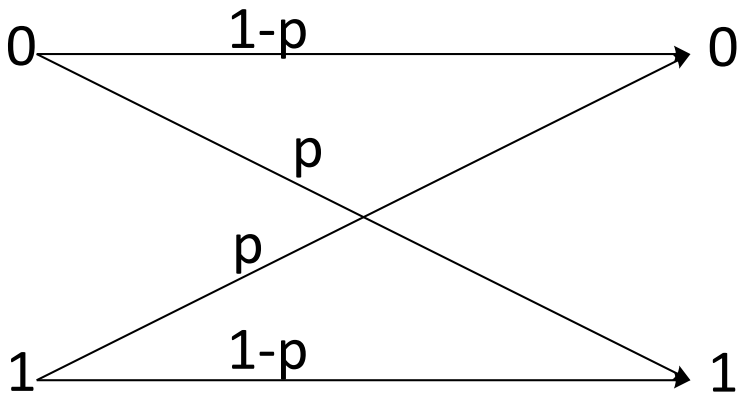
УНИВЕРСИТЕТ ИТМО

П.В. Трифонов

ОСНОВЫ ПОМЕХОУСТОЙЧИВОГО КОДИРОВАНИЯ



Санкт-Петербург
2022

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

УНИВЕРСИТЕТ ИТМО

П.В. Трифонов

ОСНОВЫ ПОМЕХОУСТОЙЧИВОГО КОДИРОВАНИЯ

УЧЕБНОЕ ПОСОБИЕ

РЕКОМЕНДОВАНО К ИСПОЛЬЗОВАНИЮ В УНИВЕРСИТЕТЕ ИТМО
по направлению подготовки 01.03.02, 10.03.01, 11.03.02
в качестве учебного пособия для реализации основных профессиональных образовательных программ высшего образования бакалавриата

Санкт-Петербург
2022

Трифонов П.В., Основы помехоустойчивого кодирования– СПб: Университет ИТМО, 2022. – 231 с.

Рецензент:
Кудряшов Борис Давидович, доктор технических наук, профессор, профессор (квалификационная категория "профессор-эксперт") факультета информационных технологий и программирования, Университета ИТМО.

Рассматриваются основные методы помехоустойчивого кодирования, применяемые в современных системах передачи информации. В частности, приведено описание кодов БЧХ и Рида-Соломона, турбо-кодов, кодов с малой плотностью проверок на четность, полярных кодов. Представлены как теоретические положения, лежащие в основе рассматриваемых конструкций, так и вопросы их практической реализации. Приведены задачи и практические задания. Издание ориентировано как на студентов, так и на инженеров, занимающихся практическое реализацией методов помехоустойчивого кодирования

Университет ИТМО – национальный исследовательский университет, ведущий вуз России в области информационных, фотонных и биохимических технологий. Альма-матер победителей международных соревнований по программированию – ICPC (единственный в мире семикратный чемпион), Google Code Jam, Facebook Hacker Cup, Яндекс.Алгоритм, Russian Code Cup, Topcoder Open и др. Приоритетные направления: IT, фотоника, робототехника, квантовые коммуникации, трансляционная медицина, Life Sciences, Art&Science, Science Communication. Входит в ТОП-100 по направлению «Автоматизация и управление» Шанхайского предметного рейтинга (ARWU) и занимает 74 место в мире в британском предметном рейтинге QS по компьютерным наукам (Computer Science and Information Systems). С 2013 по 2020 гг. – лидер Проекта 5–100.

Оглавление

Сокращения, принятые в пособии

АМ	амплитудная модуляция
БКЕР	Бал-Коке-Елинек-Равив
БЧХ	Боуз-Чоудхури-Хоквингем
ДПФ	дискретное преобразоване Фурье
КАМ	квадратурно-амплитудная модуляция
МППЧ	малая плотность проверок на четность
НОД	наибольший общий делитель
РКМ	решетчато-кодовая модуляция
РСЛОС	регистр сдвига с линейной обратной связью
ФМ	фазовая модуляция

Введение

Проблема надежной передачи информации возникла перед человеком еще в доисторические времена. Она всегда решалась путем внесения некоторой избыточности в передаваемые данные. Это делалось не только организационными методами (например, посылка нескольких гонцов с одним и тем же сообщением), но и подсознательно на уровне языка. Действительно, во многих случаях удается понять смысл предложения, даже если при его написании были допущены орфографические ошибки. Сведения о грамматических правилах, допустимых словах, взаимосвязи обозначаемых ими понятий, а также типовых ошибках при написании позволяют восстановить замысел автора текста. Вместе с тем, чем большее число ошибок допущено в документе (или чем сильнее он поврежден в результате каких-либо внешних воздействий), тем более трудоемким становится его понимание и тем больше вероятность неправильной его интерпретации. С ростом объема данных, накопленных человечеством, и развитием средств вычислительной техники потребность в обмене информацией многократно возросла. В простейших случаях избыточность естественного языка может оказаться достаточной и в случае передачи по техническим каналам связи. Однако такой подход абсолютно неприемлем при передаче больших объемов информации, а также в том случае, когда принятые данные должны интерпретироваться компьютером. Таким образом, возникла потребность в разработке математических методов помехозащиты.

Теория помехоустойчивого кодирования возникла в 1948 г. после публикации основополагающей работы К. Шеннона [62], в которой была показана принципиальная возможность обеспечения сколь угодно надежной передачи данных со скоростью, не превышающей пропускной способности канала. Однако конкретные методы, которые позволили бы достичь этого, в работе Шеннона предложены не были. В течение нескольких лет после ее публикации были предложены некоторые конструкции кодов, исправляющих ошибки, но их характеристики были чрезвычайно далеки от теоретически достижимых. С другой стороны, уровень развития вычислительной техники не позволял реализовать даже их, что привело к возникновению пессимистических настроений.

По мере развития электроники и вычислительной техники ситуация стала меняться. Одним из первых практических приложений теории кодирования стала информационная система сопровождения полета первого искусственного спутника Земли, созданная в СССР. Помехоустойчивое кодирование начало активно внедряться в системы космической связи. Постепенно, с ростом возможностей микроэлектроники, оно нашло применение в модемах, обеспечивавших взаимодействие ЭВМ, системах мобильной связи, накопителях на оптических и магнитных дисках и лентах. Была разработана богатая алгебраическая теория кодирования, которая, однако, не давала ответа на вопрос о том, как может быть практически достигнут предел Шеннона. Революция в помехоустойчивом кодировании состоялась в 1993 году, когда была предложена чрезвычайно простая конструкция турбо-кодов [18], обеспечивавшая близкую к оптимальной корректирующую способность. Позднее аналогичные свойства были обнаружены и у кодов с малой плотностью проверок на четность [50], которые были предложены еще в 1963 году [36], но не нашли применения из-за высокой сложности реализации. Наконец, в 2009 году, через 61 год после выхода работы Шеннона, была предложена конструкция полярных кодов [14], достигающих пропускной способности достаточно широкого класса каналов. Разработанный за эти годы математический аппарат нашел свое применение не только в системах хранения и передачи информации, но и в задачах обработки изображений, биометрике, биоинформатике и многих других отраслях.

Целью данного учебного пособия является систематическое изложение наиболее важных разделов теории помехоустойчивого кодирования с точки зрения инженера-разработчика системы передачи информации, решающего задачи выбора подходящих корректирующих кодов и реализации соответствующих алгоритмов, а также создания новых кодовых методов. В нем представлены конкретные численные результаты, характеризующие корректирующую способность различных кодов, которые могут быть полезны как при выборе наиболее подходящей для конкретного приложения конструкции, так и для оценки корректности реализации соответствующего алгоритма. Оно призвано также осветить недавние достижения теории кодирования, которые не отражены в классических учебниках и монографиях, изданных на русском языке. Представленный библиографический список включает статьи, оказавшие наибольшее влияние на развитие теории и практики помехоустойчивого кодирования. Несмотря на то, что некоторые из них относятся к 1960-ым

годам, представленные в них методы все еще широко используются в самых современных системах передачи и хранения информации. Вместе с тем, в список литературы включены ссылки на недавние результаты, которые уже нашли или в ближайшее время найдут практическое применение. Читателям рекомендуется самостоятельно изучить хотя бы некоторые из этих статей для более глубокого понимания рассматриваемых вопросов. Это может быть полезно при выполнении самостоятельной научной работы.

Данная книга может использоваться в качестве основного учебного пособия при изучении односеместровго курса “Основы помехоустойчивого кодирования”. Материал разбит на главы, каждая из которых соответствует одной или двум лекциям и освещает один из разделов теории кодирования. Типовая структура главы включает описание некоторого класса корректирующих кодов, их свойств, методов их кодирования и декодирования. В конце глав представлены задачи, выполнение которых способствует лучшему усвоению материалов. Предполагается, что эти задачи будут частично разобраны на практических занятиях, а частично решены обучающимися самостоятельно. Значительная часть этих задач заимствована из классических и современных учебников и монографий по теории кодирования [3, 8, 48, 1, 19, 51, 5, 7]. В приложении представлен примерный список тем для курсовой работы. Кроме того, подготовка устного сообщения по материалам статей, приведенных в списке литературы (на усмотрение преподавателя), включая строгое доказательство приведенных там утверждений, может быть альтернативой или дополнением курсовой работы или теоретического экзамена.

Пособие организовано следующим образом.

В части I изложены предпосылки развития теории кодирования и представлены её основные понятия и методы. Часть II посвящена алгебраической теории кодирования. В части III представлены результаты, касающиеся кодов на графах и вероятностных методов декодирования. Вопросы построения кодовой модуляции рассмотрены в части IV.

Автор глубоко признателен рецензенту Б.Д. Кудряшову за внимательное ознакомление с рукописью и её конструктивную критику. Автор также выражает благодарность студентам А. Горбунову, А. Маркову, А. Мишиной, А. Чугрееву за многочисленные сообщения об ошибках и опечатках.

Часть I

Теоретические основы кодирования

1. Основные понятия

1.1. Кодирование в технических системах

1.1.1. Системы хранения и передачи информации

Исторически первым приложением помехоустойчивого кодирования являются системы хранения и передачи информации. На рис. 1.1 представлена типовая структура такой системы. Первым этапом обработки информации на передающей стороне является применение некоторого алгоритма сжатия данных (кодера источника), за счет которого во многих случаях можно существенно снизить объем реально передаваемой информации. Это достигается за счет устранения статистической избыточности, которая часто встречается в данных, формируемых современными информационными системами. В некоторых случаях на этом этапе производятся необратимые преобразования данных, т.е. сжатие с потерями, которые, однако, не являются значимыми для получателя. Принципы кодирования источников изложены, например, в [6]. Шенноном было показано, что кодирование источника может выполняться независимо от канального кодирования, поэтому в дальнейшем этот этап рассматриваться не будет. Вместе с тем, будем предполагать, что статистическая избыточность в передаваемых данных уже устранена, и они представляют последовательность независимых равномерно распределенных случайных величин из некоторого множества $\mathbb{U}$.

Далее в некоторых случаях может осуществляться шифрование данных с целью их защиты от несанкционированного доступа. Иногда эту операцию ошибочно также называют кодированием. Она, однако, не является предметом изучения теории помехоустойчивого кодирования.

После добавления к передаваемым данным необходимых служебных сведений, определяемых особенностями конкретной системы (например, адреса получателя), они разбиваются на блоки фиксированной длины. Операция канального кодирования состоит во внесении в эти блоки некоторой избыточной информации, которая будет позднее использована для исправления ошибок, возникающих при передаче. Результатом этой операции являются последовательности символов c_i, называемые кодовыми словами, которые преобразуются модулятором в сигналы, пригодные для передачи по некоторому каналу. Обозначим как $\mathcal{C}(u) : \mathbb{U} \to \mathbb{X}$

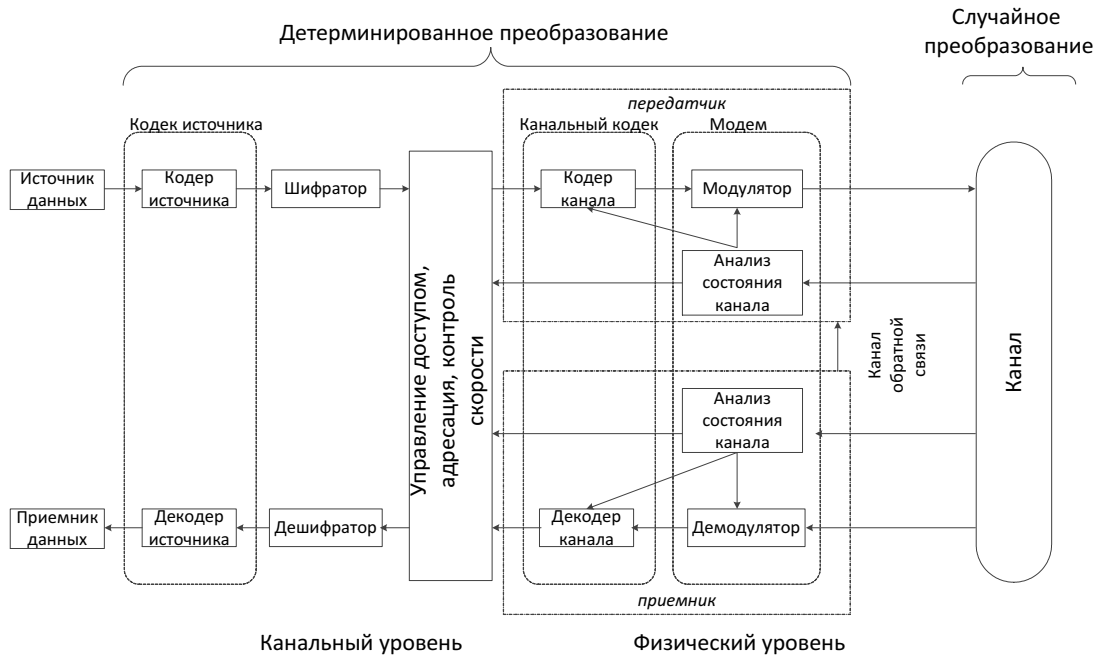


Рис. 1.1. Структура систем хранения и передачи информации

преобразование, выполняемое над сообщением u канальным кодером и модулятором. Это преобразование, как и все другие ранее описанные операции, является детерминированным.

В противоположность этому, *канал передачи информации* осуществляет стохастическое, т.е. недетерминированное преобразование данных $\mathcal{Y}(X) : \mathbb{X} \to \mathbb{Y}$, состоящее в наложении на них некоторых случайных помех. Он может быть охарактеризован некоторой условной плотностью распределения $p_{Y|X}(y|x)$, характеризующей многомерное распределение сигнала на его выходе при условии подачи на его вход последовательности сигналов x. Задачей приемника, состоящего из демодулятора и канального декодера, является нахождение $\widehat{u} : \mathcal{Y}(C(\widehat{u})) \approx y$, где y — сигнал, наблюдаемый на выходе канала. Смысл отношения $\approx$ будет пояснен в разд. 1.3.2. Демодулятор преобразует принимаемые из канала сигналы в форму, пригодную для использования декодером. В простейшем случае это могут быть оценки для каждого из символов c_i, а в наиболее общем — вероятности равенства c_i всем возможным значениям при условии наблюдения части принятого сигнала, соответствующего этому символу. С учетом этого под каналом передачи информации иногда бывает удобно понимать также и комбинацию модулятора, собственно канала и демодулятора.

Канальный декодер осуществляет собственно нахождение $\widehat{u}$. После проверки его правильности, обработки и удаления служебной информации осуществляется дешифрование и декомпрессия данных. В некоторых случаях приемник и/или передатчик могут анализировать состояние канала, динамически подстраивая свои параметры. Кроме того, иногда существует возможность организации канала обратной связи, который, например, может использоваться для запроса повторной передачи в случае неудачной попытки нахождения $\widehat{u}$.

Предметом данной книги являются вопросы построения канальных кодера и декодера. *Задачей теории кодирования является разработка математических методов внесения избыточности в данные, а также алгоритмов, использующих эту избыточность с целью восстановления сообщений, подвергшихся случайным искажениям в результате передачи по зашумленному каналу.*

1.1.2. Стеганография и защита цифровых материалов

Методы помехоустойчивого кодирования могут быть использованы для сокрытия факта передачи данных, т.е. незаметного внедрения их в некоторый объект (контейнер), который может быть передан через открытые каналы, не вызывая при этом подозрений у лиц, заинтересованных в перехвате сообщения. Очевидно, что для реализации такого внедрения необходимо минимизировать изменения, которым подвергается контейнер. Синдромное кодирование [26] позволяет решить эту проблему путем представления сообщения как синдрома некоторого зашумленного вектора. Результат декодирования этого синдрома в некотором линейном коде приводит к вектору, показывающему те изменения, которые необходимо внести в части контейнера, выбранные для внедрения информации (например, наименее значащие биты пикселов изображения), чтобы они задавали кодируемые данные, причем количество таких изменений минимизируется.

Кроме того, методы помехоустойчивого кодирования могут использоваться для выявления путей распространения нелегальных копий цифровых материалов. Для этого каждому легальному их приобретателю должна выдаваться копия, в которую скрытно внедрена метка, идентифицирующая его. Однако несколько недобросовестных приобретателей могут сравнить свои копии, выявить их отличия и, предполагая, что именно в различающихся позициях содержится идентифицирующая их информация, изготовить новую копию, в которой соответствующие сим-

волы заменены случайными или средними значениями, вычисленными по имеющимся копиям. Вообще говоря, это приводит к разрушению меток и невозможности идентификации авторов построенной таким образом пиратской копии. Но если внедренные метки являются кодовыми словами соответствующим образом подобранного кода, возникает возможность алгебраическими методами построить список возможных нарушителей авторских прав [32].

1.1.3. Поисковые системы

Коды, исправляющие ошибки, могут быть использованы для повышения надежности автоматической классификации данных. Для этого различным категориям данных могут быть сопоставлены различные кодовые слова, после чего построены распознаватели, оценивающие каждый символ кодовых слов по известным признакам классифицируемых объектов. Некоторые из этих оценок могут быть неправильными, но при надлежащем выборе кодовых слов существует возможность исправления этих ошибок и повышения точности классификации [80].

1.2. Модели каналов передачи информации

1.2.1. Понятие канала

Очевидно, что для решения задачи, сформулированной в разд. 1.1.1, необходимо иметь некоторую модель канала передачи данных. Всякий *канал передачи информации* характеризуется своим входным и выходным алфавитами $\mathbb{X}$ и $\mathbb{Y}$, а также условным распределением $P_{Y^m|X^n}\{y|x\}$ (или плотностью распределения $p_{Y^m|X^n}(y|x)$ в случае непрерывного алфавита), $y \in \mathbb{Y}^m, x \in \mathbb{X}^n$. Здесь X и Y обозначают случайные величины на входе и выходе канала, x и y — их значения, m и n — длины последовательностей. Далее, если не указано иное, будут рассматриваться *каналы без памяти*, для которых выполняется

$$\forall m : P_{Y^m|X^m}\{y_1, \ldots, y_m|x_1, \ldots, x_m\} = \prod_{i=1}^{m} P_{Y|X}\{y_i|x_i\},$$

т.е. прохождение каждого символа по нему осуществляется независимо. *Дискретным каналом* называют канал, у которого как $\mathbb{X}$, так и $\mathbb{Y}$ являются конечными множествами. Более общая модель *дискретного по времени канала* характеризуется произвольными входным и выходным алфавитами, т.е. предполагается лишь, что передаваемое и принимаемое сообщения могут быть разбиты на некоторые буквы, каждая из которых

передается в свой интервал времени. Наибольший практический интерес представляет модель *полунепрерывного канала*, характеризующаяся конечным входным алфавитом и $\mathbb{Y} = \mathbb{R}$.

В случае дискретных алфавитов распределение $P_{Y|X}\{y|x\}$ задается в виде матрицы $\Pi_{Y|X}$ переходных вероятностей, в которой строка x и столбец y содержат $P_{Y|X}\{y|x\}$. Очевидно, что сумма строк матрицы переходных вероятностей всегда равна вектору, состоящему из всех единиц. Дискретный канал называется *симметричным по входу*, если все строки матрицы переходных вероятностей являются перестановками первой строки. *симметричным по выходу*, если все столбцы матрицы переходных вероятностей являются перестановками первого столбца[1]. Дискретный канал называется *полностью симметричным*, если он симметричен по входу и выходу.

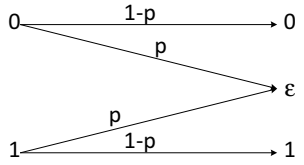
1.2.2. q-ичный стирающий канал

Простейшей моделью является *q-ичный стирающий канал*. Его входным алфавитом является $\mathbb{X} = \{0, \ldots, q-1\}$, а выходным — $\mathbb{Y} = \{0, \ldots, q-1, \epsilon\}$. Появление символа ϵ, называемого *стиранием*, на выходе канала означает безвозвратную потерю соответствующего передававшегося символа. Матрица переходных вероятностей канала имеет вид $\Pi_{Y|X} = \begin{pmatrix} 1-p & 0 & \ldots & 0 & p \\ 0 & 1-p & \ldots & 0 & p \\ \vdots & \vdots & \ddots & \ldots & \vdots \\ 0 & 0 & \ldots & 1-p & p \end{pmatrix}$, где p — вероятность стирания. Графическая модель канала при $q = 2$ представлена на рис. 1.2, а.

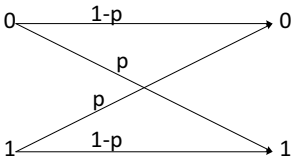
Данная модель может использоваться для описания IP-сетей, в которых, как известно, могут происходить потери пакетов. Несложно включить в пакеты, передаваемые по сети, их порядковый номер. В этом случае потеря пакета может быть легко обнаружена и интерпретирована как стирание. Заметим, что в рамках данной модели символы (например, пакеты), доставленные получателю, считаются безошибочными. Эта модель описывает также избыточные массивы жестких дисков (RAID-5,6), в которых факт отказа любого из дисков может быть легко обнаружен.

Ввиду простоты модели двоичный стирающий канал стал первым, для которого были предложены коды, достигающие его пропускной способности [49, 63]. В настоящее время эти коды нашли применение в си-

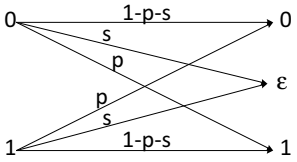
[1]К сожалению, в литературе иногда симметричный по входу канал называют симметричным по выходу и наоборот.



(a)



(b)



(c)

Рис. 1.2. Дискретные симметричные каналы:
a — двоичный стирающий канал; b — двоичный симметричный канал; c — двоичный симметричный канал со стираниями

стемах мобильной связи стандарта LTE Advanced [76] и цифровом телевидении [25].

1.2.3. q-ичный симметричный канал

Модель *q-ичного симметричного канала* предполагает $\mathbb{X} = \mathbb{Y} = \{0, \ldots, q-1\}$, причем строки матрицы переходных вероятностей являются перестановками друг друга. Это же справедливо и для ее столбцов. Данная модель характеризует каналы, искажения в которых не зависят от передаваемых данных. Как правило, рассматривается случай $q = p^m$, где p — простое число. В этом случае существует конечное поле $\mathbb{F}_q$, и выход канала может быть представлен как $Y = X + E$, где X — передаваемый символ, E — случайная величина, описывающая возможные искажения. Вероятность $p = P\{E \neq 0\}$ называют вероятностью ошибки. Наибольший интерес представляет случай $q = 2$ (см. рис. 1.2, b). Для него матрица переходных вероятностей имеет вид $\Pi_{Y|X} = \begin{pmatrix} 1-p & p \\ p & 1-p \end{pmatrix}$. В этом случае величину p называют переходной вероятностью канала или вероятностью ошибки. Данная модель в течение многих лет была основной в исследованиях по теории кодирования.

Представляет интерес также модель *q-ичного симметричного канала со стираниями*. Она характеризуется выходным алфавитом канала

$\mathbb{Y} = \mathbb{X} \cup \{\epsilon\}$. Соответствующая матрица переходных вероятностей в случае двоичного входа имеет вид $\Pi_{Y|X} = \begin{pmatrix} 1-p-s & p & s \\ p & 1-p-s & s \end{pmatrix}$, где s — вероятность стирания. Рис. 1.2, с иллюстрирует случай двоичного симметричного канала со стираниями. Стирание может искусственно вводиться приемником в том случае, если соответствующий символ был принят крайне ненадежно. Эта информация может помочь декодеру исправить ошибки в принятой последовательности, так как ему не приходится искать местоположение некоторых из них.

1.2.4. Аддитивный гауссовский канал

Намного большую практическую ценность имеет модель *аддитивного гауссовского канала* или канала с аддитивным белым гауссовским шумом. Она характеризуется некоторым, как правило, конечным входным алфавитом $\mathbb{X} \subset \mathbb{R}$ и выходным алфавитом $\mathbb{Y} = \mathbb{R}$. Принятый сигнал представляется как $Y = X + Z, X \in \mathbb{X}, Z \sim \mathcal{N}(0, \sigma^2)$. Таким образом, $p_{Y|X}(y|x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-x)^2}{2\sigma^2}}$. Заметим, что различные реализации случайной величины Z (отсчеты шумового процесса) предполагаются независимыми. Принято считать, что Z является вещественной частью комплекснозначного гауссовского случайного процесса со спектральной плотностью мощности N_0. Из этого вытекает, что $\sigma^2 = N_0/2$. Пусть $E_s = \mathbf{M}\left[|X|^2\right]$. Тогда на передачу одного символа в среднем затрачивается энергия[2] E_s. Таким образом, качество канала может быть охарактеризовано *отношением сигнал/шум на символ* E_s/N_0.

Зачастую возникает необходимость сопоставления характеристик различных методов кодирования и модуляции. Очевидно, что чем большую избыточность вносит передатчик, тем, как правило, большую помехозащищенность имеет система. Предельным случаем является m-кратное повторение каждого передаваемого символа. Однако такой подход эквивалентен m-кратному увеличению энергозатрат на передачу одного символа. Поэтому сравнение обычно производят при фиксированном *отношении сигнал/шум на бит* $\frac{E_b}{N_0} = \frac{E_s}{RN_0}$, где R — среднее количество битов полезной информации, кодируемое одним символом из $\mathbb{X}$. Отношение сигнал/шум принято выражать в децибелах как ОСШ $= 10 \lg(E_b/N_0)$[дБ].

[2]Реальные энергозатраты передатчика пропорциональны этой величине и определяются схемотехническими решениями.

Традиционный подход к построению систем передачи информации предполагает, что демодулятор осуществляет *жесткое детектирование* принятого сигнала, т.е. находит наиболее вероятное значение $\widehat{x} \in \mathbb{X}$, соответствующее принятому сигналу y. В разд. 1.3.2 будет показано, что оптимальным правилом является выбор в качестве $\widehat{x}$ ближайшего к y элемента $\mathbb{X}$. Предположим, что $\mathbb{X} = \left\{-\sqrt{E_s}, \sqrt{E_s}\right\}$ и передавался символ $x = \sqrt{E_s}$. Тогда детектирование будет ошибочным, если принятый сигнал y окажется отрицательным. Вероятность этого события равна

$$\begin{aligned} P_{AWGN} &= P\left\{Y < 0 | X = \sqrt{E_s}\right\} = \int_{-\infty}^{0} p_{Y|X}(y|\sqrt{E_s})dy = \\ &= \int_{\sqrt{E_s}}^{\infty} \frac{1}{\sqrt{\pi N_0}} e^{-\frac{y^2}{N_0}} dy = Q(\sqrt{2E_s/N_0}), \end{aligned} \tag{1.1}$$

где $Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^{\infty} e^{-t^2/2} dt$. Аналогичное выражение получается и для случая $x = -\sqrt{E_s}, \widehat{x} = \sqrt{E_s}$. Таким образом, аддитивный гауссовский канал в сочетании с процедурой жесткого детектирования может быть представлен как двоичный симметричный канал с переходной вероятностью P_{AWGN}. Величины $\widehat{x}$ являются исходными данными для декодера, который может реализовывать некоторый алгоритм *жесткого декодирования*.

Недостатком жесткого детектирования является потеря информации о надежности решений демодулятора. В связи с этим в современных системах, как правило, используется *мягкое детектирование*. Оно состоит в вычислении для каждого символа $P\{X = x | Y = y\}$. В случае канала с АБГШ и $P\left\{X = \sqrt{E_s}\right\} = P\left\{X = -\sqrt{E_s}\right\} = 1/2$ эти вероятности удобно представить в виде *логарифмического отношения правдоподобия* $L = \ln \frac{P\left\{X=\sqrt{E_s}|Y=y\right\}}{P\left\{X=-\sqrt{E_s}|Y=y\right\}}$. Воспользовавшись теоремой Байеса, получим

$$L = \ln \frac{p_{Y|X}(y|\sqrt{E_s})}{p_{Y|X}(y|-\sqrt{E_s})} = \frac{4\sqrt{E_s}}{N_0} y. \tag{1.2}$$

Полагая $E_s = 1, \sigma^2 = N_0/2$, получим $L = \frac{2}{\sigma^2} y$. Алгоритмы декодирования, способные полностью учитывать эту информацию, носят название *алгоритмов мягкого декодирования*.

Построение мягкого детектора и, в особенности, мягкого декодера, является нетривиальной задачей. Иногда удается построить декодер, способный лишь исправлять ошибки и стирания. В этом случае можно преобразовать аддитивный гауссовский канал с $\mathbb{X} = \left\{\pm\sqrt{E_s}\right\}$ в двоич-

ный симметричный канал со стираниями, заменив ими особо ненадежные принятые символы, т.е.

$$\widehat{x} = \begin{cases} 1 & L > \delta \\ \epsilon & |L| \leq \delta \\ 0 & L < -\delta \end{cases},$$

где δ — некоторый порог.

1.3. Кодирование и декодирование

1.3.1. Виды канального кодирования

Основным принципом канального кодирования является отказ от независимой передачи по каналу символов, порождаемых источником (информационных символов). Канальный кодер формирует последовательность, каждый элемент которой статистически связан со многими информационными символами. Для простоты будем считать, что как информационные, так и закодированные символы принадлежат одному и тому же алфавиту $\mathbb{X}$. В зависимости от способа построения и свойств кодовой последовательности выделяют следующие классы кодов:

- *Блоковый код* $\mathcal{C}$ является некоторым подмножеством векторного пространства $\mathbb{X}^n$, элементы которого называются кодовыми словами. Операция кодирования состоит в разбиении последовательности информационных символов на блоки фиксированной длины k, каждому из которых сопоставляется строго определенное кодовое слово. При этом кодирование информационных блоков осуществляется независимо друг от друга. Под *скоростью кода* понимается отношение $R = k/n$.
- *Древовидный код* представляет собой множество последовательностей над $\mathbb{X}$ бесконечной длины, на которое отображается множество бесконечных последовательности информационных символов. Кодер за один интервал времени отображает k_0 информационных символов в n_0 закодированных символов, причем результат этой операции зависит от предыстории. Таким образом, скорость кода равна $R = k_0/n_0$. При практической реализации древовидных кодов информационная и кодовая последовательности усекаются до конечной длины, что позволяет их рассматривать и как блоковые коды. *Сверточные коды* являются важнейшим подклассом древовидных кодов.

- *Бесскоростной* или *фонтанный* код представляет собой множество последовательностей над $\mathbb{X}$ бесконечной длины, которые однозначным образом получаются из векторов информационных символов конечной длины k.

При практической реализации как древовидных, так и фонтанных кодов кодовые последовательности некоторым образом усекаются до конечной длины.

1.3.2. Критерии декодирования

Напомним, что задачей декодера является нахождение оценки $\widehat{u}$ сообщения, закодированного кодером и подвергшегося случайным искажениям в канале. В случае цифровых сообщений естественным критерием качества декодирования является вероятность ошибки, т.е. несовпадения оценки и истинного сообщения u. Вероятность такого события равна $P_c = \sum_{u' \neq \widehat{u}} P_{U|Y}\{u'|y\} = 1 - P_{U|Y}\{\widehat{u}|y\}$, где y — последовательность принятых символов. Очевидно, что минимальная вероятность ошибки достигается при выборе

$$\widehat{u} = \arg\max_u P_{U|Y}\{u|y\}.$$

Данное правило принятия решения носит название *критерия идеального наблюдателя*. Из теоремы Байеса вытекает, что[3] $P_{U|Y}\{u|y\} = \frac{P_{Y|U}\{y|u\}P_U\{u\}}{P_Y\{y\}}$. Знаменатель этой дроби от u не зависит и может не учитываться при максимизации. Кроме того, при наличии качественно реализованного кодера источника все сообщения могут считаться равновероятными. В этом случае критерий идеального наблюдателя эквивалентен *критерию максимума правдоподобия*

$$\widehat{u} = \arg\max_u P_{Y|U}\{y|u\}. \tag{1.3}$$

В дальнейшем, если не указано иное, будет рассматриваться именно он.

Вышеуказанные критерии могут применяться не только для оценки всего сообщения u (*поблоковое декодирование*), но и отдельных его символов u_i (*посимвольное декодирование*). В общем случае при этом оценка всего сообщения $\widehat{u}$ может не совпадать с вектором посимвольных оценок $(\widehat{u}_1, \ldots, \widehat{u}_k)$. Иногда возникает необходимость найти надежность оценок для каждого из символов. В случае $\mathbb{U} = \{0, 1\}$ эту информацию

[3]Заметим, что в случае полунепрерывного канала вероятности получения конкретной последовательности y, как правило, равны нулю. Для разрешения неопределенности необходимо перейти к условным плотностям распределения.

удобно представить в виде *логарифмических отношений правдоподобия*

$$\ln \frac{P\{u_i = 1|y\}}{P\{u_i = 0|y\}} = \ln \frac{\sum_{u:u_i=1} P_{U|Y}\{u|y\}}{\sum_{u:u_i=0} P_{U|Y}\{u|y\}}. \tag{1.4}$$

В некоторых случаях может представлять интерес нахождение не единственной оценки $\widehat{u}$, максимизирующей $P_{U|Y}\{u|y\}$, а всех возможных u, для которых эта величина является достаточно большой (*списочное декодирование*). В дальнейшем можно воспользоваться дополнительной информацией (например, сведениями о структуре сообщения u) для того, чтобы выбрать наиболее подходящий элемент этого списка.

