$ или просто $W(\mathbf{x})$ число ненулевых позиций в векторе $\mathbf{x}$.

Назовем *минимальным весом* W_0 кода G минимальный вес ненулевых кодовых слов,

$$W_0 = \min_{\mathbf{a}_i \in G} W(\mathbf{a}_i).$$

Тогда справедливо следующее утверждение.

Лемма 2.1. *Минимальное расстояние линейного кода равно его минимальному весу:* $d_0 = W_0$.

Доказательство. Пусть d_0 – минимальное расстояние кода. Тогда существует два кодовых слова, $\mathbf{a}$ и $\mathbf{a}'$, таких, что $d_0 = d(\mathbf{a}, \mathbf{a}')$. Заметим, что для любого вектора $\mathbf{e}$ длины n справедливо равенство $d(\mathbf{a}, \mathbf{a}') = d(\mathbf{a} + \mathbf{e}, \mathbf{a}' + \mathbf{e})$. Примем $\mathbf{e} = \mathbf{a}'$, тогда $d(\mathbf{a} + \mathbf{a}', \mathbf{a}' + \mathbf{a}') = d(\mathbf{a} + \mathbf{a}', \mathbf{0}) = d(\hat{\mathbf{a}}, \mathbf{0}) = W(\hat{\mathbf{a}})$. Так как сумма двух слов линейного кода также является

кодовым словом, $\hat{\mathbf{a}}$ – кодовое слово и

$$d_0 = W(\hat{\mathbf{a}}) \geq W_0. \tag{2.6}$$

Теперь, пусть W_0 – минимальный вес. Тогда существует такое кодовое слово $\mathbf{a}$, для которого $W_0 = W(\mathbf{a}) = d(\mathbf{a}, \mathbf{0})$, но тогда

$$W_0 \geq d_0. \tag{2.7}$$

Из выражений (2.6) и (2.7) получаем условие леммы.

Пользуясь леммой 2.1, можно доказать следующую теорему о минимальном расстоянии линейных кодов.

Теорема 2.3. *Для того, чтобы линейный (n, k)-код с проверочной матрицей $\mathbf{H}$ имел минимальное расстояние d_0, необходимо и достаточно, чтобы:*

1. *Любые $d_0 - 1$ столбцов матрицы $\mathbf{H}$ были линейно независимы.*
2. *В матрице $\mathbf{H}$ существовали d_0 линейно зависимых столбцов.*

Доказательство. Докажем необходимость. Пусть d_0 – минимальное расстояние кода. Тогда по Лемме 2.1 в коде есть слово $\mathbf{a}$ веса $W(\mathbf{a}) = W_0 = d_0$, где W_0 – минимальный вес кода. Пусть $\{\alpha_{i_1}, \ldots, \alpha_{i_{W_0}}\}$ – все ненулевые элементы вектора $\mathbf{a}$. Так как $\mathbf{a}$ является кодовым словом, $\mathbf{a} \cdot \mathbf{H}^T = 0$, это означает, что $W_0 = d_0$ столбцов матрицы $\mathbf{H}$ с номерами $\{i_1, \ldots, i_{W_0}\}$ линейно зависимы. С другой стороны, так как W_0 – минимальный вес, в матрице $\mathbf{H}$ не может быть менее, чем d_0 линейно зависимых столбцов.

Теперь докажем достаточность. Пусть в матрице $\mathbf{H}$ есть d_0 линейно зависимых столбцов $\{\mathbf{h}_{i_1}, \ldots, \mathbf{h}_{i_{d_0}}\}$. Тогда эти столбцы задают кодовое слово веса d_0, которое содержит ненулевые элементы на позициях $\{i_1, \ldots, i_{d_0}\}$ и нули на остальных позициях. Если матрица $\mathbf{H}$ не содержит $d_0 - 1$ или менее линейно зависимых столбцов, в коде нет ненулевых слов веса, меньшего $d_0 = W_0$.

Пример 2.1. Пусть есть код с проверочной матрицей

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Так как в матрице есть нулевой столбец, минимальное расстояние этого кода $d_0 = 1$. Удалив нулевой столбец, получим код с $d_0 = 2$, так как в матрице присутствуют два одинаковых столбца.

Теперь рассмотрим код

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}. \tag{2.8}$$

В матрице (2.8) нет ни нулевого, ни двух одинаковых столбцов. Минимальное число линейно зависимых столбцов равно 3, и следовательно, $d_0 = 3$. Итак, получили код с $n = 7$, $k = 4$, $d_0 = 2t+1 = 3$, исправляющий одну ошибку.

Найдем границу Хэмминга для $n = 7$, $t = 1$:

$$\bar{N} = \frac{2^n}{\sum_{i=0}^{t} C_n^i} = \frac{2^7}{1 + C_7^1} = 16 = 2^4,$$

т. е., код с матрицей (2.8) лежит на границе Хэмминга. Это – совершенный (7,4)-код Хэмминга, столбцы проверочной матрицы которого получены из всех $2^3 - 1$ ненулевых двоичных векторов.

Теорема 2.3 дает способ нахождения минимального расстояния линейных кодов. Однако сложность этой процедуры остается экспоненциальной, так как не известно простого способа нахождения максимального числа d_0, такого, что любые $d_0 - 1$ столбцов линейно независимы.

2.5. Граница Варшамова–Гилберта для линейных кодов

Укажем процедуру для построения проверочной матрицы $\mathbf{H}$. Чтобы код имел расстояние d_0, любые $d_0 - 1$ столбцов в $\mathbf{H}$ должны быть линейно независимы. Будем строить столбец за столбцом так, чтобы никакие $d_0 - 1$ столбцов не были линейно зависимы. Процедура закончится, когда невозможно будет построить очередной линейно независимый столбец.

Первый столбец можно выбрать $2^r - 1$ способами (исключая нулевой столбец). На втором шаге запрещенными являются уже два столбца: нулевой и поставленный на предыдущем шаге. На каждом следующем шаге запрещаются все предыдущие столбцы и их линейные комбинации. Итак, пусть построено $j - 1$ столбцов. При каких условиях можно поставить j-й столбец? На данном шаге запрещено

$$1 + j - 1 + C_{j-1}^2 + \ldots + C_{j-1}^{d_0-2}$$

столбцов. Очередной j-й столбец можно выбрать в том случае, если запрещенные столбцы не исчерпывают всего множества столбцов, т. е.

$$1 + j - 1 + C_{j-1}^2 + \ldots + C_{j-1}^{d_0-2} \leq 2^r.$$

Отсюда можно определить длину кода. Выберем самое последнее число j, для которого неравенство выполняется, тогда длина кода составит $n = j - 1$. Тогда имеем

$$\sum_{i=0}^{d_0-2} C_n^i \le 2^r = 2^n/2^k,$$

отсюда

$$2^k \le \frac{2^n}{\sum_{i=0}^{d_0-2} C_n^i}. \tag{2.9}$$

Соотношение (2.9) является улучшенной границей Варшамова–Гилберта для линейных кодов.

Подсчитаем, сколько всего существует кодов с параметрами (n, k). Если считать, что каждая порождающая матрица задает код (это не совсем верно, так как у одного линейного пространства существует много базисов), имеем 2^{nk} кодов. Какая доля этих кодов лежит на нижней границе?

Будем считать, что матрица $\mathbf{H}$ выбирается случайно. Зададим вероятностный ансамбль матриц, полагая, что вероятность выбора 0 или 1 на конкретной позиции равна $1/2$. Тогда все матрицы появляются с одинаковой вероятностью 2^{-rn}.

Теперь возьмем произвольное слово $\mathbf{a} = (\alpha_1, \ldots, \alpha_n)$. Это слово будет принадлежать некоторому коду, $\mathbf{a} \in A_{\mathbf{H}}$, где $A_{\mathbf{H}}$ – код, задаваемый проверочной матрицей $\mathbf{H}$, с вероятностью $\Pr(\mathbf{a} \in A_{\mathbf{H}})$. Пусть матрица $\mathbf{H}$ состоит из столбцов $[\mathbf{h}_1, \ldots, \mathbf{h}_n]$. Тогда $\mathbf{a} \cdot \mathbf{H}^T = \alpha_1\mathbf{h}_1 + \ldots + \alpha_{n-1}\mathbf{h}_{n-1} + \alpha_n\mathbf{h}_n = \mathbf{0}$, отсюда

$$\alpha_n\mathbf{h}_n = \alpha_1\mathbf{h}_1 + \ldots + \alpha_{n-1}\mathbf{h}_{n-1}. \tag{2.10}$$

Если слово $\mathbf{a}$ – ненулевое, оно содержит хотя бы одну ненулевую позицию. Для определенности положим, что последняя позиция не равна нулю. Таким образом, вероятность того, что слово $\mathbf{a}$ принадлежит коду $A_{\mathbf{H}}$, равна вероятности выполнения равенства (2.10). Какими бы ни были столбцы $[\mathbf{h}_1, \ldots, \mathbf{h}_{n-1}]$, для выполнения равенства (2.10) достаточно правильно подобрать столбец $\mathbf{h}_n$. Тогда $\Pr(\mathbf{a} \in A_{\mathbf{H}}) = 2^{-r}$.

Теперь введем понятие «плохого» вектора. Все вектора, вес которых $W < d_{\text{ВГ}}$, будем называть плохими, где $d_{\text{ВГ}}$ получено из уравнения

$$\sum_{i=0}^{d_{\text{ВГ}}-1} C_n^i = 2^r.$$

Если код содержит хотя бы один такой вектор, то он лежит ниже границы Варшамова–Гилберта. Таких векторов:

$$\sum_{i=1}^{d_{\text{ВГ}}-1} C_n^i.$$

Вероятность того, что произвольный код $A_{\mathbf{H}}$ содержит хотя бы один «плохой» вектор $\mathbf{a}_b$, равна

$$\Pr(\mathbf{a}_b \in A_{\mathbf{H}}) = \sum_{i=1}^{d_{\text{ВГ}}-1} C_n^i 2^{-r}.$$

Если $\Pr(\mathbf{a}_b \in A_{\mathbf{H}}) < 1$, то существует хотя бы один «хороший» код. Фактически, мы получили границу Варшамова–Гилберта другим способом. На самом деле,

$$\sum_{i=1}^{d_{\text{ВГ}}-\varepsilon n} C_n^i 2^{-r} \; -$$

доля «плохих» кодов длины n. Оценим, как будет меняться эта доля с ростом n. Для этого оценим $C_n^{\alpha n}$, $\alpha < 1$, пользуясь формулой Стирлинга:

$$n! \approx \left(\frac{n}{e}\right)^n.$$

Тогда

$$C_n^{\alpha n} = \frac{n!}{(\alpha n)!(n(1-\alpha))!} \approx \frac{\left(\frac{n}{e}\right)^n}{\left(\frac{\alpha n}{e}\right)^{\alpha n}\left(\frac{n(1-\alpha)}{e}\right)^{n(1-\alpha)}} = \frac{1}{\alpha^{\alpha n}(1-\alpha)^{(1-\alpha)n}}.$$

Теперь, учитывая, что $\alpha = 2^{\log_2 \alpha}$, $(1-\alpha) = 2^{\log_2(1-\alpha)}$, получим

$$C_n^{\alpha n} = 2^{nH(\alpha)},$$

где

$$H(\alpha) = -\alpha \log \alpha - (1-\alpha)\log(1-\alpha).$$

В кодировании «хорошими» называют коды, у которых доля d/n не стремится к нулю при $n \to \infty$ и $R = \text{const}$. Пусть $d/n = \delta$. Тогда

$$\sum_{i=1}^{d-\varepsilon n} C_n^i 2^{-r} = \sum_{i=1}^{\delta n-\varepsilon n} C_n^i 2^{-(1-R)n} < nC_n^{(\delta-\varepsilon)n} 2^{-(1-R)n} \approx$$
$$\approx n2^{n(H(\delta-\varepsilon)-(1-R))} \approx 2^{n(H(\delta-\varepsilon)-H(\delta))} \approx 2^{-n\varepsilon} \xrightarrow[n\to\infty]{} 0.$$

Таким образом, доля «плохих» кодов стремится к нулю с ростом длины кода n, другими словами, почти все коды в асимптотике лежат на границе Варшамова–Гилберта. Однако данный факт не дает процедуры построения «хороших» кодов. Задача построения таких кодов является важной задачей теории кодирования.

2.6. Задачи

Для заданной порождающей матрицы кода следует:

– определить его параметры (n, k, R, d), указать число исправляемых кодом ошибок;

– выписать проверочную матрицу кода;

– построить границы Хэмминга и Варшамова–Гилберта.

1.
$$\mathbf{G} = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}.$$

2.
$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

3.
$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \end{pmatrix}.$$

4.

$$\mathbf{G} = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

5.

$$\mathbf{G} = \begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

6.

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}.$$

7.

$$\mathbf{G} = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

8.

$$\mathbf{G} = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

9.

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

10.

$$\mathbf{G} = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix}.$$

11.

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}.$$

12.

$$\mathbf{G} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}.$$

13.

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}.$$

14.

$$\mathbf{G} = \begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

15.

$$\mathbf{G} = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}.$$

16.

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}.$$

17.

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

18.

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}.$$

3. Коды с алгебраической структурой

3.1. Циклические коды

В предыдущем разделе мы рассмотрели задачу кодирования информационного сообщения из k двоичных символов в кодовое слово из n двоичных символов для повышения надежности передачи информации по каналам связи с ошибками. Количество информационных сообщений длиной k экспоненциально велико, однако использование в качестве кодов линейных подпространств позволило задать эффективную процедуру кодирования с помощью умножения информационного вектора на базис пространства G.

Если по каналу связи передаются только двоичные кодовые слова длиной n, а в результате происходящих при передаче ошибок принята может быть любая двоичная последовательность длиной n, то можно сформулировать задачу *декодирования по минимуму расстояния* как поиск кодового слова, ближайшего к принятой последовательности (например, в метрике Хэмминга).

Однако, так как кодовых слов также экспоненциально много, решить такую задачу последовательным перебором вычислительно невозможно в большинстве практических случаев. Поэтому помимо задания линейных кодов требуется обладание ими какой-то структурой, которая позволила бы осуществлять реализуемое декодирование (с полиномиальной сложностью). На практике наибольшее распространение получили так называемые *циклические коды*, которые и рассматриваются в данном разделе.

Определение 3.1. *Циклическим кодом* длины n с k информационными символами называется такой линейный (n, k)-код, у которого циклический сдвиг любого кодового слова также является кодовым словом.

Поставим в соответствие каждому вектору $f = (f_{n-1}, \ldots, f_0)$ многочлен (полином) $f(x) = f_{n-1}x^{n-1} + \ldots + f_1x + f_0$. Тогда каждому вектору длины n будет соответствовать многочлен, степень которого не превышает $n - 1$. В дальнейшем мы не будем различать вектор и соответствующий ему многочлен. Пусть $g(x)$ – многочлен степени $r = n - k$, являющийся делителем многочлена $x^n - 1$. Тогда множество многочленов степени, не превосходящей $n-1$, которые делятся на $g(x)$, образуют циклический код длины n с k информационными символами. Многочлен $g(x)$ называется *порождающим многочленом* циклического кода. Произведение любого многочлена $m(x)$ на кодовое слово $a(x)$ (умножение многочленов производится по модулю $x^n - 1$, т.е. с учетом того,

что $x^n = 1$) является кодовым словом. Кодирование информационной последовательности $m(x) = \sum_{i=0}^{k-1} m_i x^i$ циклическим кодом с порождающим многочленом $g(x) = g_r x^r + g_{r-1} x^{r-1} + \ldots + g_0$ осуществляется по правилу

$$m(x) \longrightarrow m(x)g(x) \bmod x^n - 1 = m(x)g(x). \tag{3.1}$$

Порождающая матрица циклического кода может быть представлена как

$$\mathbf{G} = \overbrace{\begin{bmatrix} g_0 & \ldots\ldots & g_r & 0 & \ldots & \ldots & 0 \\ 0 & g_0 & \ldots\ldots & g_r & 0 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & \ldots & \ldots & 0 & g_0 & \ldots\ldots & g_r \end{bmatrix}}^{n} \Bigg\updownarrow k \tag{3.2}$$

Проверочным многочленом кода с порождающим многочленом $g(x)$ называется многочлен

$$h(x) = \frac{x^n - 1}{g(x^{-1})x^r} \ , \tag{3.3}$$

где x^{-1} – величина, которая при умножении на x дает 1 ($xx^{-1} = 1$). Произведение любого кодового слова $a(x) = m(x)g(x)$ на многочлен $h(x)$ выглядит как:

$$a(x)h(x) = m(x)g(x)h(x) = 0 \bmod x^n - 1. \tag{3.4}$$

Соотношение (3.4) является основным проверочным соотношением для кодовых слов. Поскольку $x^n - 1$ делится на $h(x)$, то равенство (3.4) эквивалентно равенству

$$a(x) = 0 \bmod g(x). \tag{3.5}$$

Проверочная матрица циклического кода выражается через $h(x)$ следующим образом:

$$\mathbf{H} = \overbrace{\begin{bmatrix} h_0 & \ldots\ldots & h_k & 0 & \ldots & \ldots & 0 \\ 0 & h_0 & \ldots\ldots & h_k & 0 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & \ldots & \ldots & 0 & h_0 & \ldots\ldots & h_k \end{bmatrix}}^{n} \Bigg\updownarrow n - k = r \ . \tag{3.6}$$

Пример 3.1. Пусть $g(x) = x^4 + x^3 + 1$. Легко проверить, что $x^{15} - 1$ делится на $g(x)$. Тогда многочлен $g(x)$ порождает циклический (15,11)-код над $GF(2)$:

$$h(x) = \frac{x^{15} - 1}{(x^{-4} + x^{-3} + 1) \cdot x^4} = \frac{x^{15} - 1}{x^4 + x + 1} =$$
$$= x^{11} + x^8 + x^7 + x^5 + x^3 + x^2 + x + 1;$$

$$\mathbf{G} = \overbrace{\begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & 1 & 0 & 0 & 1 & 1 \end{bmatrix}}^{15} \Bigg\updownarrow 11 \ ;$$

$$\mathbf{H} = \overbrace{\begin{bmatrix} 1\,1\,1\,1\,0\,1\,0\,1\,1\,0\,0\,1\,0\,0\,0 \\ 0\,1\,1\,1\,1\,0\,1\,0\,1\,1\,0\,0\,1\,0\,0 \\ 0\,0\,1\,1\,1\,1\,0\,1\,0\,1\,1\,0\,0\,1\,0 \\ 0\,0\,0\,1\,1\,1\,1\,0\,1\,0\,1\,1\,0\,0\,1 \end{bmatrix}}^{15} \Bigg\updownarrow 4 \ .$$

3.2. Граница БЧХ

Введенный в предыдущем разделе порождающий многочлен циклического кода позволяет не только осуществлять процедуру кодирования, но и оценивать минимальное расстояние полученного кода.

Пусть $g(x)$ – порождающий многочлен циклического кода $\mathcal{G}$ и все корни $g(x)$ лежат в поле $GF(q)$. Тогда они могут быть записаны в виде степеней η – примитивного элемента поля $GF(q)$:

$$\eta^{i_1}, \eta^{i_2}, \dots, \eta^{i_r}. \tag{3.7}$$

Будем рассматривать последовательности подряд стоящих чисел во множестве:

$$\{i_1, i_2, \dots, i_r\}, \tag{3.8}$$

пусть самая длинная из таких последовательностей:

$$m_0, m_0 + 1, \dots, m_0 + d_0 - 2.$$

Следующая теорема дает возможность оценить расстояние циклического кода.

Теорема 3.1. *Пусть $\mathcal{G}$ – циклический код с порождающим многочленом $g(x)$, и m_0, m_0+1, …, m_0+d_0-2 – последовательность чисел, определенная выше. Тогда минимальное кодовое расстояние d кода $\mathcal{G}$*

$$d \geq d_0.$$

Доказательство. Поскольку элементы η^{m_0}, η^{m_0+1}, $\eta^{m_0+d_0-2}$ являются корнями $g(x)$, и любое кодовое слово $a(x) = \sum_{i=0}^{n-1} \alpha_i x^i$ делится на $g(x)$, имеет место система уравнений:

$$\begin{aligned}
&a(\eta^{m_0}) = \alpha_0 + \alpha_1\eta^{m_0} + \ldots + \alpha_{n-1}(\eta^{m_0})^{n-1} = 0;\\
&a(\eta^{m_0+1}) = \alpha_0 + \alpha_1\eta^{m_0+1} + \ldots + \alpha_{n-1}(\eta^{m_0+1})^{n-1} = 0;\\
&\quad\ldots\\
&a(\eta^{m_0+d_0-2}) = \alpha_0 + \alpha_1\eta^{m_0+d_0-2} + \ldots + \alpha_{n-1}(\eta^{m_0+d_0-2})^{n-1} = 0.
\end{aligned}$$

Перепишем эту систему в матричном виде:

$$\begin{aligned}
&(\alpha_0, \ldots, \alpha_{n-1})\times\\
&\times \begin{bmatrix}
1 & 1 & \ldots & 1\\
\eta^{m_0} & \eta^{m_0+1} & \ldots & \eta^{m_0+d_0-2}\\
(\eta^{m_0})^2 & (\eta^{m_0+1})^2 & \ldots & (\eta^{m_0+d_0-2})^2\\
\ldots & \ldots & \ldots & \ldots\\
(\eta^{m_0})^{n-1} & (\eta^{m_0+1})^{n-1} & \ldots & (\eta^{m_0+d_0-2})^{n-1}
\end{bmatrix} = \mathbf{0}.
\end{aligned} \tag{3.9}$$

Из равенства (3.9) следует, что фигурирующая в нем $n \times (d_0-1)$-матрица является транспонированной проверочной матрицей кода $\mathcal{G}$:

$$\mathbf{H} = \begin{bmatrix}
1 & \eta^{m_0} & \ldots & (\eta^{m_0})^{n-1}\\
1 & \eta^{m_0+1} & \ldots & (\eta^{m_0+1})^{n-1}\\
\ldots & \ldots & \ldots & \ldots\\
1 & \eta^{m_0+d_0-2} & \ldots & (\eta^{m_0+d_0-2})^{n-1}
\end{bmatrix}. \tag{3.10}$$

Отметим, что матрица (3.10) задана над полем $GF(q)$ – расширением того поля, над которым задан код. Число строк в ней отлично от $r = n - k$. До сих пор мы задавали проверочную матрицу в основном поле, и число строк в ней было строго равно r (см., например, (3.6)). Чтобы перейти от матрицы (3.10) к традиционному виду, достаточно представить элементы η^i в ней как векторы-столбцы и удалить из получившейся матрицы линейно зависимые строки.

Пользуясь матрицей $\mathbf{H}$ в виде (3.10), для доказательства теоремы достаточно доказать, что любые $d_0 - 1$ столбцов этой матрицы линейно

независимы. Для этого достаточно показать, что определитель любой $(d_0-1)\times(d_0-1)$-подматрицы матрицы $\mathbf{H}$ отличен от нуля. Составим определитель $\mathbf{D}$ из столбцов матрицы $\mathbf{H}$ с номерами $i_1,\ldots,i_{d_0-1}$:

$$\mathbf{D}=\begin{bmatrix} (\eta^{m_0})^{i_1} & (\eta^{m_0})^{i_2} & \ldots & (\eta^{m_0})^{i_{d_0-1}} \\ (\eta^{m_0+1})^{i_1} & (\eta^{m_0+1})^{i_2} & \ldots & (\eta^{m_0+1})^{i_{d_0-1}} \\ \ldots & \ldots & \ldots & \ldots \\ (\eta^{m_0+d_0-2})^{i_1} & (\eta^{m_0+d_0-2})^{i_2} & \ldots & (\eta^{m_0+d_0-2})^{i_{d_0-1}} \end{bmatrix}.$$

Вынося из каждого столбца $\eta^{m_0 i_j}$ и обозначая η^{i_j} через η_j, $\eta^{i_j}=\eta_j$, получим

$$\mathbf{D}=\begin{bmatrix} 1 & 1 & \ldots & 1 \\ \eta_1 & \eta_2 & \ldots & \eta_{d_0-1} \\ (\eta_1)^2 & (\eta_2)^2 & \ldots & (\eta_{d_0-1})^2 \\ \ldots & \ldots & \ldots & \ldots \\ (\eta_1)^{d_0-1} & (\eta_2)^{d_0-1} & \ldots & (\eta_{d_0-1})^{d_0-1} \end{bmatrix}\times \\ \times(\eta_1\cdot\eta_2\cdot\ldots\cdot\eta_{d_0-1})^{m_0}. \tag{3.11}$$

Определитель в правой части равенства (3.11) представляет собой определитель Вандермонда. Нетрудно показать, что он равен произведению разностей элементов его второй строки:

$$\begin{bmatrix} 1 & 1 & \ldots & 1 \\ \eta_1 & \eta_2 & \ldots & \eta_{d_0-1} \\ \ldots & \ldots & \ldots & \ldots \\ (\eta_1)^{d_0-1} & (\eta_2)^{d_0-1} & \ldots & (\eta_{d_0-1})^{d_0-1} \end{bmatrix}=\prod_{i>j}(\eta_i-\eta_j). \tag{3.12}$$

Из равенства (3.12) следует, что определитель $\mathbf{D}$ отличен от нуля, если все элементы η_i, $i=\overline{1,d_0-1}$, различны. Поскольку элементы η_i представляют собой различные степени примитивного элемента (или корни $g(x)$), то $\mathbf{D}\neq 0$, и теорема доказана.

Теорему 3.1 называют *БЧХ-границей* кодового расстояния циклических кодов. Она позволяет оценивать расстояние любого циклического кода посредством анализа корней его порождающего многочлена.

Пример 3.2. Рассмотрим двоичный циклический код $\mathcal{G}$ с порождающим многочленом $g(x)=x^7+x^6+x^5+x^2+x+1$. Определим параметры n, k и d этого кода. Для определения длины кода надо найти двучлен x^n-1 минимальной степени, который делится на $g(x)$. Для этого воспользуемся следующим приемом, более эффективным, чем последовательное деление многочленов x^8+1, x^9+1 и т.д. на $g(x)$. Будем искать

одночлен x^n, остаток от деления которого на $g(x)$ равен 1. Остаток от деления x^7 на $g(x)$:

$$x^7 = x^6 + x^5 + x^2 + x + 1 \bmod g(x). \tag{3.13}$$

Умножим на x по модулю $g(x)$ левую и правую части (3.13):

$$\begin{aligned} x^8 &= x^7 + x^6 + x^3 + x^2 + x = \\ &= \underline{x^6} + x^5 + \underline{x^2} + \underline{x} + 1 + \underline{x^6} + x^3 + \underline{x^2} + \underline{x} = \\ &= x^5 + x^3 + 1 \bmod g(x). \end{aligned}$$

Аналогично будем вычислять остатки от деления на $g(x)$ одночленов x^9, x^{10} и т.д., пока остаток не станет равным 1.

$$\left.\begin{aligned} x^9 &= x^6 + x^4 + x; \\ x^{10} &= x^6 + x + 1; \\ x^{11} &= x^6 + x^5 + 1; \\ x^{12} &= x^5 + x^2 + 1; \\ x^{13} &= x^6 + x^3 + x; \\ x^{14} &= x^6 + x^5 + x^4 + x + 1; \\ x^{15} &= 1. \end{aligned}\right\} \bmod g(x)$$

Таким образом, длина кода $n = 15$. Число информационних символов равно разности длины кода и степени его проверочного многочлена:

$$k = n - \deg g(x) = 15 - 7 = 8.$$

Поскольку $g(x)$ является делителем $x^n - 1$, а $n = 2^4 - 1$, то все корни $g(x)$ лежат в поле $GF(2^4)$. Пусть η – примитивный элемент этого поля. Непосредственной проверкой легко убедиться, что корнями $g(x)$ являются элементы $\eta^0 = 1$, η^5, η^6, η^9, η^{10}, η^{12}. Самая длинная последовательность из подряд стоящих степеней η во множестве корней $g(x)$ состоит всего из двух элементов η^9, η^{10} и, следовательно, расстояние рассматриваемого кода $d \geq 3$.

Теорема 3.1 дает оценку расстояния циклических кодов, которая может быть меньше истинного расстояния циклического кода. В частности, в рассмотренном примере перебор по словам кода показывает, что его расстояние равно 4. В некоторых случаях оценку БЧХ можно уточнить.

Следствие 3.1. *Циклический код, среди корней которого имеются*

$$\eta^i, \eta^{i+j}, \eta^{i+2j}, \ldots, \eta^{i+(d_0-2)j}, \tag{3.14}$$

где η *– примитивный элемент соответствующего поля, имеет минимальное расстояние* d*, не меньшее* d_0*, если* j *и* n *взаимно просты.*

Доказательство. Обозначим η^j через γ. Поскольку j и n взаимно просты, то γ – примитивный элемент того же поля, что и η. Тогда найдется m_0 такое, что $\eta^i = \gamma^{m_0}$, и последовательность корней (3.14) примет вид $\gamma^{m_0}, \gamma^{m_0+1}, \ldots, \gamma^{m_0+d_0-2}$. Теперь доказываемое утверждение сразу следует из теоремы 3.1.

3.3. БЧХ-коды

Рассмотрим следующий способ построения циклического кода длины $n = q^m - 1$ (q – степень простого числа), исправляющего t ошибок.

Выберем примитивный многочлен степени m и построим конечное поле $GF(q^m)$. Найдем минимальные многочлены $f_j(x)$ для элементов η^j, $j = 1, 2, \ldots, 2t$. Положим

$$g(x) = \text{НОК}\{f_1(x), f_2(x), \ldots, f_{2t}(x)\}.$$

В соответствии с теоремой 3.1 такой код, называемый *кодом БЧХ*, имеет минимальное расстояние $d \geq d_0 = 2t + 1$, т.е. позволяет исправлять t ошибок.

В табл. 3.1 приведены минимальные многочлены для элементов поля $GF(2^4)$ и $GF(4^2)$.

Отметим, что в соответствии со свойствами конечных полей (см. определение 1.6) минимальные многочлены для элементов η, η^2, η^4, ... совпадают, поэтому минимальные многочлены для многих элементов в табл. 3.1 одинаковы.

Пример 3.3. Построим двоичные коды БЧХ длины 15. Порождающий код для БЧХ-кода, исправляющего одну ошибку,

$$\begin{aligned} g(x) &= \text{НОК}\{f_1(x), f_2(x)\} = \\ &= \text{НОК}\{x^4 + x + 1, x^4 + x + 1\} = x^4 + x + 1. \end{aligned}$$

Мы построили $(15, 11)$-код с $d = 3$.

Для кода, исправляющего две ошибки,

$$g(x) = \text{НОК}\{f_1(x), f_2(x), f_3(x), f_4(x)\} =$$

Таблица 3.1

Минимальные многочлены для полей $GF(2^4)$ и $GF(4^2)$

Поле $GF(2^4)$		Поле $GF(4^2)$	
Элемент поля	Минимальный многочлен	Элемент поля	Минимальный многочлен
0	0	0	0
1	$x+1$	1	$x+1$
η	x^4+x+1	η	x^2+x+2
η^2	x^4+x+1	η^2	x^2+x+3
η^3	$x^4+x^3+x^2+x+1$	η^3	x^2+3x+1
η^4	x^4+x+1	η^4	x^2+x+2
η^5	x^2+x+1	η^5	$x+2$
η^6	$x^4+x^3+x^2+x+1$	η^6	x^2+2x+1
η^7	x^4+x^3+1	η^7	x^2+2x+2
η^8	x^4+x+1	η^8	x^2+x+3
η^9	$x^4+x^3+x^2+x+1$	η^9	x^2+2x+1
η^{10}	x^2+x+1	η^{10}	$x+3$
η^{11}	x^4+x^3+1	η^{11}	x^2+3x+3
η^{12}	$x^4+x^3+x^2+x+1$	η^{12}	x^2+3x+1
η^{13}	x^4+x^3+1	η^{13}	x^2+2x+2
η^{14}	x^4+x^3+1	η^{14}	x^2+3x+3

$$= \text{НОК}\{x^4+x+1, x^4+x+1,$$
$$x^4+x^3+x^2+x+1, x^4+x+1\} =$$
$$= (x^4+x+1)(x^4+x^3+x^2+x+1) =$$
$$= x^8+x^7+x^6+x^4+1.$$

Получен $(15,7)$-код с расстоянием 5.

Для кода, исправляющего 3 ошибки ($t=3$),

$$g(x) = \text{НОК}\{f_1(x), f_2(x), f_3(x), f_4(x), f_5(x), f_6(x)\} =$$
$$= x^{10}+x^8+x^5+x^4+x^2+x+1.$$

Получен (15,5)-код с расстоянием $d=7$.

Наконец, при $t=4$

$$g(x) = x^{14}+x^{13}+x12+x^{11}+x^{10}+x^9+x^8+x^7+x^6+$$
$$+x^5+x^4+x^3+x^2+x+1$$

и получен код (15,1). Этот код с повторением, действительное расстояние которого равно 15. Таким образом, на длине $n=15$ при $t \geq 4$ не существует циклического кода, отличного от кода с повторением (15,1).

Пример 3.4. Построим коды БЧХ длины 15 над полем $GF(4)$.

1. $t = 1$:

$$g(x) = \text{НОК}\{f_1(x), f_2(x)\} = (x^2 + x + 2)(x^2 + x + 3) = \\ = x^4 + x + 1.$$

Действия над коэффициентами многочленов проводятся в поле $GF(4)$. Получился порождающий многочлен кода (15,11), исправляющего одну четверичную ошибку.

2. $t = 2$:

$$g(x) = \text{НОК}\{f_1(x), f_2(x), f_3(x), f_4(x)\} = \\ = (x^2 + x + 2)(x^2 + x + 3)(x^2 + 3x + 1) = \\ = x^6 + 3x^5 + x^4 + x^3 + 2x^2 + 2x + 1.$$

Получился код (15,9) с $d = 5$.

3. $t = 3$:

$$g(x) = x^9 + 3x^8 + 3x^7 + 2x^6 + x^5 + 2x^4 + x + 2.$$

Код (15,6) с $d = 7$.

4. $t = 4$:

$$g(x) = x^{11} + x^{10} + 2x^8 + 3x^7 + 3x^6 + x^5 + 3x^4 + x^3 + x + 3.$$

Код (15.4) с $d = 9$.

5. $t = 5$:

$$g(x) = x^{12} + 2x^{11} + 3x^{10} + 2x^9 + 2x^8 + x^7 + 3x^6 + 3x^4 + 3x^3 + x^2 + 2.$$

Код (15,3) с $d = 11$.

6. $t = 6$:

$$g(x) = x^{14} + x^{13} + x^{12} + x^{11} + x^{10} + \\ + x^9 + x^8 + x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + x + 1.$$

Код (15,1) с $d = 15$.

Таким образом, можно сделать вывод, что над полем $GF(4)$ нетривиальные коды существуют для $t \leq 5$.

В общем случае для определения кода БЧХ может использоваться не только примитивный элемент поля.

Определение 3.2. Пусть γ – элемент поля $GF(q^m)$, мультипликативный порядок которого равен N, и пусть $m_0 \geq 0$ и $2 \leq d_0 \leq N$ – целые числа. Пусть, кроме того, $f_{m_0}(x)$, $f_{m_0+1}(x)$, ..., $f_{m_0+d_0-2}(x)$ – минимальные многочлены для γ^{m_0}, γ^{m_0+1}, ..., $\gamma^{m_0+d_0-2}$ соответственно, тогда циклический код с порождающим многочленом

$$g(x) = \text{НОК}\{f_{m_0+1}(x), \ldots, f_{m_0+d_0-2}(x)\}$$

называется *кодом БЧХ*.

Параметры кода БЧХ: $n = N$; $k = N - \deg g(x)$ ($\deg(\cdot)$ – степень многочлена $(\cdot)$); $d \geq d_0$.

Если $n = N = q^m - 1$, то код БЧХ называется *примитивным*.

Пример 3.5. Построим двоичный код БЧХ длиной 9 с $d = 3$. Для этого нужно выбрать элемент γ мультипликативного порядка $N = 9$. Однако число 9 нельзя представить в виде $2^m - 1$ и, следовательно, не существует поля характеристики 2, в котором γ является примитивным. Найдем минимальное поле характеристики 2, в котором содержится γ. Для этого нужно найти такое m, чтобы двучлен $x^{2^m} - 1$ делился на двучлен $x^9 - 1$. Последнее возможно, если и только если $2^m - 1$ делится на 9. Таким минимальным m будет $m = 6$, и значит, $\gamma \in GF(2^6)$. Если η – примитивный элемент поля $GF(2^6)$, то в качестве γ можно выбрать элемент η^7. Действительно,

$$\eta^9 = (\eta^7)^9 = \eta^{63} = 1.$$

Минимальным многочленом для η^7 и η^{14} в поле $GF(2^6)$ будет

$$f_7(x) = f_{14}(x) = x^6 + x^3 + 1.$$

Тогда порождающий многочлен

$$g(x) = \text{НОК}\{f_7(x), f_{14}(x)\} = x^6 + x^3 + 1.$$

Получен непримитивный БЧХ-код: $n = 9$; $k = 3$; $d = 3$.

Число проверочных символов r БЧХ-кодов (степень порождающего многочлена), построенного с помощью элемента $\gamma \in GF(q^m)$, может быть оценено произведением

$$r \leq m(d-1). \tag{3.15}$$

Неравенство (3.15) следует из того, что проверочная матрица (3.10) кода БЧХ содержит $d - 1$ строк, элементы которых представляются в виде q-ичного столбца длины m.

Для двоичных кодов оценка (3.15) может быть улучшена с учетом того, что для таких кодов корнем $g(x)$ вместе с γ является и элемент γ^2, и, следовательно, строки с четными номерами в матрице (3.10) линейно зависимы от строк с нечетными номерами. Поэтому

$$r \le mt. \tag{3.16}$$

Уточнение истинного расстояния циклических кодов, более точная оценка числа проверочных символов кодов БЧХ являются нерешенными задачами теории кодирования.

3.4. Коды Рида–Соломона

Коды Рида–Соломона (РС), предложеные в 1960 г., до сих пор остаются одними из наиболее применяемых на практике, и, несмотря на их простоту, служат основой новых, глубоких обобщений [1,11]. Начнем их обсуждение со следующего простого замечания.

Теорема 3.2 (граница Синглтона)**.** *Для любого линейного (n,k)-кода $\mathcal{G}$ с расстоянием d*

$$n - k \ge d - 1. \tag{3.17}$$

Доказательство. Поскольку в проверочной матрице $\mathbf{H}$ (n,k)-кода с расстоянием d любые $d-1$ столбцов линейно независимы, то ранг матрицы $\mathbf{H}$ по столбцам – не менее $d-1$. В то же время, ранг по строкам не больше числа строк, а так как ранги матрицы по столбцам и строкам совпадают, то для любого (n,k)-кода справедлива граница (3.17), что и требовалось доказать.