1.3.3. Метрики

Операция кодирования задает взаимно однозначное соответствие между возможными сообщениями и кодовыми словами используемого кода $\mathcal{C}$. Это позволяет переписать (1.3) как $\widehat{c} = \arg\max_{c\in\mathcal{C}} P_{Y^n|X^n}\{y_1, \ldots, y_n|c_1, \ldots, c_n\}$, где $(y_1, \ldots, y_n)$ — последовательность сигналов, возникшая на выходе канала в результате передачи кодового слова (возможно, бесконечного) $(c_1, \ldots, c_n)$. В случае канала без памяти это выражение может быть преобразовано далее в $\widehat{c} = \arg\max_{c\in\mathcal{C}} \prod_{i=1}^{n} P_{Y|X}\{y_i|c_i\}$. Во многих случаях вычисления с условными вероятностями оказываются неудобными. Попытаемся свести задачу декодирования по максимуму правдоподобия к задаче поиска кодового слова, ближайшего к принятой последовательности в некоторой метрике, т.е. *критерию минимального расстояния*.

Определение 1.1. Функция $d(x, y) : \mathbb{Y}^n \times \mathbb{Y}^n \to \mathbb{R}_+$ называется *метрикой*, если для всех $x, y, z \in \mathbb{Y}^n$:

1. $d(y, y) = 0$;
2. $d(x, y) = d(y, x)$;
3. $d(x, y) \leq d(x, z) + d(z, y)$ (неравенство треугольника).

В случае двоичного симметричного канала $P\{y_i|c_i\} = p^{\delta(y_i,c_i)}(1-p)^{1-\delta(y_i,c_i)} = \left(\frac{p}{1-p}\right)^{\delta(y_i,c_i)}(1-p)$, где $\delta(y_i, c_i)$ равно 1, если $y_i \neq c_i$, и 0 в противном случае. Если переходная вероятность канала p не превосходит $1/2$, получим, что $\widehat{c} = \arg\min_{c\in\mathcal{C}} d_H(y, c)$, где $d_H(y, c) = \sum_{i=1}^{n} \delta(y_i, c_i)$ — *расстояние Хэмминга* между последовательностями y и c, т.е. число позиций, в которых они отличаются. Критерий минимального расстояния Хэмминга применяется и в случае q-ичного симметричного канала при $q > 2$. Однако он эквивалентен критерию максимума правдоподобия,

только если все возможные ошибки $y_i - c_i$ распределены равномерно на множестве $\mathbb{Y} \setminus \{0\}$.

В случае аддитивного гауссовского канала $p_{Y|X}(y_i|c_i) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i-c_i)^2}{2\sigma^2}}$, откуда вытекает, что критерий максимума правдоподобия эквивалентен $\widehat{c} = \arg\min_{c\in\mathcal{C}} d_E(y,c)$, где $d_E(y,c) = \sqrt{\sum_{i=1}^{n}(y_i - c_i)^2}$ — *расстояние Евклида* между y и c.

Алгоритмы, позволяющие всегда найти точное решение оптимизационной задачи $\widehat{c} = \arg\min_{c\in\mathcal{C}} d_*(y,c)$ для некоторой метрики d_*, называются *алгоритмами полного декодирования*. Как правило, они имеют достаточно высокую сложность. В связи с этим на практике часто используют *алгоритмы декодирования с ограниченным расстоянием*, которые находят решение этой задачи, только если $d_*(y,\widehat{c}) < d_0$ для некоторого параметра d_0.

Несложно убедиться в том, что расстояния Хэмминга и Евклида являются метриками. Из неравенства треугольника вытекает, что если какие-то два кодовых слова расположены близко друг к другу, то расстояния между ними и принятой последовательностью также примерно равны. Если одно из них является правильным решением задачи декодирования, это означает, что вероятность ошибки может быть сопоставима с вероятностью правильного решения. Таким образом, при построении кодов необходимо обеспечить достаточно большое минимальное расстояние между кодовыми словами.

1.3.4. Параметр Бхаттачарьи

Рассмотрим канал без памяти с двоичным входом, передачу по нему равновероятных символов без кодирования и приемник по максиму-

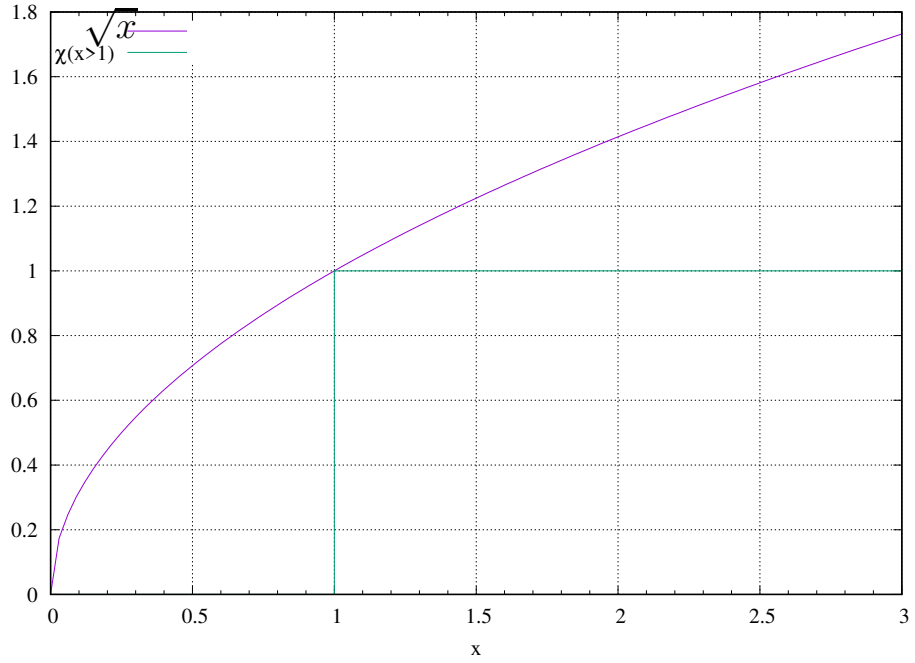


Рис. 1.3. Аппроксимация индикаторной функции

му правдоподобия. Попытаемся оценить вероятность ошибки. Она равна

$$
\begin{aligned}
P_e &= \sum_{y\in\mathbb{Y}} P\{e|y\}\, P_Y\{y\} = \sum_{x=0}^{1} P_X\{x\} \sum_{y\in\mathbb{Y}} P_{Y|X}\{y|x\}\, P\{e|y\} = \\
&= \sum_{x=0}^{1} \frac{1}{2} \sum_{y\in\mathbb{Y}} P_{Y|X}\{y|x\}\, \chi\left(P_{Y|X}\{y|\overline{x}\} > P_{Y|X}\{y|x\}\right) = \\
&= \sum_{x=0}^{1} \frac{1}{2} \sum_{y\in\mathbb{Y}} P_{Y|X}\{y|x\}\, \chi\left(\frac{P_{Y|X}\{y|\overline{x}\}}{P_{Y|X}\{y|x\}} > 1\right) \leq \\
&\leq \sum_{x=0}^{1} \frac{1}{2} \sum_{y\in\mathbb{Y}} P_{Y|X}\{y|x\} \sqrt{\frac{P_{Y|X}\{y|\overline{x}\}}{P_{Y|X}\{y|x\}}}.
\end{aligned}
$$

Здесь $\chi(A)$ — индикаторная функция, которая равна 1, если условие A выполняется, и 0 в противном случае. На рис. 1.3 представлен график индикаторной функции события $x > 1$. Видно, что эту функцию можно оценить сверху как $\sqrt{x}$, что и было использовано при получении последнего неравенства. Таким образом, $P_e \leq Z(P)$, где

$$
Z(P) = \sum_{y\in\mathbb{Y}} \sqrt{P_{Y|X}\{y|0\}\, P_{Y|X}\{y|1\}} \tag{1.5}
$$

— *параметр Бхаттачарьи* канала P.

1.4. Теоретико-информационные предпосылки

1.4.1. Теоремы кодирования

Рассмотрим две случайные величины X и Y. Если в результате испытания оказалось, что $Y = y$, то это может изменить вероятность появления $X = x$ от априорного значения $P_X\{x\}$ до апостериорного значения $P_{X|Y}\{x|y\}$. Меру этой зависимости удобно охарактеризовать величиной $I_{X;Y}(x;y) = \log_2 \frac{P_{X|Y}\{x|y\}}{P_X\{x\}} = \log_2 \frac{P_{XY}\{x,y\}}{P_X\{x\}P_Y\{y\}} = I_{Y;X}(y;x)$, которая равна количеству информации о значении X, даваемому событием $Y = y$. Усредняя ее по всем x и y, получим среднюю взаимную информацию[4]

$$I(X;Y) = \sum_{x\in\mathbb{X}}\sum_{y\in\mathbb{Y}} P_{XY}\{x,y\}\log_2\frac{P_{XY}\{x,y\}}{P_X\{x\}\,P_Y\{y\}} =$$

$$= \sum_{x\in\mathbb{X}}\sum_{y\in\mathbb{Y}} P_X\{x\}\,P_{Y|X}\{y|x\}\log_2\frac{P_{Y|X}\{y|x\}}{\sum_{x\in X} P_X\{x\}\,P_{Y|X}\{y|x\}}. \qquad (1.6)$$

Необходимо отметить, что эта величина характеризует случайные величины, а не их значения. Интуитивно понятно, что чем слабее связан выход канала передачи информации с его входом, тем меньше шансов на то, что информацию удастся достоверно передать через него.

Определение 1.2. *Пропускной способностью* дискретного канала передачи информации, заданного условным распределением $P_{Y|X}\{y|x\}$, называется величина

$$C = \max_{P_X\{x\}} I(X;Y).$$

Пропускная способность канала измеряется в битах полезной информации на один символ, передаваемый по каналу. Для случая дискретных по времени каналов связи пропускная способность определяется аналогичным образом, но при оптимизации по входным распределениям возникает необходимость наложить дополнительные ограничения на передаваемый сигнал X в виде $\mathbf{M}[f(X)] \le m$, где $f(x)$ — некоторая функция. Выбирая $f(x) = x^2$, получим ограничение на среднюю мощность передаваемого сигнала. Ниже будет показано, что существуют методы кодирования данных, обеспечивающие надежную связь со скоростью, не превышающей пропускную способность канала. Для ее достижения, однако, требуется оптимизировать распределение передаваемых символов.

[4] Для непрерывных случайных величин вероятности заменяются плотностями распределения, а сумма — интегралом.

В тех случаях, когда распределение задано и не подлежит оптимизации, пропускная способность может быть оценена просто как $C = I(X;Y)$.

Теорема 1.1 (Обратная теорема кодирования [3]). *Пусть дан дискретный стационарный источник с алфавитом объема $|\mathbb{U}| = M$ и энтропией*[5] *H. При кодировании его блоковым кодом со скоростью R для любого размера информационного блока k и передаче по дискретному каналу без памяти средняя вероятность ошибки P_b на один продекодированный информационный символ удовлетворяет неравенству*

$$P_b \log_2(M-1) + \mathcal{H}(P_b) \geq H - \frac{C}{R},$$

где $\mathcal{H}(x) = -x \log_2 x - (1-x)\log_2(1-x)$.

Аналогичное утверждение справедливо и для дискретных по времени каналов без памяти. Данная теорема показывает невозможность обеспечения надежной связи со скоростью, превышающей пропускную способность канала.

Для того, чтобы показать положительный результат для случая скоростей, меньших пропускных способностей канала, приходится рассматривать всевозможные коды, состоящие из случайно выбранных кодовых слов, и усреднять вероятность ошибки декодирования по всем таким кодам.

Теорема 1.2 (Прямая теорема кодирования, [3]). *Пусть $P_{Y|X}\{y|x\}$ — переходные вероятности для некоторого дискретного канала без памяти. Рассмотрим ансамбль блоковых кодов $\mathcal{C} \subset \mathbb{X}^n$, каждый из которых содержит $M = e^{Rn}$ кодовых слов, причем их символы выбираются независимо в соответствии с распределением $P_X\{x\}$. Тогда для любого сообщения средняя по всем кодам этого ансамбля вероятность ошибочного декодирования*[6] *по критерию максимума правдоподобия удовлетворяет неравенству*

$$P_e \leq e^{-nE_r(R)}, \tag{1.7}$$

[5] Энтропией источника U называется $H = \lim_{k\to\infty} H_k(U)$, где $H_k(U) = \frac{1}{k}\sum_{u\in\mathbb{U}^k} -P_U\{u\}\log_2 P_U\{u\}$, $P_U\{u\}$ — вероятность порождения им последовательности символов u.

[6] Сравнивать величины P_b и P_e, вообще говоря, нельзя, так как выбор неправильного кодового слова может приводить к выбору различного числа неправильных информационных символов.

где $E_r(R) = \max_{0\le\rho\le 1} \max_{P_X}(E_0(\rho, P_X) - \rho R)$ *и*

$$E_0(\rho, P_X) = -\ln \sum_{y\in\mathbb{Y}} \left(\sum_{x\in\mathbb{X}} P_X\{x\} P_{Y|X}^{\frac{1}{1+\rho}}\{y|x\} \right)^{1+\rho},$$

причем $E_r(R) > 0$ *при всех* $R, 0 \le R < C$.

Таким образом, используя достаточно длинные случайные коды, можно обеспечить в среднем сколь угодно малую вероятность ошибки. Это, однако, не означает, что произвольно выбранный код обеспечит надежную помехозащиту. Большинство случайно формируемых кодов действительно являются хорошими, но некоторые из них могут быть и плохими, т.е. вероятность ошибки декодирования при их использовании может быть велика. Средняя по всем кодам вероятность ошибки оказывается достаточно малой (1.7), поэтому можно утверждать, что существуют коды, обеспечивающие сколь угодно малую вероятность ошибки при скорости передачи данных меньшей пропускной способности канала. Прямая теорема кодирования справедлива также для каналов с памятью и дискретных по времени каналов.

Заметим, что для получения кодов, достигающих пропускной способности канала, необходимо производить оптимизацию и по распределению их символов. В реальных системах связи далеко не всегда удается реализовать оптимальные распределения.

Выражение (1.7) характеризует вероятность ошибки, которая в принципе может быть достигнута с помощью некоторых кодов. Существует, однако, и фундаментальное ограничение снизу на эту величину.

Теорема 1.3 (*Граница сферической упаковки [3]*)**.** *Для любого кода длины* n*, содержащего* $M = e^{Rn}$ *кодовых слов, вероятность ошибки декодирования по максимуму правдоподобия в дискретном канале без памяти удовлетворяет неравенству*

$$P_e \ge e^{-n(E_{sp}(R-o_1(n))+o_2(n))}, \tag{1.8}$$

где

$$E_{sp} = \sup_{\rho>0} \left(\max_{p_X\{x\}} E_0(\rho, P_X) - \rho R \right),$$

$o_1(n) = \frac{\ln 8 + |\,\mathbb{X}\,| \ln n}{n}$, $o_2(n) = \frac{\ln 8}{n} + \sqrt{\frac{2}{n}} \ln \frac{e^2}{P_0}$ *и* P_0 *— наименьшая ненулевая переходная вероятность канала.*

1.4.2. Пропускная способность некоторых каналов

Двоичный стирающий канал

Рассмотрим в качестве примера задачу нахождения пропускной способности двоичного стирающего канала (binary erasure channel, BEC) с вероятностью стирания p. Пусть $P_X\{0\} = \pi = 1 - P_X\{1\}$. Тогда $I(X;Y) = \pi(1-p)\log_2\frac{\pi(1-p)}{\pi^2(1-p)} + (1-\pi)(1-p)\log_2\frac{(1-\pi)(1-p)}{(1-\pi)^2(1-p)} + \pi p\log_2\frac{\pi p}{\pi p} + (1-\pi)p\log_2\frac{(1-\pi)p}{(1-\pi)p} = (1-p)\left(-\pi\log_2\pi - (1-\pi)\log_2(1-\pi)\right)$. Максимизируя это выражение по π, получим $\pi = 1/2$ и

$$C_{BEC} = 1 - p. \tag{1.9}$$

В данном случае декодирование сводится к попытке найти кодовое слово, совпадающее с принятой последовательностью в нестертых позициях. Ошибка может произойти в том случае, если таких кодовых слов окажется несколько. Таким образом, теорема кодирования утверждает, что при кодировании k символов блоковым кодом длины n и передаче по двоичному стирающему каналу для успешного декодирования необходимо обеспечить $k < (1-p)n$. В этом случае среднее число нестертых позиций окажется меньше k.

Двоичный симметричный канал

Для двоичного симметричного канала (binary symmetric channel, BSC) с переходной вероятностью p преобразования, аналогичные вышеприведенным, приводят к выражению

$$C_{BSC} = 1 + p\log_2 p + (1-p)\log_2(1-p). \tag{1.10}$$

При этом оптимальным распределением символов на входе канала также оказывается равномерное.

Аддитивный гауссовский канал

Можно показать [3], что пропускная способность канала с аддитивным белым гауссовским шумом (additive white Gaussian noise channel, AWGN)

$$C_{AWGN} = \frac{1}{2}\log_2\left(1 + \frac{E_s}{\sigma^2}\right) \tag{1.11}$$

достигается при использовании кодовых слов, символы которых распределены по нормальному закону с нулевым матожиданием и дисперсией E_s, где E_s — средняя энергия на символ. Ясно, что при этом $\mathbb{X} = \mathbb{R}$. Однако на практике реализовать такое сигнальное множество достаточно сложно (хотя и возможно получить достаточно хорошее приближение,

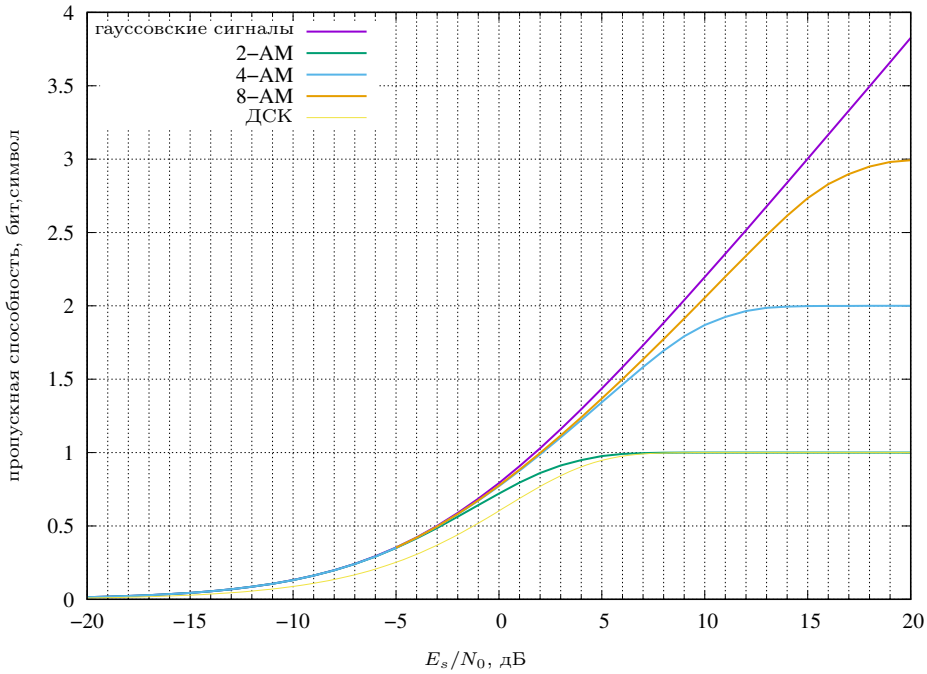


Рис. 1.4. Пропускная способность аддитивного гауссовского канала

см. [34]). В случае равномерного распределения передаваемых символов на M-элементном множестве $\mathbb{X} = \{X_i | 0 \leq i < M\}$ из (1.6) получим, что максимальная достижимая скорость передачи данных равна

$$C(\mathbb{X}) = \frac{1}{M} \sum_{i=0}^{M-1} \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-X_i)^2}{2\sigma^2}} \log_2 \frac{M e^{-\frac{(x-X_i)^2}{2\sigma^2}}}{\sum_{j=0}^{M-1} e^{-\frac{(x-X_j)^2}{2\sigma^2}}} dx.$$

На рис. 1.4 представлены графики зависимости пропускной способности от отношения сигнал/шум на символ для случая M-ичной амплитудно-импульсной модуляции, т.е. $\mathbb{X} = \left\{ (-M+1+2i)d | i \in \{0, 1, \ldots, M-1\}, d = \sqrt{\frac{3E_s}{M^2-1}} \right\}$. Кроме того, на рисунке представлена кривая пропускной способности двоичного симметричного канала, полученного как комбинация гауссовского канала и жесткого посимвольного детектора. Видно, что использование мягкого детектирования и декодирования позволяет значительно повысить пропускную способность в области малых отношений сигнал/шум. Кроме того, с ростом отношения сигнал/шум увеличивается проигрыш системы с равномерным распределением дискретных сигналов на входе по сравнению с оптимальной схемой, осуществляющей передачу с использованием нормально распределенных сигналов. Впрочем, он никогда не превосходит $\pi e/6$ или 1.53 дБ [34].

Упражнения

1. Покажите, что $I(X;Y) \geq 0$, причем $I(X;Y) = 0$ тогда и только тогда, когда X и Y являются независимыми случайными величинами.
2. Выведите формулу (1.11) и обоснуйте необходимость использования нормально распределенных сигналов для достижения пропускной способности аддитивного Гауссовского канала.
3. Постройте график, аналогичный рис. 1.4, для случая сигнального множества M-ичной фазовой и квадратурно-амплитудной модуляции.

2. Блоковые коды

2.1. Основные понятия

2.1.1. Параметры кодов

Блоковым кодом $\mathcal{C}$ называется любое подмножество векторного пространства $\mathbb{X}^n$. Элементы множества $\mathcal{C}$ называются кодовыми словами. Величина n называется длиной кода. Если код содержит $M = |\mathcal{C}|$ кодовых слов, то с помощью его кодовых слов можно представить $\log_2 M$ битов полезной информации.

Как было показано в разд. 1.3.3, вероятность ошибки декодирования зависит от расстояния между кодовыми словами. Поэтому важнейшей характеристикой кода является его минимальное расстояние

$$d = \min_{\substack{a,b\in\mathcal{C}\\ a\neq b}} d(a,b).$$

Предположим, что $\mathbb{Y} = \mathbb{X}$, и рассмотрим все элементы $\mathbb{X}^n$, находящиеся на расстоянии менее $d/2$ от кодовых слов $\mathcal{C}$. Очевидно, что никакие два шара, построенные таким образом, не пересекаются. Следовательно, если декодер использует критерий минимального расстояния и принятая последовательность y попала в один из этих шаров, решением задачи декодирования является кодовое слово, находящееся в ее центре. В случае метрики Хэмминга это означает, что код способен исправить любую конфигурацию из $\lfloor (d-1)/2 \rfloor$ ошибок. При этом, однако, некоторые векторы из $\mathbb{X}^n$ могут не попасть ни в один из шаров. Это не является проблемой для полноценного декодера, использующего критерий минимального расстояния, кроме, естественно, случая равноудаленности y от нескольких ближайших кодовых слов. К сожалению, многие алгоритмы декодирования реализуют лишь *декодирование с ограниченным расстоянием*, т.е. способны найти кодовые слова такие, что y находится в шарах радиуса r, описанных вокруг них. В большинстве случаев $r < d/2$, вследствие чего таких кодовых слов может быть не более одного. В противном случае результатом декодирования может быть список кодовых слов. Если декодеру не удалось найти ни одного кодового слова, достаточно близкого к y, фиксируется отказ от декодирования. Но даже если декодер возвратил единственное кодовое слово, невозможно гарантировать, что оно совпадает с тем, которое было передано на самом деле.

Декодирование с ограниченным расстоянием не позволяет достичь пропускной способности канала.

2.1.2. Границы

Рассмотрим некоторые фундаментальные ограничения на параметры блоковых кодов. Везде далее будет рассматриваться метрика Хэмминга и предполагаться $q = |\mathbb{X}|$.

Теорема 2.1 (*граница Хэмминга*). *Для любого q-ичного кода длины n с минимальным расстоянием $d = 2t+1$ число кодовых слов удовлетворяет неравенству*

$$A_q(n,d) \le \frac{q^n}{\sum_{i=0}^{t} C_n^i (q-1)^i}. \tag{2.1}$$

Доказательство. Код с минимальным расстоянием $d = 2t+1$ способен исправить любую конфигурацию из t ошибок, что означает возможность построения непересекающихся шаров радиуса t с центрами в кодовых словах. Каждый такой шар содержит $V = \sum_{i=0}^{t} C_n^i (q-1)^i$ q-ичных векторов. При этом общее число векторов в $\mathbb{X}^n$ не может быть меньше числа векторов, попавших в эти шары, т.е. $A_q(n,d)V$. □

Коды, для которых (2.1) выполняется с равенством, называются *совершенными*. Совершенный код при декодировании по минимальному расстоянию Хэмминга способен исправить любую конфигурацию из t ошибок и ни одной конфигурации большего веса. К сожалению, можно показать, что множество совершенных кодов исчерпывается примитивными кодами Хэмминга ($t = 1, n = \frac{q^r-1}{q-1}$ над $\mathbb{F}_q$), двоичными кодами с повторениями ($n = 2t+1$), двоичным ($n = 23, t = 3, 2^{12}$ кодовых слов) и троичным ($t = 2, n = 11, 3^6$ кодовых слов) кодами Голея.

Оценим правую часть (2.1) при больших n. Для этого запишем $A_2(n,d) \le \frac{2^n}{\sum_{i=0}^{\lfloor (d-1)/2 \rfloor} C_n^i} \le \frac{2^n}{C_n^{\lfloor (d-1)/2 \rfloor}} = \frac{2^n \lfloor (d-1)/2 \rfloor ! (n - \lfloor (d-1)/2 \rfloor)!}{n!}$. Воспользовавшись аппроксимацией Стирлинга $n! = 2^{n \log_2 n + o(1)}$, получим $A_2(n,d) \approx 2^{n + \frac{d-1}{2} \log_2 \frac{d-1}{2} + (n - \frac{d-1}{2}) \log_2 (n - \frac{d-1}{2}) - n \log_2 n} = 2^{n(1 + \frac{d-1}{2n} \log_2 \frac{d-1}{2n} + (1 - \frac{d-1}{2n}) \log_2 (1 - \frac{d-1}{2n}))}$. Таким образом,

$$R = \frac{\log_2 A_2(n,d)}{n} \le 1 - \mathcal{H}\left(\frac{d}{2n}\right) \tag{2.2}$$

Теорема 2.2 (*граница Варшамова-Гилберта*). *Существует q-ичный код длины n с минимальным расстоянием d, число кодовых слов в ко-*

тором удовлетворяет условию

$$A_q(n,d) \geq \frac{q^n}{\sum_{i=0}^{d-1} C_n^i (q-1)^i}. \tag{2.3}$$

Доказательство. В качестве доказательства попытаемся построить код $\mathcal{C}$, удовлетворяющий условию теоремы. Заметим, что если код имеет максимально возможную мощность, то для любого $x \notin \mathcal{C}$ существует $c \in \mathcal{C} : d_H(x,c) \leq d-1$, так как в противном случае такой вектор x можно было бы добавить в код без ущерба для его минимального расстояния. Введем множество возможных кодовых слов A и положим $A = \mathbb{X}, \mathcal{C} = \emptyset$. Будем выбирать по одному элементу c из A, добавлять его в $\mathcal{C}$ и исключать из A шар радиуса $d-1$ с центром в c до тех пор, пока в A не останется ни одного элемента. Видно, что на каждом шаге этой процедуры из A исключается не более $\sum_{i=0}^{d-1} C_n^i (q-1)^i$ векторов (некоторые из них попадают сразу в несколько таких шаров). Таким образом, минимальное число шагов, после которого алгоритм останавливается, удовлетворяет (2.3). □

Теорема 2.3 (*граница Синглтона*)**.** *Число кодовых слов в q-ичном коде длины n с минимальным расстоянием d удовлетворяет*

$$A_q(n,d) \leq q^{n-d+1}. \tag{2.4}$$

Доказательство. Рассмотрим множество q-ичных векторов, отличающихся друг от друга не менее, чем в d позициях. Выберем какие-нибудь $d-1$ координат и удалим их из всех этих векторов. Они останутся различными, поэтому их число не может превосходить $q^{n-(d-1)}$. □

2.2. Линейные коды

2.2.1. Основные понятия

Основной проблемой, возникающей при реализации кодера и декодера блоковых кодов, является необходимость хранить весь список кодовых слов, количество которых растет экспоненциально с ростом размера информационного блока. От этого недостатка свободны *линейные блоковые коды*.

Определение 2.1. Линейным блоковым кодом $\mathcal{C}$ называется k-мерное линейное подпространство n-мерного векторного пространства $\mathbb{X}^n$, где $\mathbb{X} = \mathbb{F}$ — некоторое поле.

Как известно, всякое линейное пространство имеет *базис*, т.е. такой линейно независимый набор элементов, что любой элемент этого пространства представим в виде линейной комбинации элементов базиса. Запишем элементы какого-либо базиса в виде строк $k \times n$ *порождающей матрицы* G. Тогда линейный код может быть представлен как $\mathcal{C} = \left\{ x = uG | u \in \mathbb{F}^k \right\}$. Таким образом, для задания кода достаточно указать лишь его порождающую матрицу, а операция кодирования сводится к умножению на нее, что радикально упрощает реализацию кодера.

Величина k называется размерностью кода. Скоростью кода называют величину $R = k/n$. Основные параметры (длину, размерность, минимальное расстояние) линейных блоковых кодов принято указывать в виде тройки (n, k, d). В тех случаях, когда минимальное расстояние неизвестно или не представляет интереса, используют запись (n, k). Очевидно, что число кодовых слов в линейном коде равно $q^k, q = |\,\mathbb{F}\,|$. Заметим, что $d = \min\limits_{\substack{a,b \in \mathcal{C} \\ a \neq b}} d_H(a, b) = \min\limits_{\substack{a,b \in \mathcal{C} \\ a \neq b}} \mathrm{wt}(a - b)$, где $\mathrm{wt}(c)$ — вес Хэмминга вектора c, т.е. число ненулевых элементов в нем. В силу линейности кода $a - b \in \mathcal{C}$. Это означает, что минимальное расстояние кода равно наименьшему из весов его ненулевых кодовых слов.

Из линейной алгебры известно, что всякому линейному векторному подпространству $\mathcal{C} \subset \mathbb{F}^n$ соответствует некоторое двойственное подпространство $\overline{\mathcal{C}} \subset \mathbb{F}^n$ такое, что скалярное произведение любых векторов $\forall c \in \mathcal{C}, \overline{c} \in \overline{\mathcal{C}}$ равно нулю. Выберем некоторый базис двойственного подпространства линейного блокового кода и запишем его элементы в виде строк $(n - k) \times n$ *проверочной матрицы* H. Это позволяет описать тот же самый линейный блоковый код как ее ядро, т.е. $\mathcal{C} = \ker H = \left\{ c \in \mathbb{F}^n \,|\, Hc^T = 0 \right\}$. Из этого также вытекает, что порождающая и проверочная матрицы связаны соотношением

$$HG^T = 0.$$

Заметим, что в дальнейшем иногда будет удобно рассматривать "избыточные" проверочные матрицы с числом строк, большим $n - k$, часть из которых при этом являются линейно зависимыми. Кроме того, будут использоваться также проверочные матрицы, содержащие элементы из некоторого расширения исходного поля $\mathbb{F}$. При этом будет неявно предполагаться, что код задан как $\ker H \cap \mathbb{F}^n$.

Теорема 2.4. *Если H — проверочная матрица кода длины n, то код имеет размерность k тогда и только тогда, когда существуют $n -$*

k линейно независимых столбцов матрицы H, а любые $n-k+1$ ее столбцов являются линейно зависимыми.

Доказательство. Утверждение теоремы эквивалентно тому, что ранг матрицы H равен $n-k$. Его справедливость вытекает из теоремы о структуре множества решений системы линейных однородных алгебраических уравнений. □

Теорема 2.5. *Линейный блоковый код имеет минимальное расстояние d тогда и только тогда, когда любые $d-1$ столбцов его проверочной матрицы линейно независимы, но при этом существуют d линейно зависимых столбцов.*

Доказательство. Вектор c является кодовым словом тогда и только тогда, когда $Hc^T = 0$, или $\sum_j H_{-,i_j} c_{i_j} = 0$, где $H_{-,l}$ обозначает l-ый столбец матрицы H, и сумма берется по всем $i_j : c_{i_j} \neq 0$. Таким образом, всякое ненулевое кодовое слово задает некоторую совокупность линейно зависимых столбцов матрицы H. □

Известно, что всякое линейное пространство имеет множество различных базисов. Из этого вытекает, что порождающая и проверочная матрицы не единственны. Все такие матрицы связаны друг с другом соотношениями $G' = UG$ и $H' = VH$, где U и V — обратимые матрицы, задающие преобразования базисов. Очевидно, что свойства и корректирующая способность кода никак не зависят от выбора порождающей и проверочной матрицы. Вместе с тем, рассматриваемые далее алгоритмы декодирования могут существенно опираться на свойства конкретной матрицы (обычно проверочной), использованной для его задания. С другой стороны, преобразование $G'' = GW$ в общем случае приводит к совершенно иному линейному блоковому коду с трудно предсказуемыми свойствами. Если матрица W содержит ровно по одному ненулевому элементу в каждой строке и каждом столбце (в двоичном случае — если W является перестановочной матрицей), код, порождаемый G'', имеет точно такую же корректирующую способность, как и исходный код, и называется *эквивалентным* ему. При этом в общем случае кодовые слова эквивалентного кода могут не являться кодовыми словами исходного кода.