Коды, достигающие границы (3.17), являются оптимальными с точки зрения минимального расстояния, и их принято называть кодами с максимально достижимым расстоянием (МДР).

Таким образом, в проверочной $(r \times n)$-матрице кода МДР любые r столбцов линейно независимы. Примером такой матрицы является матрица $\mathbf{H} = (h_{ij}) = (\alpha_j^{i-1})$, где $\alpha_1, \ldots, \alpha_n$ – различные элементы поля $GF(q)$ $(n \le q)$.

Иначе говоря, строками матрицы являются значения одночленов z^{i-1}, вычисленные в точках $\alpha_1, \ldots, \alpha_n$ поля $GF(q)$:

$$\mathbf{H}_r = \begin{pmatrix} 1 & 1 & \ldots & 1 \\ \alpha_1 & \alpha_2 & \ldots & \alpha_n \\ \ldots & & & \\ \alpha_1^{r-1} & \ldots & \ldots & \alpha_n^{r-1} \end{pmatrix}. \tag{3.18}$$

Покажем, что в матрице $\mathbf{H}$ (3.18) любые r столбцов линейно независимы. Пусть это не так и столбцы $\mathbf{h}_{j_1}, \ldots, \mathbf{h}_{j_t}$ линейно зависимы. Это равносильно тому, что $\det(\mathbf{h}_{j_1}, \ldots, \mathbf{h}_{j_t}) = 0$, а значит, линейно зависимы и строки этого минора $\mathbf{H}_{j_1,\ldots,j_t}$.

Соотношение линейной зависимости между его строками с коэффициентами $\lambda_0, \ldots, \lambda_{r-1}$ означает, что многочлен $\lambda(z) = \lambda_0 + \lambda_1 z + \ldots + \lambda_{r-1} z^{r-1}$ степени меньше r имеет r разных корней $\lambda_{j_1}, \ldots, \lambda_{j_t}$, что невозможно.

Из утверждения относительно матрицы (3.18) нетрудно получить, что в матрицах $\mathbf{H}_{a,r}$ более общего вида, строки которых задаются как значения одночленов z^j $(j = a, a+1, \ldots, a+r-1)$ на элементах α_j, любые r столбцов линейно независимы:

$$\mathbf{H}_{a,r} = \begin{pmatrix} \alpha_1^a & \ldots & \alpha_n^a \\ \alpha_1^{a+1} & \ldots & \alpha_n^{a+1} \\ \ldots & & \\ \alpha_1^{a+r-1} & \ldots & \alpha_n^{a+r-1} \end{pmatrix}. \tag{3.19}$$

Определение 3.3. Коды, задаваемые проверочными матрицами вида (3.19), называются *кодами Рида–Соломона (РС)*.

Более употребительно задание кодов РС как циклических кодов.

Пусть элементы $\alpha_1, \ldots, \alpha_n$ образуют подгруппу и, следовательно, циклическую (потому что группа по умножению всего конечного поля является циклической). Тем самым существует элемент $\beta \in GF(q)$, такой, что при подходящей перенумерации $\alpha_i = \beta^{i-1}$, $i = \overline{1,n}$. Тогда матрица $\mathbf{H}$, задаваемая (3.19), преобразуется к виду

$$\mathbf{H}_{a,r,\beta} = \begin{pmatrix} 1 & \beta^a & \ldots & \beta^{a(n-1)} \\ 1 & \beta^{a+1} & \ldots & \beta^{(a+1)(n-1)} \\ \ldots & & & \\ 1 & \beta^{a+r-1} & \ldots & \beta^{(a+r-1)(n-1)} \end{pmatrix}. \tag{3.20}$$

Заметим, что мы получили матрицу вида (3.10). Поскольку циклический сдвиг любой строки $\mathbf{H}_{a,r,\beta}$ принадлежит линейному пространству строк этой матрицы (а конкретно – получается умножением на $\beta^{a(n-1)}$ этой же строки), то код, задаваемый этой проверочной матрицей, является циклическим (как и двойственный ему код пространства строк матрицы $\mathbf{H}_{a,r,\beta}$). Условие принадлежности коду вектора $\mathbf{v}$: $\mathbf{H}_{a,r,\beta} \cdot \mathbf{v} = \mathbf{0}$ равносильно тому, что

$$v(\beta^j) = 0 \text{ для } j = \overline{a, a+r-1}. \tag{3.21}$$

Следовательно, порождающий многочлен этого кода

$$g(x) = \text{НОК}\{f_{\beta^j}(x)\}, \quad j = \overline{a, a+r-1}, \tag{3.22}$$

где $f_\gamma(x)$ – минимальный многочлен элемента γ. В данном случае $f(\gamma) = x - \gamma$ и

$$g(x) = \prod_{j=a}^{a+r-1} (x - \beta^j). \tag{3.23}$$

Наибольший интерес представляет выбор в качестве β примитивного элемента поля $GF(q)$. Тогда $n = q - 1$, и, собственно, такие коды чаще всего и называют кодами РС. Коды, задаваемые проверочными матрицами (3.18), могут иметь длину q, когда $\{\alpha_1, \ldots, \alpha_n\} = GF(q)$, а соответствующие коды называются 1-удлиненными кодами РС. Можно удлинить их и еще на 1 символ, добавив символ «∞» к множеству $\{\alpha_1, \ldots, \alpha_n\} = GF(q)$ и положив формально $\infty^j = 0$ при $j = 0, 1, \ldots, r-2$ и $\infty^{r-1} = 1$. Можно ли построить нетривиальные коды МДР (т.е. $2 < d < n$) длины $n > q + 1$, является сложной нерешенной задачей, уходящей своими корнями в проективную геометрию. Популярной является гипотеза, что таких кодов нет, за исключением кодов с $d = 4$, $n = q + 2$ и $q = 2^n$.

Пример 3.6. Построим РС-код с $n = 15$ и $t = 3$. Порождающий многочлен этого кода может быть вычислен так:

$$\begin{aligned}
g(x) &= (x - \eta)(x - \eta^2) \cdot \ldots \cdot (x - \eta^6) = \\
&= (x^2 + (\eta + \eta^2)x + \eta^3)(x^2 + (\eta^3 + \eta^4)x + \eta^7) \times \\
&\quad \times (x^2 + (\eta^5 + \eta^6)x + \eta^{11}) = \\
&= (x^2 + \eta^5 x + \eta^3)(x^2 + \eta^7 x + \eta^7)(x^2 + \eta^9 x + \eta^{11}) = \\
&= (x^4 + (\eta^5 + \eta^7)x^3 + (\eta^3 + \eta^7 + \eta^{12})x^2 + \\
&\quad + (\eta 12 + \eta^{10} x + \eta^{10}))(x^2 + \eta^9 x + \eta^{11}) = \\
&= (x^4 + \eta^{13} x^3 + \eta^6 x^2 + \eta^3 x + \eta^{10})(x^2 + \eta^9 x + \eta^{11}) = \\
&= x^6 + (\eta^{13} + \eta^9)x^5 + (\eta^{11} + \eta^7 + \eta^6)x^4 + \\
&\quad + (\eta^9 + 1 + \eta^3)x^3 + (\eta^2 + \eta^{12} + \eta^{10})x^2 + (\eta^{14} + \eta^4)x + \eta^6 = \\
&= x^6 + \eta^{10} x^5 + \eta^{14} x^4 + \eta^4 x^3 + \eta^6 x^2 + \eta^9 x + \eta^6.
\end{aligned}$$

Пример 3.7. Зададим код длины $q + 2$ с помощью систематического кодирования. А именно, набору информационных символов $a_1, \ldots, a_{q-1}$

противопоставим три проверочных символа b_0, b_1, b_2, вычисляемых по формулам

$$b_0 = \sum_{i=0}^{q-1} a_i; \quad b_1 = \sum_{i=0}^{q-1} a_i \alpha^i; \quad b_2 = \sum_{i=0}^{q-1} a_i \alpha_{2i},$$

где α – примитивный элемент поля $GF(q)$.

Несложно проверяется, что для $q = 2^m$ расстояние этого кода равно 4, т.е. это нетривиальный код МДР длины $q + 2$.

Пусть V – $(n, n-r)$-код РС, задаваемый проверочной матрицей (3.19) над полем $GF(Q)$, $Q = q^m$. Поскольку поле $GF(q)$ может быть вложено как подполе в поле $GF(q^m)$, то рассмотрим подкод V_q кода V над подмножеством $GF(q)$. Этот код есть ни что иное, как q-ичный код БЧХ. Из общих соотношений для подкода над подмножеством следует, что $|V_q| \geq q^{n-rm}$. В данном случае эту оценку можно уточнить, рассмотрев поле $GF(q)$ как m-мерное векторное пространство над $GF(q)$, а элементы матрицы $\mathbf{H}_{a,r}$ как m-мерные столбцы над полем $GF(q)$. Тогда ранг (q-ичный) новой матрицы $\mathbf{H}^{(q)}_{a,r}$ равен числу проверочных символов кода V_q . Известно, что в поле $GF(q^m)$ существует нормальный базис (определение 1.8), т.е. такой элемент γ, что элементы $\gamma, \gamma^q, \ldots, \gamma^{q^{m-1}}$ являются базисом $GF(q^m)$ как векторного пространства над полем $GF(q)$. В этом базисе «блок» из m строк матрицы $\mathbf{H}^{(q)}_{a,r}$, соответствующий строке «z^{jq}» матрицы $\mathbf{H}_{a,r}$, получается циклическим сдвигом (вниз) строк «блока» из m строк, соответствующего строке «z^j». Отсюда следует неравенство для числа $r^{(q)}$ проверочных символов q-ичного кода $V^{(q)}$ длины $n \leq q^m$ и расстоянием $\geq d$, которые мы выпишем для двух наиболее интересных частных случаев:

$$r^q \leq m(d - 1 - \lfloor (d-1)/q \rfloor) \qquad \text{для } a = 1; \tag{3.24}$$

$$r^q \leq 1 + m(d - 2 - \lfloor (d-2)/q \rfloor) \qquad \text{для } a = 0. \tag{3.25}$$

Подставив в выражение (3.24) $q = 2$ и $q = 2t+1$, получим, что число проверочных символов кода БЧХ длины n, исправляющего t ошибок, не превышает $t\lceil \log_2 n \rceil$. Отсюда и из границы Хэмминга следует, что коды БЧХ асимптотически оптимальны «по избыточности», т.е.

$$\lim_{n\to\infty} \frac{r_{\text{БЧХ}}}{r(n, 2t+1)} = 1.$$

Подставив в (3.25) $q = 3$ и $d = 5$, получим, что число проверочных символов соответствующих троичных кодов БЧХ, исправляющих две

ошибки, не превышает $1 + 2\lceil \log_3 n \rceil$, что асимптотически эквивалентно соответствующей границе Хэмминга. При $n = 3^m$ эти коды оптимальны в классе линейных кодов. Кроме этого, известно еще одно семейство кодов – четверичные коды, исправляющие две ошибки, также асимптотически удовлетворяющие границе Хэмминга для числа проверочных символов. В общем случае, неизвестны семейства кодов с другими параметрами q и t, кроме отмеченных выше, для которых бы

$$\lim_{n\to\infty} \frac{r}{t \log_q n} = 1; \tag{3.26}$$

$$q, t = \text{const}.$$

Коды БЧХ не удовлетворяют соотношению (3.26) при $q > 2$ (за исключением случая $q = 3$, $t = 2$). Для $t = 2$ построены коды асимптотически лучшие, чем q-ичные коды БЧХ, т.е. у которых левая часть (3.26) имеет большее значение.

3.5. Задачи

1. Построить порождающую и проверочную матрицы для каждого двоичного циклического (15,8)-кода.

2. Определить расстояние циклического кода с порождающим многочленом

$$g(x) = x^8 + x^7 + x^6 + x^4 + 1.$$

3. Найти порождающий и проверочный многочлены двоичного циклического (31,21)-кода, исправляющего две ошибки.

4. Найти порождающий и проверочный многочлены двоичного циклического (21,11)-кода, исправляющего две ошибки.

5. Доказать, что если в q-ичном циклическом (n, k)-коде n и q взаимно простые, то слово из всех единиц принадлежит коду тогда и только тогда, когда порождающий многочлен кода делится на $x - 1$.

6. Разложить многочлен $x^8 - 1$ над $GF(3)$. Сколько циклических кодов длины 8 имеется над $GF(3)$?

7. Построить все РС-коды длины 7 и 8.

8. Доказать, что любые k столбцов порождающей матрицы максимального кода линейно независимы.

9. Следует:

а) вычислить спектр (15,4)-кода РС;

б) определить вероятность необнаруженной ошибки, обеспечиваемой этим кодом в двоичном симметричном канале с вероятностью ошибки $p = 0.1$.

10. Построить:

а) двоичный БЧХ-код длины 15, исправляющий 1 ошибку. Выписать порождающий и проверочный многочлен, указать размерность;

б) РС-код, исправляющий 3 ошибки. Выписать порождающий многочлен, порождающую и проверочную матрицы, указать параметры получившегося кода.

Оценить параметры кода, заданного порождающим многочленом $g(x) = x^8 + x^7 + x^6 + x^4 + 1$.

11. Построить:

а) двоичный БЧХ-код длины 15, исправляющий 2 ошибки. Выписать порождающий и проверочный многочлен, указать размерность;

б) РС-код длины 8 с расстоянием 4. Выписать порождающий многочлен, порождающую и проверочную матрицы, указать параметры получившегося кода.

Оценить параметры кода, заданного порождающим многочленом $g(x) = x^9 + x^6 + x^5 + x^4 + x + 1$.

12. Построить:

а) двоичный БЧХ-код длины 15, исправляющий 3 ошибки. Выписать порождающий и проверочный многочлен, указать размерность;

б) РС-код длины 12, исправляющий 3 ошибки. Выписать порождающий многочлен, порождающую и проверочную матрицы, указать параметры получившегося кода.

Определить, является ли циклическим код, заданный порождающим многочленом $g(x) = 1+x^{10}+x^3+x^2+x^4+x^8+x^7+x^6+x^9$? Ответ обосновать. Если да, то определить параметры кода.

13. Построить:

а) двоичный БЧХ-код длины 7, исправляющий 1 ошибку. Выписать порождающий и проверочный многочлен, указать размерность;

б) РС-код с параметрами (11,7). Выписать порождающий многочлен, порождающую и проверочную матрицы, указать расстояние получившегося кода.

Найти длину циклического кода, заданного порождающим

многочленом $g(x) = x^{10} + x^9 + x^8 + x^6 + x^5 + x^3 + 1$.

14. Построить:

а) двоичный БЧХ-код с расстоянием 4. Выписать порождающий и проверочный многочлен, указать недостающие параметры;

б) РС-код с параметрами (12,7). Выписать порождающий многочлен, порождающую и проверочную матрицы, указать расстояние получившегося кода.

Найти длину циклического кода, заданного порождающим многочленом $g(x) = x^{10} + x^9 + x^7 + x + 1$.

15. Построить:

а) двоичный БЧХ-код с параметрами (15,3). Выписать порождающий и проверочный многочлен, указать расстояние;

б) РС-код с параметрами (13,к,7). Выписать порождающий многочлен, порождающую и проверочную матрицы, указать размерность получившегося кода.

Определить, является ли циклическим код, заданный порождающим многочленом $g(x) = x^{10} + x^9 + x^7 + x^6 + 1$? Ответ обосновать. Если да, то определить параметры кода.

16. Построить:

а) двоичный БЧХ-код длины 7 с минимальным расстоянием 3. Выписать порождающий и проверочный многочлен, указать размерность;

б) РС-код с параметрами (9,к,7). Выписать порождающий многочлен, порождающую и проверочную матрицы, указать размерность получившегося кода.

Найти длину циклического кода, заданного порождающим многочленом $g(x) = x^{10} + x^5 + 1$.

17. Построить:

а) двоичный БЧХ-код длины 15 с расстоянием 4. Выписать порождающий и проверочный многочлен, указать размерность;

б) РС-код с параметрами (11,5). Выписать порождающий многочлен, порождающую и проверочную матрицы, указать расстояние получившегося кода.

Найти параметры циклического кода, заданного порождающим многочленом $g(x) = 1 + x^{10} + x + x^5 + x^2 + x^4 + x^8$.

18. Построить:

а) двоичный БЧХ-код длины 15 с расстоянием 5. Выписать порождающий и проверочный многочлен, указать размерность;

б) РС-код с параметрами (11,6). Выписать порождающий многочлен, порождающую и проверочную матрицы, указать расстояние получившегося кода.

Найти параметры циклического кода, заданного порождающим многочленом $g(x) = x^7 + x^6 + x^4 + 1$.

19. Построить:

а) двоичный БЧХ-код длины 15, исправляющий 2 ошибки. Выписать порождающий и проверочный многочлен, указать размерность;

б) Построить РС-код длины 10, исправляющий 3 ошибки. Выписать порождающий многочлен, порождающую и проверочную матрицы, указать расстояние получившегося кода.

Определить, является ли циклическим код, заданный порождающим многочленом $g(x) = x^6 + x^3 + x^4 + 1$? Ответ обосновать. Если да, то указать длину кода.

4. Декодирование линейных кодов

4.1. Полное декодирование линейных блоковых кодов

В разделе 2 мы задали линейные коды, рассмотрели вопросы задания и кодирования линейных кодов, а также существующие границы. В этом разделе будут описаны общие методы декодирования линейных блоковых кодов.

Пусть $\mathbf{b}$ – принятое слово, $\{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_N\}$ – множество кодовых слов. Будем рассматривать аддитивные каналы, т. е. такие, в которых произошедшие в канале ошибки можно описать с помощью вектора ошибок $\mathbf{e}$, покомпонентно добавляемому к передаваемому кодовому слову $\mathbf{a}$:

$$\mathbf{b} = \mathbf{a} + \mathbf{e}.$$

Процедура декодирования заключается в том, чтобы по принятому слову определить, какое кодовое слово передавалось. При этом нужно максимизировать условную вероятность $P(\mathbf{a}|\mathbf{b})$. Слово, для которого вероятность будет максимальной, и будет считаться результатом декодирования. Такое декодирование называется декодированием *по максимуму правдоподобия*. Другим способом декодирования является нахождение кодового слова, «ближайшего» к принятому в терминах некоторого расстояния, например, Хэмминга, т. е. задачей является минимизация $d(\mathbf{a}, \mathbf{b})$. Такое декодирование называется декодированием *по минимуму расстояния*, оно может быть осуществлено перебором по всем 2^k кодовым словам.

Рассмотрим таблицу на рис. 4.1. Строится она следующим образом: первую строку составляют кодовые слова, вторая состоит из сумм кодовых слов и некоторого $\mathbf{e}_2$, где $\mathbf{e}_2$ выбирается как любой вектор, не вошедший в первую строку. Третья строка состоит из сумм кодовых слов и некоторого $\mathbf{e}_3$, где $\mathbf{e}_3$ выбирается как любой вектор, не вошедший в первые две строки. Построение таблицы заканчивается, когда она содержит все 2^n векторов пространства, таким образом, нельзя выбрать новый вектор $\mathbf{e}_i$ для построения новой строки.

Построенная таблица называется *стандартной расстановкой* и обладает рядом свойств, сформулируем их с помощью нескольких лемм.