С практической точки зрения представляет ценность такая реализация операции кодирования, которая позволяет получить кодовые слова, содержащие как подвектор закодированную информационную по-

следовательность. Соответствующий кодер носит название систематического. Для того, чтобы его построить, достаточно найти (например, с помощью метода Гаусса) обратимое линейное преобразование, задаваемое матрицей V, приводящее порождающую матрицу[1] к виду $\tilde{G} = VG = (I|A)$, где I — единичная матрица. Очевидно, что тогда $c = u\tilde{G} = (u|uA)$. Первые k символов такого кодового слова c называются информационными, а последние $(n-k)$ — проверочными. Соответствующая форма порождающей матрицы называется канонической. Она позволяет легко найти одну из проверочных матриц кода как $\tilde{H} = (-A^T|I)$. Несложно убедиться в том, что $\tilde{G}\tilde{H}^T = 0$. Еще раз отметим, что вероятность выбора неправильного кодового слова декодером не зависит[2] от того, какая порождающая матрица была использована кодером. Вместе с тем, для извлечения полезной информации из кодового слова, безусловно, знание порождающей матрицы необходимо. Кроме того, вероятность ошибки на информационный бит может зависеть от вида используемой порождающей матрицы.

Пример 2.1. Рассмотрим $(6,3,3)$ код над $\mathbb{F}_2$, задаваемый порождающей матрицей $G = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 \end{pmatrix}$. Применяя к ней метод Гаусса, получим следующую цепочку порождающих матриц этого же кода: $G \sim \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 \end{pmatrix} \sim \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix} \sim \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix} \sim \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}$. Последняя из них является канонической и позволяет получить проверочную матрицу как $H = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$. В этой матрице отсутствуют одинаковые столбцы, но суммы некоторых троек столбцов дают ноль. Таким образом, в силу теоремы 2.5, минимальное расстояние этого кода равно 3.

Проверочную матрицу линейного кода $\mathcal{C}$ можно рассматривать также как порождающую матрицу некоторого другого кода $\overline{\mathcal{C}}$, называемого *дуальным*. Очевидно, что $\overline{\mathcal{C}} = \ker G$ и $\forall c \in \mathcal{C}, \overline{c} \in \overline{\mathcal{C}} : c \cdot \overline{c} = 0$.

[1] Зачастую для получения матрицы указанного вида необходимо выполнить также некоторую перестановку столбцов, т.е. перейти к эквивалентному коду. Для упрощения обозначений этот случай здесь не рассматривается.

[2] Это свойство оказывается крайне полезным при отладке программной реализации алгоритмов декодирования.

Как будет показано ниже, при анализе корректирующей способности линейного кода необходимо знать не только его минимальное расстояние, но и весовой спектр $(A_0, \ldots, A_n)$, где A_i равно числу кодовых слов веса i. В общем случае нахождение весового спектра кода требует перебора всех его кодовых слов, т.е. имеет сложность $O(q^k n)$. Если $n - k < k$, эту процедуру можно упростить, вычислив весовой спектр дуального кода и воспользовавшись следующей теоремой.

Теорема 2.6 (*Мак-Вильямс [8]*). *Пусть дан двоичный линейный (n, k) код. Определим его весовую функцию как $W_{\mathcal{C}}(x, y) = \sum_{i=0}^{n} A_i x^{n-i} y^i$. Тогда*

$$W_{\overline{\mathcal{C}}}(x, y) = \frac{1}{2^k} W_{\mathcal{C}}(x + y, x - y).$$

2.2.2. Синдромное декодирование

Пусть кодовые слова линейного кода над $\mathbb{F}_q$ передаются по q-ичному симметричному каналу. Его выход может быть описан как

$$y = uG + e, \tag{2.5}$$

где G — порождающая матрица кода и e — вектор ошибки. Рассмотрим задачу декодирования вектора y по критерию минимального расстояния Хэмминга, т.е. поиск вектора e наименьшего веса, допускающего представление в виде (2.5).

Смежным классом линейного кода $\mathcal{C}$, задаваемым вектором z, называется множество $\mathcal{C} + z = \{c + z | c \in \mathcal{C}\}$. Смежные классы кода или не пересекаются, или совпадают. Действительно, если $y \in \mathcal{C} + z_1$ и $y \in \mathcal{C} + z_2$, то $y = c_1 + z_1 = c_2 + z_2, c_1, c_2 \in \mathcal{C}$, откуда следует, что $z_2 = (c_1 - c_2) + z_1 \in \mathcal{C} + z_1$, т.е. $\mathcal{C} + z_2 \subset \mathcal{C} + z_1$. Аналогичным образом показывается и обратное включение. С другой стороны, всякий вектор из $\mathbb{F}^n$ может быть отнесен к какому-либо смежному классу кода $\mathcal{C}$, т.е. семейство его смежных классов образует разбиение $\mathbb{F}^n$. Заметим, что все элементы фиксированного смежного класса $\mathcal{C} + z$ характеризуются одним и тем же значением *синдрома* $S = (c + z)H^T = zH^T$. Выпишем все возможные векторы $z \in \mathbb{F}^n$ в виде таблицы, размещая на одной строке элементы с одинаковым значением синдрома (т.е. элементы одного смежного класса). Будем считать, что первая строка соответствует $S = 0$, т.е. содержит исключительно кодовые слова рассматриваемого кода. Отсортируем все строки по весу. Будем называть лидером смежного класса с синдромом S вектор наименьшего веса в соответствующей стро-

ке таблицы. Полученная таблица носит название таблицы стандартной расстановки.

Пусть $S = yH^T = eH^T$ — синдром принятой последовательности. Заметим, что он зависит только от вектора ошибки e и не зависит от переденного кодового слова. Этот синдром может соответствовать любому вектору, находящемуся в соответствующем смежном классе кода $\mathcal{C}$. Так как рассматривается задача декодирования по минимуму расстояния Хэмминга, в качестве оценки $\widehat{e}$ может быть взят подходящий вектор наименьшего возможного веса, т.е. лидер смежного класса, соответствующего S. Заметим, что для некоторых векторов S может существовать несколько потенциальных лидеров смежного класса с наименьшим весом. Такого рода неопределенность может быть разрешена случайным образом, но, очевидно, с большой вероятностью последовательности с этими значениями синдрома будут декодироваться неправильно. Более целесообразно в таких случаях сообщать об отказе от декодирования. Коды, каждый смежный класс которых содержит единственный вектор наименьшего веса, называются *совершенными*.

Таким образом, декодирование линейных блоковых кодов сводится к построению таблицы соответствия синдромов и соответствующих им лидеров смежных классов и поиску в этой таблице синдрома принятого вектора. Для кода с минимальным расстоянием $d = 2t + 1$ размер L такой таблицы удовлетворяет[3] $\sum_{i=0}^{t} C_n^i (q-1)^i \leq L \leq q^{n-k}$. Для низкоскоростных кодов более эффективной стратегией может быть сравнение принятого вектора со всеми q^k кодовыми словами. Однако с ростом длины кода в общем случае сложность или затраты памяти для обоих методов растут экспоненциально, что приводит к необходимости поиска более эффективных алгоритмов.

Существует, однако, класс кодов, для которых процедура синдромного декодирования оказывается самой простой из возможных. Двоичный *код Хэмминга* определяется как линейный блоковый $(2^m - 1, 2^m - 1 - m, 3)$ код, столбцы проверочной матрицы которого представляют собой все различные двоичные векторы длины m. Действительно, в этом случае всякий столбец равен сумме двух других столбцов, и, в силу теоремы 2.5, минимальное расстояние кода равно 3. При реализации удобно рассматривать столбцы проверочной матрицы как двоичную запись некоторых целых чисел и упорядочить их в порядке возрастания. Тогда, если произошла одна ошибка, синдром $S = He^T$ равен номеру (в двоич-

[3]Исправимыми могут быть и некоторые конфигурации ошибок веса более t.

ной системе счисления) той позиции, которая была искажена. Непосредственной проверкой можно убедиться, что построенный таким образом код удовлетворяет границе Хэмминга (2.1) с равенством, т.е. является совершенным.

Пример 2.2. Проверочная матрица $(7, 4, 3)$ кода Хэмминга имеет вид

$$H = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}. \tag{2.6}$$

Синдром вектора $y = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$ равен $S = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}$, т.е. ошибка произошла в четвертой позиции. Исправленное кодовое слово равно $c = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$.

2.2.3. Границы

Может возникнуть подозрение, что упрощение реализации кодирования, обеспечиваемое при использовании линейных кодов, достигается за счет ухудшения их характеристик. К счастью, как будет показано в этом разделе, это не так.

Оказывается, что идея метода построения кода, использованного при доказательстве границы Варшамова-Гилберта для произвольных кодов, может быть адаптирована для линейных кодов и привести к лучшим результатам.

Теорема 2.7 (*Граница Варшамова-Гилберта*)**.** *Существует линейный* (n, k, d) *код, для которого*

$$A_q(n, d) = q^k \geq \frac{q^n}{\sum_{i=0}^{d-2} C_n^i (q-1)^i} \tag{2.7}$$

Доказательство. Попытаемся построить проверочную матрицу кода максимально возможной длины с минимальным расстоянием d и r проверочными символами. Первый ее столбец может быть выбран как произвольный ненулевой вектор. Если уже выбрано j столбцов, то, в соответствии с теоремой 2.5, в качестве очередного столбца нельзя использовать никакие линейные комбинации из $d-2$ или менее ранее выбранных столбцов, которых насчитывается $\sum_{i=0}^{d-2} C_j^i (q-1)^i$. Если эта величина превышает общее число возможных столбцов q^r, наращивание длины кода невозможно. Таким образом, если выполняется $q^r > \sum_{i=0}^{d-2} C_{n-1}^i (q-1)^i$, то существует q-ичный $(n = k + r, k, d)$ линейный код. Будем выполнять вышеописанные действия до тех пор, пока это неравенство не нарушится, т.е. не останется допустимых вариантов для столбца $n + 1$. Построенная

к этому моменту $(n-k) \times n$ проверочная матрица задает код с числом слов, удовлетворяющим (2.7). □

Заметим, что выражения (2.7) и (2.3) отличаются одним слагаемым в знаменателе, причем последнее гарантирует существование кода с большим числом кодовых слов, чем это удалось показать для случая произвольных кодов. Таким образом, существуют линейные коды с параметрами ничуть не хуже, чем у аналогичных нелинейных кодов.

Аналогично тому, как это было сделано в случае границы Хэмминга, можно получить асимптотическое выражение для границы Варшамова-Гилберта:

$$R = \frac{\log_2 A_2(n,d)}{n} \geq 1 - \mathcal{H}(d/n). \tag{2.8}$$

К линейным кодам применимы все те границы, которые были получены для произвольных блоковых кодов. Запись некоторых из них может быть упрощена. В частности, *граница Синглтона* может быть записана как

$$k \leq n - d + 1. \tag{2.9}$$

В данном случае это неравенство вытекает также из теоремы 2.5, так как ранг проверочной матрицы не может быть меньше $d-1$. Коды, удовлетворяющие (2.9) с равенством, называются *кодами с максимальным достижимым расстоянием*.

Следующая теорема дает намного более точную оценку для параметров двоичных линейных кодов.

Теорема 2.8 (*граница Грайсмера*)**.** *Пусть $N(k,d)$ — наименьшая длина линейного кода над $\mathbb{F}_q$ с размерностью k и минимальным расстоянием d. Тогда $N(k,d) \geq d + N(k-1, \lceil d/q \rceil)$.*

Доказательство. Доказательство приведем для случая $q = 2$. Предположим, что существует линейный $(N(k,d), k, d)$ код. Одна из его порождающих матриц содержит кодовое слово веса d на одной из строк. Поэтому, не ограничивая общности, будем считать, что

$$G = \begin{pmatrix} 0\,0\,\ldots\,0 & 1\,1\,\ldots\,1 \\ G_1 & G_2 \end{pmatrix},$$

где G_1 и G_2 — некоторые матрицы размерности $(k-1) \times (N(k,d)-d)$ и $(k-1) \times d$. Ранг матрицы G_1 равен $k-1$. Действительно, в противном случае с помощью операций над строками матрицы G ее можно было

бы привести к виду, в котором первая строка G_1 была бы нулевой. Тогда сумма первых двух строк преобразованной матрицы G имела бы вес, меньший d. Таким образом, G_1 может рассматриваться как порождающая матрица некоторого линейного $(N(k,d)-d, k-1, d_1)$ кода. Для того, чтобы оценить его минимальное расстояние d_1, заметим, что если вектор $(u|v)$, $u \neq 0$, принадлежит исходному коду, то $(u|(1,\ldots,1)+v)$ также принадлежит ему. При этом $\mathrm{wt}(u)+\mathrm{wt}(v) \geq d$ и $\mathrm{wt}(u)+d-\mathrm{wt}(v) \geq d$. Таким образом, $2\,\mathrm{wt}(u) \geq d$ и $d_1 \geq \lceil d/2 \rceil$, т.е. из существования $(N(k,d),k,d)$ кода вытекает существование $(N(k,d)-d, k-1, \lceil d/2 \rceil)$ кода. Возможно, что существуют и более короткие коды с размерностью $k-1$ и минимальным расстоянием $\lceil d/2 \rceil$, поэтому $N(k-1, \lceil d/2 \rceil) \leq N(k,d)-d$. □

Таким образом, граница Грайсмера может быть записана как

$$N(k,d) \geq \sum_{i=0}^{k-1} \left\lceil \frac{d}{q^i} \right\rceil . \tag{2.10}$$

Для двоичных кодов наиболее точные результаты дают границы *Мак-Элиса-Родемича-Рамсея-Велча*, которая имеет вид [8]

$$R \leq \mathcal{H}\left(\frac{1}{2} - \sqrt{\frac{d}{n}\left(1-\frac{d}{n}\right)}\right), 0 \leq d \leq n/2,$$

Бассалыго-Элайеса (см. [5])

$$R \leq 1 - \mathcal{H}\left(\frac{1-\sqrt{1-2d/n}}{2}\right),$$

а также граница линейного программирования [8], для которой в общем случае нет замкнутой формулы.

На рис. 2.1 представлены кривые асимптотических форм вышерассмотренных границ. Напомним, что граница Варшамова-Гилберта является нижней, т.е. постулирует существование кода со скоростью не ниже указанной. Все остальные границы являются верхними, т.е. не существует кодов со скоростью, превышающей наименьшую из этих величин. Вопрос о существовании кодов, параметры которых лежат в заштрихованной области, остается открытым[4].

[4]В работе [42] граница Варшамова-Гилберта была несколько улучшена. Вместе с тем, существуют алгебро-геометрические коды с параметрами, превосходящими границу Варшамова-Гилберта.

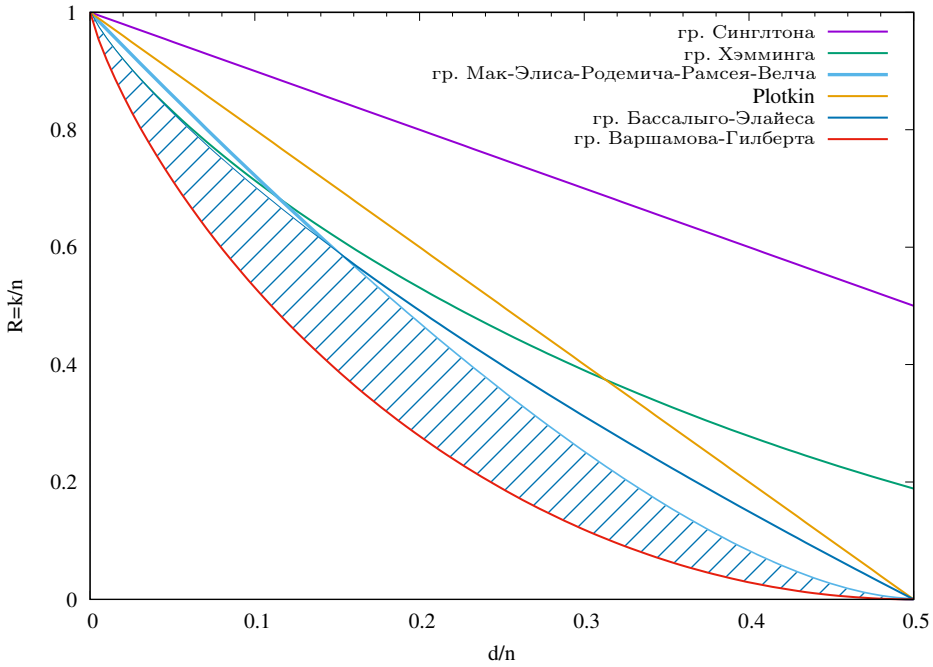


Рис. 2.1. Границы достижимой скорости двоичных линейных кодов

2.3. Анализ корректирующей способности

Всякий блоковый код и алгоритм его декодирования могут быть охарактеризованы следующими величинами:

1. Вероятностью ошибочного декодирования P_c (вероятность ошибки на блок, codeword error rate, frame error rate), т.е. вероятностью того, что информационное сообщение, восстановленное декодером, не совпадает с переданным.
2. Вероятностью битовой ошибки P_b (вероятность ошибки на бит, bit error rate), т.е. вероятностью того, что произвольно выбранный бит сообщения, восстановленного декодером, не совпадает с истинным значением. Необходимо отметить, что в случае ошибочного декодирования сообщения число искаженных информационных битов может быть самым разнообразным, причем события неправильного восстановления отдельных битов сообщения являются зависимыми.
3. Вероятностью обнаружения ошибки P_d, т.е. вероятностью выявления декодером факта несовпадения принятой и переданной последовательностей (без попытки их исправления).
4. Вероятностью необнаружения ошибки P_u, т.е. вероятностью того, что ошибки не только не будут исправлены, но даже не будет выявлен факт их наличия в принятом сообщении.

5. Вероятностью отказа от декодирования P_f, т.е. вероятностью обнаружения декодером конфигурации ошибок, которую он не способен исправить, и сигнализации об этом.

2.3.1. Обнаружение ошибок

Рассмотрим передачу кодовых слов некоторого двоичного линейного блокового (n, k, d) кода по двоичному симметричному каналу с переходной вероятностью p. Как было показано выше, принятый вектор может быть представлен как $y = uG + e$. Для обнаружения ошибок вычислим значение синдрома $S = yH^T = eH^T$. Если эта величина отлична от нуля, ошибки точно имели место. Но даже если $S = 0$, нельзя исключать того, что ненулевой вектор ошибки совпал с одним из кодовых слов. Вероятность такого события равна

$$P_u = \sum_{c \in \mathcal{C} \setminus \{0\}} P\{e = c\} = \sum_{i=1}^{n} A_i p^i (1-p)^{n-i} \leq \sum_{i=d}^{n} C_n^i p^i (1-p)^{n-i},$$

где A_i — коэффициенты весового спектра кода, и последнее неравенство получено в силу того, что не всякий вектор веса i может быть кодовым словом, т.е. $A_i \leq C_n^i$. Вероятность обнаружения ошибки равна разности вероятности искажения сообщения в канале и вероятности необнаружения ошибки, т.е.

$$P_d = \sum_{i=1}^{n} C_n^i p^i (1-p)^{n-i} - P_u = 1 - (1-p)^n - P_u.$$

2.3.2. Исправление ошибок

Двоичный симметричный канал

Как было показано выше, имеется взаимно однозначное соответствие между исправимыми конфигурациями ошибок и лидерами смежных классов кода. Таким образом, декодер, реализующий критерий минимального расстояния, ошибается, если вектор ошибки не является лидером смежного класса, т.е.

$$P_c = 1 - \sum_{j=0}^{n} a_j p^j (1-p)^{n-j},$$

где a_j — число лидеров смежных классов веса j. Однако на практике чаще используются алгоритмы, исправляющие не более $\lfloor (d-1)/2 \rfloor$

ошибок. Можно показать [51], что в этом случае

$$P_c = \sum_{j=d}^{n} A_j \sum_{l=0}^{\lfloor (d-1)/2 \rfloor} P_l^j, \tag{2.11}$$

где $P_l^j = \sum_{r=0}^{l} C_j^{l-r} C_{n-j}^r p^{j-l+2r}(1-p)^{n-j+l-2r}$ — вероятность того, что принятый вектор отличается в l позициях от некоторого кодового слова веса j. Вероятность отказа от декодирования равна

$$P_f = 1 - \sum_{j=0}^{\lfloor (d-1)/2 \rfloor} C_n^j p^j (1-p)^{n-j} - P_c.$$

Аддитивный гауссовский канал и мягкое декодирование

Рассмотрим передачу символов кодовых слов двоичного линейного блокового кода по аддитивному гауссовскому каналу с использованием сигнального множества $\mathbb{X} = \left\{-\sqrt{E_s}, \sqrt{E_s}\right\}$. Будем считать, что передаваемые символы формируются как $x_i = \sqrt{E_s}(2c_i - 1) \in \mathbb{R}, 1 \le i \le n$, где $c = uG \in \mathcal{C} \subset \mathbb{F}_2^n$. Пусть $\widehat{\mathcal{C}}$ — множество векторов, которые могут быть получены таким образом. Принятые сигналы могут быть представлены как $y_i = x_i + \eta_i, \eta_i \sim \mathcal{N}(0, N_0/2)$. Как было показано выше, декодирование по максимуму правдоподобия в этом случае эквивалентно декодированию в метрике Евклида, т.е. поиску $\widehat{x} = \arg\max_{x \in \widehat{\mathcal{C}}} \sqrt{\sum_{i=1}^n (y_i - x_i)^2} = \arg\max_{x \in \widehat{\mathcal{C}}} \sum_{i=1}^n y_i^2 + \sum_{i=1}^n x_i^2 - 2\sum_{i=1}^n y_i x_i$. Учитывая, что $x_i^2 = E_s$, и исключая слагаемые, не зависящие от x, получим эквивалентную формулировку $\widehat{x} = \arg\max_{x \in \widehat{\mathcal{C}}} \underbrace{\sum_{i=1}^{n} y_i x_i}_{L(y,x)}$. Ошибка декодирования происходит в тех случаях, когда корреляция $L(y, x')$ для какого-либо неправильного слова x' оказывается больше, чем для переданного на самом деле x, т.е. $L(y,x') - L(y,x) = \sum_{i: x'_i \neq x_i} y_i(x'_i - x_i) = \sum_{i: x'_i \neq x_i}(x_i x'_i - x_i^2 + \eta_i(x'_i - x_i)) > 0$. Учитывая, что $x_i, x'_i \in \mathbb{X}$, причем $x_i \neq x'_i$, получим, что ошибка происходит, если $\sum_{i=1}^{d_H(c,c')} \tilde{\eta}_i > 2d_H(c,c')E_s$, где c, c' — соответствующие кодовые слова $\mathcal{C}$, $\tilde{\eta}_i = \eta_{j_i}(x_{j_i} - x'_{j_i}) \sim \mathcal{N}(0, 2E_s N_0)$ и j_i — номер i-й различающейся позиции кодовых слов. Левая часть неравенства распределена по закону $\mathcal{N}(0, 2d_H(c,c')E_s N_0)$. Таким образом, переходная вероятность равна $P(c,c') = Q(\sqrt{2d_H(c,c')E_s/N_0})$. Однако существует множество различных неправильных кодовых слов c', причем может оказаться так, что одновременно для нескольких из них выполняется $L(y,c') > L(y,c)$.

Кроме того, величины $L(y, c')$ вычисляются с использованием одних и тех же значений y_i и потому являются статистически зависимыми. Точный учет этих зависимостей является крайне сложной задачей. Тем не менее, можно получить оценку сверху для вероятности ошибки как *(объединенная верхняя граница)*

$$P_c = P\left\{\bigcup_{c'\in\mathcal{C}\setminus\{c\}}\{L(y,c') > L(y,c)\}\right\} \le \sum_{c'\in\mathcal{C}\setminus\{c\}} P(c',c) = \\ = \sum_{i=d}^{n} A_i Q\left(\sqrt{2i\frac{E_s}{N_0}}\right) = \sum_{i=d}^{n} A_i Q\left(\sqrt{2iR\frac{E_b}{N_0}}\right). \qquad (2.12)$$

Данная граница является точной только в области больших отношений сигнал/шум, когда вероятность того, что правильное решение оказывается хуже сразу нескольких неправильных, пренебрежимо мала. В большинстве случаев при вычислениях достаточно ограничиться лишь одним или двумя первыми ненулевыми слагаемыми. Более точные оценки корректирующей способности можно получить с помощью, например, *касательной сферической границы Полтырева* [56].

Упражнения

1. Постройте порождающую матрицу для кода с проверочной матрицей

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

2. Код, совпадающий с дуальным к нему, называется *самодуальным*. Докажите, что код является самодуальным тогда и только тогда, когда его проверочная матрица равна (с точностью до перестановки столбцов) $H = (-P|I)$, где $PP^T = I$.
3. Вычислите весовой спектр кода Хэмминга $(7, 4, 3)$.
4. Какова минимальная длина двоичного кода с $k = 7, d = 7$?
5. Докажите теорему 2.8 для произвольного q.
6. Преобразованием Адамара отображения $f : \mathbb{F}_2^n \to \mathbb{F}_2$ называется функция $\widehat{f}(u) = \sum_{v\in\mathbb{F}_2^n}(-1)^{u\cdot v} f(v)$. Докажите, что для двоичного линейного кода справедливо $\sum_{u\in\overline{\mathcal{C}}} f(u) = \frac{1}{|\mathcal{C}|}\sum_{u\in\mathcal{C}} \widehat{f}(u)$.
7. Воспользуйтесь вышедоказанным свойством преобразования Адамара для доказательства теоремы 2.6.
8. Докажите выражение (2.11).

3. Универсальные алгоритмы декодирования

3.1. Декодирование по информационным совокупностям

3.1.1. Поиск ближайшего кодового слова

Рассмотрим задачу декодирования линейного кода в метрике Хэмминга, т.е. нахождения кодового слова $\widehat{c} = \arg\min_{c:cH^T=0} d_H(c, y)$, где H — проверочная матрица кода. В общем случае данная задача является NP-полной [15]. Непосредственный метод ее решения состоит в переборе всех кодовых слов и сравнении их с принятым вектором y, что требует $O(q^k n)$ операций. В разд. 2.2.2 был описан альтернативный подход, состоящий в предварительном построении таблицы соответствия синдромов и лидеров смежных классов, имеющей размер $O(q^{n-k})$. Эта таблица позволяет произвести декодирование путем вычисления синдрома $S = yH^T$ и нахождения соответствующей записи в таблице, что требует $O(nk)$ операций. Попытаемся построить алгоритм с меньшей сложностью и потребностями в памяти.

Определение 3.1. *Информационной совокупностью* для линейного (n, k, d) кода $\mathcal{C} \subset \mathbb{F}_q^n$ называется множество $\gamma = \{i_1, \ldots, i_k\} \subset \{1, \ldots, n\}$ такое, что для любого $z \in \mathbb{F}_q^k$ существует единственное кодовое слово $c \in \mathcal{C}$ такое, что $c_{i_j} = z_j, 1 \leq j \leq k$.

Если γ — информационная совокупность, то множество $\phi = \{1, \ldots, n\} \setminus \gamma$ называется *проверочной совокупностью*. Заметим, что подматрица $G(\gamma)$ порождающей матрицы G, содержащая столбцы с номерами из γ, является обратимой. Если бы это было не так, то существовали бы ненулевые векторы $u : uG(\gamma) = 0$, и для любого кодового слова c можно было бы найти другие кодовые слова $c' = c + uG$, которые совпадали бы с ним на информационной совокупности. Таким образом, зная значения символов кодового слова на информационной совокупности, можно восстановить все слово как $c = z(G(\gamma))^{-1}G$. С вычислительной точки зрения операцию нахождения $G_\gamma = (G(\gamma))^{-1}G$ удобно реализовать путем приведения матрицы G к единичной (например, с помощью метода Гаусса) на позициях γ. Так как не все $k \times k$ подматрицы порождающей матрицы обязательно являются обратимыми, далеко не любой набор из k позиций образует информационную совокупность.

Пусть $y = c + e$ — принятая зашумленная последовательность. Информационная совокупность γ называется свободной от ошибок, если $e_{i_j} = 0$ для всех $i_j \in \gamma$. Если ее удалось найти, то кодовое слово может быть восстановлено как $c = (y_{i_1}, \ldots, y_{i_k})G_\gamma$. Это позволяет сформулировать следующий *алгоритм декодирования по информационным совокупностям*:

1. Пусть $\widehat{c} = 0$.
2. Выбрать новую информационную совокупность $\gamma = \{i_1, \ldots, i_k\}$.
3. С помощью операций над строками матрицы G построить матрицу G_γ, содержащую единичную подматрицу на позициях γ.
4. Вычислить $c_\gamma = (y_{i_1}, \ldots, y_{i_k})G_\gamma$. Если $d_H(c_\gamma, y) < d_H(\widehat{c}, y)$, присвоить $\widehat{c} = c_\gamma$.
5. Если еще остаются непроверенные информационные совокупности, перейти к шагу 2.
6. Возвратить $\widehat{c}$.

Очевидно, что после нахождении $\widehat{c} : d_H(\widehat{c}, y) < d/2$, вычисления можно прекратить.

Теорема 3.1. *Алгоритм декодирования по информационным совокупностям является алгоритмом полного декодирования.*

Доказательство. Необходимо доказать, что для всякого исправимого вектора ошибок найдется информационная совокупность, свободная от ошибок. Пусть c — единственное истинное решение задачи декодирования, $E = \mathrm{supp}(e)$ — множество ненулевых позиций вектора ошибок $e = y - c$, причем $|E| \leq n - k$. Предположим, что множество $\{1, \ldots, n\} \setminus E$ не содержит информационных совокупностей. Это означает, что существуют кодовые слова, отличающиеся друг от друга исключительно в позициях E, что противоречит предположению о единственности c. □

Вышеописанный алгоритм находит решение задачи декодирования, как только ему попадается информационная совокупность, свободная от ошибок, или, эквивалентно, проверочная совокупность, покрывающая все искаженные позиции.

Определение 3.2. (n, r, t)-*покрытием* называется набор F из подмножеств ϕ_i мощности r множества $N_n = \{1, \ldots, n\}$ таких, что всякое t-элементное подмножество N_n содержится хотя бы в одном из $\phi_i \in F$.

Если допустима замена полного декодирования на декодирование с исправлением не более чем t ошибок, достаточно лишь построить

$(n, n-k, t)$-покрытие F и рассматривать информационные совокупности, соответствующие проверочным совокупностям $\phi_i \in F$. Необходимо отметить, что далеко не всякое покрытие таково, что все его элементы являются допустимыми проверочными совокупностями конкретного кода. Это связано с тем, что не всякая $k \times k$ подматрица порождающей матрицы имеет ранг k, как того требует шаг 3 вышеприведенного алгоритма.

Пример 3.1. Рассмотрим декодирование вектора $y = \begin{pmatrix}0 & 1 & 1 & 0 & 1 & 0 & 0\end{pmatrix}$ в $(7,4,3)$-коде с порождающей матрицей $G = \begin{pmatrix}1 & 1 & 1 & 0 & 0 & 0 & 0\\ 1 & 0 & 0 & 1 & 1 & 0 & 0\\ 0 & 1 & 0 & 1 & 0 & 1 & 0\\ 1 & 1 & 0 & 1 & 0 & 0 & 1\end{pmatrix}$. Для этого построим $(7,3,1)$-покрытие $F = \{\{1,2,4\},\{5,6,7\},\{3,4,5\}\}$. Видно, что все конфигурации ошибок веса 1 покрываются хотя бы одним из его элементов. Приводя порождающую матрицу к единичной на информационных совокупностях $\gamma_i = \{1,\ldots,7\} \setminus \phi_i$, получим $G_{\{3,5,6,7\}} = \begin{pmatrix}1 & 1 & 1 & 0 & 0 & 0 & 0\\ 1 & 0 & 0 & 1 & 1 & 0 & 0\\ 0 & 1 & 0 & 1 & 0 & 1 & 0\\ 1 & 1 & 0 & 1 & 0 & 0 & 1\end{pmatrix}, G_{\{1,2,3,4\}} = \begin{pmatrix}1 & 0 & 0 & 0 & 0 & 1 & 1\\ 0 & 1 & 0 & 0 & 1 & 0 & 1\\ 0 & 0 & 1 & 0 & 1 & 1 & 0\\ 0 & 0 & 0 & 1 & 1 & 1 & 1\end{pmatrix}, G_{\{1,2,6,7\}} = \begin{pmatrix}1 & 0 & 0 & 1 & 1 & 0 & 0\\ 0 & 1 & 1 & 1 & 1 & 0 & 0\\ 0 & 0 & 1 & 0 & 1 & 1 & 0\\ 0 & 0 & 1 & 1 & 0 & 0 & 1\end{pmatrix}$. Выделяя подвекторы y, задаваемые этими информационными совокупностями, и умножая их на матрицы G_{γ_i}, получим кодовые слова $c_{\{3,5,6,7\}} = c_{\{1,2,6,7\}} = \begin{pmatrix}0 & 1 & 1 & 1 & 1 & 0 & 0\end{pmatrix}, c_{\{1,2,3,4\}} = \begin{pmatrix}0 & 1 & 1 & 0 & 0 & 1 & 1\end{pmatrix}$. Таким образом, ближайшее к вектору y кодовое слово равно $c_{\{1,2,6,7\}}$, и единственная ошибка расположена в четвертой позиции. Заметим, что оно было получено по двум различным информационным совокупностям.

Для коротких кодов соответствующее покрытие содержит небольшое число элементов и может быть построено заранее, причем сложность алгоритма декодирования по информационным совокупностям может быть существенно меньше, чем у других методов. Для длинных кодов предпочтительным является перебор информационных совокупностей в таком порядке, который позволял бы максимально переиспользовать результаты диагонализации порождающей матрицы, избегая обращения к трудоемкому методу Гаусса на каждом шаге. Подробнее этот подход будет рассмотрен в следующем разделе.

3.1.2. Поиск кодового слова наименьшего веса

Задача декодирования линейного (n,k,d)-кода в метрике Хэмминга может быть сформулирована как поиск вектора e наименьшего веса,

такого что $y = uG' + e$ для некоторого u. Если $0 < \mathrm{wt}(e) < d$, то эта проблема эквивалентна поиску ненулевого кодового слова наименьшего веса в линейном коде с порождающей матрицей $G = \begin{pmatrix} G' \\ y \end{pmatrix}$. Впрочем, последняя задача представляет самостоятельный интерес и возникает, например, при анализе свойств построенных кодов. Один из наиболее эффективных алгоритмов ее решения является обобщением метода декодирования по информационным совокупностям [20].