Лемма 4.1. *Стандартная расстановка содержит все 2^n векторов пространства, причем каждый элемент – ровно один раз.*

Доказательство. Стандартная расстановка содержит все элементы пространства по построению. Так как все кодовые слова (первая строка стандартной расстановки) различны, и при построении i-й строки к

каждому слову прибавляется один и тот же вектор $\mathbf{e}_i$, то одна строка не может содержать двух одинаковых векторов. Наконец, предположим, что два элемента таблицы в строках i, j (пусть для определенности $i < j$) и столбцах t, s совпадают, т.е.

$$\mathbf{e}_i + \mathbf{a}_t = \mathbf{e}_j + \mathbf{a}_s.$$

Тогда $\mathbf{e}_j = \mathbf{e}_i + (\mathbf{a}_t + \mathbf{a}_s)$. Так как $\mathbf{a}_t$ и $\mathbf{a}_s$ – кодовые слова линейного кода, их сумма также является кодовым словом, скажем $\mathbf{a}_x$. Тогда $\mathbf{e}_j = \mathbf{e}_i + \mathbf{a}_x$, чего не может быть по построению стандартной расстановки.

Из доказанной леммы следует, что стандартная расстановка содержит в точности $2^n/2^k = 2^r$ строк, называемых *смежными классами*. Вектор $\mathbf{e}_i$, используемый для построения i-го смежного класса, назовем *лидером* смежного класса [11, 12].

Лемма 4.2. *Выбор лидера смежного класса не влияет на элементы этого класса, а лишь на их порядок внутри класса.*

Доказательство. Действительно, пусть у нас есть смежный класс с лидером $\mathbf{e}_i$. Предположим, при построении стандартной расстановки мы выбрали бы какой-то другой элемент этого смежного класса, скажем $\mathbf{e}_i + \mathbf{a}_x$. Но тогда любой элемент «нового» смежного класса имел бы вид $(\mathbf{e}_i + \mathbf{a}_x) + \mathbf{a}_j$, где $\mathbf{a}_j$ – некоторое кодовое слово. Так как код линеен, то $(\mathbf{a}_x + \mathbf{a}_j)$ также является кодовым словом, и следовательно, $\mathbf{e}_i + (\mathbf{a}_x + \mathbf{a}_j)$ принадлежит также и «старому» смежному классу. Если $\mathbf{a}_j$ пробегает все кодовые слова, то $(\mathbf{a}_x + \mathbf{a}_j)$ при фиксированном $\mathbf{a}_x$ также пробегает все кодовые слова, но в другом порядке, таким образом, «старый» и «новый» смежные классы отличаются только порядком своих элементов, что и требовалось доказать.

Итак, мы получили, что линейный (n, k)-код, т. е. k-мерное подпространство, разбивает n-мерное пространство на 2^{n-k} непересекающихся множеств (смежных классов). При этом выбор лидеров смежных классов влияет только на порядок «выписывания» смежных классов и на порядок элементов в них, но не на саму принадлежность элементов n-мерного пространства смежным классам.

Заметим, что синдром любого вектора в смежном классе с лидером $\mathbf{e}$ одинаков:

$$\mathbf{S}(\mathbf{a}_i + \mathbf{e}) = \mathbf{a}_i \cdot \mathbf{H}^T + \mathbf{e} \cdot \mathbf{H}^T = \mathbf{e} \cdot \mathbf{H}^T; \quad i = \overline{1, 2^k}.$$

Таким образом, если вычислить синдром (2.5) по принятому вектору $\mathbf{b} = \mathbf{a} + \mathbf{e}$, где $\mathbf{a}$ – передававшееся кодовое слово, $\mathbf{e}$ – вектор ошибки, это позволит определить смежный класс, в котором лежит вектор $\mathbf{e}$, но

$\mathbf{a}_1$	$\mathbf{a}_2$	...	$\mathbf{a}_{2^k}$
$\mathbf{e}_2$	$\mathbf{a}_2 + \mathbf{e}_2$	...	$\mathbf{a}_{2^k} + \mathbf{e}_2$
...	...	...	...
$\mathbf{e}_{2^r}$	$\mathbf{a}_2 + \mathbf{e}_{2^r}$	...	$\mathbf{a}_{2^k} + \mathbf{e}_{2^r}$

Рис. 4.1. Стандартная расстановка

не сам вектор $\mathbf{e}$. Будем считать, что вектором ошибки является лидер смежного класса, чей синдром совпадает с синдромом принятого слова. Тогда можно сформулировать следующую процедуру декодирования:

1. Вычисляется синдром $\mathbf{S}$ принятого вектора $\mathbf{b}$.

2. В таблице стандартной расстановки ищется синдром $\mathbf{S}$ и определяется соответствующий $\mathbf{S}$ лидер смежного класса $\mathbf{e}$.

3. Вычисляется вектор $\hat{\mathbf{a}} = \mathbf{b} + \mathbf{e}$ – декодированный вариант принятого вектора $\mathbf{b}$.

Описанный метод декодирования называется *синдромным декодированием*, или декодированием по стандартной расстановке.

Таблица на рис. 4.1 описывает все векторы, которые может исправить код – это лидеры смежных классов. Ошибка декодирования произойдет тогда, когда реальный вектор ошибки не являлся лидером смежного класса. Однако свойства стандартной расстановки позволяют, выбирая различные вектора в качестве лидеров и не меняя при этом самих смежных классов, приспособить стандартную расстановку к конкретному каналу связи таким образом, чтобы лидерами были самые вероятные в данном канале векторы. Например, в канале ДСК более вероятными векторами являются векторы меньшего веса Хэмминга, и, выбрав в качестве лидеров самые легкие векторы смежного класса, получим, что в данном случае декодирование по минимуму расстояния совпадает с декодированием по максимуму правдоподобия.

Декодирование, исправляющее все ошибки-лидеры смежных классов, называется *полным*, так как оно позволяет исправить все ошибки, которые может исправить код. Это требует хранения таблицы из 2^r лидеров. Таким образом, декодирование может осуществляться перебором по всем 2^k кодовым словам, или всем 2^r синдромам. Тогда общая сложность декодирования составит

$$\chi = \min(2^k, 2^r). \tag{4.1}$$

Исправление не всех лидеров смежных классов, а только какого-то

их подмножества (например, векторов веса, не превышающего некоторой величины), может позволить для некоторых классов кодов построить процедуры декодирования, имеющие меньшую (полиномиальную) сложность, и влечет за собой увеличение вероятности ошибочного декодирования. Однако, такие декодеры требуют дополнительных ограничений на структуру кода.

Помимо процедуры декодирования, стандартная расстановка позволяет оценить вероятность ошибки декодирования $P_{\text{ош}}$. Эта вероятность равна вероятности того, что в канале произойдет ошибка, не являющаяся лидером смежного класса.

В следующем разделе мы рассмотрим общий метод декодирования, применимый к любому линейному коду, позволяющий уменьшить сложность (4.1).

4.2. Декодирование по информационным совокупностям

В этом разделе рассматриваются вопросы, связанные с методом общего декодирования линейных кодов, использующего информационные совокупности (и.с.) [8].

Ключевым при изучении многих общих методов декодирования является понятие информационной совокупности (и.с.) кода. Мы уже знаем из раздела 2.3, что для систематического кода задание k первых символов $\{\alpha_0, \ldots, \alpha_k\}$ полностью определяет слово $(\alpha_0, \ldots, \alpha_k, \alpha_{k+1}, \ldots, \alpha_n)$ (n, k)-кода. Это означает, что в коде есть только одно слово, у которого на позициях с номерами $\{0, \ldots, k-1\}$ стоят символы $\alpha_0, \ldots, \alpha_{k-1}$.

Множество номеров $0, \ldots, k-1$ не является, однако, исключительным. Для каждого кода есть еще целый ряд множеств $\{j_1, \ldots, j_k\}$ $(0 \leq j_1 < \ldots < j_k \leq n-1)$ таких, что каковы бы ни были значения $\alpha_{j_1}, \ldots, \alpha_{j_k}$, в коде имеется одно и только одно слово, у которого на позициях $\{j_1, \ldots, j_k\}$ стоят символы $\alpha_{j_1}, \ldots, \alpha_{j_k}$, т. е., задание символов на позициях $\{j_1, \ldots, j_k\}$ так же полностью определяет кодовое слово, как и задание первых его символов.

Определение 4.1. Множество номеров $\{j_1, \ldots, j_k\}$ $(0 \leq j_1 < \ldots < j_k \leq n-1)$ называется *информационной совокупностью* (и.с.) кода V, если задание компонент $\alpha_{j_1}, \ldots, \alpha_{j_k}$ однозначно определяет слово из V.

Пусть $\mathbf{G}$ – порождающая матрица кода V. Обозначим через $\mathbf{G}(\gamma)$ матрицу, составленную из столбцов матрицы $\mathbf{G}$, номера которых входят в множество γ.

Утверждение 4.1. Множество позиций $\gamma = \{j_1, \ldots, j_k\}$ является и.с. тогда и только тогда, когда матрица $\mathbf{G}(\gamma)$ является невырожденной.

Если в принятом с ошибками кодовом слове имеется хотя бы одна и.с., в символах которой (т.е. в символах с номерами из этой и.с.) ошибок нет (и.с., *свободная от ошибок*), то передававшееся слово может быть правильно восстановлено по этой и.с. Процесс декодирования при этом может быть интерпретирован как процесс поиска и.с., свободной от ошибок. Важным при организации такого декодирования становится вопрос «об остановке», т. е. выборе правила, руководствуясь которым, можно определить, что найдена свободная от ошибок и.с. В дальнейшем, если не сделано специальной оговорки, мы будем рассматривать декодирование ошибок, кратность которых не превышает t.

Опишем теперь алгоритм декодирования по и.с. Пусть $\gamma = \{j_1, \ldots, j_k\}$ – и.с. кода V, а матрицы $\mathbf{G}$ и $\mathbf{H}$ – порождающая и проверочная матрицы этого кода. Обозначим через $\mathbf{G}_\gamma$ матрицу

$$\mathbf{G}_\gamma = (\mathbf{G}(\gamma))^{-1} \cdot \mathbf{G}. \tag{4.2}$$

Ясно, что столбцы матрицы $\mathbf{G}_\gamma$ с номерами $\{j_1, \ldots, j_k\}$ составляют единичную $(k \times k)$-матрицу. Умножив вектор $(\alpha_{j_1}, \ldots, \alpha_{j_k})$ на матрицу $\mathbf{G}_\gamma$, мы получим кодовое слово, у которого на позициях $\{j_1, \ldots, j_k\}$ расположены символы $\alpha_{j_1}, \ldots, \alpha_{j_k}$:

$$(\alpha_{j_1}, \ldots, \alpha_{j_k}) \cdot \mathbf{G}_\gamma = (\alpha_{j_1}, \ldots, \alpha_{j_k}) \cdot \begin{array}{c} \begin{matrix} \quad & j_1 & \quad & j_2 & \quad & j_k & \quad \end{matrix} \\ \begin{bmatrix} \ldots & 1 & \ldots & 0 & \ldots & 0 & \ldots \\ \ldots & 0 & \ldots & 1 & \ldots & 0 & \ldots \\ \ldots & & \ldots & & \ldots & & \ldots \\ \ldots & 0 & \ldots & 0 & \ldots & 1 & \ldots \end{bmatrix} \end{array} =$$

$$= (\ldots \alpha_{j_1} \ldots \alpha_{j_2} \ldots \alpha_{j_k} \ldots).$$

Матрице $\mathbf{G}_\gamma$ поставим в соответствие проверочную матрицу

$$\mathbf{H}_\gamma = (\mathbf{H}(\bar{\gamma}))^{-1} \cdot \mathbf{H}, \tag{4.3}$$

где $\bar{\gamma} = \{1, 2, \ldots, n\} \backslash \gamma$, т.е. множество номеров позиций кодового слова, не вошедших в γ, а $\mathbf{H}(\bar{\gamma})$ – матрица, составленная из столбцов матрицы $\mathbf{H}$ с номерами из $\bar{\gamma}$. В матрице $\mathbf{H}_\gamma$ в столбцах, не вошедших в и.с. γ, стоит единичная $(r \times r)$-матрица. Поэтому, если все ненулевые компоненты некоторого вектора ошибки $\mathbf{e}$ расположены в множестве $\bar{\gamma}$ (т.е. если γ свободна от ошибок), то вес синдрома

$$\mathbf{S}_\gamma(\mathbf{e}) = \mathbf{e} \cdot \mathbf{H}_\gamma^T \tag{4.4}$$

равен весу вектора ошибки. При декодировании ошибок кратности, не превышающей $t = \lfloor (d-1)/2 \rfloor$, вес синдрома, вычисленного по и.с., свободной от ошибок, не превышает t. Следовательно, алгоритм декодирования по и.с. ошибок кратности $v \leq t$ можно сформулировать следующим образом.

Пусть $\Gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_e\}$ – множество и.с. кода, достаточное для декодирования ошибок кратности v.

1. По принятому вектору $\mathbf{b} = \mathbf{a} + \mathbf{e}$, $\mathbf{b} = (\beta_0, \beta_1, \ldots, \beta_{n-1})$, где $\mathbf{a}$ – передававшийся вектор, $\mathbf{e}$ – вектор ошибки, вычисляются синдромы $\mathbf{S}_{\gamma_i}(b)$ до тех пор, пока не найдется такая и.с. $\gamma^* = \{j_1, j_2, \ldots, j_k\}$, что вес синдрома

$$W(\mathbf{S}_{\gamma^*}(\mathbf{b})) \leq v. \tag{4.5}$$

2. При выполнении (4.5) кодовое слово

$$\hat{\mathbf{a}} = \mathbf{b}(\gamma^*) \cdot \mathbf{G}_{\gamma^*} = (\beta_{j_1}, \beta_{j_2}, \ldots, \beta_{j_k}) \cdot \mathbf{G}_{\gamma^*} \tag{4.6}$$

считается декодированным вариантом принятого слова.

3. Если ни для одного γ из Γ соотношение (4.5) не выполняется, считается, что обнаружена неисправляемая ошибка.

Пример 4.1. Рассмотрим декодирование двоичного кода $(7, 4)$ с $d = 3$. Порождающая $\mathbf{G}$ и проверочная $\mathbf{H}$ матрицы этого кода задаются как

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}; \quad \mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

Множество символов $\gamma' = \{0, 1, 2, 6\}$ является и.с. рассматриваемого кода, а множество $\gamma'' = \{0, 1, 2, 4\}$ не является и.с., так как определитель матрицы $\mathbf{G}(\gamma')$

$$|\mathbf{G}(\gamma')| = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \neq 0,$$

а определитель матрицы $\mathbf{G}(\gamma'')$

$$|\mathbf{G}(\gamma'')| = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} = 0.$$

Аналогично можно проверить, что $\gamma^{(0)} = \{0,1,2,3\}$, $\gamma^{(1)} = \{3,4,5,6\}$, $\gamma^{(2)} = \{0,1,2,6\}$ являются и.с. Для них $\mathbf{H}_{\gamma^{(0)}} = \mathbf{H}$; $\mathbf{G}_{\gamma^{(0)}} = \mathbf{G}$;

$$\mathbf{H}_{\gamma^{(1)}} = (\mathbf{H}(0,1,2))^{-1} \cdot \mathbf{H} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}^{-1} \cdot \mathbf{H} =$$

$$= \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix};$$

$$\mathbf{G}_{\gamma^{(1)}} = (\mathbf{G}(3,4,5,6))^{-1} \cdot \mathbf{G} = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix};$$

$$\mathbf{H}_{\gamma^{(2)}} = (\mathbf{H}(3,4,5))^{-1} \cdot \mathbf{H} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix};$$

$$\mathbf{G}_{\gamma^{(2)}} = (\mathbf{G}(0,1,2,6))^{-1} \cdot \mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Множество и.с. $\Gamma = \left\{\gamma^{(0)}, \gamma^{(1)}, \gamma^{(2)}\right\}$ позволяет декодировать любые однократные ошибки в коде $(7,4)$. Пусть передавалось кодовое слово $\mathbf{a} = (0000000)$, а принято слово $\mathbf{b} = (0001000)$. Продекодируем слово $\mathbf{b}$ в соответствии с изложенным выше алгоритмом:

$$\mathbf{S}_{\gamma^{(0)}}(\mathbf{b}) = \mathbf{b} \cdot \mathbf{H}^T_{\gamma^{(0)}} = (011) \Rightarrow W(\mathbf{S}_{\gamma^{(0)}}(\mathbf{b})) = 2 > 1 = \frac{d-1}{2};$$

$$\mathbf{S}_{\gamma^{(1)}}(\mathbf{b}) = \mathbf{b} \cdot \mathbf{H}^T_{\gamma^{(1)}} = (101) \Rightarrow W(\mathbf{S}_{\gamma^{(1)}}(\mathbf{b})) = 2 > 1 = \frac{d-1}{2};$$

$$\mathbf{S}_{\gamma^{(2)}}(\mathbf{b}) = \mathbf{b} \cdot \mathbf{H}^T_{\gamma^{(2)}} = (100) \Rightarrow W(\mathbf{S}_{\gamma^{(2)}}(\mathbf{b})) = 1 = \frac{d-1}{2}.$$

По и.с. $\gamma^{(2)}$ вычисляем декодированный вариант принятого слова

$$\hat{\mathbf{a}} = (\beta_0, \beta_1, \beta_2, \beta_6) \cdot \mathbf{G}_{\gamma^{(2)}} =$$

$$= (0000) \cdot \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} = (0000000) = \mathbf{a}.$$

В общем случае множество Γ – и.с., необходимых для декодирования, быстро растет при увеличении длины кода и числа ошибок, которые он может исправлять. А так как для каждой γ из Γ описанный алгоритм требует хранения или вычисления матриц $\mathbf{G}_\gamma$ и (или) $\mathbf{H}_\gamma$, то растет и сложность алгоритма декодирования. Упрощение реализации декодирования по и.с. связано с использованием двух его модификаций: перестановочного декодирования и декодирования с использованием покрывающих полиномов.

4.3. Перестановочное декодирование и декодирование с помощью покрывающих полиномов

Напомним, что перестановка π – это взаимно однозначное отображение множества $\{1, 2, \ldots, n\}$ на себя. Произвольная перестановка переводит слово кода V в некоторое другое, вообще говоря, не кодовое слово. Однако могут быть и такие перестановки, которые любое кодовое слово переводят также в кодовое слово. Такую перестановку называют *сохраняющей код*, а код – *инвариантным* относительно данной перестановки.

Пример 4.2. Рассмотрим двоичный линейный $(3, 2)$-код V, состоящий из 4 слов, (000), (110), (100), (010). Перестановка местами первого и второго символов любого кодового слова переводит его в кодовое слово, перестановка же второго и третьего символов переводит слово (110) в слово (101), которого в коде нет.

Пусть $\mathbf{G}$ – порождающая, а $\mathbf{H}$ – проверочная матрица кода V:

$$\mathbf{G} = \begin{bmatrix} g_1 \\ \vdots \\ g_k \end{bmatrix}; \quad \mathbf{H} = \begin{bmatrix} h_1 \\ \vdots \\ h_r \end{bmatrix}.$$

Утверждение 4.2. Для того, чтобы некоторая перестановка π сохраняла код V, необходимо и достаточно, чтобы матрица

$$\pi(\mathbf{G}) = \begin{bmatrix} \pi(g_1) \\ \vdots \\ \pi(g_k) \end{bmatrix}$$

удовлетворяла условию $\pi(\mathbf{G}) \cdot \mathbf{H}^T = \mathbf{0}$.

Пусть π – перестановка, сохраняющая код V. Тогда для любого вектора $\mathbf{b} = \mathbf{a} + \mathbf{e}$, $\mathbf{a} \in V$, $\pi(\mathbf{b}) = \pi(\mathbf{a}) + \pi(\mathbf{e}) = \mathbf{a}' + \mathbf{e}'$, где $\mathbf{a}'$ – какое-то

кодовое слово; а $\mathbf{e}'$ – вектор ошибок, вес которого равен весу вектора $\mathbf{e}$. Если вес вектора $\mathbf{e}$ не превышает t, то и вес вектора $\mathbf{e}'$ не превышает t.

С помощью перестановок, сохраняющих код, декодирование по и.с. может быть ограничено следующим образом. Пусть $\mathbf{G}$ и $\mathbf{H}$ – порождающая и проверочная матрицы кода V в систематическом виде, т.е. соответствуют и.с. $\gamma_0 = \{0, 1, \ldots, k-1\}$. Пусть, кроме того, $\{\pi_1, \ldots, \pi_e\}$ – множество перестановок, сохраняющих код. Будем вычислять синдромы векторов $\pi_i(\mathbf{b})$ по и.с. γ_0:

$$\mathbf{S}(\pi_i(\mathbf{b})) = (\pi_i(\mathbf{b})) \cdot \mathbf{H}^T, \tag{4.7}$$

и вычислять вес синдромов (4.7). Если какая-нибудь перестановка π переведет вектор $\mathbf{e}$ в вектор $\mathbf{e}'$, у которого все ненулевые компоненты расположены на позициях $\{k, \ldots, n-1\}$, то и.с. γ_0 окажется свободной от ошибок, а вес соответствующего синдрома при этом будет не более, чем t. Тогда для восстановления передававшегося слова достаточно будет применить перестановку π^{-1}, обратную к π, к вектору $\mathbf{a}'$.

Пример 4.3. Рассмотрим перестановочное декодирование $(7, 4)$-кода с $d = 3$, заданного в примере 4.1. Данный код инвариантен относительно циклической перестановки T ($T(i) = i + 1 \bmod 7$). Действительно,

$$T(\mathbf{G}) = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix},$$

и выполняется условие $T(\mathbf{G}) \cdot \mathbf{H}^T = \mathbf{0}$. Проведем декодирование слова $\mathbf{b} = \mathbf{a} + \mathbf{e} = (1011000) + (0100000) = (1111000)$. Будем вычислять $\mathbf{S}_{\gamma_0}(T^i(\mathbf{b}))$ для $i = 0, 1, \ldots, 6$, поскольку $T^7 = E = T^0$. Имеем

$$\mathbf{S}_{\gamma_0}(\mathbf{b}) = \mathbf{b} \cdot \mathbf{H}^T = (111) \Rightarrow W(\mathbf{S}_{\gamma_0}(\mathbf{b})) = 3 > 1 = \frac{d-1}{2};$$

$$\mathbf{S}_{\gamma_0}(T(\mathbf{b})) = T(\mathbf{b}) \cdot \mathbf{H}^T = (110) \Rightarrow W(\mathbf{S}_{\gamma_0}(T(\mathbf{b}))) = 2 > 1 = \frac{d-1}{2};$$

$$\mathbf{S}_{\gamma_0}(T^2(\mathbf{b})) = T^2(\mathbf{b}) \cdot \mathbf{H}^T = (011) \Rightarrow W(\mathbf{S}_{\gamma_0}(T^2(\mathbf{b}))) = 2 > 1 = \frac{d-1}{2};$$

$$\mathbf{S}_{\gamma_0}(T^3(\mathbf{b})) = T^3(\mathbf{b}) \cdot \mathbf{H}^T = (100) \Rightarrow W(\mathbf{S}_{\gamma_0}(T^3(\mathbf{b}))) = 1 = \frac{d-1}{2}.$$

Условие (4.5) выполнено. По и.с. γ_0 вычисляем слово $T^3(\hat{\mathbf{a}})$ ($\hat{\mathbf{a}}$ – декодированный вариант принятого слова):

$$\mathbf{a}' = T^3(\hat{\mathbf{a}}) = (0001) \cdot \mathbf{G} = (0001011)$$

и наконец

$$\hat{\mathbf{a}} = T^{-3}(\mathbf{a}') = (1011000) = \mathbf{a}.$$

Другим способом «очистки» и.с. от ошибок является попытка перебором «накрыть» ошибки на и.с. Этот метод получил название декодирования с помощью покрывающих полиномов[1].

Пусть $\mathbf{\Theta}$ – вектор (покрывающий полином), совпадающий с вектором ошибки $\mathbf{e}$ на позициях и.с. γ, и имеющий нули на всех остальных позициях. Тогда для вектора $(\mathbf{e} - \mathbf{\Theta})$ и.с. γ является свободной от ошибок, а соответствующий синдром $\mathbf{S}_\gamma(\mathbf{b} - \mathbf{\Theta}) = (\mathbf{e} - \mathbf{\Theta}) \cdot \mathbf{H}^T$ имеет вес

$$W(\mathbf{S}_\gamma(\mathbf{b} - \mathbf{\Theta})) \le t - W(\mathbf{\Theta}). \tag{4.8}$$

Перебирая по векторам $\mathbf{\Theta}$ в порядке возрастания их веса (начиная с $\mathbf{\Theta}_0 = (0 \dots 0)$) до тех пор, пока для какого-нибудь $\mathbf{\Theta}^*$ не выполнится условие (4.8), мы по вектору $\mathbf{b}^* = (\mathbf{b} - \mathbf{\Theta}^*)$ и и.с. γ восстановим передававшийся вектор (если, конечно, вес вектора ошибки не превосходит t).

Пример 4.4. Применим декодирование с помощью покрывающих полиномов по и.с. γ_0 в ситуации, рассмотренной в примере 4.1. Множество покрывающих полиномов выберем в виде $\mathbf{\Theta}_0 = (0000000)$; $\mathbf{\Theta}_1 = (1000000)$; $\mathbf{\Theta}_2 = (0100000)$; $\mathbf{\Theta}_3 = (0010000)$; $\mathbf{\Theta}_4 = (0001000)$. Принятый вектор $\mathbf{b} = (0001000) = \mathbf{a} + \mathbf{e} = (0000000) + (0001000)$. Будем последовательно вычислять синдромы

$$\begin{aligned}
&\mathbf{S}_{\gamma_0}(\mathbf{b} - \mathbf{\Theta}_0) = (0001000) \cdot \mathbf{H}^T = (011) \Rightarrow \\
&\qquad \Rightarrow W(\mathbf{S}_{\gamma_0}(\mathbf{b} - \mathbf{\Theta}_0)) = 2 > 1 = (d-1)/2; \\
&\mathbf{S}_{\gamma_0}(\mathbf{b} - \mathbf{\Theta}_1) = (1001000) \cdot \mathbf{H}^T = (110) \Rightarrow \\
&\qquad \Rightarrow W(\mathbf{S}_{\gamma_0}(\mathbf{b} - \mathbf{\Theta}_1)) = 2 > 0 = (d-1)/2 - W(\mathbf{\Theta}_1); \\
&\mathbf{S}_{\gamma_0}(\mathbf{b} - \mathbf{\Theta}_2) = (0101000) \cdot \mathbf{H}^T = (100) \Rightarrow \\
&\qquad \Rightarrow W(\mathbf{S}_{\gamma_0}(\mathbf{b} - \mathbf{\Theta}_2)) = 1 > 0 = \frac{d-1}{2} - W(\mathbf{\Theta}_2); \\
&\mathbf{S}_{\gamma_0}(\mathbf{b} - \mathbf{\Theta}_3) = (0011000) \cdot \mathbf{H}^T = (101) \Rightarrow \\
&\qquad \Rightarrow W(\mathbf{S}_{\gamma_0}(\mathbf{b} - \mathbf{\Theta}_3)) = 2 > 0 = \frac{d-1}{2} - W(\mathbf{\Theta}_3); \\
&\mathbf{S}_{\gamma_0}(\mathbf{b} - \mathbf{\Theta}_4) = (0000000) \cdot \mathbf{H}^T = (000) \Rightarrow \\
&\qquad \Rightarrow W(\mathbf{S}_{\gamma_0}(\mathbf{b} - \mathbf{\Theta}_4)) = 0 = \frac{d-1}{2} - W(\mathbf{\Theta}_4).
\end{aligned}$$

[1] Термин «покрывающий полином», традиционно используемый в теории кодирования, не совсем точен, более точным был бы термин «покрывающий вектор» или «покрывающее слово».

Тогда $\mathbf{\Theta}^* = \mathbf{\Theta}_4$; $\mathbf{b}^* = \mathbf{b} - \mathbf{\Theta}^* = (0000000)$, и по и.с. γ_0 и вектору $\mathbf{b}^*$ мы восстанавливаем переданное слово:

$$\hat{\mathbf{a}} = (\beta_0^*, \ldots, \beta_{k-1}^*) \cdot \mathbf{G}_{\gamma_0} = (0000) \cdot \mathbf{G}_{\gamma_0} = (0000000) = \mathbf{a}.$$

Лучшие практические результаты рассмотренные выше алгоритмы дают при их совместном использовании.

Пусть $\Gamma = \{\gamma_0, \ldots, \gamma_{m-1}\}$ – некоторое множество из m и.с. кода V и $\Theta^{(0)}, \ldots, \Theta^{(m-1)}$ – множества покрывающих полиномов $\Theta^{(0)} = \{\mathbf{\Theta}_{00}, \ldots, \mathbf{\Theta}_{0l_0}\}, \ldots, \Theta^{(m-1)} = \{\mathbf{\Theta}_{m-10}, \ldots, \mathbf{\Theta}_{m-1l_{m-1}}\}$ для и.с. $\gamma_0, \ldots, \gamma_{m-1}$, соответственно, причем $\mathbf{\Theta}_{j0} = \mathbf{0}$, и $W(\mathbf{\Theta}_{j0}) < W(\mathbf{\Theta}_{j1}) \leq \ldots \leq W(\mathbf{\Theta}_{jl_j})$ для $j = 0, \ldots, m-1$. Алгоритм декодирования, основанный на совместном использовании и.с. и покрывающих полиномов, состоит в следующем:

1. По принятому вектору $\mathbf{b}$ для каждой пары γ_i, $\mathbf{\Theta}_{ij}$ ($i = \overline{0, m-1}$, $j = \overline{0, l_i}$) вычисляется вектор $\tilde{\mathbf{b}}_{ij} = \mathbf{b} - \mathbf{\Theta}_{ij}$ и синдром $\mathbf{S}_{\gamma_i}(\tilde{\mathbf{b}}_{ij}) = \tilde{\mathbf{b}}_{ij} \cdot \mathbf{H}_{\gamma_i}^T$ до тех пор, пока не найдется такая пара чисел i^* и j^*, для которой выполняется условие

$$W(\mathbf{S}_{i^*j^*}(\mathbf{b})) \leq t - W(\mathbf{\Theta}_{i^*j^*}). \tag{4.9}$$

2. При выполнении условия (4.9) вычисляется декодированный вариант принятого слова:

$$\hat{\mathbf{a}} = \tilde{\mathbf{b}}(\gamma_{i^*}) \cdot \mathbf{G}_{\gamma_{i^*}},$$

где $\tilde{\mathbf{b}}(\gamma_{i^*})$ – подвектор вектора $\tilde{\mathbf{b}}$, составленный из элементов последнего с координатами, входящими в и.с. γ_{i^*}.

3. Если соотношение (4.9) не выполнится ни для одной пары γ_i и $\mathbf{\Theta}_{ij}$, то считается, что произошла неисправимая ошибка.

Совокупность множества Γ и множества покрывающих полиномов $\Theta = \left\{\Theta^{(0)}, \ldots, \Theta^{(m-1)}\right\}$, используемых в описанной процедуре, мы будем называть *декодирующей совокупностью* (д.с.) и обозначать через ДС $= \{\Gamma, \Theta\}$. В дальнейшем будем использовать для декодирования по и.с. $\boldsymbol{\gamma}$ все полиномы веса до v, т.е. векторы, все ненулевые элементы которых расположены на множестве $\boldsymbol{\gamma}$, и их число не превышает v. Такие полиномы мы будем обозначать $\Theta_{\boldsymbol{\gamma}}(v)$.

Пример 4.5. Для $(7, 4)$-кода рассмотрим $\boldsymbol{\gamma} = \{1, 2, 3, 4\}$. Множество полиномов $\Theta_{\boldsymbol{\gamma}}(1)$ состоит из пяти полиномов: $\mathbf{\Theta}_{\boldsymbol{\gamma}0} = (0000000)$; $\mathbf{\Theta}_{\boldsymbol{\gamma}1} = (0001000)$; $\mathbf{\Theta}_{\boldsymbol{\gamma}2} = (0000100)$; $\mathbf{\Theta}_{\boldsymbol{\gamma}3} = (0000010)$; $\mathbf{\Theta}_{\boldsymbol{\gamma}4} = (0000001)$.

При использовании множества полиномов веса до v по всем и.с. из Γ мы не будем писать значок и.с. при обозначении множества покрывающих полиномов, т. е. писать ДС$= \{\Gamma, \Theta(v))\}$.

Нетрудно видеть, что если γ является и.с. кода V, а π – перестановка, сохраняющая этот код, то множество номеров $\pi(\gamma)$ также является и.с. кода V.

Пример 4.6. Рассмотрим декодирование $(15, 5)$-кода с порождающей матрицей (4.10) и проверочной матрицей (4.11), с расстоянием $d = 7$:

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}; \qquad (4.10)$$

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \qquad (4.11)$$

Для декодирования этого кода может быть использована декодирующая совокупность ДС$=\{\Gamma, \Theta(1)\}$; $\Gamma = \{\gamma_0, \gamma_1 = T^5(\gamma_0)\}$, где $\gamma = \{0, 1, 2, 3, 4\}$, а $T^5(\gamma_0) = \{5, 6, 7, 8, 9\}$ – циклический сдвиг и.с. γ_0 на 5 позиций. Пусть слово $\mathbf{b} = \mathbf{a} + \mathbf{e} = (001110110010100) + (010100010000000)$, т.е. в принятом слове ошибки произошли на первой, третьей и седьмой позициях. Будем последовательно вычислять $\mathbf{S}_{0j}(\mathbf{b})$ по и.с. γ_0:

$\tilde{\mathbf{b}}_{00} = \mathbf{b} + \mathbf{\Theta}_{00} = \mathbf{b}$;

$\mathbf{S}_{00}(\mathbf{b}) = \tilde{\mathbf{b}}_{00} \cdot \mathbf{H}^T_{\gamma_0} = (0011110110) \Rightarrow W(\mathbf{S}_{00}) > 3 = \dfrac{d-1}{2} - W(\mathbf{\Theta}_{00})$;

$\tilde{\mathbf{b}}_{01} = \mathbf{b} + \mathbf{\Theta}_{01} = \mathbf{b} + (100000000000000)$;

$\mathbf{S}_{01}(\mathbf{b}) = \tilde{\mathbf{b}}_{01} \cdot \mathbf{H}^T_{\gamma_0} = (1101000100) \Rightarrow W(\mathbf{S}_{01}) > 2 = \dfrac{d-1}{2} - W(\mathbf{\Theta}_{01})$;

$\ldots$

$\tilde{\mathbf{b}}_{05} = \mathbf{b} + \mathbf{\Theta}_{05} = \mathbf{b} + (000010000000000)$;

$S_{05}(\mathbf{b}) = \tilde{\mathbf{b}}_{05} \cdot \mathbf{H}^T_{\gamma_0} = (1110010011) \Rightarrow W(\mathbf{S}_{05}) > 2 = \dfrac{d-1}{2} - W(\mathbf{\Theta}_{05})$.

Перейдем теперь к и.с. $\gamma_1 = T^5(\gamma_0)$, ей соответствуют $\mathbf{G}_{\gamma_1} = T^5(\mathbf{G}_{\gamma_0})$ и $\mathbf{H}_{\gamma_1} = T^5(\mathbf{H}_{\gamma_0})$:

$$\mathbf{G}_{\gamma_1} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \end{bmatrix};$$

$$\mathbf{H}_{\gamma_1} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Вычислим $\mathbf{S}_{1j}(\mathbf{b})$ по и.с. γ_1:

$\tilde{\mathbf{b}}_{10} = \mathbf{b} + \mathbf{\Theta}_{10} = b + (000000000000000)$;

$\mathbf{S}_{10}(\mathbf{b}) = \tilde{\mathbf{b}}_{10} \cdot \mathbf{H}^T_{\gamma_1} = (1101010100) \Rightarrow W(\mathbf{S}_{10}) = 5 > 3 = \dfrac{d-1}{2} - W(\mathbf{\Theta}_{10})$;

$\tilde{\mathbf{b}}_{11} = \mathbf{b} + \mathbf{\Theta}_{11} = \mathbf{b} + (000010000000000)$;

$\mathbf{S}_{11}(\mathbf{b}) = \tilde{\mathbf{b}}_{11} \cdot \mathbf{H}^T_{\gamma_1} = (0011100110) \Rightarrow W(\mathbf{S}_{11}) = 5 > 2 = \dfrac{d-1}{2} - W(\mathbf{\Theta}_{11})$;

$\ldots$

$\tilde{\mathbf{b}}_{13} = \mathbf{b} + \mathbf{\Theta}_{13} = \mathbf{b} + (000000010000000) = (011010110010100)$;

$\mathbf{S}_{13}(\mathbf{b}) = \tilde{\mathbf{b}}_{13} \cdot \mathbf{H}^T_{\gamma_1} = (0000001010) \Rightarrow W(\mathbf{S}_{13}) = 2 = \dfrac{d-1}{2} - W(\mathbf{\Theta}_{13})$.

По слову $\tilde{\mathbf{b}}_{13}$ и и.с. γ_1 восстанавливается слово $\hat{\mathbf{a}}$:

$$\hat{\mathbf{a}} = \tilde{\mathbf{b}}_{13}(\gamma_1) \cdot \mathbf{G}_{\gamma_1} = (001110110010100) = \mathbf{a}.$$

В рассматриваемом примере декодирующую совокупность можно реализовать, используя не вторую и.с. γ_1, а производя перестановку T^5 принятого вектора. При этом алгоритм декодирования сводится к последовательному вычислению $\mathbf{S}_{0j}(\mathbf{b})$, а затем к вычислению $\mathbf{S}_{0j}(T^5(\mathbf{b}))$.

Сложность декодирования (n, k)-кода с использованием декодирующих совокупностей при $n \to \infty$ можно оценить как

$$\chi(n, k) \approx 2^{n((1-R)R+O(1))},$$

где $O(1)$ – величина, убывающая с ростом n; R – скорость кода.

4.4. Лемма Евсеева

В заключение раздела, посвященного общим методам декодирования линейных блоковых кодов, приведем лемму, связывающую расстояние Варшамова–Гилберта $d_{\text{ВГ}}$, полученное из решения (2.9) относительно d_0, и вероятность ошибочного декодирования P_0 при осуществлении декодирования по всем лидерам смежного класса.