Основная идея *алгоритма Канто-Шабада* состоит в целенаправленном поиске кодовых слов, которые имели бы небольшой вес на одном подмножестве символов γ и нулевой вес на другом подмножестве символов $\sigma : |\sigma| = s$. Для простоты, ограничимся случаем двоичных линейных кодов. Более конкретно, пусть дана некоторая информационная совокупность γ. Построим соответствующую порождающую матрицу $G_\gamma = M_\gamma G$, где G — порождающая матрица рассматриваемого кода и M_γ — подходящая обратимая матрица. Выделим некоторое подмножество $\sigma \subset N_n \setminus \gamma$. Разобъем множество строк матрицы G_γ на два непересекающихся подмножества примерно одинакового размера и для каждого из них рассмотрим все линейные комбинации не более чем t строк матрицы G_γ. Для каждой такой линейной комбинации, задаваемой вектором u_1 или u_2, вычислим символы соответствующих кодовых слов на позициях, входящих в σ. При этом важно не производить вычисление всех символов этих кодовых слов. Пусть $V(\lambda)$ — множество линейных комбинаций строк из первого подмножества, приводящих к подвектору λ на позициях σ. Если линейная комбинация строк из второго подмножества также приводит к подвектору λ, то линейная комбинация, задаваемая $u_1 + u_2$ (т.е. использующая строки из обоих подмножеств), приводит к кодовому слову, имеющему не более $2t$ единиц на γ и нулевые символы на позициях σ, т.е. его вес сравнительно невелик. В этом случае имеет смысл вычислить все символы этого кодового слова и найти его вес. Таким образом, из рассмотрения автоматически исключаются кодовые слова, имеющие априори большой вес. После проверки всех кодовых слов, порождаемых этой процедурой для информационной совокупности γ, необходимо перейти к другой информационной совокупности и повторить описанные действия.

Наиболее трудоемким этапом этой процедуры, как и вышеописанного алгоритма декодирования по информационным совокупностям, является приведение порождающей матрицы к диагональному виду. Эту

операцию можно упростить, если рассматривать последовательность информационных совокупностей $\gamma_i = (\gamma_{i-1} \setminus j') \cup j'', j' \in \gamma_{i-1}, j'' \notin \gamma_{i-1}$. В этом случае матрица $G_{\gamma_{i-1}}$ является "почти диагональной" на множестве столбцов γ_i: для приведения ее к полностью диагональному виду достаточно лишь обработать столбец j''. Необходимо отметить, что для получения информационной совокупности γ_i необходимо обеспечить обратимость соответствующей подматрицы. Это достигается путем выбора j'', такого что матрица $G_{\gamma_{i-1}}$ содержит единицу в столбце j'' в той же строке, в которой она присутствует в столбце j'.

Параметрами этого алгоритма являются величины s и t. При условии оптимального их выбора асимптотическая сложность алгоритма составляет $O(2^{0{,}12n\mathcal{H}(k/n+0{,}031)+10})$.

3.2. Метод порядковых статистик

3.2.1. Алгоритм

Обобщение метода декодирования по информационным совокупностям на случай метрики Евклида приводит к одному из наиболее эффективных универсальных алгоритмов мягкого декодирования двоичных линейных блоковых кодов, известному как *метод порядковых статистик* [35]. Пусть даны логарифмические отношения правдоподобия принятых символов[1] $L_i = \ln \frac{P_{X|Y}\{c_i=1|y_i\}}{P_{X|Y}\{c_i=0|y_i\}}, 1 \le i \le n$. Их абсолютная величина характеризует надежность соответствующих символов. Построим вектор жестких решений $\widehat{y} = (\mathbf{1}\{L_1 > 0\}, \ldots, \mathbf{1}\{L_n > 0\})$, где $\mathbf{1}\{A\}$ — индикаторная функция, равная 1 тогда и только тогда, когда выполняется условие A, и 0 в противном случае. Упорядочим символы в порядке убывания их надежности, т.е. найдем перестановку $(i_1, \ldots, i_n) : |L_{i_j}| \ge |L_{i_{j+1}}|$. Величина $|L_{i_j}|$ носит название j-ой порядковой статистики. Попытаемся привести порождающую матрицу кода к диагональному виду на позициях $i_1, \ldots, i_k$. Возможно, они не образуют информационную совокупность, т.е. на каком-то из шагов (например, j') метода Гаусса не удастся получить единичное значение на диагонали матрицы. В этом случае поменяем местами элементы $i_{j'}$ и $i_{j''}$, где j'' — наименьшее целое число, большее j' (т.е. надежность соответствующего символа является наибольшей) такое, что становится возможным продолжение работы

[1] В случае аддитивного гауссовского канала $L_i = \frac{2\sqrt{E_s}}{\sigma^2} y_i$. Так как в дальнейшем эти величины будут только суммироваться и сравниваться друг с другом, множитель $2\sqrt{E_s}/\sigma^2$ можно опустить.

метода Гаусса. Таким образом, формируется наиболее надежная информационная совокупность γ и соответствующая матрица G_γ. Однако γ может быть не свободна от ошибок, хотя, так как в информационную совокупность входят достаточно надежные символы, их число, скорее всего, невелико. Для их исправления организуем перебор по всем пробным векторам ошибки $e = (e_1, \ldots, e_k) : 0 \le \mathrm{wt}(e) \le t$ на найденной информационной совокупности. По каждому из рассмотренных векторов $z = (\widehat{y}_{j_1} + e_1, \ldots, \widehat{y}_{j_k} + e_k)$ восстановим кодовое слово $c = zG_\gamma$, построим соответствующий вектор сигналов $x : x_i = \sqrt{E_s}(2c_i - 1)$ и сравним его с принятой последовательностью. В качестве окончательного решения выберем то кодовое слово, которое является ближайшим к ней в метрике Евклида. Очевидно, что сложность непосредственной реализации этого метода составляет $O(\sum_{i=0}^{t} C_k^i n)$ операций.

3.2.2. Методы ускорения

Сложность метода порядковых статистик может быть снижена за счет отсечения заведомо плохих пробных конфигураций ошибок [77]. Ограничимся рассмотрением случая каналов без памяти. Введем эллипсоидный вес кодового слова c как $E(c, y) = \sum_{i:c_i \neq \widehat{y}_i} |L_i|$.

Лемма 3.1. *Вероятность того, что кодовое слово c является решением задачи декодирования, убывает с ростом $E(c, y)$.*

Доказательство. Из определения логарифмического отношения правдоподобия вытекает, что $P_{X|Y}\{1|y_i\} = \frac{e^{L_i}}{1+e^{L_i}}$ и $P_{X|Y}\{0|y_i\} = \frac{1}{1+e^{L_i}}$, т.е. $P_{X|Y}\{c_i|y_i\} = \frac{e^{L_i/2}}{1+e^{L_i}} e^{(2c_i-1)L_i/2}$.

Вероятность того, что было передано кодовое слово c при условии того, что на выходе канала без памяти наблюдаются сигналы y, равна $P\{c|y\} = \prod_{i=1}^{n} P_{X|Y}\{c_i|y_i\} = \prod_{i=1}^{n} \frac{e^{L_i/2} e^{(2c_i-1)L_i/2}}{1+e^{L_i}}$. Отсюда видно, что вероятность того, что c является решением задачи декодирования, возрастает с ростом корреляции $L(c, y) = \sum_{i=1}^{n}(2c_i - 1)L_i = \sum_{i:c_i=\widehat{y}_i} |L_i| - \sum_{i:c_i \neq \widehat{y}_i} |L_i| = \sum_{i=1}^{n} |L_i| - 2E(c, y)$. Первое слагаемое не зависит от c, откуда и вытекает утверждение леммы. □

Таким образом, декодирование сводится к поиску кодового слова с наименьшим эллипсоидальным весом. Целесообразно упорядочить пробные векторы ошибок в порядке возрастания $E(e, y) = \sum_{i:e_i \neq 0} |L_{j_i}|$. Для векторов фиксированного веса это может быть сделано путем их сортировки (или генерации) в обратном лексикографическом порядке. Тогда можно останавливать перебор, если все последующие пробные

векторы удовлетворяют $E(e,y) > E(\widehat{c},y)$, где $\widehat{c}$ — текущее наилучшее кодовое слово. Кроме того, если уже найдено некоторое кодовое слово $\widehat{c}$, то все остальные кодовые слова c отличаются от него не менее чем в d позициях, где d — минимальное расстояние кода. Так как $d_H(\widehat{c},\widehat{y}) + d_H(c,\widehat{y}) \geq d_H(\widehat{c},c) \geq d$, $d_H(c,\widehat{y}) \geq d - d_H(\widehat{c},\widehat{y})$. Если кодовое слово c отличается от y в j позициях на наиболее надежной информационной совокупности γ (пусть e равно разности этих векторов на γ), число различий на проверочной совокупности не может быть меньше, чем $d - d_H(\widehat{c},\widehat{y}) - j$. Так как $\widehat{y}_i \neq c_i$ в двоичном случае означает $\widehat{y}_i = \widehat{c}_i$, получим, что $E(c,y) \geq E(e,y) + R[d - d_H(\widehat{c},\widehat{y}) - \mathrm{wt}(e)]$, где $R[l]$ — сумма l наименьших значений $|L_{j_i}|$ таких, что $\widehat{c}_{j_i} = \widehat{y}_{j_i}$. Если правая часть этого неравенства превосходит $E(\widehat{c},y)$, пробный вектор e может быть исключен из рассмотрения.

Очевидно, что корректирующая способность метода порядковых статистик возрастает с увеличением t, что сопровождается быстрым ростом сложности. Существует возможность размена числа арифметических операций на объем используемой оперативной памяти. Для этого следует дополнить наиболее надежную информационную совокупность s следующими по надежности символами и рассмотреть все конфигурации ошибок веса не более $2t$ на этом множестве. При этом такие конфигурации могут получаться как суммы пробных векторов веса не более t (алгоритм *Box and Match* [77]).

3.3. Декодирование по обобщенному минимальному расстоянию

3.3.1. Исправление ошибок и стираний

Напомним, что *стиранием* называется особый вид ошибки, местоположение которой точно известно. Стирание s символов эквивалентно их выкалыванию при передаче (см. разд. 12.1.2). При этом минимальное расстояние Хэмминга кода становится равным $d' \geq d - s$. Укороченный код способен исправить $t \leq \frac{d'-1}{2}$ ошибок. Таким образом, код с минимальным расстоянием Хэмминга d способен исправить t ошибок и s стираний, если выполняется неравенство

$$2t + s + 1 \leq d. \tag{3.1}$$

Для некоторых кодов известны эффективные алгоритмы, способные одновременно исправлять ошибки и стирания. Если такой алгоритм недоступен, в случае двоичных кодов можно воспользоваться следую-

щим подходом. Пусть $y \in (\mathbb{F}_2 \cup \{\epsilon\})^n$ — декодируемый вектор. Построим вспомогательные векторы y' и y'', в которых символы стирания заменены на 0 и 1, соответственно. Продекодируем эти векторы с помощью алгоритма, исправляющего только ошибки. Пусть c', c'' — кодовые слова, возвращенные этим декодером. Выберем в качестве решения задачи декодирования то из них, которое является ближайшим к y (стертые позиции при вычислении расстояния Хэмминга не учитываются). Данный метод позволяет достичь границы (3.1). Действительно, если в результате описанных действий в y' были внесены τ дополнительных ошибок, то их число в y'' равно $s - \tau$. Декодирование с исправлением только ошибок является успешным, если выполняются неравенства $2(t + \tau) < d$ и $2(t+s-\tau) < d$, соответственно. Видно, что если одно из них нарушается, но при этом верно (3.1), то другое выполняется автоматически.

3.3.2. Метод Форни

Существенным недостатком метода порядковых статистик является его высокая вычислительная сложность. В тех случаях, когда рассматриваемый код обладает некоторой алгебраической структурой, делающей возможным использование эффективных алгоритмов декодирования в метрике Хэмминга, мягкое декодирование может быть реализовано с помощью сравнительно простой надстройки над такими алгоритмами жесткого декодирования.

Пусть дан двоичный (n, k, d)-код над $\mathbb{F}_q$. Предположим, что существует алгоритм декодирования этого кода в метрике Хэмминга, исправляющий ошибки и стирания, т.е. способный для любого $(z_1, \ldots, z_n) \in (\mathbb{F}_q \cup \{\epsilon\})^n$ находить $\widehat{c} = \arg\min_{c \in \mathcal{C}} |\{i | z_i \neq \epsilon, z_i \neq c_i\}|$. Пусть $(z_1, \ldots, z_n) \in \mathbb{F}_q^n$ и $(r_1, \ldots, r_n) \in \mathbb{R}_+^n$— вектора жестких решений и их надежности. Например, в случае $q = 2$ $z_i = \mathbf{1}\{L_i > 0\}$ и $r_i = |L_i|$, где L_i — логарифмические отношения правдоподобия принятых символов. Алгоритм *декодирования по обобщенному минимальному расстоянию* имеет следующий вид:

1. Упорядочить принятые символы в порядке возрастания их надежности, т.е. найти перестановку $j_i : r_{j_i} \leq r_{j_{i+1}}, 1 \leq i < n$.
2. Пусть $s = 1 - (d \bmod 2)$, $\widehat{c} = 0$.
3. Объявить стертыми s наименее надежных символов, т.е. $z_{j_i} = \epsilon, 1 \leq i \leq s$.

4. Пусть c' — результат декодирования z с помощью декодера, исправляющего ошибки и стирания. Если c' более правдоподобен, чем $\widehat{c}$, присвоить $\widehat{c} = c'$.
5. Присвоить $s = s + 2$. Если $s < d$, перейти к шагу 3.

Данный алгоритм применим для достаточно широкого класса возможных определений понятий "надежность" и "правдоподобность", поэтому при его описании был принят неформальный стиль. Анализ области декодирования этого метода представлен в [33, 1].

3.4. Метод Чейза

Еще одним простым способом построения мягкого декодера на базе жесткого является метод Чейза [21]. Существуют следующие три типа *декодера Чейза*:

1. Сформируем вектор жестких решений $(z_1, \ldots, z_n)$, а также все возможные пробные векторы, отличающиеся от него не более чем в $d-1$ позиций. Продекодируем эти векторы с помощью декодера, исправляющего ошибки в метрике Хэмминга, и выберем наиболее правдоподобное из получившихся кодовых слов. Высокая сложность делает этот подход практически неприменимым.
2. Сформируем вектор жестких решений $(z_1, \ldots, z_n)$, а также все возможные пробные векторы, отличающиеся от него не более чем в τ наименее надежных позициях. Продекодируем эти векторы с помощью декодера, исправляющего ошибки в метрике Хэмминга, и выберем наиболее правдоподобное из получившихся кодовых слов. В некоторых случаях при этом удается многократно переиспользовать результаты промежуточных вычислений, что позволяет получить декодер с достаточно малой сложностью [16, 82]. Классический декодер Чейза предполагает $\tau = \lfloor d/2 \rfloor$, но ничто не мешает выбрать большее значение τ, улучшив таким образом корректирующую способность за счет увеличения сложности.
3. Этот тип применим к двоичным кодам. Сформируем вектор жестких решений $(z_1, \ldots, z_n)$, а также пробные векторы, полученные путем инвертирования $s_0 = 1 - (d \bmod 2), s_0 + 2, s_0 + 4, \ldots, d - 1$ наименее надежных символов. Продекодируем эти векторы с помощью декодера, исправляющего ошибки в метрике Хэмминга, и выберем наиболее правдоподобное из получившихся кодовых слов. Данный метод схож с декодированием по обобщенному минимальному расстоянию.

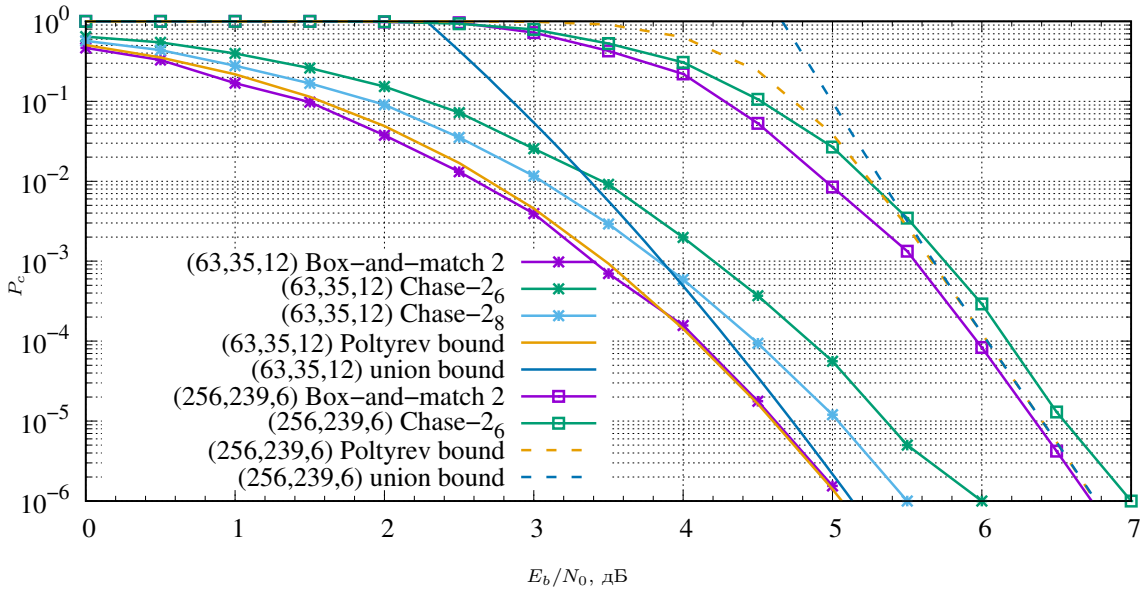


Рис. 3.1. Корректирующая способность кодов БЧХ $(63, 35, 12)$ и $(256, 239, 6)$

Для высокоскоростных кодов метод Чейза обеспечивает декодирование почти по максимуму правдоподобия и при этом весьма прост.

На рис. 3.1 представлены результаты статистического моделирования, иллюстрирующие корректирующую способность декодеров Чейза типа 2 с $\tau = 6, 8$ и Box-and-Match с $t = 2$ для случая кодов БЧХ $(63, 35, 12)$ и $(256, 239, 6)$. Кроме того, на графике представлены объединенная верхняя граница (2.12) и верхняя граница Полтырева для вероятности ошибки декодирования по максимуму правдоподобия. Видно, что алгоритм Box-and-Match обеспечивает декодирование почти по максимуму правдоподобия. С помощью метода Чейза можно получить весьма хорошее приближение к декодеру по максимуму правдоподобия. Объединенная верхняя граница дает приемлемую точность только на высоких отношениях сигнал/шум, в то время как граница Полтырева дает намного более информативные результаты.

Упражнения

1. Пусть дан код с порождающей матрицей $G = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$. Продекодируйте в нем

вектор $y = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$ в метрике Хемминга.

Часть II

Алгебраические методы кодирования

4. Конечные поля

При построении и декодировании многих корректирующих кодов используется аппарат полей Галуа.

4.1. Некоторые алгебраические структуры

4.1.1. Кольца, тела, поля

Определение 4.1. *Кольцом* называется алгебра

$$\mathcal{R} = (R, +, \cdot, \mathbf{0}, \mathbf{1}),$$

сигнатура которой состоит из двух бинарных и двух нульарных операций, для которых выполняются равенства (аксиомы кольца):

1. $a + (b + c) = (a + b) + c$;
2. $a + b = b + a$;
3. $a + \mathbf{0} = a$;
4. $\forall a \in R : \exists a' : a + a' = 0$;
5. $a \cdot (b \cdot c) = (a \cdot b) \cdot c$;
6. $a \cdot \mathbf{1} = \mathbf{1} \cdot a = a$;
7. $a \cdot (b + c) = a \cdot b + a \cdot c$
8. $(b + c) \cdot a = b \cdot a + c \cdot a$.

Операции $+$ и $\cdot$ называются сложением и умножением, $\mathbf{0}$ и $\mathbf{1}$ — нулем и единицей кольца. Можно заметить, что алгебра $(R, \mathbf{0}, +)$ является абелевой группой, называемой аддитивной группой кольца $\mathcal{R}$. При этом обратный элемент $a' : a + a' = \mathbf{0}$ обозначают как $-a$, а операцию сложения с ним называют вычитанием $a + (-b) = a - b$. С другой стороны, алгебра $(R, \mathbf{1}, \cdot)$ является мультипликативным моноидом кольца. Связь между сложением кольца и умножением кольца устанавливается аксиомой дистрибутивности. Кольцо называется коммутативным, если его операция умножения коммутативна.

Пример 4.1. Алгебра $(\mathbb{Z}, +, \cdot, 0, 1)$ является коммутативным кольцом. При этом $(\mathbb{N}_0, +, \cdot, 0, 1)$ кольцом не является, т.к. $(\mathbb{N}_0, +)$ — коммутативный моноид, но не группа.

Пример 4.2. Множество всех квадратных матриц фиксированного порядка с операциями сложения и умножения является некоммутативным кольцом.

Теорема 4.1. *В любом кольце выполняются следующие тождества:*

1. $\mathbf{0} \cdot a = a \cdot \mathbf{0} = \mathbf{0}$

2. $(-a)\cdot b = -(a\cdot b) = a\cdot(-b)$

3. $(a-b)\cdot c = a\cdot c - b\cdot c, c\cdot(a-b) = c\cdot a - c\cdot b$

Доказательство. 1. $a + \mathbf{0}\cdot a = \mathbf{1}\cdot a + \mathbf{0}\cdot a = (\mathbf{1}+\mathbf{0})\cdot a = \mathbf{1}\cdot a = a \Rightarrow \mathbf{0}\cdot a = a - a = \mathbf{0}$. Равенство $a\cdot\mathbf{0} = \mathbf{0}$ доказывается аналогично.

2. $a\cdot(-b) + a\cdot b = a\cdot((-b)+b) = a\cdot\mathbf{0} = \mathbf{0} \Rightarrow a\cdot(-b) = -(a\cdot b)$

3. $a\cdot(b-c) = a\cdot(b+(-c)) = a\cdot b + a\cdot(-c) = a\cdot b - a\cdot c$.

□

Ненулевые элементы a, b кольца называются *делителями нуля*, если $a\cdot b = \mathbf{0} \vee b\cdot a = \mathbf{0}$. Примерами кольца с делителями нуля являются кольцо вычетов целых чисел по модулю составного числа ($2 * 3 \equiv 0 \bmod 6$) и кольцо квадратных матриц порядка не ниже 2. Коммутативное кольцо без делителей нуля называется *областью целостности*.

Пример 4.3. Областями целостности являются множества целых чисел $\mathbb{Z}$, многочленов с вещественными коэффициентами $\mathbb{R}[x]$ и Гауссовы целые (комплексные числа с целочисленными компонентами).

Ненулевой элемент кольца u называется обратимым, если для него существует обратный элемент $v : uv = 1$. Элемент u кольца называется *неразложимым* или *неприводимым*, если он не может быть представлен в виде произведения двух или более нетривиальных (т.е. отличных от $\mathbf{1}$ и других обратимых) элементов. Неразложимыми являются простые числа в кольце целых чисел и неприводимые многочлены.

Пусть задана некоторая область целостности D. Будем говорить что элемент $a \in D$ делит другой элемент $b \in D$ (или b является кратным a, a является делителем b, $a|b$), если существует $q \in D : b = qa$. В силу аксиомы ассоциативности, если $a|b$ и $b|c$, то $a|c$. Если $a|b$, то $\forall c \in D : a|(cb)$. Если $a|b \wedge a|c$, то $a|(b+c), a|(b-c)$. Обратимые элементы являются делителями всех элементов кольца, т.к. $\forall b : b = b\cdot\mathbf{1} = (b\cdot a^{-1})\cdot a$. Если $a|b$ и $b|a$, то элементы a и b называются *ассоциированными*. Элементы являются ассоциированными тогда и только тогда, когда существует обратимый элемент $u : au = b$. Например, в кольце многочленов с вещественными коэффициентами обратимыми являются все элементы вида $f(x) = a, a \in \mathbb{R}\setminus\{0\}$, а многочленом, ассоциированным с некоторым $f(x)$, является $af(x), a \neq 0$.

Определение 4.2. *Полукольцом* называется алгебра

$$\mathcal{S} = (S, +, \cdot, \mathbf{0}, \mathbf{1}),$$

такая, что для произвольных элементов a, b, c множества S выполняются следующие равенства (аксиомы полукольца):

1. $a + (b + c) = (a + b) + c$;
2. $a + b = b + a$;
3. $a + \mathbf{0} = a$;
4. $a \cdot (b \cdot c) = (a \cdot b) \cdot c$;
5. $a \cdot \mathbf{1} = \mathbf{1} \cdot a = a$;
6. $a \cdot (b + c) = a \cdot b + a \cdot c$;
7. $(b + c) \cdot a = b \cdot a + c \cdot a$;
8. $a \cdot \mathbf{0} = \mathbf{0} \cdot a = \mathbf{0}$;

Видно, что кольцо — частный случай полукольца. Если кольцо является абелевой группой по сложению, то полукольцо — лишь коммутативный моноид.

Пример 4.4. Алгебра логики $(\{0, 1\}, \vee, \wedge)$ является полукольцом.

Поставим вопрос: в каких случаях кольцо образует группу относительно операции умножения? Ясно, что все элементы кольца, в котором $\mathbf{0} \neq \mathbf{1}$, не могут образовывать группу по умножению, т.к. $\mathbf{0}$ не имеет обратного. Действительно, если существует $\mathbf{0}^{-1} : \mathbf{0} \cdot \mathbf{0}^{-1}$, то $\mathbf{0} = \mathbf{0} \cdot \mathbf{0}^{-1} = \mathbf{1}$. Таким образом, нулевой элемент не может входить в мультипликативную группу. Если в кольце имеются делители нуля, то подмножество ненулевых элементов не может быть группой по умножению, т.к. оно не замкнуто.

Определение 4.3. Кольцо, в котором множество всех ненулевых элементов образует группу по умножению, называется *телом*. Коммутативное тело называется *полем*. Группа ненулевых элементов тела (поля) по умножению называется *мультипликативной группой* тела (поля).

Таким образом, в любом поле $\mathbb{F}$ выполняются следующие тождества:

1. $a + (b + c) = (a + b) + c$;
2. $a + b = b + a$;
3. $a + \mathbf{0} = a$;
4. $\forall a \in F : \exists a' : a + a' = 0$;
5. $a \cdot (b \cdot c) = (a \cdot b) \cdot c$;
6. $a \cdot b = b \cdot a$; в случае тела имеет место лишь $a \cdot \mathbf{1} = \mathbf{1} \cdot a = a$;
7. $a \cdot (b + c) = a \cdot b + a \cdot c, (b + c) \cdot a = b \cdot a + c \cdot a$;
8. $\forall a \neq \mathbf{0}\, \exists a^{-1} : a \cdot a^{-1} = a^{-1} \cdot a = \mathbf{1}$.

Пример 4.5. Алгебра $(\mathbb{Q}, \cdot, +, 0, 1)$ называется полем рациональных чисел. Алгебра $(\mathbb{R}, \cdot, +, 0, 1)$ называется полем вещественных чисел.

Теорема 4.2. *Если A — конечное множество и $f : A \to A$ — инъекция, то она также является сюръекцией и, следовательно, биекцией.*

Доказательство. Предположим, что f не является сюръективным, т.е. существует такой элемент $a \in A : \nexists x \in A : f(x) = a$. Пусть $x_1, \ldots, x_n$ — последовательность различных элементов A, где $n = |A|$. Рассмотрим последовательность $f(x_1), \ldots, f(x_n)$. Эта последовательность не может содержать элемент a, т.е. в ней имеется не более $n - 1$ различных элементов. Это означает, что в ней имеются повторяющиеся элементы, т.е. $\exists i, j : i \neq j \wedge f(x_i) = f(x_j)$, что противоречит предположению об инъективности отображения. □

Теорема 4.3. *Конечная область целостности является полем.*

Доказательство. Т.к. область целостности $\mathbb{F}$ по определению является коммутативным кольцом, достаточно только показать наличие мультипликативно обратных у всех ее ненулевых элементов, т.е. что $\forall x \in \mathbb{F} : x \neq \mathbf{0} : \exists x^{-1} : xx^{-1} = \mathbf{1}$.

Построим отображение $f_a(x) = a \cdot x, a, x \neq \mathbf{0}$. Ясно, что $f_a(x) : \mathbb{F} \setminus \{0\} \to \mathbb{F} \setminus \{\mathbf{0}\}$. Это отображение является инъекцией, т.к. из $a \cdot x = a \cdot y$ следует $a \cdot (x - y) = \mathbf{0}$. В силу отсутствия делителей нуля, это означает, что $x = y$. В силу конечности $\mathbb{F}$ из теоремы 4.2 вытекает, что $f_a(x)$ является сюръекцией, а следовательно, и биекцией. Следовательно, $\forall y \in \mathbb{F} \setminus \{\mathbf{0}\} : \exists! x : y = ax$. В частности, при $y = \mathbf{1}$ соответствующий x является элементом, обратным к a. □

4.1.2. Идеалы

Определение 4.4. Подмножество I кольца R (в общем случае некоммутативного) называется *правым идеалом*, если

1. $(I, \{+\})$ является подгруппой $(R, \{+\})$
2. $\forall r \in R, \forall x \in I : xr \in I$.

Если последнее условие заменить на $\forall r \in R, \forall x \in I : rx \in I$, то множество I называется *левым идеалом*. Если I является и левым, и правым идеалом, оно называется *двусторонним идеалом* или просто *идеалом*.

Ясно, что в коммутативных кольцах все идеалы являются двусторонними. Для простоты будем далее рассматривать именно этот случай.

Если $A \subset R$, то идеалом, порождаемым A, называется наименьший идеал, содержащий A, который обозначается $< A >$. Этот идеал состоит из всех элементов вида $\sum_i r_i a_i, a_i \in A, r_i \in R$. Идеал I называется *конечно порожденным*, если существует конечное множество $A : I =< A >$.

Пример 4.6. 1. Множество четных чисел является идеалом в кольце $\mathbb{Z}$, порожденным элементом 2.
2. Множество многочленов с вещественными коэффициентами, делящихся на x^2+1, является идеалом в $\mathbb{R}[x]$.
3. Множество квадратных матриц, у которых последний столбец является нулевым, является левым идеалом в кольце квадратных матриц, но не является правым идеалом.
4. Кольцо непрерывных функций вещественного аргумента $C(\mathbb{R})$ содержит идеал непрерывных функций, таких что $f(1)=0$.
5. $\{\mathbf{0}\}$ и R являются идеалами в любом кольце R.
6. $< x, y >$ является идеалом, содержащим все многочлены от двух переменных с нулевым свободным членом.

Идеал называется *главным*, если он порождается единственным элементом. Область целостности, в которой все идеалы являются главными, называется *кольцом главных идеалов*.

Определение 4.5. Идеал I кольца R называется *максимальным*, если $I \neq R$ и всякий идеал J, содержащий его, равен или I, или R.

Другими словами, идеал называется максимальным, если он отличен от своего кольца и не содержится ни в каком ином идеале. Заметим, что если $f_1, f_2, \ldots, f_n$ — система образующих максимального идеала I, то добавление в нее $f_0 \notin I$ приведет к $< f_0, \ldots, f_n >= R$.

Пример 4.7. Идеал $< 7 >\subset \mathbb{Z}$ является максимальным, т.к. числа, кратные 7, невозможно получить иначе как умножением 7 на целые числа.

Определение 4.6. Пусть дана полугруппа $\mathcal{G} = (G, \cdot)$. Бинарное отношение $\sim\subset G^2$ называется отношением *конгруэнтности*, если оно является эквивалентностью на G и $\forall a, b, c \in G : (a \sim b) \Rightarrow a \cdot c \sim b \cdot c$.

Пример 4.8. Множество целых чисел является коммутативным моноидом по умножению. Заданное на нем бинарное отношение сравнимости по модулю другого целого числа является отношением конгруэнтности (сравнимости). Действительно, это отношение является отношением эквивалентности. Если $a \equiv b \bmod d$, то $(a = pd+r) \wedge (b = qd + r)$ и $\forall c \in \mathbb{Z} : (ac \equiv rc \bmod d) \wedge (bc \equiv rc \bmod d)$, откуда в силу транзитивности следует $ac \equiv bc \bmod d$.

Пусть дано кольцо R и идеал $I \subset R$. Определим бинарное отношение $\sim= \{(a, b) \in R^2 | b - a \in I\}$. Ясно, что оно является отношением конгруэнтности. Если $a \sim b$, то говорят, что a и b конгруэнтны (сравнимы) по модулю I. Данное бинарное отношение задает разбиение кольца на классы эквивалентности $[a] = a + I = a \bmod I = \{a + r | r \in I\}$. Множество этих классов эквивалентности называется *факторкольцом* или *кольцом вычетов R по модулю I*, и обозначается R/I. Факторкольцо

действительно является кольцом, если на нем определить операции следующим образом:

1. $(a+I)+(b+I)=\underbrace{(a+b)}_{\in R}+I$
2. $-(a+I)=(-a)+I$
3. $(a+I)(b+I)=\underbrace{(ab)}_{\in R}+I$
4. Нулевой элемент $\mathbf{0}+I=I$
5. Единичный элемент $\mathbf{1}+I$.

Пример 4.9. Рассмотрим идеал целых чисел, кратных 3, т.е. $<3>$. Факторкольцо $\mathbb{Z}/<3>$ представляет собой множество из трех классов эквивалентности $\{\{0,\pm3,\pm6,\pm9,\ldots\},\{1,-2,4,-5,7,-8,\ldots\},\{2,-1,5,-4,8,-7,\ldots\}\}$.

Аналогичным образом можно построить кольцо вычетов $R[x]$ по модулю некоторого многочлена $f(x)$.

Если R — кольцо главных идеалов, и $<a>\in R$, то соответствующее факторкольцо обозначают R/aR.