Лемма 4.3. *Пусть E – множество из всех 2^n возможных векторов ошибки аддитивного канала, E_V – множество из 2^r самых вероятных векторов ошибки, и E_r – множество из 2^r лидеров смежного класса для некоторого линейного (n, k)-кода. Если обозначить через $P_0 = \Pr(E \backslash E_r)$ вероятность ошибки при декодировании по всем лидерам, то*

$$\Pr(E \backslash (E_r \cap E_V)) \leq 2P_0.$$

Доказательство. Разберем условие леммы. Предположим, что используется декодирование по стандартной расстановке, где в каждом смежном классе в качестве лидера выбирается наиболее вероятный вектор канала связи. В этом случае декодирование является декодированием по максимуму правдоподобия. Вероятность ошибки при таком декодировании – это вероятность того, что вектор произошедшей ошибки не входит в множество лидеров, это и есть величина P_0 из условия леммы. При этом заметим, что даже при таком выборе лидеров смежных классов множество E_V наиболее вероятных векторов и множество E_r лидеров могут не совпадать, так как возможна ситуация, при которой наиболее вероятные векторы окажутся в одном смежном классе.

Далее, пересечение множеств $(E_r \cap E_V)$ – это те высоковероятные векторы из E_V, которые вошли в множество лидеров E_r, а $\Pr(E \backslash (E_r \cap E_V))$ – вероятность того, что произошла ошибка, не вошедшая в это пересечение. Таким образом, утверждается, что вероятность этого события не более чем в два раза превосходит вероятность того, что ошибка не является лидером смежного класса.

Заметим (рис. 4.2), что

$$E \backslash (E_r \cap E_V) = (E \backslash E_r) \cup (E_r \backslash (E_r \cap E_V)).$$

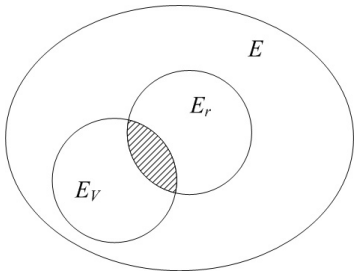

Рис. 4.2. К доказательству леммы Евсеева

Запишем вероятность этого события:

$$\Pr(E\backslash(E_r \cap E_V)) = \Pr(E\backslash E_r) + \Pr(E_r\backslash(E_r \cap E_V)).$$

Так как $|E_V| = |E_r| = 2^r$ и E_V – множество самых вероятных векторов, то вероятность получить вектор ошибки, являющийся лидером смежного класса, не может превышать вероятности получить один из 2^r самых вероятных векторов $\Pr(E_r) \leq \Pr(E_V)$ и, следовательно,

$$\Pr(E_r\backslash(E_r \cap E_V)) \leq \Pr(E_V\backslash(E_r \cap E_V)) \leq \Pr(E\backslash E_r).$$

Отсюда

$$\Pr(E\backslash(E_r \cap E_V)) \leq P_0 + \Pr(E_V\backslash(E_r \cap E_V) \leq 2P_0,$$

что и требовалось доказать.

Заметим, что при доказательстве леммы нигде не использовался ДСК, поэтому полученный результат справедлив для любого аддитивного канала.

Смысл доказанной леммы состоит в том, что если при декодировании исправлять не всех лидеров смежных классов, а только тех, которые являются еще и наиболее вероятными векторами канала, то вероятность ошибки возрастет не более чем в 2 раза. При этом, так как P_0 стремится к нулю, ухудшение в 2 раза можно считать незначительным.

Но из границы Варшамова–Гилберта мы знаем, что количество векторов, имеющих вес $d_{\text{ВГ}}$, не превышает 2^r. Предполагая, что канал связи согласован с метрикой Хэмминга, т. е. что более вероятны векторы ошибок, имеющие меньший вес (это справедливо, например, для ДСК при $p < 1/2$), получим, что выбор 2^r самых вероятных векторов канала соответствует векторам, вес которых не превышает $d_{\text{ВГ}}$.

Таким образом, если исправлять только тех лидеров, вес которых не превышает $d_{\text{ВГ}}$, такое декодирование практически совпадет с полным декодированием по максимуму правдоподобия.

4.5. Задачи

1. Для заданной порождающей матрицы кода и модели ДСК с вероятностью ошибки p:

– выбрать более простой способ декодирования (по минимуму расстояния или по стандартной расстановке), выписать соответствующую таблицу;

– вычислить вероятность необнаружения ошибки;

– вычислить вероятность ошибки декодирования при полном декодировании;

– вычислить вероятность ошибки декодирования при декодировании ошибок веса до t;

– вычислить вероятность ошибки декодирования при декодировании ошибок веса до $d_{\text{ВГ}}$.

1.1.
$$\mathbf{G} = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}.$$

1.2.
$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

1.3.
$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \end{pmatrix}.$$

1.4.

$$\mathbf{G} = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

1.5.

$$\mathbf{G} = \begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

1.6.

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}.$$

1.7.

$$\mathbf{G} = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

1.8.

$$\mathbf{G} = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

1.9.

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

1.10.

$$\mathbf{G} = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix}.$$

1.11.

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}.$$

1.12.

$$\mathbf{G} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}.$$

1.13.

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}.$$

1.14.

$$\mathbf{G} = \begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

1.15.

$$\mathbf{G} = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}.$$

1.16.

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}.$$

1.17.

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

1.18.

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}.$$

2. Дан код над полем GF(11) с примитивным элементом $\alpha = 2$, с порождающей и проверочной матрицами:

$$\mathbf{G} = \begin{pmatrix} 1 & 8 & 5 & 3 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 8 & 5 & 3 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 8 & 5 & 3 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 8 & 5 & 3 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 8 & 5 & 3 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 8 & 5 & 3 & 1 \end{pmatrix};$$

$$\mathbf{H} = \begin{pmatrix} 1 & 2 & 4 & 8 & 5 & 10 & 9 & 7 & 3 & 6 \\ 1 & 4 & 5 & 9 & 3 & 1 & 4 & 5 & 9 & 3 \\ 1 & 8 & 9 & 6 & 4 & 10 & 3 & 2 & 5 & 7 \\ 1 & 5 & 3 & 4 & 9 & 1 & 5 & 3 & 4 & 9 \end{pmatrix}.$$

Произвести декодирование полученного слова $\mathbf{b}$ при помощи декодирования по информационным совокупностям:

2.1. $\mathbf{b} = (1, 2, 3, 8, 5, 6, 9, 0, 6, 3)$.
2.2. $\mathbf{b} = (4, 8, 6, 8, 3, 9, 4, 5, 1, 0)$.
2.3. $\mathbf{b} = (3, 8, 9, 1, 5, 10, 6, 0, 3, 10)$.
2.4. $\mathbf{b} = (5, 4, 10, 10, 10, 2, 7, 5, 10, 2)$.
2.5. $\mathbf{b} = (5, 3, 5, 3, 2, 4, 7, 0, 10, 1)$.
2.6. $\mathbf{b} = (0, 1, 9, 10, 9, 5, 6, 2, 10, 9)$.
2.7. $\mathbf{b} = (5, 9, 6, 10, 2, 3, 0, 1, 0, 5)$.
2.8. $\mathbf{b} = (5, 9, 5, 6, 4, 9, 6, 1, 10, 5)$.
2.9. $\mathbf{b} = (1, 8, 9, 1, 4, 2, 6, 1, 7, 8)$.
2.10. $\mathbf{b} = (0, 8, 2, 10, 8, 9, 1, 8, 5, 8)$.
2.11. $\mathbf{b} = (3, 8, 1, 8, 5, 8, 6, 8, 1, 1)$.
2.12. $\mathbf{b} = (9, 1, 6, 5, 1, 6, 2, 2, 2, 2)$.
2.13. $\mathbf{b} = (7, 4, 5, 7, 1, 8, 1, 0, 10, 8)$.
2.14. $\mathbf{b} = (1, 0, 5, 8, 2, 5, 8, 0, 4, 9)$.
2.15. $\mathbf{b} = (2, 5, 8, 9, 5, 2, 2, 8, 1, 9)$.

5. Декодирование циклических кодов

5.1. Алгоритм Питерсона–Горенстейна–Цирлера

БЧХ-коды, и особенно РС-коды, находят широкое применение на практике. Это объясняется не только тем, что эти коды обладают хорошими дистанционными и скоростными характеристиками, но и тем, что для их декодирования разработаны эффективные алгебраические процедуры.

Пусть $\mathcal{G}$ – циклический (n, k)-код БЧХ над полем $GF(q)$ с расстоянием $d = 2t + 1$, и $g(x)$ – порождающий многочлен этого кода. Пусть, кроме того, η, η^2, …, η^{2t} – корни многочлена $g(x)$. Тогда проверочная матрица кода $\mathcal{G}$ может быть записана в виде

$$\mathbf{H} = \begin{bmatrix} 1 & \eta & \eta^2 & \dots & \eta^{n-1} \\ 1 & \eta^2 & (\eta^2)^2 & \dots & (\eta^2)^{n-1} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & \eta^{2t} & (\eta^{2t})^2 & \dots & (\eta^{2t})^{n-1} \end{bmatrix}. \tag{5.1}$$

Пусть $b(x)$ – принятое слово, равное сумме кодового слова $a(x)$ и ошибки $e(x)$, а именно: $b(x) = a(x) + e(x)$, причем многочлен ошибки

$$e(x) = \varepsilon_0 + \varepsilon_1 x + \ldots + \varepsilon_{n-1} x^{n-1} \tag{5.2}$$

содержит ровно $v \le t$ ненулевых коэффициентов ε_{i_1}, …, ε_{i_v}. Умножая $b(x)$ на $\mathbf{H}$, получим *вектор синдрома* с компонентами

$$S_j = b(\eta^j), \quad j = 1, \ldots, 2t. \tag{5.3}$$

Поскольку $a(\eta^j) = 0$ для любого кодового слова, равенство (5.3) перепишется в виде

$$S_j = \varepsilon_{i_1}(\eta^j)^{i_1} + \varepsilon_{i_2}(\eta^j)^{i_2} + \ldots + \varepsilon_{i_v}(\eta^j)^{i_v}. \tag{5.4}$$

Многочлен ошибки (5.2) полностью описывается множеством пар $\{\varepsilon_{i_1}, i_1\}$, …, $\{\varepsilon_{i_v}, i_v\}$. Обозначим ε_{i_s} через Y_s, а η^{i_s} через X_s, тогда

$$S_j = X_1^j Y_1 + \ldots + X_v^j Y_v, \quad j = 1, \ldots, 2t. \tag{5.5}$$

Величины Y_s мы будем называть *значениями* ошибок, а величины X_s – *локаторами* ошибок.

Величины S_j непосредственно могут быть вычислены по принятому вектору, следовательно, равенство (5.5) можно рассматривать как

систему из $2t$ нелинейных уравнений относительно $2v$ неизвестных X_1, X_2, …, X_v, Y_1, …, Y_v. Можно показать, что система (5.5) имеет единственное решение, которое и надо найти, чтобы продекодировать принятое слово.

Однако систему нелинейных уравнений трудно решать непосредственно. Для того, чтобы обойти эти трудности, введем в рассмотрение многочлен $\sigma(x)$:

$$\sigma(x) = 1 + \sigma_1 x + \ldots + \sigma_v x^v. \tag{5.6}$$

Это – многочлен минимальной степени, такой, что его корнями являются X_1^{-1}, …, X_v^{-1} – величины, обратные локаторам ошибок:

$$\sigma(x) = (1 - xX_1)(1 - xX_2) \cdot \ldots \cdot (1 - xX_v). \tag{5.7}$$

Многочлен $\sigma(x)$ называют *многочленом локаторов ошибок*.

Если коэффициенты этого многочлена σ_1, …, σ_v известны, то для вычисления локаторов нужно найти его корни. Поэтому задача определения локаторов ошибок может теперь решаться в два этапа: находятся коэффициенты $\sigma(x)$, а затем вычисляются его корни.

Для того, чтобы указать способ определения σ_1, …, σ_v, нужно указать связь этих коэффициентов (неизвестных величин) с известными компонентами синдрома S_j, $j = 1, \ldots, 2t$.

Подставляя в многочлен локаторов ошибок элементы X_1^{-1}, …, X_v^{-1}, получим систему уравнений

$$\begin{aligned} 1 + \sigma_1(X_i^{-1}) + \sigma_2(X_i^{-1})^2 + \ldots + \sigma_v(X_i^{-1})^v = 0, \\ i = 1, \ldots, v. \end{aligned} \tag{5.8}$$

Помножим левую и правую части равенства (5.8) на $Y_i X_i^{v+1}$:

$$\begin{cases} Y_1 X_1^{v+1} + \sigma_1 Y_1 X_1^v + \ldots + \sigma_v Y_1 X_1 = 0; \\ \ldots \\ Y_v X_v^{v+1} + \sigma_1 Y_v X_v^v + \ldots + \sigma_v Y_v X_v = 0. \end{cases} \tag{5.9}$$

Складывая левые и правые части равенства (5.9), получим

$$\begin{aligned} (Y_1 X_1^{v+1} + \ldots + Y_v X_v^{v+1}) + \sigma_1(Y_1 X_1^v + \ldots + Y_v X_v^v) + \\ + \ldots + \sigma_v(Y_1 X_1 + \ldots + Y_v X_v) = 0, \end{aligned}$$

откуда с учетом (5.5) имеем

$$S_{v+1} + \sigma_1 S_v + \ldots + \sigma_v S_1 = 0. \tag{5.10}$$

Домножая равенства (5.8) на $Y_iX_i^{v+2}$, $Y_iX_i^{v+3}$ и т.д., и действуя аналогично тому, как мы действовали при получении (5.10), будем иметь систему уравнений, задающую искомую связь между компонентами синдрома и коэффициентами многочлена локаторов ошибки:

$$\begin{cases} S_1\sigma_v + S_2\sigma_{v-1} + \ldots + S_v\sigma_1 = -S_{v+1}; \\ S_2\sigma_v + S_3\sigma_{v-1} + \ldots + S_{v+1}\sigma_1 = -S_{v+2}; \\ \ldots \\ S_v\sigma_v + S_{v+1}\sigma_{v-1} + \ldots + S_{2v-1}\sigma_1 = -S_{2v}. \end{cases} \tag{5.11}$$

Система уравнений (5.11) является, в отличие от системы (5.5), линейной, и для ее решения имеются регулярные методы.

Однако для того, чтобы составить эту систему, нужно знать величину v. Следующая теорема позволяет определить v.

Теорема 5.1. *Для того, чтобы определитель матрицы* $\mathbf{M}_\mu$ –

$$\mathbf{M}_\mu = \begin{bmatrix} S_1 & S_2 & \ldots & S_\mu \\ S_2 & S_3 & \ldots & S_{\mu+1} \\ \ldots & \ldots & \ldots & \ldots \\ S_\mu & S_{\mu+1} & \ldots & S_{2\mu-1} \end{bmatrix}$$

был отличен от нуля, $|\mathbf{M}_\mu| \neq 0$*, необходимо, чтобы* μ *было равно числу произошедших ошибок,* $\mu = v$*. При* $\mu > v$*,* $|\mathbf{M}_\mu| = 0$*.*

Доказательство. Пусть $X_j = 0$ для $j > v$ и пусть

$$\mathbf{W}_\mu = \begin{bmatrix} 1 & 1 & \ldots & 1 \\ X_1 & X_2 & \ldots & X_\mu \\ \ldots & \ldots & \ldots & \ldots \\ X_1^{\mu-1} & X_2^{\mu-1} & \ldots & X_\mu^{\mu-1} \end{bmatrix} -$$

матрица Вандермонда. Обозначим через $\mathbf{D}_\mu$ диагональную матрицу

$$\mathbf{D}_\mu = \begin{bmatrix} Y_1X_1 & 0 & \ldots & 0 \\ 0 & Y_2X_2 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & Y_\mu X_\mu \end{bmatrix}.$$

Непосредственной проверкой легко убедиться, что с учетом равенств (5.5)

$$\mathbf{M}_\mu = \mathbf{W}_\mu \cdot \mathbf{D}_\mu \cdot \mathbf{W}_\mu^T,$$

где $\mathbf{W}_\mu^T$ – транспонированная матрица Вандермонда. Тогда определитель матрицы $\mathbf{M}_\mu$

$$|\mathbf{M}_\mu| = |\mathbf{W}_\mu| \cdot |\mathbf{D}_\mu| \cdot |\mathbf{W}_\mu^T|. \tag{5.12}$$

Если $\mu = v$, то все элементы $X_1, \ldots, X_\mu, Y_1, \ldots, Y_\mu$ отличны от нуля и $X_1, \ldots, X_\mu$ различны. Тогда все определители, входящие в правую часть соотношения (5.12), отличны от нуля, и $|\mathbf{M}_\mu| \neq 0$. Если $\mu > v$, то $|\mathbf{D}_\mu| = 0$ и $|\mathbf{M}_\mu| = 0$.

Используя теорему 5.1, можно указать следующую процедуру определения v.

Нужно последовательно рассматривать матрицы $\mathbf{M}_\mu$ для $\mu = t, t-2$ и т.д. до тех пор, пока не найдется такое μ_0, что матрица $|\mathbf{M}_{\mu_0}| \neq 0$. Тогда $\mu_0 = v$ – истинное число ошибок, которое произошло.

Зная v, можно решить систему (5.11) методом Гаусса или обращением матрицы $\mathbf{M}_v$ и получить коэффициенты полинома $\sigma(x)$. Для определения локаторов теперь достаточно найти корни $\sigma(x)$, что в конечном поле можно сделать, последовательно подставляя в $\sigma(x)$ все элементы поля (такая процедура поиска корней полинома носит название *процедуры Ченя*).

Для вычисления величин $Y_1, \ldots, Y_v$ достаточно решить систему из v первых уравнений системы (5.5) после подстановки туда известных теперь величин $X_1, \ldots, X_v$.

Таким образом, алгоритм декодирования БЧХ-кодов, который называют обычно *прямым* алгоритмом декодирования БЧХ-кодов [12], или *алгоритмом Питерсона–Горенстейна–Цирлера*, или *прямым методом* декодирования БЧХ-кодов, сводится к четырем этапам [7]:

– Вычисление компонент синдрома S_j.

– Определение коэффициентов $\sigma_1, \ldots, \sigma_v$ полинома $\sigma(x)$.

– Определение корней многочлена $\sigma(x)$.

– Вычисление $Y_1, \ldots, Y_v$ – значений ошибок.

Формально прямой алгоритм состоит в следующем:

1. По принятому многочлену $b(x)$ вычисляются компоненты синдрома $S_j = b(\eta^j)$, $j = \overline{1, 2t}$.

2. Рассматриваются определители $|\mathbf{M}_t|, |\mathbf{M}_{t-1}|, \ldots$ до тех пор, пока не найдется v такое, что $|\mathbf{M}_v| \neq 0$.

3. Вычисляются коэффициенты многочлена локаторов ошибок по формуле

$$(\sigma_v, \ldots, \sigma_1) = (-S_{v+1}, \ldots, -S_{2v}) \cdot (\mathbf{M}_v^{-1})^T.$$

4. Последовательной подстановкой всех элементов поля определяются корни многочлена $\sigma(x)$ (процедура Ченя).

5. Определяются значения ошибок из соотношения

$$\begin{bmatrix} Y_1 \\ Y_2 \\ \dots \\ Y_v \end{bmatrix} = \begin{bmatrix} X_1 & X_2 & \dots & X_v \\ X_1^2 & X_2^2 & \dots & X_v^2 \\ \dots & \dots & \dots & \dots \\ X_1^v & X_2^v & \dots & X_v^v \end{bmatrix}^{-1} \cdot \begin{bmatrix} S_1 \\ S_2 \\ \dots \\ S_v \end{bmatrix}. \qquad (5.13)$$

Пример 5.1. Прямой алгоритм декодирования кода Рида–Соломона (15,9) над $GF(16)$, исправляющего три ошибки.