Теорема 4.4. *Если $I\subset R$ является максимальным идеалом, то R/I является полем.*

Доказательство. То, что факторкольцо является кольцом, очевидно. Необходимо только показать, что для всякого ненулевого $a\in R/I$ существует мультипликативно обратный элемент $b: ab=\mathbf{1}$. Если $a\neq\mathbf{0}$, то $a\notin I$. Рассмотрим идеал $J=\{ax+m|x\in R, m\in I\}$. Это множество действительно является идеалом, т.к. $(ax_1+m_1)\pm(ax_2+m_2)=a(x_1\pm x_2)+(m_1\pm m_2), (x_1\pm x_2)\in R, (m_1\pm m_2)\in I$ и $(ax+m)y=a(xy)+(my), xy\in R, my\in I$ при любом $y\in R$. Полагая $x=0$, можно получить все элементы исходного идеала I, откуда следует, что $I\subset J$. В то же время $a\in J, a\notin I$. Т.к. I — максимальный идеал, это означает, что $J=R$, т.е. $\mathbf{1}\in J$. Таким образом, для всякого ненулевого a найдутся $b,m: \mathbf{1}=ab+m$, откуда $ab-\mathbf{1}\in I$, т.е. в факторкольце справедливо $ab=\mathbf{1}$. Таким образом, для каждого ненулевого элемента факторкольца по модулю максимального идеала существует обратный элемент, т.е. оно является полем. □

Пример 4.10. Идеал $<x,y>\subset R[x,y]$ является максимальным. Следовательно, $R[x,y]/<x,y>$ является полем.

Теорема 4.5. *Пусть R — кольцо главных идеалов и p — его неприводимый элемент. Тогда факторкольцо R/pR является полем.*

Доказательство. Докажем, что идеал $I=<p>$ является максимальным. Предположим, что существует идеал $J\neq R: I\subset J$. Т.к.

R — кольцо главных идеалов, этот идеал также является главным и $\exists q \in R : J =< q >$. Если q — обратимый элемент кольца, то $J = R$. В противном случае получим, что т.к. $p \in J$, найдется $s \in R : p = qs$. Но по предположению p является неприводимым элементом, откуда s является обратимым элементом кольца. Но тогда $s^{-1} \in R$ и $q = s^{-1}p$, т.е. $J \subset I$, откуда $I = J$. Таким образом, идеал $< p >= pR$ является максимальным и R/pR является полем. □

В дальнейшем подобные поля будут обозначаться просто R/p.

Пример 4.11. Пусть p — простое число. Ниже будет показано, что кольцо целых чисел $\mathbb{Z}$ является кольцом главных идеалов. Тогда $\mathbb{Z}/p$ является полем и обозначается $GF(p)$. Действительно, сравнение вида $ax \equiv 1 \bmod p, 0 < a < p$ всегда имеет единственное решение.

Пример 4.12. Многочлен x^2+1 является неприводимым над полем вещественных чисел $\mathbb{R}$. Как будет показано ниже, кольцо $\mathbb{R}[x]$ является кольцом главных идеалов. Следовательно, факторкольцо $\mathbb{R}[x]/(x^2+1)$ является полем. На самом деле оно совпадает с полем комплексных чисел $\mathbb{C}$. Действительно, $(a+bx)(c+dx) = ac+(ad+bc)x+bdx^2 \equiv (ac-bd)+(ad+bc)x \bmod (x^2+1)$, что соответствует произведению комплексных чисел $a + \mathbf{i}\, b$ и $c + \mathbf{i}\, d$. Обозначая $\mathbf{i} = [x]_{\equiv_{x^2+1}}$, получим $\mathbf{i} \cdot \mathbf{i} = x^2 \equiv -1 \bmod (x^2 + 1)$.

Пример 4.13. Рассмотрим многочлен $x^2 + x + 1$, неприводимый над полем $GF(2)$. Кольцо $GF(2)[x]$ является кольцом главных идеалов. Следовательно, $GF(2)/(x^2 + x + 1)$ является полем и обозначается $GF(2^2)$ или $GF(4)$. Элементами этого поля являются классы многочленов $0, 1, x, x + 1$ по модулю $x^2 + x + 1$. Все операции в этом поле выполняются по модулю $x^2 + x + 1$. Например, $x \cdot (x + 1) = x^2 + x \equiv 1 \bmod (x^2 + x + 1)$. Аналогично случаю комплексных чисел, можно ввести элемент $\beta \in GF(2^2)$, такой что $\beta^2 + \beta + 1 = 0$.

Таким образом, расширенные поля образуются присоединением к исходному полю мнимого корня некоторого неприводимого многочлена.

4.2. Алгебраические свойства конечных полей

4.2.1. Основные факты о конечных полях

Напомним, что в соответствии с теоремой 4.5, если R является кольцом главных идеалов и p — его неприводимым элементом, то факторкольцо R/pR является полем. В зависимости от свойств кольца R, это поле может содержать конечное или бесконечное число элементов. Поле, содержащее конечное число элементов q, называется *конечным* и обозначается $GF(q)$. Конечные поля называются также полями Галуа. Конечное поле является областью целостности, т.к. если допустить, что $\exists a, b \neq 0 : ab = \mathbf{0}$, то $\mathbf{0} = (((ab)b^{-1})a^{-1}) = \mathbf{1}$.

Если поле конечно, то не могут быть различными все элементы $\mathbf{1}, \mathbf{1}+\mathbf{1}, \mathbf{1}+\mathbf{1}+\mathbf{1}, \ldots$. Следовательно, существует наименьшее число $p : \underbrace{\mathbf{1}+\mathbf{1}+\ldots+\mathbf{1}}_{p \text{ раз}} = \mathbf{0}$. Оно называется *характеристикой поля*. Число p является простым, т.к. если допустить, что $p = st$ и $s, t > 1$, то обозначая $S = \underbrace{\mathbf{1}+\mathbf{1}+\ldots+\mathbf{1}}_{s \text{ раз}}, T = \underbrace{\mathbf{1}+\mathbf{1}+\ldots+\mathbf{1}}_{t \text{ раз}}$, получим $\mathbf{0} = \underbrace{\mathbf{1}+\mathbf{1}+\ldots+\mathbf{1}}_{p \text{ раз}} = \underbrace{S+S+\ldots+S}_{t \text{ раз}} = S(\underbrace{\mathbf{1}+\mathbf{1}+\ldots+\mathbf{1}}_{t \text{ раз}}) = ST$, откуда следует, что $S = 0 \vee T = 0$, что противоречит предположению о минимальности p. Бесконечные поля называют полями характеристики нуль.

Заметим, что если $p(x)$ — неприводимый многочлен степени m над полем $GF(q)$, то с помощью него можно построить поле $GF(q)[x]/p(x)$, состоящее из всех классов полиномиальных вычетов по модулю $p(x)$. Ясно, что таковых имеется q^m штук.

Теорема 4.6. *Пусть $\mathbb{F}$ — поле из q элементов. Тогда $q = p^m$, где p — простое, а m — натуральное число.*

Доказательство. Как было показано выше, элемент $\mathbf{1}$ образует аддитивную циклическую подгруппу исходного поля, причем эта подгруппа имеет простой порядок p. Из аксиом поля вытекает, что все его элементы также образуют аддитивную группу из q элементов, откуда в силу теоремы Лагранжа следует $p|q$. Кроме того, элементы подгруппы, порождаемой $\mathbf{1}$, изоморфны кольцу целых чисел по модулю p. Следовательно, $\mathbb{F}$ содержит подполе, изоморфное $GF(p)$.

Будем называть элементы $\alpha_1, \ldots, \alpha_m$ линейно независимыми с коэффициентами из $GF(p)$, если[1] $\{(x_1, x_2, \ldots, x_m) \in GF(p)^m | \sum_{i=1}^m x_i\alpha_i = 0\} = \{(0, \ldots, 0)\}$. Среди всех линейно независимых подмножеств $\mathbb{F}$ выделим подмножество $\{\alpha_1, \ldots, \alpha_m\}$ с максимальным числом элементов. Следовательно, $\forall \beta \in \mathbb{F}$ множество $\{\beta, \alpha_1, \ldots, \alpha_m\}$ линейно зависимо, т.е. $\exists x_1, \ldots, x_m \in GF(p) : \beta = \sum_{i=1}^m x_i\alpha_i$. В силу линейной независимости элементов α_i различные значения x_i приведут к различным значениям β. Всего различных значений x_i существует p штук, т.е. число различных элементов поля равно p^m. □

Следствие 4.1. *Поле $GF(q^m)$ образует m-мерное линейное пространство над полем $GF(q)$.*

[1]Не следует путать m-мерное векторное пространство $GF(p)^m$ и конечное поле $GF(p^m)$. Их взаимосвязь будет исследована ниже.

Замечание 4.1. Описанный выше способ построения поля $GF(q^m)$ с помощью неприводимого над $GF(q)$ многочлена применим как в случае простого q, так и в случае $q = p^l$.

Замечание 4.2. Необходимо понимать, что конечное поле $GF(p^m), m > 1$ не имеет ничего общего с кольцом $\mathbb{Z}_{p^m}$ целых чисел по модулю p^m, которое полем не является.

Поля, содержащие простое число элементов, называются простыми. Поля вида $GF(p^m)$, где $m > 1$ и p — простое, называются расширенными.

Лемма 4.1. *Пусть $\mathcal{G} = (G, \cdot)$ — конечная группа и элементы $g, h \in G$ имеют порядок r, s соответственно, причем $(r, s) = 1$. Тогда элемент gh имеет порядок rs.*

Доказательство. То, что $(gh)^{rs} = \mathbf{1}$, очевидно. Следовательно, истинный порядок p элемента gh должен быть делителем числа rs. Пусть $p|(rs)$ и $(gh)^p = 1$. Тогда $(gh)^{pr} = h^{pr} = \mathbf{1}$. Из этого следует, что $s|(pr)$, откуда $s|p$. Аналогично можно показать, что $r|p$. Т.к. $(r, s) = 1$, получаем $(rs)|p$, что означает $p = rs$. □

Теорема 4.7. *Множество ненулевых элементов конечного поля $GF(q)$ образует конечную циклическую группу по умножению.*

Доказательство. То, что ненулевые элементы образуют конечную группу по умножению, следует из аксиом поля. Выберем в этой группе элемент α с максимально возможным порядком r. Пусть l — порядок некоторого другого элемента $\beta \neq 0$. Пусть π — простое число, такое что $r = \pi^a r'$ и $l = \pi^b l'$ и $(r', \pi) = (l', \pi) = 1$. Тогда элемент α^{π^a} имеет порядок r', а $\beta^{l'}$ имеет порядок π^b. Из леммы 4.1 следует, что порядок $\gamma = \alpha^{\pi^a}\beta^{l'}$ равен $\pi^b r'$. Т.к. r — максимально возможный порядок, получим $\pi^b r' \leq \pi^a r'$ и $b \leq a$. Это справедливо для всех простых π. Таким образом, если некоторая степень простого числа является делителем l, то она является и делителем r. Следовательно, $l|r$. Таким образом, получаем, что все элементы конечного поля удовлетворяют соотношению $x^r - 1 = 0$. Многочлен $x^r - 1$ принадлежит Евклидову кольцу $GF(q)[x]$ и разлагается на множители единственным образом. Следовательно, $\forall \beta \in GF(q)\backslash\{0\} : (x-\beta)|(x^r-1)$, откуда $r \geq q - 1$. Но порядок элемента в группе не может превышать порядка самой группы, откуда $r = q - 1$. Таким образом, в группе ненулевых элементов поля существует образующий элемент, т.е. она является циклической. □

Образующий элемент α, существование которого было установлено в предыдущей теореме, называется *примитивным*. В случае простого q этот элемент равен первообразному корню по модулю q.

Следствие 4.2. *Для всякого ненулевого $\beta \in GF(q)$ выполняется $\beta^{q-1} = 1$.*

Следствие 4.3. *Все элементы поля $GF(q)$ удовлетворяют уравнению $x^q - x = 0$.*

Доказательство. Ненулевые элементы удовлетворяют уравнению $x^{q-1} - 1 = 0$, а следовательно и $x(x^{q-1} - 1) = 0$. Ясно, что последнему уравнению удовлетворяет и нулевой элемент. □

Следствие 4.4. *Порядок любого ненулевого $\beta \in GF(q)$ делит $q - 1$.*

Следствие 4.5. *Всякий ненулевой элемент поля β образует некоторую циклическую подгруппу по умножению, порядок которой совпадает с порядком β.*

Теорема 4.8. *В поле характеристики $p > 1$ справедливо*

$$(x + y)^p = x^p + y^p$$

Доказательство. Воспользуемся соотношением

$$(x + y)^p = \sum_{i=0}^{p} C_p^i x^{p-i} y^i = x^p + y^p + \sum_{i=1}^{p-1} C_p^i x^{p-i} y^i.$$

Заметим, что $C_p^i = \frac{p!}{i!(p-i)!} = \frac{p(p-1)(p-2)\cdots(p-i+1)}{1\cdot 2\cdots i} \equiv 0 \bmod p, 0 < i < p.$ □

4.2.2. Минимальные многочлены

Как было показано выше, все элементы конечного поля $GF(p^m)$ удовлетворяют уравнению $x^{p^m} - x = 0$. Они, однако, могут быть корнями и многочленов меньшей степени.

Определение 4.7. *Минимальным многочленом* элемента $\beta \in GF(p^m)$ над $GF(p)$ называется нормированный многочлен $M(x) \in GF(p)[x]$ наименьшей возможной степени, такой что $M(\beta) = 0$.

Пусть $M(x)$ — минимальный многочлен некоторого $\beta \in GF(p^m)$.

Теорема 4.9. *$M(x)$ неприводим над $GF(p)$.*

Доказательство. Если $M(x) = M_1(x)M_2(x), \deg M_i(x) < \deg M(x), M_i(x) \in GF(p)[x]$ и $M(\beta) = 0$, то $M_1(\beta) = 0$ или $M_2(\beta) = 0$, откуда следует, что степень $M(x)$ не минимальна. □

Теорема 4.10. *Если $f(x) \in GF(p)[x]$ и $f(\beta) = 0$, то $M(x)|f(x)$.*

Доказательство. Разделив $f(x)$ на $M(x)$ с остатком, получим $f(x) = q(x)M(x) + r(x), \deg r(x) < \deg M(x)$, $0 = f(\beta) = q(\beta)0 + r(\beta)$. Если $r(x) \neq 0$, то оказалось бы, что степень $r(x) \in GF(p)[x]$, имеющего корень β, меньше степени минимального многочлена β. Следовательно, $r(x) = 0$. □

Теорема 4.11. $M(x)|(x^{p^m} - x)$

Доказательство. Утверждение непосредственно вытекает из предыдущей теоремы. □

Теорема 4.12. $\deg M(x) \leq m$

Доказательство. $GF(p^m)$ образует некоторое m-мерное линейное пространство над $GF(p)$. Следовательно, любые $m+1$ элементов $GF(p^m)$ линейно зависимы над $GF(p)$. В частности, $\forall\beta : \exists a_0, a_1, \ldots, a_m \in GF(p) : \sum_{i=0}^{m} a_i\beta^i = 0$. Таким образом, $M(x) = \sum_{i=0}^{m} a_i x^i \in GF(p)[x]$ имеет элемент β своим корнем. Возможно, этот многочлен можно разложить на сомножители меньшей степени. □

Теорема 4.13. *Если α — примитивный элемент $GF(p^m)$, то степень его минимального многочлена равна m.*

Доказательство. Пусть $\pi(x)$ — минимальный многочлен примитивного элемента α. Его степень d не превосходит m. Заметим, что $|GF(p)[x]/\pi(x)| = p^d$. Т.к. $\pi(x)$ неприводим, он порождает поле $GF(p^d)$. Но это поле содержит также элемент $\alpha : \pi(\alpha) = 0$, а, значит, и все поле $GF(p^m)$. Следовательно, $d \geq m$, откуда $d = m$. □

Минимальный многочлен примитивного элемента поля называется *примитивным*. Не все неприводимые многочлены являются примитивными.

Теорема 4.14. *Все конечные поля $GF(p^m)$ изоморфны.*

Доказательство. Пусть F и G — поля порядка p^m и пусть α — примитивный элемент поля F с минимальным многочленом $\pi(x)$. Из теоремы 4.11 следует, что $\pi(x)|(x^{p^m} - x)$. Следовательно, $\exists\beta \in G : \pi(\beta) = 0$. Теперь F можно рассматривать как множество многочленов от α степени

не более $m-1$, а G — как множество многочленов от β степени не более $m-1$. Тогда соответствие $\alpha \leftrightarrow \beta$ задает изоморфизм полей F и G. $\square$

Пример 4.14. Рассмотрим два способа задания поля $GF(2^3)$.

Через многочлен x^3+x+1	Через многочлен x^3+x^2+1
$(000)=0$	$(000)=0$
$(001)=1=\alpha^0$	$(001)=1=\gamma^0$
$(010)=\alpha$	$(010)=\gamma$
$(100)=\alpha^2$	$(100)=\gamma^2$
$(011)=\alpha^3=\alpha+1$	$(101)=\gamma^3=\gamma^2+1$
$(110)=\alpha^4=\alpha^2+\alpha$	$(111)=\gamma^4=\gamma^2+\gamma+1$
$(111)=\alpha^5=\alpha^2+\alpha+1$	$(011)=\gamma^5=\gamma+1$
$(101)=\alpha^6=\alpha^2+1$	$(110)=\gamma^6=\gamma^2+\gamma$

Заметим, что $\alpha^3+\alpha+1=0$ и $(\gamma^3)^3+\gamma^3+1=\gamma^2+\gamma^3+1=0$, т.е. $\pi(\alpha)=\pi(\gamma^3)=0$, где $\pi(x)=x^3+x+1$. Таким образом, соответствие $\alpha \leftrightarrow \gamma^3$ задает изоморфизм между этими двумя полями.

Пример 4.15. Многочлен $\pi(x)=x^4+x^3+x^2+x+1$ неприводим над полем $GF(2)$, но не является примитивным, т.к. выполняется сравнение $x^5 \equiv 1 \bmod (x^4+x^3+x^2+x+1)$. Тем не менее, факторкольцо $GF(2)[x]/\pi(x)$ является полем $GF(2^4)$. Примитивным элементом этого поля является многочлен $\alpha=x+1$ (т.е. класс многочленов, сравнимых с ним по модулю $\pi(x)$).

Лемма 4.2. *Если $n,r,s \in \mathbb{Z} : n \geq 2, r \geq 1, s \geq 1$, то $(n^s-1)|(n^r-1)$ тогда и только тогда, когда $s|r$*

Доказательство. Пусть $r=Qs+R, 0 \leq R < s$. Тогда $\frac{n^r-1}{n^s-1} = n^R\frac{n^{Qs}-1}{n^s-1} + \frac{n^R-1}{n^s-1}$. $n^{Qs}-1$ всегда делится на n^s-1. n^R-1 делится на n^s-1 только при $R=0$. $\square$

Теорема 4.15. *$GF(p^r)$ содержит подполе $GF(p^s)$ тогда и только тогда, когда $s|r$.*

Доказательство. Если $s|r$, то p^r-1 делится на p^s-1 и многочлен $x^{p^r}-x$ делится на $x^{p^s}-x$. Т.к. $GF(p^r)[x]$ является областью однозначного разложения на множители, а все его элементы являются корнями $x^{p^r}-x$, этот многочлен не имеет кратных корней. Следовательно, таковых нет и у $x^{p^s}-x$. Покажем, что корни последнего образуют поле.

Действительно, $\mathbf{0}, \pm\mathbf{1}$ являются его корнями. Если a,b — корни этого многочлена, то $a+b$ и ab также является таковыми, т.к. $(a+b)^{p^s}-(a+b)=a^{p^s}-a+b^{p^s}-b=\mathbf{0}$ и $(ab)^{p^s}-ab=(a^{p^s}-a)(b^{p^s}-b)+ba^{p^s}-ab+ab^{p^s}-ab=\mathbf{0}$. Таким образом, множество корней $x^{p^s}-x$ замкнуто по умножению и сложению. Свойства коммутативности и ассоциативности выполняются в силу того, что эти корни принадлежат также $GF(p^r)$. Если $aa^{-1}=\mathbf{1}$, то $(a^{-1})^{p^s}-a^{-1}=(a^{p^s})^{-1}-a^{-1}=a^{-1}-a^{-1}=\mathbf{0}$,

т.е. для всех ненулевых корней существует обратный элемент, также являющийся корнем. Следовательно, корни рассматриваемого многочлена образуют поле $GF(p^s)$

Если β — примитивный элемент $GF(p^s) \subset GF(p^r)$, то $\beta^{p^s-1} = \mathbf{1}, \beta^{p^r-1} = \mathbf{1}$, откуда $(p^s-1)|(p^r-1)$, что возможно только при $s|r$. □

Пример 4.16. Поле $GF(2^{12})$ содержит подполя $GF(2), GF(2^2), GF(2^4), GF(2^3), GF(2^6)$. $GF(2^4)$ не является подполем $GF(2^6)$. Однако $GF(2^2)$ является подполем $GF(2^6)$.

Конечные поля могут быть построены для любого простого числа p и любого $m > 0$. В силу вышедоказанной теоремы, расширенные поля могут строиться итеративно, т.е. на основе поля $GF(p^m)$ можно построить поле $GF(p^n)$.

Теорема 4.16. *Минимальные многочлены элементов $\beta \in GF(p^m)$ и β^p совпадают.*

Доказательство. Если $0 = M(\beta) = \sum_{i=0}^{\deg M(x)} a_i \beta^i$, то $0 = M(\beta)^p = \sum_{i=0}^{\deg M(x)} a_i^p \beta^{pi}$. Т.к. $M(x) \in GF(p)[x]$, $a_i \in GF(p)$ и $a_i^p = a_i$. Следовательно, $0 = M(\beta^p)$. □

Таким образом, множество ненулевых корней минимального многочлена имеет вид β^j, где j пробегает все элементы некоторого *циклотомического класса* относительно умножения на p по модулю $p^m - 1$

$$\{s, sp, sp^2, \ldots, sp^{m_s-1}\}, sp^{m_s} \equiv s \bmod (p^m - 1),$$

причем число m_s выбирается наименьшим возможным.

Пример 4.17. Циклотомические классы относительно умножения на 2 по модулю 15 имеют вид

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

Элементы, являющиеся корнями одного и того же минимального многочлена, называются *сопряженными*. Таким образом, все минимальные многочлены по модулю $p^m - 1$ (кроме x) могут быть получены как

$$M_s(x) = \prod_{j \in C_s} (x - \alpha^j),$$

где α — примитивный элемент $GF(p^m)$. Кроме того, из теоремы 4.11 следует, что неприводимые над $GF(p)$ сомножители в разложении многочлена $x^{p^m} - x$ также являются минимальными многочленами.

Теорема 4.17. *Многочлен $x^{p^m} - x$ равен произведению всех нормированных неприводимых над $GF(p)$ многочленов, степени которых делят m.*

Доказательство этой теоремы оставляется читателю в качестве упражнения.

4.3. Вычисления в конечных полях

При использовании конечных полей в приложениях необходимо реализовать, как минимум, операции сложения, вычитания, умножения и деления. В зависимости от характеристики поля, степени его расширения, доступных аппаратных возможностей и специфики решаемой задачи могут использоваться различные методы или их комбинации.

В случае простых полей вычисления могут производиться как с обычными целыми числами по модулю простого числа. Реализация арифметики расширенных полей может быть менее тривиальной. В общем случае расширенные поля могут рассматриваться как факторкольца многочленов, т.е. все операции могут производиться по правилам работы с многочленами с последующим приведением по модулю неприводимого многочлена. Однако этот способ в большинстве случаев оказывается крайне неэффективным.

4.3.1. Логарифмы

Как было показано выше, все ненулевые элементы конечного поля $GF(p^m)$ представимы как $\beta = \alpha^i$, где α — примитивный элемент поля. Следовательно, умножение и деление двух ненулевых элементов может быть выполнено как $\alpha^i \cdot \alpha^j = \alpha^{i+j}$ и $\frac{\alpha^i}{\alpha^j} = \alpha^{i-j}$. При практической реализации целесообразно не допускать неограниченного роста показателей степеней, поэтому их приходится приводить по модулю $p^m - 1$. Таким образом, ненулевые элементы конечного поля могут быть представлены в виде их логарифмов (или индексов) $i = \mathrm{ind}_\alpha \beta$. При этом предполагается $\mathrm{ind}(\mathbf{0}) = -\infty$. Однако данный метод не позволяет эффективно реализовать сложение и вычитание.

Проблема сложений может быть решена с помощью *логарифмов Зеча*, определяемых уравнением $1 + \alpha^n = \alpha^{Z(n)}$. В этом случае элементы также можно хранить в виде их обычных логарифмов, а для сложения использовать тождество $\alpha^m + \alpha^n = \alpha^m(1 + \alpha^{n-m}) = \alpha^{m+Z(n-m)}$. При этом

для вычитания придется вычислять $-\alpha^i = \alpha^{\mathrm{ind}_\alpha -1}\alpha^i$, т.е. необходимо заранее найти $\mathrm{ind}_\alpha(-1)$. Однако в практически наиболее важном случае полей характеристики 2 имеет место $\mathbf{1} = -\mathbf{1}$ и сложение совпадает с вычитанием.

4.3.2. Стандартный базис

Конечное поле $GF(p^m)$ может рассматриваться как линейное пространство $GF(p)^m$. Каждое конечномерное линейное пространство обладает базисом. Некоторые базисы конечных полей представляют особый практический и теоретический интерес.

Как было показано выше, во всяком конечном поле $GF(p^m)$ существует примитивный элемент α, являющийся корнем некоторого примитивного многочлена $\pi(x) \in GF(p)[x]$ степени m. Это означает, что набор элементов $\alpha^0, \ldots, \alpha^{m-1}$ линейно независим над $GF(p)$, а значит все элементы $GF(p^m)$ могут быть представлены как линейные комбинации $\alpha^i, i = 0..m-1$ с коэффициентами из $GF(p)$. Таким образом, элементы $\alpha^i, i = 0..m-1$ образуют базис, называемый *стандартным*. Каждый элемент $\beta \in GF(p^m) : \beta = \sum_{i=0}^{m-1} b_i\alpha^i$ однозначно представляется в виде вектора $(b_0, \ldots, b_{m-1})$. Тогда сложение и вычитание могут выполняться по правилам векторной алгебры в поле $GF(p)$. На практике в большинстве случаев m мало и $p = 2$, так что каждый элемент $GF(2^m)$ может быть представлен в виде одного машинного слова, что позволяет реализовать сложение в виде одной операции XOR.

Умножение может осуществляться как $\beta \cdot \gamma = \sum_{i=0}^{m-1} b_i(\alpha^i\gamma)$. Заметим, что элемент $\alpha^i\gamma$ может быть получен сдвигом векторного представления γ на i позиций с последующим приведением по модулю примитивного многочлена $\pi(x)$. Часто операции сдвига и приведения по модулю совмещают в регистрах сдвига с линейной обратной связью.

Однако на практике более эффективным приемом оказывается использование таблиц логарифмов и антилогарифмов, которые по заданному вектору $(b_0, \ldots, b_{m-1})$ позволяют найти $i = \mathrm{ind}_\alpha \beta$, а также выполнить обратное преобразование. В этом случае умножение сводится к нахождению по таблице логарифмов перемножаемых элементов, их сложению по модулю $p^m - 1$ и восстановлению с помощью таблицы антилогарифмов векторного представления произведения.

Упражнения

1. Построить поле $GF(2^4)$, задаваемое примитивным многочленом $x^4 + x + 1$, и вычислить $\frac{\alpha^2+\alpha^{20}\alpha^3}{\alpha^9}$, где α — примитивный элемент поля.
2. Вычислить $\frac{\beta^{293}+\beta^{395}\beta^{614}}{\beta^{712}}$, где β — примитивный корень многочлена $T^{10}+T^8+T^6+T^4+T^2+T+1$ по модулю 2.
3. Найти многочлен, неприводимый над полем $GF(2^2)$, и построить с помощью него поле $GF(2^4)$.

5. Коды Боуза-Чоудхури-Хоквингема

Использование линейных кодов позволяет значительно упростить реализацию операции кодирования. Однако сложность универсальных алгоритмов декодирования линейных блоковых кодов, рассмотренных выше, в большинстве случаев оказывается слишком высокой для практического использования. С другой стороны, процедура построения хороших линейных кодов, приведенная в доказательстве теоремы 2.7, также имеет слишком большую сложность. Как будет показано в этой и последующих главах, наложение дополнительных ограничений на структуру кода позволяет не только легко строить коды с заданными параметрами, но и эффективно их декодировать.

5.1. Циклические коды

5.1.1. Основные понятия

Определение 5.1. Блоковый код называется *циклическим*, если он линеен и всякий циклический сдвиг любого его кодового слова дает другое кодовое слово этого же кода.

Кодовые слова $(c_0, \ldots, c_{n-1})$ циклических кодов удобно представлять в виде многочленов $c(x) = \sum_{i=0}^{n-1} c_i x^i$, где x — формальная переменная. Такая запись позволяет описать операцию циклического сдвига на одну позицию как $xc(x) \equiv c_{n-1} + c_0 x + c_1 x^2 + \ldots + c_{n-2} x^{n-1} \bmod x^n - 1$. Таким образом, вводится соответствие между кодовыми словами и остатками от деления многочленов на $x^n - 1$. Множество последних образует факторкольцо $\mathbb{F}[x]/(x^n - 1)$. В дальнейшем будут одновременно использоваться и векторное, и многочленное представления кодовых слов, причем понятие кодового многочлена будет использоваться наравне с понятием кодового слова.

Теорема 5.1. *Подмножество $\mathcal{C} \subset \mathbb{F}[x]/(x^n - 1)$ образует циклический код тогда и только тогда, когда $\mathcal{C}$ образует группу по сложению и $\forall a(x) \in \mathbb{F}[x]/(x^n - 1) : c(x) \in \mathcal{C} \Rightarrow a(x)c(x) \in \mathcal{C}$.*

Доказательство. Предположим, что $\mathcal{C}$ обладает указанными свойствами. Первое из них означает, что $\mathcal{C}$ является линейным подпространством $\mathbb{F}^n$ (т.е. линейным кодом), а второе означает, в частности, что всякий

циклический сдвиг кодового слова (т.е. $a(x) = x^i$) также является кодовым словом.

Обратно, всякий линейный код образует группу по сложению. Операцию умножения $c(x)$ на $a(x)$ можно представить как взвешенную сумму различных циклических сдвигов кодового слова, задаваемого $c(x)$, которая также является кодовым словом в силу линейности кода. □

Выберем ненулевой многочлен $g(x) \in \mathcal{C}$ наименьшей возможной степени r такой, что $g_r = 1$. Он является единственным и называется *порождающим многочленом* кода. Все $c(x) \in \mathcal{C}$ делятся на него. Действительно, если предположить противное, т.е. существование $c(x) \in \mathcal{C}$, такого что $c(x) = a(x)g(x) + r(x), \deg r(x) < \deg g(x), r(x) \neq 0$, то получим, что $r(x)$ также принадлежит $\mathcal{C}$. Это противоречит условию минимальности степени $g(x)$.

Так как $\mathcal{C}$ содержит многочлены степени не более $n-1$, операцию кодирования можно определить как $c(x) = u(x)g(x)$, где $u(x)$ — информационный многочлен такой, что $\deg u(x) < n - r$. Таким образом, размерность циклического кода равна $k = n - \deg g(x)$. Соответствующая порождающая матрица имеет вид

$$G = \begin{pmatrix} g_0 & g_1 & \cdots & g_{n-k} & 0 & \cdots & 0 & 0 \\ 0 & g_0 & g_1 & \cdots & g_{n-k} & 0 & \cdots & 0 \\ 0 & 0 & g_0 & g_1 & \cdots & g_{n-k} & 0 & \vdots \\ \vdots & \ddots & & \ddots & \ddots & & \ddots & \\ 0 & \cdots & 0 & 0 & g_0 & \cdots & g_{n-k-1} & g_{n-k} \end{pmatrix}.$$

Теорема 5.2. *Циклический код длины n с порождающим многочленом $g(x)$ существует тогда и только тогда, когда $g(x)|(x^n - 1)$.*

Доказательство. Рассмотрим циклический код с порождающим многочленом $g(x)$ и разделим $x^n - 1$ на $g(x)$ с остатком, т.е. представим $x^n - 1 = q(x)g(x) + r(x), \deg r(x) < \deg g(x)$. Многочлен $a(x) \equiv q(x)g(x) \bmod x^n - 1$ является кодовым, так как получен из кодового многочлена $g(x)$. Но в этом случае $r(x) \equiv -a(x) \bmod x^n - 1$, т.е. $r(x)$ также является кодовым многочленом. Это возможно, только если $r(x) = 0$.

Обратно, можно построить циклический код как множество $c(x) = a(x)g(x)$ для какого-нибудь делителя $x^n - 1$. □

Таким образом, существует такой $h(x)$, что $x^n - 1 = h(x)g(x)$. Многочлен $h(x)$ называется *проверочным*, так как для любого $c(x) \in \mathcal{C}$ справедливо $c(x)h(x) = a(x)g(x)h(x) \equiv 0 \bmod x^n - 1$. Это сравнение можно

переписать как $c(x)h(x) = a(x)(x^n - 1)$. Так как $\deg a(x) < k$, в многочлене $c(x)h(x)$ коэффициенты при $x^k, \ldots, x^{n-1}$ равны нулю. Таким образом, кодовые слова циклического кода удовлетворяют проверочным уравнениям

$$\sum_{i=0}^{k} h_i c_{l-i} = 0, k \le l \le n-1.$$

Это позволяет записать проверочную матрицу кода в виде

$$H = \begin{pmatrix} h_k & h_{k-1} & \cdots & h_0 & 0 & \cdots & 0 & 0 \\ 0 & h_k & h_{k-1} & \cdots & h_0 & 0 & \cdots & 0 \\ 0 & 0 & h_k & h_{k-1} & \cdots & h_0 & 0 & \vdots \\ \vdots & \ddots & & \ddots & \ddots & & \ddots & \\ 0 & \cdots & 0 & 0 & h_k & \cdots & h_1 & h_0 \end{pmatrix}.$$

На практике желательно иметь систематический кодер. Потребуем, чтобы информационные символы размещались в позициях $n-k \ldots, n-1$ кодового слова, т.е. будем строить кодовый многочлен как

$$c(x) = a(x) + x^{n-k}u(x). \tag{5.1}$$

Многочлен $a(x)$, соответствующий проверочным символам, должен быть подобран таким образом, чтобы $c(x)$ делился на $g(x)$. Очевидное решение состоит в выборе $a(x) = -x^{n-k}u(x) \bmod g(x)$, где $u(x)$ — многочлен, задающий кодируемую информационную последовательность. Заметим, что $\deg a(x) < n - k = \deg g(x)$.

Существуют и другие методы кодирования циклических кодов, соответствующие иным формам порождающей матрицы. Однако корректирующая способность и алгоритмы исправления ошибок не зависят от выбора метода кодирования.

5.1.2. Декодер Меггитта

Рассмотрим декодирование циклического кода в метрике Хэмминга. Принятая последовательность может быть представлена как

$$y(x) = u(x)g(x) + e(x),$$

где $e(x)$ — многочлен, соответствующий вектору ошибок. Введем синдромный многочлен $S(x) \equiv y(x) \equiv e(x) \bmod g(x)$. Рассмотрим произвольный циклический сдвиг вектора ошибки, который можно представить как $e'(x) = a(x)(x^n - 1) + x^i e(x)$, где $a(x)$ — некоторый многочлен.

В силу свойств сравнений и теоремы 5.2 получим, что $e'(x) \equiv x^i e(x) \equiv x^i S(x) \bmod g(x)$. Таким образом, синдромы циклических сдвигов вектора ошибки тесно связаны между собой. За счет этого можно уменьшить размер таблицы, используемой синдромным декодером (см. разд. 2.2.2). Из нее можно удалить все записи $[S^{(i)}(x), e^{(i)}(x)]$, соответствующие лидерам смежных классов $e^{(i)}(x)$, которые являются циклическими сдвигами друг друга. Процедура декодирования состоит в вычислении $S(x)$ и поиске в таблице записи, первая компонента которой была бы равна $S^{(i)}(x) \equiv x^j S(x) \bmod g(x), 0 \leq j < n$. Вектор ошибки равен $e(x) = x^{n-j_0} e^{(i)}(x) \bmod x^n - 1$, где j_0 — то значение j, на котором поиск завершился успешно.

Пример 5.1. Рассмотрим декодирование циклического $(15, 11, 3)$ кода с порождающим многочленом $g(x) = x^4 + x + 1$. Код способен исправлять одну ошибку, т.е. общее число исправимых векторов ошибки равно 15. Используя вышеописанный подход, достаточно построить лишь таблицу с одной записью, например $[x^3 + 1, x^{14}]$ (так как $x^3 + 1 \equiv x^{14} \bmod g(x)$). Предположим, что принятый многочлен имеет вид $y(x) = x^{14} + x^2 + 1$. Синдромный многочлен равен $S(x) \equiv y(x) \equiv x^3 + x^2 \bmod g(x)$. Рассматривая все возможные многочлены $x^j S(x) \bmod g(x)$, обнаружим, что $x^3 + 1 \equiv x^8 S(x) \bmod g(x)$, откуда $e(x) = x^7 \cdot x^{14} \equiv x^6 \bmod x^{15} - 1$, т.е. исправленный кодовый многочлен равен $c(x) = y(x) - e(x) = x^{14} + x^6 + x^2 + 1$.

5.1.3. Корни порождающего многочлена

Как было показано выше, все кодовые многочлены циклического кода над $\mathbb{F}_q$ делятся на порождающий многочлен $g(x)$, который, в свою очередь, является делителем $x^n - 1$. Ограничимся рассмотрением случая, когда многочлен $x^n - 1$ свободен от квадратов, т.е. в разложении $x^n - 1 = \prod_{i=1}^{l} f_i(x)$, где $f_i(x)$ — неприводимые над $\mathbb{F}_q$ многочлены, отсутствуют повторяющиеся элементы. Многочлен $g(x)$ может быть представлен как произведение некоторых из этих многочленов, откуда вытекает, что всего существует $2^l - 2$ нетривиальных циклических кодов длины n.

Циклический код длины n над $\mathbb{F}_q$ называют *примитивным*, если $n = q^m - 1$ для некоторого m. Известно, что в этом случае $x^{q^m-1} - 1 = \prod_{i=1}^{q^m-1}(x - \beta_j)$, где β_j — различные ненулевые элементы конечного поля $\mathbb{F}_{q^m}$, а неприводимые многочлены $f_i(x) \in \mathbb{F}_q[x]$ являются *минимальными многочленами* соответствующих элементов из $\mathbb{F}_{q^m}$, т.е. многочленами наименьшей возможной степени с коэффициентами из $\mathbb{F}_q$, имеющими эти элементы своими корнями. Несложно убедиться в том, что каждый

такой многочлен имеет вид

$$f_i(x) = \prod_{j=0}^{m_i-1} (x - \beta_i^{q^j}), \tag{5.2}$$

где m_i — наименьшее положительное число такое, что $\beta_i = \beta_i^{q^{m_i}}$. С другой стороны, все ненулевые элементы конечного поля могут быть представлены как степени некоторого примитивного элемента α. Тогда $f_i(x) = \prod_{j \in C_i}(x - \alpha^j)$, где $C_i = \left\{s_i, s_i q, s_i q^2, \ldots, s_i q^{m_i - 1}\right\}$ — *циклотомический класс* над $\mathbb{F}_q$ по модулю n, причем $s_i q^{m_i} \equiv s_i \bmod n$. Элементы конечного поля, минимальные многочлены которых совпадают, называются сопряженными.

Вышеописанный подход может быть использован и в тех случаях, когда циклический код не является примитивным.

Теорема 5.3. *Пусть дано конечное поле $\mathbb{F}_q$. Если n и q взаимно просты, то существует m такое, что $(x^n - 1) | (x^{q^m - 1} - 1)$ и многочлен $x^n - 1$ имеет n различных корней в $\mathbb{F}_{q^m}$.*

Доказательство. Докажем вначале существование $m : n | (q^m - 1)$. Деля q^i на n с остатком, получим $q^i = Q_i n + s_i, 1 \leq i \leq n+1, 0 \leq s_i < n$. Найдется по крайней мере одна пара $(i, j) : i > j$, такая что $s_i = s_j$. Тогда $q^j(q^{i-j} - 1) = (Q_i - Q_j)n$. Так как $(q, n) = 1$, получим, что $n | (q^m - 1)$, где $m = i - j$.

Таким образом, возможно разложение $x^{q^m - 1} - 1 = x^{nr} - 1 = (x^n - 1)(1 + x^n + x^{2n} + \ldots + x^{n(r-1)})$, где $r = \frac{q^m - 1}{n}$. В силу свойств конечного поля $\mathbb{F}_{q^m}$, все корни многочлена $x^{q^m - 1} - 1$ различны. □

Если $\beta_1, \ldots, \beta_{n-k} \in \mathbb{F}_{q^m}$ — корни порождающего многочлена $g(x) \in \mathbb{F}_q[x]$, то эти элементы являются также и корнями всех кодовых многочленов циклического кода. Таким образом, все кодовые слова удовлетворяют системе линейных уравнений

$$\sum_{i=0}^{n-1} \beta_j^i c_i = 0, 1 \leq j \leq n - k. \tag{5.3}$$

Это позволяет определить проверочную матрицу кода над расширенным полем как $H = \begin{pmatrix} \beta_1^0 & \beta_1^1 & \ldots & \beta_1^{n-1} \\ \beta_2^0 & \beta_2^1 & \ldots & \beta_2^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_{n-k}^0 & \beta_{n-k}^1 & \ldots & \beta_{n-k}^{n-1} \end{pmatrix}$.

В силу того, что порождающий многочлен циклического кода равен произведению некоторых минимальных многочленов, для его задания достаточно лишь указать по одному корню β_i каждого из этих многочленов; все их сопряженные элементы $\beta_i^{q^j}$ окажутся его корнями автоматически в силу (5.2). Это позволяет исключить некоторое число строк из вышеприведенной проверочной матрицы. Вместе с тем, для эффективной работы рассматриваемых ниже алгоритмов декодирования может быть необходимо использование полной проверочной матрицы. Необходимо отметить, что эта матрица содержит элементы из некоторого расширенного конечного поля $\mathbb{F}_{q^m}$, хотя рассматривается код $\mathcal{C}$ над $\mathbb{F}_q$. В принципе, она задает также некоторый код $\mathcal{C}'$ над $\mathbb{F}_{q^m}$, причем исходный код $\mathcal{C}$ является *ограничением кода* $\mathcal{C}'$ *на подполе* $\mathbb{F}_q$, т.е. представляет собой множество таких кодовых слов $\mathcal{C}'$, все элементы которых принадлежат $\mathbb{F}_q$.

Пусть дан некоторый базис $\gamma_0, \ldots, \gamma_{m-1}$ конечного поля $\mathbb{F}_{q^m}$. Разложим по этому базису все элементы $\beta_j^i = \sum_{l=0}^{m-1} b_{ijl}\gamma_l, b_{ijl} \in \mathbb{F}_q$. Тогда система (5.3) может быть переписана как $\sum_{i=0}^{n-1} b_{ijl}c_i = 0$. Элементы $b_{ijl}, 1 \le j \le n-k, 0 \le i < n, 0 \le l < m$ также образуют $(n-k)m \times n$ проверочную матрицу кода $\mathcal{C}$. Зачастую эта матрица содержит линейно зависимые строки. Тем не менее, ее можно использовать для того, чтобы получить оценку для размерности циклического кода. Последняя не может быть меньше, чем $n - (n-k)m$.

Пример 5.2. Рассмотрим циклический $(7, 4, 3)$ код с порождающим многочленом $g(x) = x^3 + x + 1$. Его корнями являются элементы $\alpha, \alpha^2, \alpha^4 \in \mathbb{F}_8$, где α — примитивный элемент[1] этого поля. Таким образом, проверочная матрица этого кода над расширенным полем имеет вид $H = \begin{pmatrix} \alpha^0 & \alpha^1 & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \\ \alpha^0 & \alpha^2 & \alpha^4 & \alpha^6 & \alpha^1 & \alpha^3 & \alpha^5 \\ \alpha^0 & \alpha^4 & \alpha^1 & \alpha^5 & \alpha^2 & \alpha^6 & \alpha^3 \end{pmatrix}$. Раскладывая ее элементы по стандартному базису $(\alpha^0, \alpha^1, \alpha^2)$, получим еще одну проверочную матрицу этого кода $H' = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}$. Несложно убедиться в том, что ее

[1] Многочлен $g(x)$ оказался примитивным и может быть использован для построения $\mathbb{F}_{2^3}$.

строки 4 – 9 являются линейными комбинациями строк 1 – 3, а потому могут быть исключены. Это свидетельствует об избыточности строк 2–3 матрицы H, которые соответствуют сопряженным корням $g(x)$. Столбцы, образованные элементами первых трех строк матрицы H', представляют собой все различные двоичные векторы длины 3. Таким образом, этот код эквивалентен коду Хэмминга.

5.2. Коды БЧХ

5.2.1. Свойства и конструкция кодов

Теорема 5.4 (*Граница Боуза-Чоудхури-Хоквингема*)**.** *Пусть дан (n, k, d) циклический код над $\mathbb{F}_q$ с порождающим многочленом $g(x)$. Пусть $\mathbb{F}_{q^m}$ — наименьшее расширение поля, содержащее все корни $g(x)$, и $\beta \in \mathbb{F}_{q^m}$ — образующий элемент мультипликативной циклической группы порядка n. Если для некоторого b корнями $g(x)$ являются элементы $\beta^b, \beta^{b+1} \ldots, \beta^{b+\delta-2}$, то $d \geq \delta$.*

Доказательство. Все кодовые слова рассматриваемого кода удовлетворяют системе линейных уравнений

$$\underbrace{\begin{pmatrix} \beta^0 & \beta^b & \ldots & \beta^{b(n-1)} \\ \beta^0 & \beta^{(b+1)} & \ldots & \beta^{(b+1)(n-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \beta^0 & \beta^{(b+\delta-2)} & \ldots & \beta^{(b+\delta-2)(n-1)} \end{pmatrix}}_{H} c^T = 0. \tag{5.4}$$

Предположим, что существует ненулевое кодовое слово веса $w \leq \delta - 1$. Пусть $i_1, \ldots, i_w$ — номера позиций его ненулевых элементов. Их значения могут быть найдены как решение системы

$$\begin{pmatrix} \beta^{bi_1} & \beta^{bi_2} & \ldots & \beta^{bi_w} \\ \beta^{(b+1)i_1} & \beta^{(b+1)i_2} & \ldots & \beta^{(b+1)i_w} \\ \vdots & \vdots & \ddots & \vdots \\ \beta^{(b+\delta-2)i_1} & \beta^{(b+\delta-2)i_2} & \ldots & \beta^{(b+\delta-2)i_w} \end{pmatrix} \begin{pmatrix} c_{i_1} \\ c_{i_2} \\ \vdots \\ c_{i_w} \end{pmatrix} = 0. \tag{5.5}$$

Рассмотрим подматрицу H' этой системы, составленную из первых ее w строк. Ее определитель равен

$$\det(H') = \beta^{i_1 b + \ldots + i_w b} \det \begin{pmatrix} 1 & 1 & \ldots & 1 \\ \beta^{i_1} & \beta^{i_2} & \ldots & \beta^{i_w} \\ \vdots & \vdots & \ddots & \vdots \\ \beta^{i_1(w-1)} & \beta^{i_2(w-1)} & \ldots & \beta^{i_w(w-1)} \end{pmatrix}.$$

Последний определитель известен как определитель Вандермонда. Он отличен от нуля, если все β^{i_j} различны. Это свойство обеспечивается за счет того, что порядок β равен n и $0 \le i_1 < i_2 < \ldots < i_w < n$. Следовательно, матрица H' обратима и единственным решением системы (5.5) является $c_{i_1} = \ldots = c_{i_w} = 0$, т.е. код не содержит слов веса меньше δ. □

Полученную границу можно использовать для построения кодов с заданным минимальным расстоянием.

Определение 5.2. *Кодом БЧХ* над полем $\mathbb{F}_q$ длины n с конструктивным расстоянием δ называется наибольший q-ичный циклический код, корни порождающего многочлена которого включают $\beta^b, \ldots, \beta^{b+\delta-2}$, где $\beta \in \mathbb{F}_{q^m}$ — элемент порядка n, $n|(q^m - 1)$.

Коды, полученные при $b = 1$, называются *кодами БЧХ в узком смысле*. Коды, полученные при $m = 1$, называются *кодами Рида-Соломона*. Величины $\beta^{b+i}, 0 \le i \le \delta - 2$, называются нулями кода.

Таким образом, построение кода БЧХ сводится к нахождению порождающего многочлена $g(x) \in \mathbb{F}_q[x]$, имеющего корни $\beta^{b+i} \in \mathbb{F}_{q^m}$. С учетом результатов разд. 5.1.3, можно записать

$$g(x) = LCM(f_{\beta^b}(x), f_{\beta^{b+1}}(x), \ldots, f_{\beta^{b+\delta-2}}(x)), \tag{5.6}$$

где $f_\gamma(x)$ — минимальный многочлен элемента α над $\mathbb{F}_q$, и $LCM(\cdot)$ — наименьшее общее кратное многочленов. Фактически оно равно произведению всех *различных* указанных многочленов.

Разложим все элементы проверочной матрицы H над $\mathbb{F}_{q^m}$ (см. (5.4)) в каком-нибудь его базисе. Это приведет к $((\delta - 1)m) \times n$ матрице H', содержащей элементы $\mathbb{F}_q$. Таким образом, размерность кода БЧХ может быть оценена как $k \ge n - m(\delta - 1)$. В случае двоичных кодов для любого $c(x) \in \mathbb{F}_2[x]$ из равенства $c(\beta^{(b+i)}) = 0$ всегда вытекает $c(\beta^{2(b+i)}) = 0$. Поэтому при $0 \le b \le 1$ не менее половины строк матрицы H' можно исключить как линейно зависимые. Это приводит к уточненной оценке для размерности кода $k \ge n - m \left\lfloor \frac{\delta-1}{2} \right\rfloor$. Известна точная формула для размерности кодов БЧХ в узком смысле [8], которая, однако, достаточно сложна для практического использования. Практически более полезными оказываются вышеприведенные простые оценки, а также таблицы кодов БЧХ [9], содержащие не только значения (n, k, d), но и готовые порождающие многочлены. Впрочем, современные системы компьютерной алгебры позволяют достаточно легко построить такие многочлены, руководствуясь лишь правилом (5.6).

В некоторых случаях истинное минимальное расстояние кодов БЧХ может несколько превышать конструктивное. Однако рассматриваемые ниже алгоритмы декодирования не способны непосредственно использовать этот факт для увеличения числа исправляемых ошибок, вследствие чего на практике такую возможность обычно игнорируют. С другой стороны, известно, что не существует ансамблей двоичных примитивных кодов БЧХ, для которых одновременно выполнялись бы условия $\lim_{n\to\infty} d/n > 0$ и $\lim_{n\to\infty} k/n > 0$. Поэтому коды БЧХ являются асимптотически плохими [8]. Вместе с тем, на небольших длинах (до нескольких сотен) они относятся к числу наилучших известных.

Пример 5.3. Рассмотрим построение $(15, 7, 5)$ примитивного кода БЧХ в узком смысле. Пусть α — примитивный элемент поля $\mathbb{F}_{2^4}$, являющийся корнем примитивного многочлена x^4+x+1. Выберем $\beta = \alpha$. Корнями порождающего многочлена должны быть величины $\beta, \beta^2, \beta^3, \beta^4$. Тогда $f_\beta(x) = f_{\beta^2}(x) = f_{\beta^4}(x) = f_{\beta^8}(x) = x^4 + x + 1$, $f_{\beta^3} = \prod_{j=1}^{4}(x - \beta^{3\cdot 2^j}) = x^4+x^3+x^2+x+1$. Таким образом, $g(x) = x^8+x^7+x^6+x^4+1$. Так как порождающий многочлен кода соответствует также одному из кодовых слов, видно, что минимальное расстояние этого кода в точности равно 5.

5.2.2. Декодирование кодов БЧХ

Рассмотрим задачу декодирования кодов БЧХ в метрике Хэмминга. Пусть $y(x) = \underbrace{a(x)g(x)}_{c(x)} + e(x)$ — многочлен, соответствующий принятому зашумленному кодовому слову. Построим вектор синдрома $(S_0, \ldots, S_{\delta-2})$, где

$$S_i = y(\beta^{b+i}) = e(\beta^{b+i}), \tag{5.7}$$

и δ — конструктивное расстояние кода. Предположим, что ошибки произошли в позициях $j_1, \ldots, j_w$, т.е. $e_{j_l} \neq 0$. Это означает, что $S_i = \sum_{l=1}^{w} e_{j_l}\beta^{(b+i)j_l}$. Пусть $X_l = \beta^{j_l}$ — *локатор* соответствующей ошибки, и $Y_l = e_{j_l}$ — ее значение. Тогда получим

$$S_i = \sum_{l=1}^{w} Y_l X_l^{b+i}. \tag{5.8}$$

Решение этой нелинейной системы $\delta - 1$ уравнений с $2w$ неизвестными Y_l, X_l позволяет найти как местоположение искаженных символов, так и их значения. Однако непосредственно решить ее достаточно сложно.

Введем *многочлен локаторов ошибок*

$$\Lambda(x) = \prod_{l=1}^{w}(1 - X_l x) = 1 + \sum_{l=1}^{w} \Lambda_l x^l. \tag{5.9}$$

Корнями его являются величины, обратные к локаторам ошибок, т.е. имеет место $\Lambda(X_l^{-1}) = 1 + \sum_{j=1}^{w} \Lambda_j X_l^{-j} = 0, 1 \le l \le w$. Домножая эти тождества на $Y_l X_l^{b+i+w}, 0 \le i \le \delta - 2$, получим

$$0 = Y_l X_l^{b+i+w} + Y_l \sum_{j=1}^{w} \Lambda_j X_l^{b+i+w-j}.$$

Суммирование их по l приводит с учетом (5.8) к системе уравнений

$$-S_{i+w} = \sum_{j=1}^{w} \Lambda_j S_{i+w-j}, 0 \le i \le \delta - 2 - w. \tag{5.10}$$

Таким образом, вместо нелинейной системы уравнений (5.8) можно решить систему линейных неоднородных алгебраических уравнений (5.10) относительно коэффициентов многочлена $\Lambda(x)$, найти его корни X_l^{-1} (нелинейная операция!) и решить систему (5.8), которая является линейной относительно Y_l. Некоторые затруднения вызывает лишь то, что неизвестна размерность w системы (5.10).

Теорема 5.5. *Матрица* $M^{(\mu)} = \begin{pmatrix} S_0 & S_1 & S_2 & \dots & S_{\mu-1} \\ S_1 & S_2 & S_3 & \dots & S_{\mu} \\ S_2 & S_3 & S_4 & \dots & S_{\mu+1} \\ \vdots & \dots & \vdots & \ddots & \vdots \\ S_{\mu-1} & S_{\mu} & S_{\mu+1} & \dots & S_{2\mu-2} \end{pmatrix}$ *обратима, если* $\mu = w$, *и вырождена, если* $\mu > w$.

Доказательство. Каждый элемент этой матрицы с учетом (5.8) может быть представлен как $M_{ij}^{(\mu)} = S_{i+j} = \sum_{l=1}^{\mu} Y_l X_l^{b+i+j} = \sum_{l=1}^{\mu} X_l^i Y_l X_l^b X_l^j = (ABA^T)_{ij}$, где $X_l = 0, w < l \le \mu$, $A = \begin{pmatrix} 1 & 1 & \dots & 1 \\ X_1 & X_2 & \dots & X_{\mu} \\ \vdots & \vdots & \ddots & \vdots \\ X_1^{\mu-1} & X_2^{\mu-1} & \dots & X_{\mu}^{\mu-1} \end{pmatrix}$, $B = \mathrm{diag}(Y_1 X_1^b, \dots, Y_{\mu} X_{\mu}^b)$. Матрица A является матрицей Вандермонда, которая обратима, если все X_l различны. Матрица B обратима, если все $X_l \ne 0$. Произведение нескольких квадратных матриц $M^{(\mu)} = ABA^T$ вырождено тогда и только тогда, когда вырожден хотя бы один из его сомножителей. □

Эта теорема позволяет сформулировать *алгоритм Питерсона-Горенстайна-Цирлера* декодирования кодов БЧХ:

1. Вычислить компоненты вектора синдрома S_i в соответствии с (5.7).
2. Если $S_0 = \ldots = S_{\delta-2} = 0$, завершить работу.
3. Положить $w \leftarrow \lfloor (\delta - 1)/2 \rfloor$.
4. Проверить обратимость матрицы $M^{(w)}$.
5. Если $M^{(w)}$ вырождена, положить $w \leftarrow w - 1$, перейти к п. 4.
6. Найти коэффициенты многочлена локаторов ошибок как решение системы

$$M^{(w)}(\Lambda_w \, \Lambda_{w-1} \, \ldots \, \Lambda_1)^T = (-S_w \, - S_{w+1} \, \ldots \, - S_{2w-1})^T.$$

7. Найти корни $X_l^{-1}, 1 \le l \le w$, многочлена $\Lambda(x)$.
8. Найти значения ошибок как решение системы (5.8) относительно Y_l.
9. Если $X_l = \beta^{j_l}$, вычесть Y_l из y_{j_l}.

Заметим, что этап нахождения значений ошибок не нужен в случае двоичных кодов.

Ясно, что максимальное число ошибок, которое может быть исправлено с помощью этого метода, не превосходит $\lfloor (\delta - 1)/2 \rfloor$. Если истинное минимальное расстояние кода превышает δ, принципиально возможно исправление и большего числа ошибок. Для некоторых кодов известны "надстройки" над вышеописанным алгоритмом, позволяющие сделать это, но универсальные методы декодирования циклических кодов и, в частности, кодов БЧХ до их истинного минимального расстояния, которые обладали бы при этом приемлемой сложностью, пока не известны.

Число операций, выполняемых на шаге 4 в случае его реализации с помощью алгоритма Гаусса, составляет $O(w^3)$, что делает его наиболее трудоемким этапом этого алгоритма. Ниже будут приведены чрезвычайно эффективные алгоритмы, позволяющие выполнить шаги 3–6 со сложностью $O(w^2)$. Этап нахождения значений ошибок также может быть упрощен. Для этого введем синдромный многочлен $S(x) = \sum_{i=0}^{\delta-2} S_i x^i = \sum_{l=1}^{w} Y_l X_l^b \sum_{i=0}^{\delta-2} (X_l x)^i$. Заметим, что $(1 - X_l x) \sum_{i=0}^{\delta-2} (X_l x)^i = 1 - (X_l x)^{\delta-1} \equiv 1 \bmod x^{\delta-1}$. Таким образом, $S(x) \equiv \sum_{l=1}^{w} \frac{Y_l X_l^b}{(1 - X_l x)}$. Введем *многочлен значений ошибок*

$$\Gamma(x) = \sum_{l=1}^{w} Y_l X_l^b \prod_{p \ne l} (1 - X_p x) = \Lambda(x) \sum_{l=1}^{w} \frac{Y_l X_l^b}{1 - X_l x}.$$

Видно, что выполняется сравнение

$$\Gamma(x) \equiv \Lambda(x) S(x) \bmod x^{\delta-1}, \deg \Gamma(x) < \deg \Lambda(x) \le \lfloor (\delta - 1)/2 \rfloor . \quad (5.11)$$

Оно называется *ключевым уравнением декодирования кодов БЧХ*. Зная $\Lambda(x)$, с помощью него можно найти $\Gamma(x)$. Впрочем, в разд. 5.2.5 будет описан метод, позволяющий одновременно найти из этого сравнения $\Lambda(x)$ и $\Gamma(x)$. Из определения многочлена значений ошибок и условия (5.11) вытекает, что правильное решение задачи декодирования удовлетворяет $\Gamma_j = \sum_{i=0}^{w} \Lambda_i S_{j-i} = 0, w \le j \le \delta - 2$, т.е. (5.10) является следствием ключевого уравнения.

Непосредственной проверкой легко убедиться, что значения ошибок могут быть найдены как (*алгоритм Форни*)

$$Y_l = \frac{X_l^{-b}\Gamma(X_l^{-1})}{\prod_{j \ne l}(1 - X_j X_l^{-1})}. \tag{5.12}$$

Видно, что сложность алгоритма Форни равна $O(w^2)$.

5.2.3. Спектральное описание декодера кодов БЧХ

Несколько более простое описание (а в некоторых случаях и ведущее к более эффективной реализации) алгоритма декодирования кодов БЧХ можно получить, воспользовавшись спектральными методами [1]. Напомним, что *дискретным преобразованием Фурье* (ДПФ) вектора $(f_0, \ldots, f_{n-1})$ (или многочлена $f(x) = \sum_{i=0}^{n-1} f_i x^i$) называется вектор $(F_0, \ldots, F_{n-1})$, где $F_j = f(\beta^j) = \sum_{i=0}^{n-1} f_i \beta^{ij}$, причем элемент β (ядро ДПФ) должен иметь порядок n. Отметим, что в конечных полях такой элемент удается найти далеко не для всех n. Известно множество эффективных алгоритмов вычисления ДПФ [1, 11, 12, 22]. Обратное дискретное преобразование Фурье определяется как $f_i = \frac{1}{n}\sum_{j=0}^{n-1} f_j \beta^{-ij}$. Важнейшим свойством ДПФ является

Теорема 5.6 (*о свертке*)**.** *Предположим, что существует ДПФ длины n и $e_i = f_i g_i, 0 \le i < n$. Тогда*

$$E_j = \frac{1}{n}\sum_{k=0}^{n-1} F_{j-k \bmod n} G_k, 0 \le j < n.$$

Пусть $y(x) = a(x)g(x) + e(x)$ — многочлен, соответствующий принятому зашумленному кодовому слову. Вычисляя его ДПФ порядка n, получим

$$Y_i = y(\beta^i) = A_i G_i + E_i, 0 \le i < n.$$

Заметим, что из конструкции кодов БЧХ вытекает, что $G_i = g(\beta^i) = 0, b \le i \le b + \delta - 2$. Таким образом, некоторые компоненты вектора Y_i

зависят исключительно от вектора ошибки. i-ая компонента обратного ДПФ многочлена локаторов ошибок (5.9) $\lambda_i = \Lambda(\beta^{-i})$ равна нулю тогда и только тогда, когда $e_i \neq 0$. Таким образом, справедливо равенство $e_i\lambda_i = 0, 0 \leq i < n$. В силу теоремы о свертке, это тождество можно переписать как $\sum_{j=0}^{n-1} \Lambda_j E_{i-j} = 0, 0 \leq i < n$. Учитывая, что $\Lambda_0 = 1, \deg\Lambda(x) = w$, получим

$$E_i = -\sum_{j=1}^{t} \Lambda_j E_{i-j}, 0 \leq i < n. \tag{5.13}$$

Система уравнений (5.10) вытекает из этих тождеств в силу того, что $S_i = Y_{b+i} = E_{b+i}, 0 \leq i \leq \delta - 2$. Заметим, что после того, как найдены коэффициенты Λ_j, можно подставить компоненты вектора синдрома в (5.13) и рекуррентно вычислить все E_i, после чего найти вектор ошибки как их обратное ДПФ. В некоторых случаях такой подход может оказаться несколько проще, чем описанный в разд. 5.2.2 метод, основанный на поиске корней многочлена локаторов ошибок и нахождении их значений с помощью алгоритма Форни.

Спектральная интерпретация позволяет просто получить обобщение алгоритма декодирования на случай исправления ошибок и стираний. Напомним, что *стиранием* называется искажение, местоположение которого известно, а значение — нет. Введем *многочлен локаторов стираний* $\Phi(x) = \prod_{i=1}^{s}(1 - \beta^{l_i}x) = \sum_{j=0}^{s} \Phi_j x^j$, где $l_i, 1 \leq i \leq s$ — номера стертых символов в принятой последовательности. Заменим символы стирания в ней произвольным (например, нулевым) элементом $\mathbb{F}_q$. Тогда она может быть представлена как $y_j = c_j + e_j + v_j$, где $v(x) = \sum_{i=1}^{s} v_{l_i} x^{l_i}$ и v_{l_i} соответствуют истинным значениям стертых символов. Заметим, что компоненты $\phi_j = \frac{1}{n}\Phi(\beta^{-j})$ обратного ДПФ многочлена локаторов стираний равны нулю тогда и только тогда, когда $j = l_i$ для некоторого i. Таким образом, $y'_j = y_j\phi_j = c_j\phi_j + e_j\phi_j = c'_j + e'_j$, причем e'_j отличны от нуля на тех и только на тех позициях, на которых $e_j \neq 0$. Рассуждения, аналогичные тем, которые были проведены для случая исправления только ошибок, приводят к системе уравнений $\sum_{j=0}^{n-1} \Lambda_j E'_{i-j} = 0, 0 \leq i < n$, где Λ_j — коэффициенты многочлена локаторов ошибок. Заметим, что ДПФ последовательности c'_j в многочленном виде имеет вид $C'(x) \equiv C(x)\Phi(x) \bmod x^n - 1$, причем $C_i = 0, b \leq i \leq b + \delta - 2$. Следовательно, $C'_i = 0, b + s \leq i \leq b + \delta - 2$, т.е. $E'_i = Y'_i, b + s \leq i \leq b + \delta - 2$. Таким образом, получена система

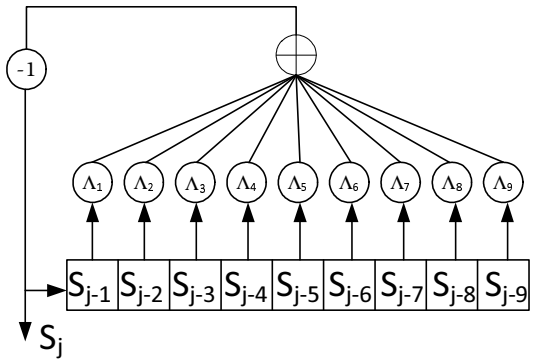


Рис. 5.1. Регистр сдвига с линейной обратной связью

уравнений

$$-S'_{i+w} = \sum_{j=1}^{w} \Lambda_j S'_{i+w-j}, s \le i \le \delta - 2 - w, \tag{5.14}$$

где $S'_i = \sum_{j=0}^{s} \Phi_j S_{i-j}$ и S_i — синдром, вычисленный по последовательности y_j. Для однозначной разрешимости этой системы необходимо, чтобы уравнений в ней $\delta - 1 - w - s$ было не меньше, чем число неизвестных w, т.е. должно выполняться условие $2w + s + 1 \le \delta$, которое совпадает с ограничением на число ошибок и стираний, исправимых кодом с минимальным расстоянием δ.

5.2.4. Алгоритм Берлекэмпа-Месси

В данном разделе будет представлен эффективный алгоритм решения системы уравнений (5.10). Снижение сложности достигается за счет того, что ее матрица является сильно структурированной. Идея *алгоритма Берлекэмпа-Месси* основана на том наблюдении, что если коэффициенты многочлена $\Lambda(x)$ известны, то некоторые компоненты вектора синдрома могут быть рекуррентно выражены через предшествующие, т.е. $S_j = -\sum_{i=1}^{w} \Lambda_i S_{j-i}, w \le j \le \delta - 2$. Эту зависимость можно реализовать с помощью регистра сдвига с линейной обратной связью (РСЛОС), представленного на рис. 5.1. Таким образом, задачу нахождения коэффициентов многочлена локаторов ошибок можно сформулировать как поиск РСЛОС, порождающего заданную последовательность S_j по ее начальной части $S_0, \ldots, S_{w-1}$. Вообще говоря, существует множество таких РСЛОС. Интерес представляет кратчайший из них, соответствующий многочлену $\Lambda(x)$ наименьшей степени. Будем говорить, что фильтр

$(L, \Lambda^{(n)}(x) = 1 + \sum_{i=1}^{L} \Lambda_i^{(n)} x^i)$ порождает последовательность S_0^{n-1}, если

$$S_k = -\sum_{i=1}^{L} \Lambda_i^{(n)} S_{k-i}, L \le k \le n-1. \tag{5.15}$$

Параметры L и $\Lambda^{(n)}(x)$ называются длиной фильтра (РСЛОС) и многочленом связей, соответственно. Заметим, что в общем случае $\deg \Lambda^{(n)}(x) \le L$.