Пусть α – примитивный элемент поля $GF(16)$. Тогда

$$\begin{aligned} g(x) &= (x+\alpha)(x+\alpha^2)\dots(x+\alpha^6) = \\ &= x^6 + \alpha^{10}x^5 + \alpha^{14}x^4 + \alpha^4 x^3 + \alpha^6 x^2 + \alpha^9 x + \alpha^6; \end{aligned}$$

$$a(x) = 0;$$

$$b(x) = a(x) + e(x) = \alpha^2 x^5 + \alpha x;$$

$$S_1 = \alpha^2 \cdot \alpha^5 + \alpha \cdot \alpha = \alpha^7 + \alpha^2 = \alpha^{12};$$

$$S_2 = \alpha^{10}, \quad S_3 = \alpha^{10}, \quad S_4 = \alpha^{13};$$

$$S_5 = \alpha^4, \quad S_6 = \alpha^{12}.$$

Пусть $v = 3$. Тогда

$$\mathbf{M}_3 = \begin{bmatrix} S_1 & S_2 & S_3 \\ S_2 & S_3 & S_4 \\ S_3 & S_4 & S_5 \end{bmatrix} = \begin{bmatrix} \alpha^{12} & \alpha^{10} & \alpha^{10} \\ \alpha^{10} & \alpha^{10} & \alpha^{13} \\ \alpha^{10} & \alpha^{13} & \alpha^4 \end{bmatrix}.$$

Определитель матрицы

$$\begin{aligned} |\mathbf{M}_3| &= \alpha^{11} + \underline{\alpha^3} + \underline{\alpha^3} + 1 + \alpha^8 + \alpha^9 = \\ &= \underline{\alpha^3} + \underline{\alpha^2} + \underline{\alpha} + \underline{1} + \underline{\alpha^2} + \underline{1} + \underline{\alpha^3} + \underline{\alpha} = 0. \end{aligned}$$

Значит, ошибок меньше чем 3. Пусть $v = 2$:

$$\mathbf{M}_2 = \begin{bmatrix} S_1 & S_2 \\ S_2 & S_3 \end{bmatrix} = \begin{bmatrix} \alpha^{12} & \alpha^{10} \\ \alpha^{10} & \alpha^{10} \end{bmatrix}.$$

Определитель этой матрицы не равен нулю, значит, произошли две ошибки. Далее

$$\mathbf{M}_2^{-1} = \begin{bmatrix} \alpha^{12} & \alpha^{12} \\ \alpha^{12} & \alpha^{14} \end{bmatrix};$$

$$\begin{bmatrix}\sigma_2\\ \sigma_1\end{bmatrix} = \mathbf{M}_2^{-1}\cdot\begin{bmatrix}S_3\\ S_4\end{bmatrix} = \begin{bmatrix}\alpha^{12} & \alpha^{12}\\ \alpha^{12} & \alpha^{14}\end{bmatrix}\cdot\begin{bmatrix}\alpha^{10}\\ \alpha^{13}\end{bmatrix} = \begin{bmatrix}\alpha^6\\ \alpha^2\end{bmatrix};$$

$$\sigma(x) = \alpha^6x^2 + \alpha^2x + 1 = \alpha^6(x+\alpha^{14})(x+\alpha^{10}).$$

Итак, α^{14} и α^{10} – величины, обратные локаторам ошибок $X_1 = \alpha$, $X_2 = \alpha^5$. Значения ошибок Y_1 и Y_2 определяются по формуле

$$\begin{bmatrix}Y_1\\ Y_2\end{bmatrix} = \begin{bmatrix}X_1 & X_2\\ X_1^2 & X_2^2\end{bmatrix}^{-1}\cdot\begin{bmatrix}S_1\\ S_2\end{bmatrix} = \begin{bmatrix}\alpha & \alpha^5\\ \alpha^2 & \alpha^{10}\end{bmatrix}\cdot\begin{bmatrix}\alpha^{12}\\ \alpha^{10}\end{bmatrix} =$$

$$= \begin{bmatrix}\alpha^2 & \alpha^{12}\\ \alpha^9 & \alpha^8\end{bmatrix}\cdot\begin{bmatrix}\alpha^{12}\\ \alpha^{10}\end{bmatrix} = \begin{bmatrix}\alpha\\ \alpha^2\end{bmatrix};$$

$$Y_1 = \alpha; \quad Y_2 = \alpha^2.$$

Получен $e(x) = \alpha x + \alpha^2 x^5$.

5.2. Ключевое уравнение

Шаги прямого алгоритма, представленные в разделе 5.1, были исторически первыми, описанными для декодирования циклических кодов, и они сочетаются со способом решения системы уравнений (5.5) путем сведения ее к линейной системе (5.11), что и является центральным в алгоритме Питерсона–Горенстейна–Цирлера.

Однако на сегодняшний день известны более эффективные алгоритмы декодирования циклических кодов, шаги которых, по сути, повторяют шаги прямого алгоритма, однако реализуются они иначе.

Основным отличием является использование так называемого *ключевого уравнения* вместо решения системы уравнений, для нахождения не только многочлена локаторов $\sigma(x)$, но и некоторого *многочлена значений ошибок* $\omega(x)$. Кроме того, на последнем шаге для нахождения значений ошибок вместо рассмотрения системы (5.13) выполняется *процедура Форни*, использующая многочлен значений $\omega(x)$.

Приведем шаги улучшенного алгоритма.

1. *Вычисление синдрома.* Пусть $\mathbf{b} = \mathbf{a} + \mathbf{e}$ – принятый вектор, где $\mathbf{a}$ – кодовое слово, $\mathbf{e}$ – вектор ошибки, $W(\mathbf{e}) \le t$. Первый шаг декодирования, как и прежде, состоит в вычислении синдрома $S(x)$. Заметим, что эта процедура может быть выполнена с линейной сложностью с помощью схемы Горнера, вычисляющей значение многочлена $f(x)$ степени n в точке, используя следующую группировку операций:

$$f(x) = f_0 + f_1x + f_2x^2 + \ldots + f_nx^n =$$
$$= f_0 + x(f_1 + x(f_2 + x(\ldots(f_{n-1} + f_nx)))).$$

2. *Решение ключевого уравнения.* Вторая фаза декодирования состоит в решении *ключевого уравнения*, т. е. нахождении многочлена локаторов $\sigma(x)$ и многочлена *значений ошибок* $\omega(x)$, удовлетворяющих следующему соотношению:

$$\frac{\omega(x)}{\sigma(x)} = S(x) \bmod x^r. \tag{5.14}$$

Многочлен $\omega(x)$ используется на шаге 4 для нахождения значений ошибок.

3. *Нахождение локаторов ошибок (процедуря Ченя).* Этот шаг полностью аналогичен описанному в разделе 5.1 и заключается в нахождении корней $\sigma(x)$ с помощью перебора по элементам конечного поля. Заметим только, что для его реализации также может быть использована схема Горнера.

4. *Нахождение значений ошибок (процедура Форни).* На последнем шаге значения ошибок могут быть найдены с помощью многочлена $\omega(x)$, полученного на шаге 2:

$$\varepsilon_i = -\frac{\omega(\alpha^{-i})}{\sigma'(\alpha^{-i})}, \tag{5.15}$$

где $\sigma'(x)$ – формальная производная многочлена $\sigma(x)$ (см. формулу (1.16)); i – позиция ошибки.

В следующих разделах мы рассмотрим наиболее эффективные способы решения ключевого уравнения.

5.3. Алгоритм Берлекэмпа–Месси

Основную сложность в прямом алгоритме декодирования представляет этап, связанный с решением системы уравнений (5.11), шаги 2 и 3 прямого алгоритма. Эта система, однако, не произвольна – ее коэффициенты обладают определенной структурой. На использовании этой структуры и основаны все известные способы упрощения прямого метода декодирования, в том числе рассмотренные в разделе 5.2. Однако эти методы ничего не несут для улучшения понимания процесса декодирования БЧХ-кодов, поэтому мы приведем в дальнейшем один из наиболее известных сегодня алгоритмов упрощения декодирования БЧХ-кодов, не обосновывая его подробно.

Лучшим способом осуществления второго и третьего этапов прямого декодирования кодов БЧХ считается *итеративный алгоритм Берлекэмпа–Месси* [1, 11]. Смысл этого алгоритма заключается в том, что

многочлен $\sigma(x)$ ищется методом последовательных приближений, начиная с $\sigma^{(0)}(x)$, $\sigma^{(1)}(x)$ и т.д., до $\sigma^{(v)}(x) = \sigma(x)$, выбирая $\sigma^{(j)}(x)$ как уточнение $\sigma^{(j-1)}(x)$.

При этом алгоритм Берлекэмпа–Месси может использоваться как вместо второго и третьего этапов прямого декодирования из раздела 5.1, так и для второго этапа – решения ключевого уравнения (5.14), описанного в разделе 5.2.

Из системы (5.11) видно, что для компонент синдрома имеет место рекуррентное уравнение

$$S_j = -\sum_{i=1}^{v} \sigma_i S_{j-i}, \quad j = v+1, \ldots, 2v. \tag{5.16}$$

Будем говорить, что многочлен $\sigma^{(j)}(x)$ *порождает* S_1, …, S_j, если для всех этих величин выполняется равенство

$$S_j = -\sum_{i=1}^{v^{(j)}} \sigma_i^{(j)} S_{j-i},$$

где $v^{(j)}$ – степень многочлена $\sigma^{(j)}(x)$, а $\sigma_i^{(j)}$ – коэффициенты этого многочлена. В полях характеристики 2 это условие можно переписать как

$$\Delta^{(j)} = S_j + \sum_{i=1}^{v^{(j)}} \sigma_i^{(j)} S_{j-i},$$

если $\Delta^{(j)} = 0$, то $\sigma^{(j)}(x)$ порождает S_1, …, S_j и $\Delta^{(j)} \neq 0$ в противном случае.

Задача решения системы уравнений (5.11) может интерпретироваться тогда как задача подбора многочлена $\sigma(x)$ минимальной степени, который порождает последовательность компонент синдрома S_1, S_2, …, S_{2t}.

Итеративный процесс отыскания $\sigma(x)$ начинается с $\sigma^{(0)}(x) = 1$ и упрощенно состоит в следующем: при заданном многочлене $\sigma^{(j-1)}(x)$, порождающем S_1, …, S_{j-1}, проверяется, не является ли $\sigma^{(j-1)}(x)$ порождающим и для S_j (т. е. выполняется ли $\Delta^{(j)} = 0$); если является, то $\sigma^{(j)}(x)$ полагается равным $\sigma^{(j-1)}(x)$, в противном случае $\sigma^{(j)}(x)$ уточняется так, чтобы он генерировал последовательность S_1, …, S_{j-1}, S_j. Процесс продолжается до тех пор, пока не отыщется многочлен, порождающий все компоненты синдрома.

Алгоритм Берлекэмпа–Месси имеет множество вариаций [1, 3, 11]. На рис. 5.1 приведена одна из версий. При этом, помимо многочлена локаторов $\sigma(x)$ алгоритм находит также многочлен значений $\omega(x)$, который затем может использоваться для нахождения значений ошибок, как это было описано в разделе 5.2. Отметим, что при использовании алгоритма Берлекэмпа–Месси нет нужды определять количество произошедших ошибок – алгоритм выполняет столько шагов, сколько у него на входе компонент синдрома.

Пример 5.2. Алгоритм Берлекэмпа–Месси при декодировании (15,9)-кода Рида–Соломона, исправляющего тройные ошибки:

$$\begin{aligned}
&a(x) = 0;\\
&b(x) = a(x) + e(x) = e(x) = \alpha x^7 + \alpha^5 x^5 + \alpha^{11} x^2;\\
&S_1 = \alpha \cdot \alpha^7 + \alpha^5 \cdot \alpha^5 + \alpha^{11} \cdot \alpha^2 = \alpha^{12};\\
&S_2 = 1;\ S_3 = \alpha^{14};\ S_4 = \alpha^{13};\ S_5 = 1;\ S_6 = \alpha^{11}.
\end{aligned}$$

В табл. 5.1 приведены шаги алгоритма Берлекэмпа–Месси. Таким образом, многочлен локаторов ошибок равен

$$\sigma(x) = 1 + \alpha^{14} x + \alpha^{11} x^2 + \alpha^{14} x^3 = (1 + \alpha^7 x)(1 + \alpha^5 x)(1 + \alpha^2 x),$$

отсюда его корни – α^{-7}, α^{-5}, α^{-2}, локаторы – α^2, α^5, α^7.

Теперь применим процедуру Форни (5.15) для нахождения значений ошибок. Заметим, что $\sigma'(x) = \alpha^{14} x^2 + \alpha^{14}$.

Найдем значение ошибки в позиции 2 (которой соответствует локатор α^2). Подставим в формулу (5.15) значение $i = 2$, с учетом того, что в поле $GF(2^4)$ $\alpha^{-2} = \alpha^{13}$:

$$\varepsilon_2 = -\frac{\omega\left(\alpha^{-2}\right)}{\sigma'\left(\alpha^{-2}\right)} = \frac{\alpha^8 \left(\alpha^{13}\right)^2 + \alpha^{12}\alpha^{13} + \alpha^{12}}{\alpha^{14} \left(\alpha^{13}\right)^2 + \alpha^{14}} = \frac{\alpha^7}{\alpha^{11}} = \alpha^{11}.$$

Аналогично для $i = 5$:

$$\varepsilon_5 = -\frac{\omega\left(\alpha^{-5}\right)}{\sigma'\left(\alpha^{-5}\right)} = \frac{\alpha^8 \left(\alpha^{10}\right)^2 + \alpha^{12}\alpha^{10} + \alpha^{12}}{\alpha^{14} \left(\alpha^{10}\right)^2 + \alpha^{14}} = \frac{\alpha^{14}}{\alpha^9} = \alpha^5,$$

и для $i = 7$:

$$\varepsilon_7 = -\frac{\omega\left(\alpha^{-7}\right)}{\sigma'\left(\alpha^{-7}\right)} = \frac{\alpha^8 \left(\alpha^8\right)^2 + \alpha^{12}\alpha^8 + \alpha^{12}}{\alpha^{14} \left(\alpha^8\right)^2 + \alpha^{14}} = \frac{\alpha^4}{\alpha^3} = \alpha.$$

Исходные данные: коэффициенты S_1, S_2, …, S_N.
Цель: вычисление полиномов $\sigma(x)$ и $\omega(x)$.

Обозначения: j – номер шага; $v^{(j)}$ – степень полинома $\sigma^{(j)}(x) = \sum_{i=0}^{v^{(j)}} \sigma_i^{(j)} x^i$; $k^{(j)}$, $L^{(j)}$ и $\Theta^{(j)}(x)$, $\Omega^{(j)}(x)$ — промежуточные переменные на j-м шаге.
Начальные условия: $j = 0$; $k^{(0)} = -1$; $L^{(0)} = 0$; $\sigma^{(0)}(x) = \sigma_0^{(0)} = 1$; $\Theta^{(0)}(x) = x$; $\omega^{(0)}(x) = 0$; $\Omega^{(0)}(x) = -1$.
Алгоритм:

1. Выполнить $j = j + 1$.
2. Вычислить
$$\Delta^{(j)} = S_j + \sum_{i=1}^{v^{(j-1)}} \sigma_i^{(j-1)} S_{j-i};$$
$$\sigma^{(j)}(z) = \sigma^{(j-1)}(x) - \Delta^{(j)} \Theta^{(j-1)}(x);$$
$$\omega^{(j)}(z) = \omega^{(j-1)}(x) - \Delta^{(j)} \Omega^{(j-1)}(x).$$
3. Оценить j, если $j = N$, перейти к п.7, при $j < N$ – к п.4.
4. Принять
$$L^{(j)} = \begin{cases} L^{(j-1)}, & \text{если } \Delta^{(j)} = 0; \\ \max(L^{(j-1)}, j - 1 - k^{(j-1)}), & \text{если } \Delta^{(j)} \neq 0. \end{cases}$$
5. Вычислить $\Theta^{(j)}(x)$, $\Omega^{(j)}(x)$ и $k^{(j)}$ в зависимости от следующих условий:

   если $L^{(j)} = L^{(j-1)}$, то
$$\Theta^{(j)}(x) = x\Theta^{(j-1)}(x);$$
$$\Omega^{(j)}(x) = x\Omega^{(j-1)}(x);$$
$$k^{(j)} = k^{(j-1)};$$

   если $L^{(j)} > L^{(j-1)}$, то
$$\Theta^{(j)}(x) = x\sigma^{(j-1)}(x)/\Delta^{(j)};$$
$$\Omega^{(j)}(x) = x\omega^{(j-1)}(x)/\Delta^{(j)};$$
$$k^{(j)} = j - 1 - L^{(j-1)}.$$
6. Вернуться к п.1.
7. Конец алгоритма. Возвратить $\sigma(x) = \sigma^{(N)}(x)$, $\omega(x) = \omega^{(N)}(x)$.

Рис. 5.1. Алгоритм Берлекэмпа–Месси

Таблица 5.1

Шаги декодирования алгоритма Берлекэмпа–Месси

j	Δ	$\sigma(x)$	$\omega(x)$	L	$\Theta(x)$	$\Omega(x)$	k
0	–	1	0	0	x	-1	-1
1	α^{12}	$1+\alpha^{12}x$	α^{12}	1	$\alpha^3 x$	0	0
2	α^7	$1+\alpha^3 x$	α^{12}	1	$\alpha^3 x^2$	0	0
3	1	$1+\alpha^3 x+\alpha^3 x^2$	α^{12}	2	$x+\alpha^3 x^2$	$\alpha^{12}x$	1
4	1	$1+\alpha^{14}x$	$\alpha^{12}+\alpha^{12}x$	2	$x^2+\alpha^3 x^3$	$\alpha^{12}x^2$	1
5	α^{11}	$1+\alpha^{14}x+$ $+\alpha^{11}x^2+\alpha^{14}x^3$	$\alpha^{12}+\alpha^{12}x+\alpha^8 x^2$	3	$\alpha^4 x+\alpha^3 x^2$	$\alpha x+\alpha x^2$	2
6	0	$1+\alpha^{14}x+$ $\alpha^{11}x^2+\alpha^{14}x^3$	$\alpha^{12}+\alpha^{12}x+\alpha^8 x^2$	–	–	–	–

Мы получили вектор ошибки:

$$e(x) = \alpha x^7 + \alpha^5 x^5 + \alpha^{11} x^2,$$

что завершает декодирование.

5.4. Решение ключевого уравнения: алгоритм Евклида

В этом разделе мы приведем еще один способ решения ключевого уравнения, описанного в разделе 5.2, использующий для нахождения многочлена локаторов и многочлена значений ошибок обобщенный алгоритм Евклида.

Перепишем ключевое уравнение (5.14) в другом виде [1, 11]:

$$S(x)\sigma(x) + x^r q(x) = \omega(x),$$

где $q(x)$ – некоторый многочлен. Сформулируем теперь обобщенный алгоритм Евклида для многочленов [5, 11].

Теорема 5.2 (обобщенный алгоритм Евклида)**.** *Для двух многочленов* $r_{-1}(x)$ *и* $r_0(x)$*, таких, что* $\deg r_{-1}(x) \geq \deg r_0(x)$*, существует разложение*

$$b(x)r_{-1}(x) + a(x)r_0(x) = d(x), \tag{5.17}$$

где $d(x) = НОД(r_{-1}(x), r_0(x))$*,* u $\deg a(x), \deg b(x) < \deg r_{-1}(x)$.