Лемма 5.1. *Пусть фильтры* $(L_{n-1}, \Lambda^{(n-1)}(x))$ *и* $(L_n, \Lambda^{(n)}(x))$ *порождают последовательности* S_0^{n-2} *и* S_0^{n-1} *соответственно, причем* $(L_{n-1}, \Lambda^{(n-1)}(x))$ *не порождает* S_0^{n-1}*, и величины* L_{n-1} *и* L_n *являются наименьшими возможными. Тогда*

$$L_n \ge \max(L_{n-1}, n - L_{n-1}). \tag{5.16}$$

Доказательство. Фильтр, порождающий S_0^{n-1}, обязан порождать и S_0^{n-2}, поэтому $L_n \ge L_{n-1}$. Покажем, что если фильтр $(L_{n-1}, \Lambda^{(n-1)}(x))$ порождает S_0^{n-2}, но не порождает S_0^{n-1}, то $L_n \ge n - L_{n-1}$. Предположим, что это не так. Тогда

$$S_{n-1} \ne -\sum_{i=1}^{L_{n-1}} \Lambda_i^{(n-1)} S_{n-1-i} = \sum_{i=1}^{L_{n-1}} \Lambda_i^{(n-1)} \sum_{k=1}^{L_n} \Lambda_k^{(n)} S_{n-1-k-i}.$$

Последний переход возможен в силу того, что для всех i выполняется $L_n \le n - L_{n-1} - 1 \le n - 1 - i \le n - 2$, т.е. все величины S_{n-1-i} могут быть порождены с помощью $(L_n, \Lambda^{(n)}(x))$ в соответствии с (5.15). Меняя порядок суммирования, получим

$$S_{n-1} \ne \sum_{k=1}^{L_n} \Lambda_k^{(n)} \sum_{i=1}^{L_{n-1}} \Lambda_i^{(n-1)} S_{n-1-k-i} = -\sum_{k=1}^{L_n} \Lambda_k^{(n)} S_{n-1-k} = S_{n-1}.$$

Из полученного противоречия вытекает, что $L_n \ge n - L_{n-1}$. □

Алгоритм Берлекэмпа-Месси итеративно строит последовательность РСЛОС, каждый элемент которой способен порождать все более длинную последовательность величин S_i. Покажем, что если текущий РСЛОС оказывается неспособен породить очередное значение S_n, то его можно скорректировать, воспользовавшись другим РСЛОС, который пришлось корректировать ранее, когда с помощью него также не удалось породить один из предшествующих элементов последовательности. Будем считать, что длина фильтра является наименьшей, если (5.16) выполняется с равенством.

Теорема 5.7. *Предположим, что РСЛОС* $(L_i, \Lambda^{(i)}(x))$ *порождает* $S_0^{i-1}, 1 \le i \le r-1$, *причем величины* L_i *являются наименьшими возможными. Тогда РСЛОС с многочленом связей*

$$\Lambda^{(r)}(x) = \begin{cases} \Lambda^{(r-1)}(x), & \text{если } \Delta_r^{(r-1)} = 0 \\ \Lambda^{(r-1)}(x) - \frac{\Delta_r^{(r-1)}}{\Delta_m^{(m-1)}} x^{r-m} \Lambda^{(m-1)}(x), & \text{если } \Delta_r^{(r-1)} \neq 0 \end{cases} \quad (5.17)$$

порождает S_0^{r-1} *и имеет наименьшую длину. Здесь* $\Delta_r^{(v)} = \sum_{j=0}^{L_{r-1}} \Lambda_j^{(v)} S_{r-1-j}$ *— невязка*[2], m *— наибольшее число, меньшее* r *такое, что* $L_{m-1} < L_m$, *и* $\Delta_0^{(0)} = 1$.

Доказательство. Утверждение теоремы фактически задает итеративный алгоритм нахождения требуемых РСЛОС. Будем считать, что при $r = 0$ $\Lambda^{(0)}(x) = 1$, и покажем, что на каждом шаге правило коррекции (5.17) приводит к РСЛОС наименьшей длины. Очевидно, что при $\Delta_r = 0$ в изменении фильтра нет необходимости и, в соответствии с (5.16), свойство минимальности выполняется. Проверим теперь, что изменения, выполняемые в противном случае, приводят к корректному результату. Из факта изменения длины РСЛОС на шаге m вытекает, что $L_{r-1} = L_m = m - L_{m-1}$ и $L_{m-1} < L_{r-1}$. Видно, что $\deg \Lambda^{(r)}(x) \le \max(\deg \Lambda^{(r-1)}(x), r - m + \deg \Lambda^{(m-1)}(x)) \le \max(L_{r-1}, r - m + L_{m-1}) = \max(L_{r-1}, r - L_{r-1})$. Таким образом, степень многочлена $\Lambda^{(r)}(x)$ позволяет рассматривать его как многочлен связей для РСЛОС с оптимальной длиной $L_r = \max(L_{r-1}, r - L_{r-1})$. Невязка, соответствующая модифицированному РСЛОС, равна

$$\begin{aligned} \Delta_r^{(r)} &= \sum_{j=0}^{L_{r-1}} \Lambda_j^{(r-1)} S_{r-1-j} - \frac{\Delta_r^{(r-1)}}{\Delta_m^{(m-1)}} \sum_{j=0}^{L_{m-1}} \Lambda_j^{(m-1)} S_{r-1-j-(r-m)} = \\ &= \Delta_r^{(r-1)} - \frac{\Delta_r^{(r-1)}}{\Delta_m^{(m-1)}} \Delta_m^{(m-1)} = 0. \end{aligned}$$

Таким образом, модифицированный фильтр действительно порождает S_{r-1}. Он порождает также и необходимые предшествующие элементы, так как $\Delta_k^{(r)} = \sum_{j=0}^{L_r} \Lambda_j^{(r)} S_{k-1-j} = \sum_{j=0}^{L_{r-1}} \Lambda_j^{(r-1)} S_{k-1-j} - \frac{\Delta_r^{(r-1)}}{\Delta_m^{(m-1)}} \sum_{j=0}^{L_{m-1}} \Lambda_j^{(m-1)} S_{k-1-j-(r-m)} = \Delta_k^{(r-1)} - \frac{\Delta_r^{(r-1)}}{\Delta_m^{(m-1)}} \Delta_{k-r+m}^{(m-1)} = 0, L_{r-1} +$

[2]Видно, что $\Delta_r^{(v)}$ равна разности истинного значения S_{r-1} и значения, вычисленного с помощью $(L_v, \Lambda^{(v)}(x))$

```
BerlekampMassey(S_0, ..., S_{δ-2})
 1  Λ(x) ← 1, r ← 1, m ← 0, L ← 0, B(x) ← 1
 2  while r ≤ δ − 1
 3  do Δ_r ← Σ_{j=0}^{L} Λ_j S_{r−1−j}
 4     if Δ_r ≠ 0
 5        then T(x) ← Λ(x) − Δ_r x^{r−m} B(x)
 6             if 2L ≤ r − 1
 7                then B(x) ← Δ_r^{−1} Λ(x)
 8                     Λ(x) ← T(x)
 9                     L ← r − L
10                     m ← r
11                else Λ(x) ← T(x)
12     r ← r + 1
13  return (L, Λ(x))
```

Рис. 5.2. Алгоритм Берлекэмпа-Месси

$1 \le L_r+1 \le k \le r-1$. Последнее равенство вытекает из того, что РСЛОС $(L_{r-1}, \Lambda^{(r-1)}(x))$ порождает S_0^{r-2}, т.е. $\Delta_k^{(r-1)} = 0, L_{r-1}+1 \le k \le r-1$, и $(L_{m-1}, \Lambda^{(m-1)}(x))$ порождает S_0^{m-2}, т.е. $\Delta_k^{(m-1)} = 0, L_{m-1}+1 \le k \le m-1$. □

На рис. 5.2 представлен псевдокод данного алгоритма. Для упрощения обозначений, а также ввиду того, что в действительности необходимо хранить лишь одну пару значений $(\Delta_r^{(r-1)}, \Delta_m^{(m-1)})$, верхние индексы в нем не указаны. Его аргументами является синдромный вектор $(S_0, \ldots, S_{\delta-2}), S_i \in \mathbb{F}_{q^m}$. Алгоритм возвращает длину найденного РСЛОС, порождающего последовательность компонент синдромного вектора, и соответствующий многочлен связей. Не всякий результат работы алгоритма Берлекэмпа-Месси задает правильное решение задачи декодирования. Если оказалось, что $L \ne \deg \Lambda(x)$ или число корней $\Lambda(x)$ в $\mathbb{F}_{q^m}$ не равно $\deg \Lambda(x)$, то можно утверждать, что предположение $w \le \lfloor (\delta-1)/2 \rfloor$ было нарушено, и данная конфигурация ошибок с помощью методов, рассматриваемых в этом разделе, исправлена быть не может.

Видно, что на шаге r алгоритма выполняется $O(L)$ операций, связанных с вычислением невязки и коррекцией многочлена связей, причем

Таблица 5.1

Пример работы алгоритма Берлекэмпа-Месси

r	Δ	L	$\Lambda(x)$	$B(x)$
1	11	1	$1+2x$	6
2	9	1	1	6
3	6	2	$1+3x^2$	11
4	0	2	$1+3x^2$	11
5	12	3	$1+x^2$	$12+10x^2$
6	6	3	$1+6x+x^2+5x^3$	$12+10x^2$

$L \le r/2$. Следовательно, сложность алгоритма может быть оценена как $O(\delta^2)$.

Пример 5.4. Рассмотрим декодирование $(12, 6, 7)$ кода БЧХ (Рида-Соломона) над $\mathbb{F}_{13}$ с порождающим многочленом $g(x) = \prod_{i=3}^{8}(x-\beta^i)$, где $\beta = 2$ — примитивный элемент поля $\mathbb{F}_{13}$. Пусть принята зашумленная последовательность $y = (6\,1\,7\,8\,7\,6\,8\,0\,9\,11\,3\,1)$. Соответствующий вектор синдрома равен $(11\,0\,6\,0\,7\,6)$. Последовательность промежуточных результатов, получаемых при обработке S алгоритмом Берлекэмпа-Месси, представлена в табл. 5.1. Отметим, что на второй итерации длина регистра сдвига оказалась больше степени многочлена связей, что является вполне нормальным явлением. Последний найденный $\Lambda(x)$ является многочленом локаторов ошибок. Непосредственной проверкой можно убедиться, что его корнями являются $1, 7, 10$. Следовательно, локаторы ошибок равны $2^0, 1/7 \equiv 2^1 \bmod 13, 1/10 \equiv 2^2 \bmod 13$, т.е. ошибки имели место в позициях $0, 1, 2$. Из (5.11) получим, что многочлен значений ошибок равен $\Gamma(x) = 4x^2 + x + 11$. Согласно (5.12), значения ошибок равны 1, 5 и 4, соответственно. Многочлен, соответствующий исправленному кодовому слову, имеет вид $c(x) = x^{11} + 3x^{10} + 11x^9 + 9x^8 + 8x^6 + 6x^5 + 7x^4 + 8x^3 + 3x^2 + 9x + 5$.

В случае исправления ошибок и стираний можно пропустить несколько начальных шагов алгоритма Берлекэмпа-Месси, произведя инициализацию $\Lambda(x) = B(x) = \Phi(x)$.

5.2.5. Алгоритм Сугиямы

Рассмотрим метод декодирования кодов БЧХ, основанный на непосредственном поиске решения ключевого уравнения (5.11). Для этого вначале рассмотрим основы теории аппроксимации Паде.

Классическим методом нахождения наибольшего общего делителя двух многочленов $a(x)$ и $b(x)$ с коэффициентами из некоторого поля является *алгоритм Евклида*. Он представляет собой процедуру нахождения последовательности остатков от деления

$$r_{i-1}(x) = q_i(x)r_i(x) + r_{i+1}(x), \deg r_{i+1}(x) < \deg r_i(x),$$

где $r_0(x) = a(x), r_{-1}(x) = b(x)$. Наибольший общий делитель находится как последний ненулевой остаток $r_i(x)$. Действия, выполняемые на каждом шаге алгоритма Евклида, можно записать в матричном виде как

$$\begin{pmatrix} r_{i+1}(x) \\ r_i(x) \end{pmatrix} = \underbrace{\begin{pmatrix} -q_i(x) & 1 \\ 1 & 0 \end{pmatrix}}_{A^{(i)}} \begin{pmatrix} r_i(x) \\ r_{i-1}(x) \end{pmatrix} = \underbrace{A^{(i)} A^{(i-1)} \cdots A^{(0)}}_{B^{(i)}} \begin{pmatrix} r_0(x) \\ r_{-1}(x) \end{pmatrix}. \tag{5.18}$$

Алгоритм Евклида, дополненный операциями, необходимыми для вычисления матриц $B^{(i)}$, называется *расширенным алгоритмом Евклида*. Из (5.18) вытекает, что на каждом его шаге выполняется тождество $r_i(x) = u_i(x)a(x) + v_i(x)b(x)$, где $u_i(x) = B_{21}^{(i)} = B_{11}^{(i-1)}, v_i(x) = B_{22}^{(i)} = B_{12}^{(i-1)}$, или $r_i(x) \equiv u_i(x)a(x) \bmod b(x)$. Это сравнение весьма похоже на (5.11).

Лемма 5.2. *Пусть $R(x), U(x)$ — многочлены такие, что $R(x) = U(x)a(x) + V(x)b(x)$, причем $\deg R(x) + \deg U(x) < \deg b(x)$. Тогда существует j такое, что*

$$\deg r_j(x) \le \deg R(x) < \deg r_{j-1}(x), \tag{5.19}$$

и $R(x) = w(x)r_j(x), U(x) = w(x)u_j(x), V(x) = w(x)v_j(x)$, где $r_j(x), u_j(x), v_j(x)$ — промежуточные результаты, найденные в ходе обработки многочленов $a(x)$ и $b(x)$ с помощью расширенного алгоритма Евклида.

Доказательство. Отметим, что на каждом шаге расширенного алгоритма Евклида выполняются условия $\deg r_{i-1}(x) = \deg q_i(x) + \deg r_i(x)$, $u_{i+1}(x) = -q_i(x)u_i(x) + u_{i-1}(x)$, причем $u_0(x) = 1, u_{-1}(x) = 0$. Таким образом, $\deg u_i(x) = \sum_{i=0}^{i-1} \deg q_i(x) = \deg r_{-1}(x) - \deg r_{i-1}(x)$, т.е. последовательности $\deg u_i(x)$ и $\deg r_i(x), i \ge 0$, являются, соответственно, строго возрастающей и строго убывающей. Из этого вытекает достижимость условия (5.19).

Покажем, что $u_j(x)V(x) = U(x)v_j(x)$. Предположим, что это не так. Тогда система уравнений

$$\begin{pmatrix} u_j(x) & v_j(x) \\ U(x) & V(x) \end{pmatrix} \begin{pmatrix} a(x) \\ b(x) \end{pmatrix} = \begin{pmatrix} r_j(x) \\ R(x) \end{pmatrix}$$

может быть однозначно разрешена. В соответствии с правилом Крамера,

$$b(x) = \frac{\det \begin{pmatrix} u_j(x) & r_j(x) \\ U(x) & R(x) \end{pmatrix}}{\det \begin{pmatrix} u_j(x) & v_j(x) \\ U(x) & V(x) \end{pmatrix}} = \frac{u_j(x)R(x) - r_j(x)U(x)}{u_j(x)V(x) - v_j(x)U(x)}.$$

Степень правой части не превосходит $\max(\deg u_j(x) + \deg R(x), \deg r_j(x) + \deg U(x)) \leq \max(\deg b(x) - \deg r_{j-1}(x) + \deg R(x), \deg R(x) + \deg U(x)) < \deg b(x)$. Из полученного противоречия вытекает, что $u_j(x)V(x) = U(x)v_j(x)$.

Заметим, что для всех i $\det A^{(i)} = -1$, откуда $\det B^{(i)} = (-1)^{i+1} = B_{11}^{(i)}B_{22}^{(i)} - B_{12}^{(i)}B_{21}^{(i)} = u_{i+1}(x)v_i(x) - v_{i+1}(x)u_i(x)$. Это означает, что НОД$(u_i(x), v_i(x)) = 1$. Следовательно, $u_j(x)|U(x), v_j(x)|V(x)$, откуда и вытекает утверждение леммы. □

Теорема 5.8. *Пусть дано сравнение* $R(x) \equiv U(x)a(x) \bmod b(x)$. *Для любого* $k : 0 \leq k \leq \deg b(x)$ *расширенный алгоритм Евклида находит его решение* $R(x), U(x)$ *такое, что* $\deg R(x) < k, \deg U(x) \leq \deg b(x) - k$, *причем степени многочленов* $R(x)$, $U(x)$ *являются наименьшими возможными.*

Доказательство. Применим расширенный алгоритм Евклида к многочленам $a(x)$ и $b(x)$ и положим $R(x) = r_j(x), U(x) = u_j(x)$, где j — наименьшее натуральное число такое, что $\deg r_j(x) < k$ (т.е. $\deg r_{j-1}(x) \geq k$). Тогда $\deg u_j(x) = \deg b(x) - \deg r_{j-1}(x) \leq n - k$. Ясно, что данная пара многочленов действительно является решением рассматриваемого сравнения. В силу леммы 5.2, любое другое его решение удовлетворяет $R'(x) = w(x)R(x), U'(x) = w(x)U(x)$, т.е. имеет не меньшую степень. □

В случае $b(x) = x^n$ данная теорема позволяет восстановить рациональную функцию $\frac{R(x)}{U(x)}$ по первым n членам (представленным в виде многочлена $a(x)$) ее разложения в ряд Тейлора в окрестности нуля (аппроксимация Паде). В частности, полагая $a(x) = S(x), b(x) = x^{\delta-1}$, получим, что решение ключевого уравнения декодирования кодов БЧХ также может быть найдено с помощью расширенного алгоритма Евклида. Данный подход носит название *алгоритма Сугиямы*. Он проиллюстрирован на рис. 5.3. В описании алгоритма используется функция $Divide(b(x), a(x))$, которая возвращает пару многочленов $q(x), r(x)$ таких, что $b(x) = q(x)a(x) + r(x), \deg r(x) < \deg a(x)$. Алгоритм возвращает пару $(\Gamma(x), \Lambda(x))$, являющуюся решением уравнения (5.11).

SUGIYAMA$(S(x), \delta)$
1 $u(x) \leftarrow 1, u'(x) \leftarrow 0, b(x) \leftarrow x^{\delta-1}, a(x) \leftarrow S(x)$
2 **while** $\deg a(x) \geq \frac{\delta-1}{2}$
3 **do** $(q(x), r(x)) \leftarrow$ DIVIDE$(b(x), a(x))$
4 $v(x) \leftarrow -q(x)u(x) + u'(x)$
5 $u'(x) \leftarrow u(x), u(x) \leftarrow v(x)$
6 $b(x) \leftarrow a(x), a(x) \leftarrow r(x)$
7 **return** $(a(x), u(x))$

Рис. 5.3. Алгоритм Сугиямы

Таблица 5.2

Пример работы алгоритма Сугиямы

$a(x)$	$b(x)$	$u(x)$	$u'(x)$
$x^4 + 12x^3 + 12x^2 + 9x + 9$	$11 + 6x^2 + 7x^4 + 6x^5$	$2x + 2$	1
$6x^3 + 4x^2 + 11x + 11$	$x^4 + 12x^3 + 12x^2 + 9x + 9$	$x^2 + x + 1$	$2x + 2$
$12x^2 + 3x + 7$	$6x^3 + 4x^2 + 11x + 11$	$2x^3 + 3x^2 + 5x + 3$	$x^2 + x + 1$

Пример 5.5. Продолжим рассмотрение примера 5.4. Последовательность промежуточных результатов, вычисляемых алгоритмом Сугиямы при обработке синдромного многочлена $S(x) = 11 + 6x^2 + 7x^4 + 6x^5$, представлена в табл. 5.2. Заметим, что многочлены $\Lambda(x) = 2x^3 + 3x^2 + 5x + 3$ и $\Gamma(x) = 12x^2 + 3x + 7$ отличаются от полученных ранее с помощью алгоритма Берлекэмпа-Месси домножением на константу 3, что должно быть учтено на этапе поиска значений ошибок.

5.2.6. Вопросы реализации

При условии использования представленных выше эффективных алгоритмов решения уравнений (5.10) и (5.11), наиболее трудоемкими этапами декодирования оказываются вычисление компонент синдрома S_i и нахождение корней многочлена локаторов ошибок. В соответствии с теоремой Абеля [2], корни многочленов степени выше четвертой в общем случае не могут быть выражены через суммы, произведения, частные и корни их коэффициентов. Таким образом, остаются только алгоритмические способы поиска X_l. Известны алгоритмы факторизации многочленов над конечным полем, которые позволяют построить разложение (5.9) с полиномиальной сложностью [78]. Однако для практически значимых параметров кодов более эффективным оказывается подход, состоящий

в подстановке всех возможных β^i в $\Lambda(x)$ и проверке на равенство нулю полученного значения (*процедура Ченя*) $\Lambda(\beta^i) = \sum_{l=0}^{w} \Lambda_l \beta^{il}$. Эффективные алгоритмы решения этой задачи были описаны в [30, 31]. Возможно также использование алгоритмов быстрого преобразования Фурье.

Задача вычисления S_i может быть сформулирована как вычисление неполного дискретного преобразования Фурье принятого вектора. Одним из простейших способов решения этой задачи является использование *алгоритма Герцеля*. Он основан на том, что если $r(x) \equiv y(x) \bmod f_{\beta^i}(x)$, то $S_i = y(\beta^i) = r(\beta^i)$, где $f_{\beta^i}(x)$ — минимальный многочлен β^i. Ускорение достигается за счет того, что один и тот же минимальный многочлен может соответствовать нескольким различным $\beta^i \in \mathbb{F}_{q^m}$, а также того, что коэффициенты многочленов $f_{\beta^i}(x)$ принадлежат подполю $\mathbb{F}_q \subset \mathbb{F}_{q^m}$, что упрощает деление на них. Заметим, что все используемые при данном подходе многочлены являются делителями порождающего многочлена кода (см. (5.6)). Методы дальнейшего снижения сложности этого этапа описаны в [24].

Упражнения

1. Покажите, что многочлен $(x^3+x+1)^2$ порождает циклический $(14, 8, 3)$ код над $\mathbb{F}_2$. Какой двоичный циклический код порождает многочлен $(x^3+x+1)^{2^s}$?
2. Пусть дан $(15, 7, 5)$ циклический код над $\mathbb{F}_2$ с порождающим многочленом $g(x) = 1+x^4+x^6+x^7+x^8$. Закодируйте в нем систематическим образом информационный вектор (1001011).
3. Продекодируйте в вышеуказанном коде вектор $y = (100110101111011)$.
4. Постройте примитивный двоичный код БЧХ длины 31 с минимальным расстоянием 7. Какова его размерность?
5. Покажите, что если конструктивное расстояние $\delta = 2t+1$ двоичного кода БЧХ в узком смысле делит его длину n, то оно равно минимальному расстоянию этого кода.
6. Продекодируйте с помощью алгоритмов Берлекэмпа-Месси и Сугиямы в коде, построенном в предыдущем упражнении, любой вектор веса 28.
7. Докажите, что при декодировании двоичных кодов БЧХ в узком смысле на четных итерациях алгоритма Берлекэмпа-Месси всегда получается нулевая невязка Δ_r.
8. Докажите, что обратное ДПФ от вектора, полученного как ДПФ $f = (f_0, \ldots, f_{n-1})$, равно исходному вектору f.
9. Докажите теорему о свертке.

6. Коды Рида-Соломона и производные от них

6.1. Коды Рида-Соломона

6.1.1. Конструкции кодов

Как было указано выше, коды БЧХ над $\mathbb{F}_q$ длины $n|(q-1)$ называются *кодами Рида-Соломона*. Из границы БЧХ вытекает, что минимальное расстояние (n,k)-кода Рида-Соломона не может быть меньше $n-k+1$, но граница Синглтона исключает существование кодов с большим минимальным расстоянием. Таким образом, оно в точности равно $n-k+1$.

Однако существует и иной (в общем случае неэквивалентный) способ определить эти коды.

Теорема 6.1. *Существует взаимно однозначное соответствие между кодовыми словами $(n,k,n-k+1)$ кода Рида-Соломона в узком смысле с порождающим многочленом $g(x)=\prod_{i=1}^{n-k}(x-\beta^i)$ и множеством векторов вида $F=(f(\beta^0),\ldots,f(\beta^{n-1}))$, где $f(x)=\sum_{i=0}^{k-1}f_ix^i, f_i\in\mathbb{F}_q$ — некоторый многочлен и β — элемент порядка n.*

Доказательство. Проверим, что вектор указанного вида действительно является кодовым словом. Для этого построим соответствующий ему многочлен $c(x)=\sum_{i=0}^{n-1}f(\beta^i)x^i$. Видно, что $c(\beta^j)=\sum_{i=0}^{n-1}\sum_{l=0}^{n-1}f_l\beta^{i(l+j)}=\sum_{l=0}^{n-1}f_{-l \bmod n}\sum_{i=0}^{n-1}\beta^{i(j-l)}=f_{-j}$. Последний переход справедлив в силу того, что β является элементом порядка n и $0=\beta^{rn}-1=(\beta^r-1)\left(\sum_{i=0}^{n-1}\beta^{ri}\right)$. Если $r\not\equiv 0 \bmod n$, то первый сомножитель не может быть равен нулю, т.е. нулю должна быть равна вторая сумма. Так как $\deg f(x)<k$, $c(\beta^j)=0, 1\le j\le n-k$, т.е. все корни порождающего многочлена $g(x)=\prod_{j=1}^{n-k}(x-\beta^j)$ являются корнями $c(x)$. Таким образом, каждый вектор F является кодовым словом кода Рида-Соломона. Число различных векторов F указанного в условии теоремы вида совпадает с числом кодовых слов кода Рида-Соломона. □

Доказанная теорема позволяет сформулировать еще одно определение кодов Рида-Соломона, которое оказывается полезным при описании многих рассматриваемых далее алгоритмов декодирования.

Определение 6.1. $(n, k, n-k+1)$ кодом Рида-Соломона над $\mathbb{F}_q$ называется множество

$$\left\{ (f(a_1), \ldots, f(a_n)) | f(x) = \sum_{i=0}^{k-1} f_i x^i, f_i, a_i \in \mathbb{F}_q, a_i \text{ различны} \right\}.$$

Необходимо отметить, что определенные таким образом коды не обязательно являются кодами БЧХ и даже циклическими. С другой стороны, коды Рида-Соломона в широком смысле (см. разд. 5.2.1) не соответствуют определению 6.1. Небольшое изменение конструкции кодов позволяет унифицировать терминологию.

Определение 6.2. *Обобщенным кодом Рида-Соломона* над $\mathbb{F}_q$ называется множество

$$GRS_k(a, v) = \left\{ (f(a_1)v_1, \ldots, f(a_n)v_n) | f(x) = \sum_{i=0}^{k-1} f_i x^i, f_i, v_i \in \mathbb{F}_q, v_i \neq 0 \right\}, \tag{6.1}$$

где $a = (a_1, \ldots, a_n)$, $a_i \in \mathbb{F}_q$ — различные величины, называемые *локаторами кода*.

Очевидно, что домножение всех символов кодовых слов кода Рида-Соломона на фиксированные ненулевые элементы v_i не влияет на их вес. Таким образом, минимальное расстояние обобщенного (n, k) кода Рида-Соломона также равно $n - k + 1$.

Выбор a_i редко оказывает значимое влияние на свойства кода. Так как математический аппарат циклических кодов предоставляет такое значимое преимущество, как простую процедуру систематического кодирования (см. (5.1)), на практике удобно рассматривать коды Рида-Соломона как коды БЧХ, а при необходимости использования алгоритмов, основанных на определениях 6.1 или 6.2 — подбирать подходящие параметры в последнем (см. упражнение 3).

Теорема 6.2. *Кодом, дуальным к обобщенному коду Рида-Соломона* $GRS_k(a, v)$, *также является обобщенный код Рида-Соломона* $GRS_{n-k}(a, u)$ *с некоторым весовым вектором* v'.

Доказательство. Рассмотрим вначале случай $k = n - 1$. Очевидно, что дуальный код имеет размерность 1 и его порождающая матрица может быть представлена как $H = (u_1, \ldots, u_n)$. Вектор H может быть найден как ненулевое решение системы линейных однородных алгебраических

уравнений $0 = GH^T$, которую можно переписать как

$$0 = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ a_1 & a_2 & \cdots & a_n \\ a_1^2 & a_2^2 & \cdots & a_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ a_1^{n-2} & a_2^{n-2} & \cdots & a_n^{n-2} \end{pmatrix} \begin{pmatrix} v_1 u_1 \\ v_2 u_2 \\ \vdots \\ v_n u_n \end{pmatrix}, \tag{6.2}$$

где G — порождающая матрица $GRS_k(a, v)$, соответствующая (6.1). Предположим, что некоторый u_i (например, u_1) равен нулю. Тогда можно удалить соответствующий ему столбец матрицы этой системы, после чего она станет квадратной матрицей Вандермонда. По построению все a_i различны, поэтому решением этой системы является вектор $v_i u_i = 0, 2 \leq i \leq n$. Так как $v_i \neq 0, 1 \leq i \leq n$, получим, что $H = 0$, что противоречит предположению о том, что он является проверочной матрицей дуального кода. Таким образом, $u_i \neq 0, 1 \leq i \leq n$.

При $k \leq n - 1$ величины u_i могут быть найдены из условия $G_i H_j^T = 0$, где G_i, H_j — соответствующие строки порождающих матриц $GRS_k(a, v)$ и $GRS_{n-k}(a, u)$, которое приводит к системе уравнений

$$0 = \sum_{l=1}^{n} a_l^i v_l a_l^j u_l = \sum_{l=1}^{n} a_l^{i+j} v_l u_l, 0 \leq i \leq k-1, 0 \leq j \leq n-k-1.$$

Эта система совпадает с (6.2), для которой существование полностью ненулевого решения $(u_1, \ldots, u_n)$ было установлено выше. □

6.2. Декодирование до половины минимального расстояния

Декодирование кодов Рида-Соломона, являющихся кодами БЧХ, может быть выполнено с помощью любого из алгоритмов, описанных в разд. 5.2. В этом разделе будут представлены методы кодов Рида-Соломона в смысле определения 6.1, которые концептуально существенно отличаются от рассмотренных ранее.

6.2.1. Алгоритм Гао

Рассмотрим проблему декодирования кодов Рида-Соломона в метрике Хэмминга с числом исправляемых ошибок $w \leq \lfloor (n-k)/2 \rfloor$. Пусть $(y_1, \ldots, y_n) \in \mathbb{F}_q^n$ — принятая зашумленная последовательность. Результатом декодирования должно стать кодовое слово или соответствующий ему многочлен $f(x)$ такой, что $f(a_i) = y_i$ в $\tau = n - w$ неискаженных по-

зициях i. Введем *многочлен ошибок*[1] такой, что $\sigma(a_i) = 0$ тогда и только тогда, когда $f(a_i) \neq y_i$. Таким образом, справедлива система уравнений

$$f(a_i)\sigma(a_i) = y_i\sigma(a_i), 1 \leq i \leq n. \tag{6.3}$$

Пусть $\phi(x) = \prod_{i=1}^{n}(x - a_i)$ и $T(x) : T(a_i) = y_i$. Тогда эта система эквивалентна сравнению

$$p(x) = f(x)\sigma(x) \equiv T(x)\sigma(x) \bmod \phi(x), \tag{6.4}$$

причем $\deg\sigma(x) = w \leq \lfloor (n-k)/2 \rfloor$, $\deg p(x) < k + w \leq \lfloor (n+k)/2 \rfloor$. В соответствии с теоремой 5.8, решение $(p(x), \sigma(x))$ этого сравнения может быть найдено с помощью расширенного алгоритма Евклида, после чего многочлен, задающий кодовое слово, может быть найден как $f(x) = \frac{p(x)}{\sigma(x)}$. Данный подход носит название *алгоритма Гао*. Очевидно, что концептуально он намного проще рассмотренных в разд. 5.2 подходов, основанных на вычислении синдрома, поиске многочлена локаторов ошибок, его корней и значений ошибок. Однако необходимость работы с многочленами $T(x), f(x)$, имеющими сравнительно большую степень, во многих случаях делает его менее эффективным.

6.2.2. Декодирование с помощью многомерной интерполяции

Рассмотрим интерпретацию алгоритма Гао, которая позволит в дальнейшем обобщить его на случай списочного декодирования. Система уравнений (6.3) может быть переписана как

$$Q(a_i, y_i) = 0, 1 \leq i \leq n, \tag{6.5}$$

где $Q(x, y) = p(x) - \sigma(x)y = \sum_{j=0}^{k-1+w} q_{0j}x^j + \sum_{j=0}^{w} q_{1j}x^j y$ — неизвестный многочлен от двух переменных. Ее можно интерпретировать как систему линейных однородных алгебраических уравнений. Если удалось найти ее ненулевое решение, то задающий кодовое слово многочлен $f(x)$ может быть найден из условия $Q(x, f(x)) = 0$. Задача нахождения $Q(x, y)$ является частным случаем задачи многомерной интерполяции. Решив ее, можно найти многочлен $f(x)$ от одной переменной, проходящий через *некоторые* из интерполяционных точек (x_i, y_i). Таким образом, получен алгоритм решения задачи интерполяции по зашумленным данным.

Принцип декодирования по минимальному расстоянию требует минимизировать w, т.е. найти многочлен $Q(x, y) = q_0(x) + q_1(x)y$, для которого степени $q_0(x)$ и $q_1(x)$ были бы минимально возможными. Для

[1] Многочлен локаторов ошибок $\Lambda(x)$, задаваемый (5.9), связан с ним соотношением $\Lambda(x) = x^w\sigma(x^{-1})$, т.е. его коэффициенты записаны в обратном порядке.

того, чтобы формализовать это требование, придется ввести некоторые понятия из коммутативной алгебры [4].