Доказательство. Всякий многочлен $r_{j-2}(x)$ можно единственным образом разделить с остатком на многочлен $r_{j-1}(x)$, получая представление

$$r_{j-2}(x) = q_j(x)r_{j-1}(x) + r_j(x), \tag{5.18}$$

где $q_j(x)$ – частное от деления, $r_j(x)$ – остаток от деления многочлена $r_{j-2}(x)$ на многочлен $r_{j-1}(x)$, $\deg r_j(x) < \deg r_{j-1}(x)$. Очевидно, что если $d(x) = \text{НОД}(r_{j-2}(x), r_{j-1}(x))$, то также $d(x) = \text{НОД}(r_{j-1}(x), r_j(x))$. Так как степени многочленов на каждом j-м шаге, начиная с $j = 1$, строго уменьшаются, вычисление (5.18) завершится. Последний ненулевой остаток и будет равен $d(x)$.

Теперь, пусть

$$\begin{aligned} a_{-1}(x) &= 0; & a_0(x) &= 1; \\ b_{-1}(x) &= 1; & b_0(x) &= 0. \end{aligned}$$

Определим

$$\begin{aligned} a_i(x) &= q_i(x)a_{i-1}(x) + a_{i-2}(x); \\ b_i(x) &= q_i(x)b_{i-1}(x) + b_{i-2}(x). \end{aligned}$$

Тогда

$$\begin{aligned} \begin{bmatrix} a_i(x) & a_{i-1}(x) \\ b_i(x) & b_{i-1}(x) \end{bmatrix} &= \begin{bmatrix} a_{i-1}(x) & a_{i-2}(x) \\ b_{i-1}(x) & b_{i-2}(x) \end{bmatrix} \begin{bmatrix} q_i(x) & 1 \\ 1 & 0 \end{bmatrix} = \ldots = \\ &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} q_1(x) & 1 \\ 1 & 0 \end{bmatrix} \cdots \begin{bmatrix} q_i(x) & 1 \\ 1 & 0 \end{bmatrix}. \end{aligned} \tag{5.19}$$

Аналогично

$$\begin{bmatrix} r_{i-2}(x) \\ r_{i-1}(x) \end{bmatrix} = \begin{bmatrix} q_i(x) & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} r_{i-1}(x) \\ r_i(x) \end{bmatrix};$$

$$\begin{bmatrix} r_{i-3}(x) \\ r_{i-2}(x) \end{bmatrix} = \begin{bmatrix} q_{i-1}(x) & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} q_{i-2}(x) & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} r_{i-1}(x) \\ r_i(x) \end{bmatrix};$$

$$\begin{aligned} \begin{bmatrix} r_{i-1}(x) \\ r_0(x) \end{bmatrix} &= \begin{bmatrix} q_1(x) & 1 \\ 1 & 0 \end{bmatrix} \cdots \begin{bmatrix} q_{i-1}(x) & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} q_i(x) & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} r_{i-1}(x) \\ r_i(x) \end{bmatrix} = \\ &= \begin{bmatrix} a_i(x) & a_{i-1}(x) \\ b_i(x) & b_{i-1}(x) \end{bmatrix} \begin{bmatrix} r_{i-1}(x) \\ r_i(x) \end{bmatrix}. \end{aligned} \tag{5.20}$$

Определитель в правой части (5.19) равен $(-1)^i$. Отсюда с учетом (5.19) и (5.20)

$$\begin{bmatrix} r_{i-1}(x) \\ r_i(x) \end{bmatrix} = (-1)^i \begin{bmatrix} b_{i-1}(x) & -a_{i-1}(x) \\ -b_i(x) & a_i(x) \end{bmatrix} \begin{bmatrix} r_{-1}(x) \\ r_0(x) \end{bmatrix}. \tag{5.21}$$

Тогда

$$r_j(x) = (-1)^i(-b_j(x)r_{-1}(x) + a_j(x)r_0(x)),$$

откуда следует (5.18). Кроме того,

$$\begin{aligned}
&\deg a_i = \sum_{k=1}^{i} \deg q_k;\\
&\deg r_{i-1} = \deg r_{-1} - \sum_{k=1}^{i} \deg q_k;\\
&\deg a_i = \deg r_{-1} - \deg r_{i-1} < \deg r_{-1}.
\end{aligned}$$

Аналогичные соотношения могут быть выписаны для b_i.

Описанный алгоритм применим для решения ключевого уравнения (5.14). При известных $S(x)$ и x^r алгоритм Евклида для многочленов позволяет легко найти $\sigma(x)$ и $\omega(x)$, более формально:

$$r_{-1}(x) = x^r; \quad r_0(x) = S(x).$$

Далее действуем согласно алгоритму Евклида, пока на шаге k не выполнится

$$\begin{aligned}
&\deg r_{k-1}(x) \geq r/2;\\
&\deg r_k(x) \leq r/2 - 1.
\end{aligned} \tag{5.22}$$

Тогда искомые многочлен локаторов и многочлен значений

$$\begin{aligned}
&\sigma(x) = \delta a_k(x); && \deg \sigma(x) \leq r/2;\\
&\omega(x) = (-1)^k \delta r_k(x); && \deg \omega(x) \leq r/2 - 1,
\end{aligned}$$

где нормирующий коэффициент δ выбирается так, чтобы свободный член в $\sigma(x)$ был равен единице.

Теорема 5.3. *Многочлены $\sigma(x)$ и $\omega(x)$ в (5.22) являются единственным решением (5.14) при $\sigma(0) = 1$, $\deg \sigma \leq r/2$, $\deg \omega \leq r/2 - 1$ и наименьшей возможной степени многочлена $\sigma(x)$.*

Доказательство. Покажем, что если существуют два решения (5.14), например, (σ, ω), и (σ', ω'), при выполнении условий $\deg \sigma \leq r/2$, $\deg \omega \leq r/2 - 1$, $\deg \sigma' \leq r/2$, $\deg \omega' \leq r/2 - 1$, то для некоторого многочлена μ выполняется $\sigma = \mu\sigma'$ и $\omega = \mu\omega'$.

Действительно, если $\omega \equiv \sigma S \bmod x^r$, $\omega' \equiv \sigma' S \bmod x^r$, то $\omega\sigma' \equiv \omega'\sigma \bmod x^r$.

Но, так как степень многочленов в обеих частях сравнения меньше r, имеем $\omega\sigma' = \omega'\sigma$, т. е. $\mu = \omega/\omega' = \sigma/\sigma'$.

Далее, если в решении (σ, ω), задаваемом (5.22), степень $\sigma(x)$ не является наименьшей, тогда $\sigma = \mu\sigma'$ и $\omega = \mu\omega'$, где (σ', ω') – также решение (5.14), а μ – некоторый многочлен. Тогда из (5.21) и (5.22) имеем

$$\omega(x) = (-1)^k \sigma r_k(x) = -\sigma b_k(x) x^r + \sigma(x) S(x);$$
$$\mu(x)\omega'(x) = -\delta b_k(x) x^r + \mu(x)\sigma'(x) S(x).$$

Кроме того, из (5.14) для некоторого $\psi(x)$

$$\omega'(x) = \sigma'(x) S(x) + \psi(x) x^r.$$

Следовательно, $\mu(x)$ делит $b_k(x)$. Но $\mu(x)$ делит также и $\sigma(x) = \delta a_k(x)$. Так как $a_k(x)$ и $b_k(x)$ взаимно просты, $\mu(x)$ является константой.

Пример 5.3. Рассмотрим декодирование принятого слова $u(x)$ из примера 5.2:

$$u(x) = \alpha x^7 + \alpha^5 x^5 + \alpha^{11} x^2;$$
$$S(x) = \alpha^{11} x^5 + x^4 + \alpha^{13} x^3 + \alpha^{14} x^2 + x + \alpha^{12}.$$

В табл. 5.2 приведены шаги вычисления $\sigma(x)$ и $\omega(x)$. Заметим, что для вычисления $\sigma(x)$ нам достаточно в выражении (5.17) найти только $a(x)$.

Отнормировав $a_3(x)$ и $r_3(x)$ умножением на $\delta = \alpha^{14}$, получим

$$\sigma(x) = \alpha^{14} x^3 + \alpha^{11} x^2 + \alpha^{14} x + 1;$$
$$\omega(x) = \alpha^8 x^2 + \alpha^{12} x + \alpha^{12}.$$

Процедуры Ченя и Форни выполняются аналогично примеру 5.2.

Таблица 5.2

Шаги декодирования алгоритма Евклида

i	$r_i(x)$	$q_i(x)$	$a_i(x)$
-1	x^6		
0	$\alpha^{11} x^5 + x^4 + \alpha^{13} x^3 + \alpha^{14} x^2 +$ $+ x + \alpha^{12}$		
1	$x^4 + \alpha^2 x^3 + \alpha^3 x^2 + \alpha^{10} x + \alpha^5$	$\alpha^4 x + \alpha^8$	$\alpha^4 x + \alpha^8$
2	$x^3 + \alpha^{12} x^2 + x + 1$	$\alpha^{11} x + \alpha^6$	$x^2 + \alpha^2 x + \alpha^3$
3	$\alpha^9 x^2 + \alpha^{13} x + \alpha^{13}$	$x + \alpha^7$	$x^3 + \alpha^{12} x^2 + x + \alpha$

5.5. Задачи

1. Используется двоичный код БЧХ (15,5) с $d = 7$. Принято слово

$$b(x) = x^{14} + x^8 + x^5 + x^4 + x^2 + x + 1.$$

Для принятого слова:

а) продекодировать $b(x)$ прямым алгоритмом;

б) определить $\sigma(x)$ методом Берлекэмпа–Месси;

в) сравнить число операций для вычисления $\sigma(x)$ при использовании алгоритмов Питерсона и Берлекэмпа.

2. Модифицировать прямой алгоритм декодирования на случай исправления стираний, ошибок и стираний.

3. Определить $\sigma(x)$ при декодировании двоичного кода (15,7), если

$$b(x) = x^{12} + x^{11} + x^8 + x^7 + x^6 + x^4 + 1.$$

4. Используется БЧХ-код (15,9) над $GF(4)$ с порождающим многочленом $g(x) = x^6 + 3x^5 + x^4 + x^3 + 2x^2 + 2x + 1$. Принято слово

$$b(x) = x^{11} + 2x^{10} + 3x^9 + x^8 + 2x^7 + 2x^6 + x^5.$$

Продекодировать $b(x)$.

5. Продекодировать принятое слово РС-кода, исправляющего три ошибки, если $S_1 = \alpha^7$; $S_2 = \alpha^{12}$; $S_3 = \alpha^6$; $S_4 = \alpha^{12}$; $S_5 = \alpha^{14}$; $S_6 = \alpha^{14}$.

6. Дан код Рида–Соломона над полем GF(11) с примитивным элементом $\alpha = 2$, с порождающим многочленом

$$g(x) = x^4 + 3x^3 + 5x^2 + 8x + 1$$

и порождающей и проверочной матрицами:

$$\mathbf{G} = \begin{pmatrix} 1 & 8 & 5 & 3 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 8 & 5 & 3 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 8 & 5 & 3 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 8 & 5 & 3 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 8 & 5 & 3 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 8 & 5 & 3 & 1 \end{pmatrix};$$

$$\mathbf{H} = \begin{pmatrix} 1 & 2 & 4 & 8 & 5 & 10 & 9 & 7 & 3 & 6 \\ 1 & 4 & 5 & 9 & 3 & 1 & 4 & 5 & 9 & 3 \\ 1 & 8 & 9 & 6 & 4 & 10 & 3 & 2 & 5 & 7 \\ 1 & 5 & 3 & 4 & 9 & 1 & 5 & 3 & 4 & 9 \end{pmatrix}.$$

В данном коде необходимо произвести декодирование полученного слова b при помощи: а) декодера Питерсона–Горенстейна–Цирлера; б) декодера Берлекэмпа–Месси; в) алгоритма Евклида.

6.1.

а) $\mathbf{b} = (2, 8, 2, 9, 4, 9, 1, 6, 2, 5)$;
б) $\mathbf{b} = (1, 2, 3, 8, 5, 6, 9, 0, 6, 3)$;
в) $\mathbf{b} = (2, 10, 1, 4, 5, 0, 7, 6, 10, 9)$.

6.2.

а) $\mathbf{b} = (2, 3, 8, 5, 0, 2, 5, 2, 8, 3)$;
б) $\mathbf{b} = (4, 8, 6, 8, 3, 9, 4, 5, 1, 0)$;
в) $\mathbf{b} = (6, 6, 9, 4, 5, 10, 0, 5, 0, 7)$.

6.3.

а) $\mathbf{b} = (5, 0, 1, 2, 5, 6, 0, 8, 8, 4)$;
б) $\mathbf{b} = (3, 8, 9, 1, 5, 10, 6, 0, 3, 10)$;
в) $\mathbf{b} = (0, 9, 4, 2, 10, 3, 1, 2, 5, 10)$.

6.4.

а) $\mathbf{b} = (9, 5, 5, 9, 0, 5, 9, 1, 4, 9)$;
б) $\mathbf{b} = (5, 4, 10, 10, 10, 2, 7, 5, 10, 2)$;
в) $\mathbf{b} = (7, 5, 2, 4, 2, 4, 0, 8, 2, 9)$.

6.5.

а) $\mathbf{b} = (10, 5, 1, 3, 8, 1, 10, 7, 9, 1)$;
б) $\mathbf{b} = (5, 3, 5, 3, 2, 4, 7, 0, 10, 1)$;
в) $\mathbf{b} = (7, 10, 2, 4, 2, 9, 7, 2, 2, 9)$.

6.6.

а) $\mathbf{b} = (7, 10, 5, 8, 0, 9, 1, 2, 4, 5)$;
б) $\mathbf{b} = (0, 1, 9, 10, 9, 5, 6, 2, 10, 9)$;
в) $\mathbf{b} = (8, 1, 2, 6, 7, 7, 1, 4, 1, 10)$.

6.7.

а) $\mathbf{b} = (8, 9, 3, 1, 3, 5, 9, 10, 0, 10)$;
б) $\mathbf{b} = (5, 9, 6, 10, 2, 3, 0, 1, 0, 5)$;
в) $\mathbf{b} = (2, 5, 10, 2, 0, 5, 10, 4, 10, 0)$.

6.8.

а) $\mathbf{b} = (2, 4, 0, 9, 6, 2, 4, 8, 9, 7)$;
б) $\mathbf{b} = (5, 9, 5, 6, 4, 9, 6, 1, 10, 5)$;
в) $\mathbf{b} = (5, 8, 10, 3, 10, 5, 5, 7, 9, 5)$.

6.9.

а) $\mathbf{b} = (10, 9, 1, 6, 8, 4, 6, 1, 9, 8)$;
б) $\mathbf{b} = (1, 8, 9, 1, 4, 2, 6, 1, 7, 8)$;
в) $\mathbf{b} = (3, 4, 8, 4, 0, 8, 9, 2, 5, 7)$.

6.10.

а) $\mathbf{b} = (2, 1, 10, 0, 2, 4, 5, 9, 10, 4)$;
б) $\mathbf{b} = (0, 8, 2, 10, 8, 9, 1, 8, 5, 8)$;
в) $\mathbf{b} = (1, 2, 6, 2, 0, 3, 9, 4, 3, 10)$.

6.11.

а) $\mathbf{b} = (1, 3, 1, 3, 5, 3, 4, 6, 10, 9)$;
б) $\mathbf{b} = (3, 8, 1, 8, 5, 8, 6, 8, 1, 1)$;
в) $\mathbf{b} = (7, 2, 9, 9, 7, 3, 4, 8, 10, 2)$.

6.12.

а) $\mathbf{b} = (7, 8, 9, 5, 1, 0, 10, 8, 0, 5)$;
б) $\mathbf{b} = (9, 1, 6, 5, 1, 6, 2, 2, 2, 2)$;
в) $\mathbf{b} = (1, 10, 2, 2, 6, 8, 9, 0, 6, 3)$.

6.13.

а) $\mathbf{b} = (3, 1, 9, 5, 6, 0, 7, 7, 2, 3)$;
б) $\mathbf{b} = (7, 4, 5, 7, 1, 8, 1, 0, 10, 8)$;
в) $\mathbf{b} = (3, 6, 7, 3, 3, 0, 5, 1, 5, 5)$.

6.14.

а) $\mathbf{b} = (8, 7, 4, 0, 8, 3, 6, 2, 9, 6)$;
б) $\mathbf{b} = (1, 0, 5, 8, 2, 5, 8, 0, 4, 9)$;
в) $\mathbf{b} = (8, 1, 4, 9, 3, 8, 9, 1, 7, 2)$.

6.15.

а) $\mathbf{b} = (3, 4, 5, 8, 4, 7, 4, 10, 1, 0)$;
б) $\mathbf{b} = (2, 5, 8, 9, 5, 2, 2, 8, 1, 9)$;
в) $\mathbf{b} = (9, 6, 6, 10, 9, 0, 5, 4, 2, 0)$.

Литература

1. *Блейхут Р.* Теория и практика кодов, контролирующих ошибки. М.:Мир, 1986.
2. *Виноградов И.* Основы теории чисел: учеб. для вузов. М.:Лань, 2009.
3. *Галлагер Р.* Теория информации и надежная связь. М.:Советское радио, 1974.
4. *Касами Т.*, *Токура Н.*, *Ивадари Е. и др.* Теория кодирования. М.:Мир, 1978.
5. *Кнут Д.* Искусство программирования для ЭВМ. Т.2. Получисленные алгоритмы. М.:Вильямс, 2000.
6. *Колесник В.Д.*, *Полтырев Г.Ш.* Курс теории информации. М.:Наука, 1982.
7. *Крук Е.А.* Алгебраическое декодирование циклических кодов. СПб.:Нестор, 2002.
8. *Крук Е.А.*, *Овчинников А.А.* Лекции по теории кодирования. СПб.:ГУАП, 2004.
9. *Лазарева С.В.*, *Овчинников А.А.* Математические основы криптологии. Тесты простоты и факторизация. СПб.:ГУАП, 2006.
10. *Лидл Р.*, *Нидеррайтер Г.* Конечные поля. В 2-х т. М.:Мир, 1988.
11. *Мак-Вильямс Ф. Дж.*, *Слоэн Н. Дж. А.* Теория кодов, исправляющих ошибки. М.:Связь, 1979.
12. *Питерсон У.*, *Уэлдон Э.* Коды, исправляющие ошибки. М.:Мир, 1976.
13. *Шеннон К.* Работы по теории информации и кибернетике. М.:Иностранная литература, 1963.
14. *McEliece R.J.* Finite fields for computer scientists and engineers. Kluwer Academic publishers, 1986.
15. *Shannon C. E.* A Mathematical Theory of Communication. Bell System Technical Journal, 1948.

Содержание

Учебное издание

Крук Евгений Аврамович
Овчинников Андрей Анатольевич

Основы теории кодирования

Учебное пособие

В авторской редакции

Сдано в набор 6.12.13. Подписано в печать 16.12.13. Формат 60 × 84 1/16.
Бумага офсетная. Печать офсетная. Усл. печ. л. 6,2. Уч.-изд. л. 6,6.
Тираж 300 экз. Заказ №635.

Редакционно-издательский центр ГУАП

190000, Санкт-Петербург, ул. Б. Морская, 67