(a,b)-*взвешенной степенью* одночлена $cx^iy^j, c \neq 0$, называется величина $\mathrm{wdeg}_{(a,b)}(cx^iy^j) = ai + bj$. Взвешенная степень может быть использована для упорядочения членов многочлена от нескольких переменных. Определим (a,b)-*взвешенное мономиальное упорядочение* как $cx^iy^j \prec_{(a,b)} dx^uy^v \Leftrightarrow (\mathrm{wdeg}_{(a,b)}(cx^iy^j) < \mathrm{wdeg}_{(a,b)}(dx^uy^v)) \vee (\mathrm{wdeg}_{(a,b)}(cx^iy^j) = \mathrm{wdeg}_{(a,b)}(dx^uy^v)) \wedge (cx^iy^j \prec_{lex} dx^uy^v)$. Здесь для разрешения неопределенности в случае равенства взвешенных степеней используется *лексикографическое мономиальное упорядочение*, которое определяется как[2] $cx^iy^j \prec_{lex} dx^uy^v \Leftrightarrow (j < v) \vee (j = v) \wedge (i < u)$. *Старшим членом* многочлена $Q(x,y)$ называется ненулевой его член, наибольший в смысле некоторого фиксированного мономиального упорядочения, и обозначаемый как $\mathrm{LT}\, Q(x,y)$. (a,b)-взвешенной степенью многочлена $Q(x,y)$ называется (a,b)-взвешенная степень его старшего члена относительно $\prec_{(a,b)}$.

Пример 6.1. Старшим членом многочлена $Q(x,y) = xy + y + x^4 + x + 1$ относительно лексикографического упорядочения является xy. При выборе $(1,2)$-взвешенного лексикографического упорядочения старшим членом становится x^4.

Сформулированный выше метод декодирования предполагает нахождение многочлена $Q(x,y) = q_0(x) + q_1(x)y$, такого что $\deg q_0(x) - \deg q_1(x) \leq k-1$, и степени $q_0(x), q_1(x)$ — наименьшие возможные. Эти условия можно сформулировать как поиск $Q(x,y)$ с наименьшей $(1,k-1)$-взвешенной степенью, такого что $\mathrm{LT}\, Q(x,y) = ax^wy, a \neq 0$.

Определение 6.3. *Модулем* M над кольцом $\mathbf{R}$ называется абелева группа $(M,+)$, на которой задана операция $\cdot : \mathbf{R} \times M \dot{\to} M$ такая, что для всех $r,s \in \mathbf{R}, x,y \in M$ выполняются условия

1. $r \cdot (x+y) = r \cdot x + r \cdot y$;
2. $(r+s) \cdot x = r \cdot x + s \cdot x$;
3. $(rs) \cdot x = r \cdot (s \cdot x)$;
4. $1_{\mathbf{R}} \cdot x = x$.

Понятие модуля является обобщением концепции линейного векторного пространства. Отличие состоит в том, что компонентами векторов линейного пространства должны быть элементы некоторого поля, в то время как в случае модуля достаточно лишь, чтобы они принадле-

[2] В рассматриваемых ниже алгоритмах удобнее вначале сравнивать степени одночленов при y, а затем — при x.

жали некоторому кольцу. В данном случае в качестве кольца $\mathbf{R}$ будет рассматриваться множество $\mathbb{F}_q[x]$ многочленов от одной переменной, а в качестве модуля — множество $M = \mathbb{F}_q[x, y]$ многочленов от двух переменных вида $\sum_{j=0}^{\rho} q_j(x)y^j$ (пока будем полагать $\rho = 1$). Заметим, что множество многочленов, удовлетворяющих (6.5), образует модуль.

Аналогично случаю линейного пространства, можно ввести понятие *базиса модуля* как набора элементов $m_1, \ldots, m_l \in M$, с помощью которого можно породить любой его элемент как $m = \sum_{i=1}^{l} a_i m_i, a_i \in \mathbf{R}$. Рассматриваемый модуль $M_{1,1}$ многочленов, удовлетворяющих (6.5), имеет базис $[\phi(x), y - T(x)]$, где $\phi(x) = \prod_{i=1}^{n}(x - a_i)$ и $T(x) : T(a_i) = y_i$. Действительно, произвольный многочлен $Q(x, y) = q_0(x) + q_1(x)y \in M_{1,1}$ может быть представлен как $Q(x, y) = q_1(x)(y - T(x)) + \phi(x)\frac{q_0(x)+q_1(x)T(x)}{\phi(x)}$. Числитель последней дроби делится нацело на ее знаменатель, так как, по предположению, $Q(a_i, y_i) = q_0(a_i) + q_1(a_i)y_i = 0$ и $T(a_i) = y_i$, т.е. все a_i являются корнями числителя. С другой стороны, они же являются простыми корнями знаменателя.

Особый интерес представляет *базис Грёбнера* $\mathcal{B}$, который определяется как базис модуля M такой, что

$$\forall m \in M \exists Q \in \mathcal{B} : \operatorname{LT} Q | \operatorname{LT} m.$$

Данное условие означает, что базис Грёбнера содержит в некотором смысле "маленькие" элементы модуля. В частности, можно показать, что наименьший в смысле заданного упорядочения интерполяционный полином, удовлетворяющий (6.5), всегда содержится в базисе Грёбнера соответствующего модуля [61]. Таким образом, необходимо найти базис Грёбнера модуля $M_{1,1}$ относительно $(1, k-1)$-взвешенного лексикографического упорядочения.

Различным мономиальным упорядочениям соответствуют различные базисы Грёбнера. Более того, даже для фиксированного мономиального упорядочения можно построить множество различных базисов Грёбнера. Для того, чтобы некоторый базис $\{Q_0(x, y), \ldots, Q_\rho(x, y)\}$ модуля был его базисом Грёбнера, достаточно, чтобы все величины $\operatorname{ydeg} Q_i(x, y)$ были различны, где $\operatorname{ydeg} Q(x, y) = j \Leftrightarrow \operatorname{LT} Q(x, y) = ax^i y^j, a \neq 0$. Действительно, если это не так, и, например, $\operatorname{ydeg} Q_1(x, y) = \operatorname{ydeg} Q_2(x, y)$ (будем считать для определенности, что $\operatorname{LT} Q_1(x, y) \prec \operatorname{LT} Q_2(x, y)$), то многочлен $\frac{\operatorname{LT} Q_2(x,y)}{\operatorname{LT} Q_1(x,y)} Q_1(x, y) - Q_2(x, y)$ имеет старший член, меньший, чем $\operatorname{LT} Q_1(x, y)$ и $\operatorname{LT} Q_2(x, y)$. Эти рассуждения приводят к следующему очевидному способу нахождения базиса

REDUCE($[S_0(x,y),\ldots,S_{i-1}(x,y)], P(x,y)$)

1 $S_i(x,y) \leftarrow P(x,y)$
2 **while** $\exists j : (0 \le j < i) \wedge (\operatorname{ydeg} S_j(x,y) = \operatorname{ydeg} S_i(x,y))$
3 **do if** $\operatorname{LT} S_i(x,y) | \operatorname{LT} S_j(x,y)$
4 **then** $W(x,y) \leftarrow S_j(x,y) - \frac{\operatorname{LT} S_j(x,y)}{\operatorname{LT} S_i(x,y)} S_i(x,y)$
5 $S_j(x,y) \leftarrow S_i(x,y)$
6 $S_i(x,y) \leftarrow W(x,y)$
7 **else** $S_i(x,y) \leftarrow S_i(x,y) - \frac{\operatorname{LT} S_i(x,y)}{\operatorname{LT} S_j(x,y)} S_j(x,y)$
8 **if** $S_i(x,y) = 0$
9 **then** $i \leftarrow i - 1$
10 **return** $[S_0(x,y),\ldots,S_i(x,y)]$

Рис. 6.1. Многомерный алгоритм Евклида

Грёбнера модуля: можно взять произвольный его базис и применять к нему обратимые преобразования, приводящие к сокращению старших членов, до тех пор, пока это возможно. Данный алгоритм удобно сформулировать следующим образом: пусть уже построен базис Грёбнера $[S_0(x,y),\ldots,S_{i-1}(x,y)]$ некоторого подмодуля M' такой, что $\operatorname{LT} S_j(x,y) = j$. Тогда для того, чтобы построить базис Грёбнера модуля $M = \{Q(x,y) + a(x)P(x,y) | Q(x,y) \in M', a(x) \in \mathbb{F}[x]\}$, достаточно произвести сокращения между старшими членами $S_j(x,y)$ и членами $P(x,y)$, пока это возможно. На рис. 6.1 представлен алгоритм, реализующий этот подход. Можно заметить, что выполняемые этим алгоритмом действия очень похожи на те, которые осуществляются расширенным алгоритмом Евклида. Поэтому будем называть его многомерным алгоритмом Евклида. Искомый базис Грёбнера $M_{1,1}$ может быть найден как $[q_{00}(x) + q_{10}y, q_{01}(x) + q_{11}y] = Reduce([\phi(x)], y - T(x))$. Эту операцию можно представить как

$$\mathcal{Q}(x) = \begin{pmatrix} q_{00}(x) & q_{01}(x) \\ q_{10}(x) & q_{11}(x) \end{pmatrix} = \underbrace{\begin{pmatrix} \phi(x) & -T(x) \\ 0 & 1 \end{pmatrix}}_{\mathcal{Q}'(x)} U_1 U_2 \cdots U_l,$$

где U_i — матрицы, описывающие преобразования, выполняемые многомерным алгоритмом Евклида. Видно, что $\det U_i = \pm 1$, а потому

$$\det \mathcal{Q}(x) = q_{00}(x)q_{11}(x) - q_{01}(x)q_{10}(x) = \det \mathcal{Q}'(x) = \pm\phi(x). \qquad (6.6)$$

В силу того, что результатом применения *Reduce* является базис Грёбнера относительно $(1, k-1)$-взвешенного лексикографического упорядочения, можно считать, что $\deg q_{11}(x)+k-1 \geq q_{01}(x)$, $\deg q_{10}(x)+k-1 < \deg q_{00}(x)$. Отсюда $\deg q_{11}(x) - \deg q_{10}(x) > \deg q_{01}(x) - \deg q_{00}(x)$. Тогда в силу (6.6) получим, что $\deg q_{00}(x) + \deg q_{11}(x) = n$. С другой стороны, искомый многочлен $q_{10}(x) + yq_{11}(x)$ является наименьшим тогда и только тогда, когда $\deg q_{11}(x) + k - 1 < \deg q_{00}(x) = n - \deg q_{11}(x)$, т.е. $\deg q_{11}(x) \leq \frac{n-k}{2}$. Таким образом, если число произошедших ошибок w не превосходит $(n-k)/2$, то описанный алгоритм найдет многочлен $q_{10}(x) + q_{11}(x)y = p(x) - \sigma(x)y$.

Пример 6.2. Рассмотрим декодирование $(4, 2, 3)$ кода Рида-Соломона над $\mathbb{F}_5$. Пусть $a_i = i, 1 \leq i \leq 4$, и принятая последовательность равна $y = (1, 2, 3, 0)$. Ей соответствует интерполяционный многочлен $T(x) \equiv x^3 + 4x^2 + 2x + 4 \bmod 5$. Задача декодирования сводится к поиску многочлена $f(x)$ степени 1, значения которого в точках a_i отличались бы от y не более чем в одной позиции. Для этого необходимо найти многочлен $Q(x, y)$ наименьшей $(1, 1)$-взвешенной степени такой, что $Q(a_i, y_i) = 0$. Такой многочлен присутствует в базисе Грёбнера модуля $M_{1,1} = \{Q(x, y) | Q(a_i, y_i) = 0\}$. Воспользуемся для его построения многомерным алгоритмом Евклида. Начальным приближением к искомому базису является $[\phi(x)]$, где $\phi(x) = (x-1)(x-2)(x-3)(x-4) = x^4 + 4$. При вызове $Reduce([\phi(x)], y - T(x))$ будут выполнены следующие действия:

1. $S_0(x, y) = x^4 + 4, S_1(x, y) = y + 4x^3 + x^2 + 3x + 1$.
2. Так как $\mathrm{LT}\, S_0(x, y) = x^4, \mathrm{LT}\, S_1(x, y) = 4x^3$, старший член $S_0(x, y)$ можно сократить путем сложения $S_0(x, y)$ с $xS_1(x, y)$, что приводит к $S_0(x, y) = xy + x^3 + 3x^2 + x + 4$.
3. Так как $\mathrm{LT}\, S_0(x, y) = x^3, \mathrm{LT}\, S_1(x, y) = 4x^3$, старший член $S_0(x, y)$ можно сократить путем сложения $S_0(x, y)$ с $S_1(x, y)$, что приводит к $S_0(x, y) = (x+1)y + 4x^2 + 4x$. Теперь $\mathrm{ydeg}\, S_0(x, y) = 1$, поэтому вычисления прекращаются.

Решение задачи декодирования получается из $S_0(x, y)$ как $f(x) = -\frac{4x^2+4x}{x+1} = x$, т.е. исправленное кодовое слово имеет вид $(1, 2, 3, 4)$.

6.3. Списочное декодирование

6.3.1. Алгоритм Гурусвами-Судана

Попытаемся обобщить идею метода декодирования с помощью интерполяции, описанного в разд. 6.2.2, на случай исправления большего числа ошибок. Если радиус декодирования t превышает $\lfloor (d-1)/2 \rfloor$, может существовать несколько решений задачи декодирования. Если пытаться найти все такие решения в виде $f^{(j)}(x) : Q(x, f^{(j)}(x))$, то очевидно, что степень многочлена $Q(x, y)$ относительно y должна быть не меньше, чем число l кодовых слов, находящихся на расстоянии t от произвольного вектора. В этом случае должно выполняться равенство

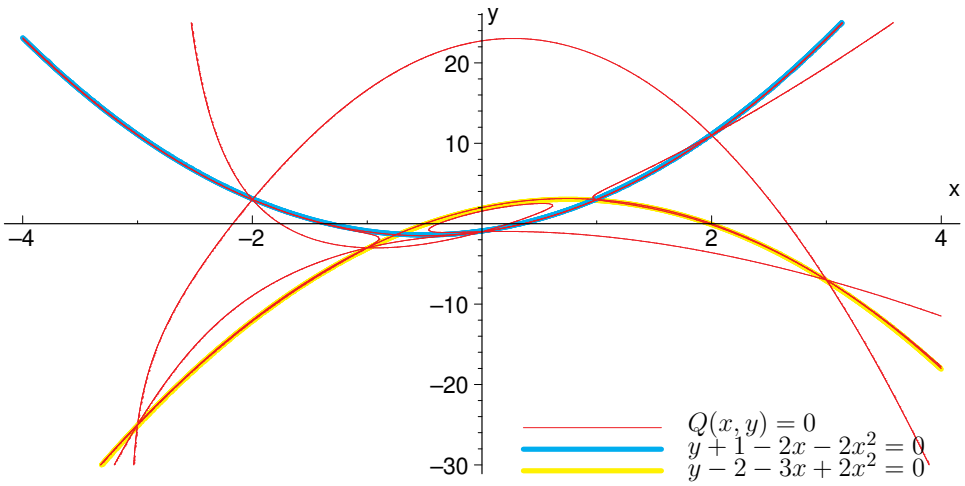


Рис. 6.2. Пример работы алгоритма Гурусвами-Судана

$Q(x,y)=(y-f^{(1)}(x))(y-f^{(2)}(x))\cdots(y-f^{(l)}(x))Q'(x,y)$. Однако с ростом t и l растет вероятность совпадения некоторых символов нескольких кодовых слов, т.е. того, что $y_i-f^{(j)}(x_i)=0$ сразу для нескольких j. Это означает, что кратность корня (x_i,y_i) многочлена $Q(x,y)$ должна быть достаточно большой, что необходимо предусмотреть на этапе его построения.

Этот подход реализует *алгоритм Гурусвами-Судана* списочного декодирования кодов Рида-Соломона. На рис. 6.2 представлен пример его использования для декодирования вектора $y=(-25,3,-3,-1,3,11,-7)$ в $(7,3,5)$-коде Рида-Соломона над $\mathbb{R}$ с вектором локаторов $a=(-3,-2,-1,0,1,2,3)$. Был построен некоторый многочлен $Q(x,y)$ такой, что точки (a_i,y_i) являлись его корнями кратности 4. Этот многочлен задает некоторую кривую, заданную в неявном виде, которая показана тонкими линиями. Декодирование сводится к поиску парабол (так как код имеет размерность 3), которыми можно было бы покрыть различные ветви этой кривой. В данном случае нашлись две такие параболы (показаны толстыми линиями), которые соответствуют кодовым словам $(-25,-12,-3,2,3,0,-7)$ и $(13,3,-1,-1,3,11,23)$.

Наличие у многочлена корня (a_i,y_i) кратности r означает, что в разложении многочлена в окрестности этой точки в ряд Тейлора отсутствуют члены порядка, меньшего r, т.е.

$$Q(x,y)=\sum_{j_1+j_2\geq r} Q^{[j_1,j_2]}(a_i,y_i)(x-a_i)^{j_1}(y-y_i)^{j_2}. \qquad (6.7)$$

Коэффициенты $Q^{[j_1,j_2]}(a_i,y_i)$ в этом разложении носят название частных производных Хассе $Q(x,y)$ порядка j_1 по x и порядка j_2 по y в точке (a_i,y_i). Производные Хассе связаны с обычными производными соотношением $Q^{[j_1,j_2]}(a_i,y_i)=\frac{1}{j_1!j_2!}Q^{(j_1,j_2)}(a_i,y_i)$. Необходимость их использования обусловлена тем, что последние в случае конечных полей могут обращаться в ноль, даже если точка не является корнем соответствующей кратности. Например, очевидно, что многочлен x^2 имеет единственный нулевой корень кратности 2. Однако обычная производная $(x^2)'=2x$ сравнима с нулем в любом поле характеристики 2 для любых x. Выражение (6.7) показывает, что производные Хассе возникают из формулы Тейлора намного более естественным образом, чем обычные производные. Из формулы бинома Ньютона несложно получить следующее выражение для производных Хассе многочлена $Q(x,y)=\sum_{i,j\geq 0}q_{ij}x^iy^j$:

$$Q^{[j_1,j_2]}(a_i,y_i)=\sum_{j_1'\geq j_1}\sum_{j_2'\geq j_2}C_{j_1'}^{j_1}C_{j_2'}^{j_2}q_{j_1',j_2'}a_i^{j_1'-j_1}y_i^{j_2'-j_2}. \tag{6.8}$$

Алгоритм Гурусвами-Судана списочного декодирования вектора $(y_1,\ldots,y_n)$ в $(n,k,n-k+1)$ коде Рида-Соломона с вектором локаторов $(a_1,\ldots,a_n)$ включает следующие шаги:

1. (Интерполяция) Построение многочлена $Q(x,y)=\sum_{j=0}^{\rho}q_j(x)y^j$: $Q^{[j_1,j_2]}(a_i,y_i)=0, j_1+j_2<r$, такого что его $(1,k-1)$-взвешенная степень не превосходит l.
2. (Факторизация) Поиск всех $f^{(j)}(x):Q(x,f^{(j)}(x))=0,\deg f(x)<k$.
3. Выбор тех $f^{(j)}(x)$, для которых $f^{(j)}(a_i)=y_i$ не менее чем для $\tau=n-t$ позиций i.

Корректность этого алгоритма и взаимосвязь его параметров ρ,r,l,τ будут обоснованы в следующих леммах.

Лемма 6.1. *Если* $Q^{[j_1,j_2]}(a_i,y_i)=0, j_1+j_2<r$ *и* $f(a_i)=y_i$, *то* $(x-a_i)^r|Q(x,f(x))$.

Доказательство. Если $f(a_i)=y_i$, то $f(x)-y_i=(x-a_i)g(x)$. Из (6.7) вытекает, что $Q(x,f(x))=\sum_{j_1+j_2\geq r}Q^{[j_1,j_2]}(a_i,y_i)(x-a_i)^{j_1}(f(x)-y_i)^{j_2}=\sum_{j_1+j_2\geq r}Q^{[j_1,j_2]}(a_i,y_i)(x-a_i)^{j_1+j_2}g^{j_2}(x)=(x-a_i)^rh(x)$. □

Лемма 6.2. *Если многочлен* $Q(x,y)=\sum_j q_j(x)y^j$ *удовлетворяет условиям* $Q^{[j_1,j_2]}(a_i,y_i)=0, j_1+j_2<r, 1\leq i\leq n$, $\mathrm{wdeg}_{(1,k-1)}Q(x,y)<r\tau$, *то для любого многочлена* $f(x)$, *такого что* $\deg f(x)<k$ *и* $|\{i|f(a_i)=y_i\}|\geq\tau$, *выполняется* $Q(x,f(x))=0$.

Доказательство. В соответствии с леммой 6.1, многочлен $\prod_{i:f(a_i)=y_i}(x-a_i)^r$, степень которого по условию леммы не меньше τr, является делителем $Q(x, f(x)) = \sum_{j\geq 0} q_j(x)f^j(x)$. Степень этого многочлена не превосходит $\mathrm{wdeg}_{(1,k-1)} Q(x, y) < \tau r$. Многочлен большей степени может быть делителем многочлена меньшей степени, только если последний тождественно равен нулю. □

Теорема 6.3. *Если*

$$\left(\left\lfloor \frac{l}{k-1} \right\rfloor + 1\right)\left(l + 1 - \frac{k-1}{2}\left\lfloor \frac{l}{k-1} \right\rfloor\right) > n\frac{r(r+1)}{2},$$

то алгоритм Гурусвами-Судана обеспечивает списочное декодирование кодов Рида-Соломона с числом исправляемых ошибок

$$t < n - \sqrt{n(k-1)}.$$

Доказательство. Как видно из (6.8), построение многочлена $Q(x, y)$ сводится к решению системы линейных однородных алгебраических уравнений $Q^{[j_1,j_2]}(a_i, y_i) = 0, j_1 + j_2 < r, 1 \leq i \leq n$. Общее число таких уравнений равно $n\frac{r(r+1)}{2}$. Число неизвестных равно числу одночленов в этом полиноме, $(1, k-1)$-взвешенная степень которых не превосходит l. Очевидно, что можно положить $\rho = \left\lfloor \frac{l}{k-1} \right\rfloor$. Тогда число таких одночленов равно $N(l) = \sum_{j=0}^{\rho}(l+1-(k-1)j) = (\rho+1)(l+1) - (k-1)\frac{\rho(\rho+1)}{2} = \left(\left\lfloor \frac{l}{k-1} \right\rfloor + 1\right)\left(l + 1 - \frac{k-1}{2}\left\lfloor \frac{l}{k-1} \right\rfloor\right)$. Если число неизвестных превосходит число уравнений, то такая система имеет ненулевое решение, которое соответствует искомому многочлену. Это условие может быть записано как $N(l) \geq \left(\left\lfloor \frac{l}{k-1} \right\rfloor + 1\right)(l + 1 - l/2) \geq \frac{l(l+2)}{2(k-1)} > n\frac{r(r+1)}{2}$. Выбирая $l = r\tau - 1$, получим $n(k-1)r^2 + n(k-1)r < r^2\tau^2 - 1$. Решая это неравенство относительно r, получим

$$r > \frac{n(k-1) + \sqrt{n^2(k-1)^2 + 4(\tau^2 - n(k-1))}}{2(\tau^2 - n(k-1))}, \tag{6.9}$$

причем знаменатель этой дроби должен быть неотрицателен, откуда $n - t = \tau > \sqrt{n(k-1)}$. □

Необходимо отметить, что оценка (6.9) является сильно завышенной, что приводит к бесполезным вычислительным затратам. Их можно значительно сократить, если первоначально выбрать r на основе (6.9), а затем постепенно его уменьшать до тех пор, пока не нарушится условие $N(r\tau - 1) > n\frac{r(r+1)}{2}$.

ITERATIVEINTERPOLATION$(n, \{(a_i, y_i), 1 \le i \le n\}, r, \rho)$

```
 1  for i ← 0 to ρ − 1
 2  do Q_i(x, y) ← y^i;
 3  for i ← 1 to n
 4  do for β ← 0 to r − 1
 5      do for α ← 0 to r − β − 1
 6          do Δ_j ← Q_j^[α,β](a_i, y_i), 0 ≤ j ≤ ρ − 1
 7              m ← arg min_{j:Δ_j≠0} Q_j(x, y)
 8              for j ≠ m
 9              do Q_j(x, y) ← Q_j(x, y) − (Δ_j/Δ_m) Q_{j_0}(x, y)
10              Q_m(x, y) ← Q_m(x, y)(x − a_i)
11  return (Q_0(x, y), ..., Q_{ρ−1}(x, y))
```

Рис. 6.3. Итеративный интерполяционный алгоритм

6.3.2. Быстрая интерполяция

Непосредственное решение системы линейных уравнений $Q^{[j_1,j_2]}(a_i, y_i) = 0, j_1 + j_2 < r, 1 \le i \le n$ с помощью метода Гаусса требует порядка $O(n^3 r^6)$ операций, что недопустимо много для практической реализации. В связи с этим возникает необходимость построения более эффективной реализации интерполяционного шага. Это может быть реализовано с помощью *итеративного интерполяционного алгоритма* [46, 52], представленного на рис. 6.3. Основная идея этого алгоритма состоит в построении нескольких многочленов, удовлетворяющих интерполяционным условиям, причем на каждом шаге алгоритма выполняется условие $\mathrm{LT}\, Q_j(x, y) = x^{t_j} y^j$ для наименьшего возможного значения $t_j + (k-1)j$. Интерполяционные условия обрабатываются последовательно, так что на каждом шаге многочлены $Q_j(x, y)$ удовлетворяют всем уже обработанным условиям. Можно заметить, что преобразования, выполняемые на шагах 9 и 10 алгоритма, подобраны таким образом, чтобы обнулить очередную производную Хассе.

На каждом шаге алгоритма число членов в полиномах примерно равно числу уже обработанных интерполяционных условий. Сложность вычисления производных Хассе и коррекции многочленов пропорциональна числу членов в них, поэтому сложность алгоритма можно оценить как $O(n^2 \rho r^4) = O(n^2 r^5)$. Существуют, однако, и более эффективные методы решения этой задачи [69].

```
RECONSTRUCT(Q(x,y), k, i, Φ)
 1  Найти наибольшее r : x^{-r}Q(x,y) является полиномом
 2  M(x,y) ← x^{-r}Q(x,y)
 3  Найти корни γ_i ∈ 𝔽, i = 1..v, многочлена M(0,y)
 4  for j ← 1 to v
 5  do Φ[i] ← γ_j
 6     if i = k − 1
 7        then PRINT(Σ_{j=0}^{i} Φ[j]x^j)
 8        else  M̂(x,y) ← M(x, y + γ_j)
 9              M̃(x,y) ← M̂(x, xy)
10              RECONSTRUCT(M̃(x,y), k, i + 1, Φ)
```

Рис. 6.4. Поиск $f(x) : Q(x, f(x)) = 0$

6.3.3. Быстрый поиск функциональных корней

Этап факторизации, или поиска функциональных корней $f(x)$ таких, что $Q(x, f(x)) = 0$ в алгоритме Гурусвами-Судана может быть выполнен с помощью *алгоритма Рота-Рукенштейн*, который основан на следующей идее. Поделим вначале $Q(x, y)$ на наибольшую возможную степень x. Если $Q(x, f(x)) = 0$, то справедливо и равенство $0 = Q(0, f(0)) = Q(0, f_0)$, которое является уравнением с одним неизвестным. Оно может быть разрешено, например, с помощью процедуры Ченя. Для каждого найденного таким образом значения f_0 справедливо равенство $0 = Q(x, f_0 + xf'(x)) = Q'(x, f'(x))$, где $f(x) = f_0 + xf'(x)$ и $Q'(x, y) = Q(x, f_0 + xy)$. Таким образом, задача поиска $f(x)$ свелась к аналогичной задаче поиска $f'(x)$, для решения которой можно рекуррентно воспользоваться этим же методом. На рис. 6.4 представлен алгоритм, реализующий этот подход [58]. Можно показать, что он имеет сложность $O((\rho \log^2 \rho)k(n + \rho \log q))$.

6.4. Альтернантные коды

6.4.1. Конструкция кодов

Определение 6.4. *Альтернантным кодом* над $\mathbb{F}_q$ называется код с проверочной матрицей

$$H = \begin{pmatrix} a_1^0 & a_2^0 & \dots & a_n^0 \\ a_1^1 & a_2^1 & \dots & a_n^1 \\ \vdots & \vdots & \ddots & \vdots \\ a_1^{r-1} & a_2^{r-1} & \dots & a_n^{r-1} \end{pmatrix} \operatorname{diag}(u_1, u_2, \dots, u_n),$$

где $a_i \in \mathbb{F}_{q^m}$ — различные элементы, $u_i \in \mathbb{F}_{q^m} \setminus \{0\}$.

Величины a_i называются локаторами кода. Из этого определения видно, что альтернантный код состоит из всех кодовых слов обобщенного кода Рида-Соломона $GRS_{n-r}(a, v)$ над $\mathbb{F}_{q^m}$ с подходящими параметрами, все символы которых принадлежат $\mathbb{F}_q$, т.е. он является ограничением на подполе $\mathbb{F}_q$ кода $GRS_{n-r}(a, v)$. Связь величин u и v была исследована в доказательстве теоремы 6.2.

Минимальное расстояние альтернантного кода не может быть меньше минимального расстояния соответствующего обобщенного кода Рида-Соломона, т.е. $d \geq r + 1$. Аналогично случаю кодов БЧХ, размерность может быть оценена как $k \geq n - mr$. В отличие от кодов БЧХ, которые асимптотически плохи, существуют хорошие длинные альтернантные коды.

Теорема 6.4. *Пусть $m|(n-h)$. Существует альтернантный $(n, k \geq h, d \geq \delta)$ код над $\mathbb{F}_q$ такой, что*

$$\sum_{i=1}^{\delta-1} (q-1)^i C_n^i < (q^m - 1)^{(n-h)/m}. \tag{6.10}$$

Доказательство. В ходе доказательства будем предполагать, что вектор локаторов $(a_1, \dots, a_n)$ фиксирован.

Пусть $y \in \mathbb{F}_q^n$ — произвольный ненулевой вектор. Оценим число обобщенных кодов Рида-Соломона, содержащих его. Вектор принадлежит $GRS_{k'}(a, v)$ тогда и только тогда, когда $y_i/v_i = f(a_i), 1 \leq i \leq n, \deg f(x) < k'$. Выберем значения $v_i, 1 \leq i \leq k'$, произвольным образом и построим по точкам $(a_i, y_i/v_i), 1 \leq i \leq k'$, интерполяционный полином $f(x)$, после чего найдем оставшиеся v_i как $v_i = \frac{y_i}{f(a_i)}, k' < i \leq n$.

Таким образом, получим описание некоторого обобщенного кода Рида-Соломона, содержащего кодовое слово y. При этом иногда может оказаться так, что $y_i \neq 0 = f(a_i)$ или $y_i = 0 \neq f(a_i)$. В этом случае будем считать выбор начальных значений v_i неудачным. Так как число возможных вариантов выбора каждого из начальных значений v_i равно $q^m - 1$, общее число обобщенных кодов Рида-Соломона, содержащих y, не превосходит $(q^m - 1)^{k'}$.

Рассмотрим далее альтернантные коды, полученные как ограничение на $\mathbb{F}_q$ $GRS_{k'}(a, v), k' = n - (n - h)/m$. Их размерность $k \geq h$. Число различных таких альтернантных кодов, содержащих фиксированный вектор y, не превосходит число соответствующих обобщенных кодов Рида-Соломона, т.е. $(q^m-1)^{n-(n-h)/m}$. Таким образом, число альтернантных кодов, имеющих минимальное расстояние менее δ, не превосходит $(q^m-1)^{n-(n-h)/m} \sum_{i=1}^{\delta} (q-1)^i C_n^i$. Общее число обобщенных кодов Рида-Соломона равно числу различных векторов v, т.е. $(q^m - 1)^n$. Если оно превосходит число плохих кодов, т.е.

$$(q^m - 1)^{n-(n-h)/m} \sum_{i=1}^{\delta} (q-1)^i C_n^i < (q^m - 1)^n,$$

то код с требуемыми параметрами существует. □

Сопоставляя (6.10) и границу Варшамова-Гилберта (2.7), можно увидеть, что существуют альтернантные коды, лежащие на границе Варшамова-Гилберта. Однако доказанная теорема не дает конкретных способов их отыскания.

6.4.2. Коды Гоппы

Одним из наиболее интересных классов альтернантных кодов являются коды Гоппы.

Определение 6.5. Пусть заданы вектор $(a_1, \ldots, a_n)$ и *многочлен Гоппы* $G(x) = \sum_{i=0}^{r} g_i x^i \in \mathbb{F}_{q^m}[x] : G(a_i) \neq 0, 1 \leq i \leq n$. *Код Гоппы* состоит из всех векторов $(c_1, \ldots, c_n) \in \mathbb{F}_q^n$ таких, что

$$\sum_{i=1}^{n} \frac{c_i}{x - a_i} \equiv 0 \bmod G(x), \tag{6.11}$$

где a_i — *локаторы кода*.

Для того, чтобы показать принадлежность кодов Гоппы к числу альтернантных кодов, заметим, что многочлен $x - a_i$ обратим по модулю

$G(x)$, причем

$$(x-a_i)^{-1} = -\frac{G(x)-G(a_i)}{x-a_i} G^{-1}(a_i).$$

Видно, что $\frac{G(x)-G(a_i)}{x-a_i} = g_r(x^{r-1} + x^{r-2}a_i + \ldots + a_i^{r-1}) + g_{r-1}(x^{r-2} + \ldots + a_i^{r-2}) + \ldots + g_2(x + a_i) + g_1$. Группируя члены при степенях x, (6.11) можно переписать как $0 = \sum_{i=1}^{n} \frac{c_i}{G(a_i)} \cdot \frac{G(x)-G(a_i)}{x-a_i} = \sum_{i=1}^{n} \frac{c_i}{G(a_i)} \sum_{j=0}^{k} g_{r-k+j} a_i^j, 0 \le k < r$. Таким образом, проверочная матрица кода имеет вид

$$H' = \begin{pmatrix} g_r & 0 & \ldots & 0 \\ g_{r-1} & g_r & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ g_1 & g_2 & \ldots & g_r \end{pmatrix} \begin{pmatrix} a_1^0 & a_2^0 & \ldots & a_n^0 \\ a_1^1 & a_2^1 & \ldots & a_n^1 \\ \vdots & \vdots & \ddots & \vdots \\ a_1^{r-1} & a_2^{r-1} & \ldots & a_n^{r-1} \end{pmatrix} \operatorname{diag}\left(\frac{1}{G(a_1)}, \ldots, \frac{1}{G(a_n)}\right).$$

Первая матрица в этом произведении является обратимой, поэтому ее можно исключить из рассмотрения. Оставшиеся два сомножителя образуют проверочную матрицу альтернантного кода. Таким образом, размерность и минимальное расстояние кода Гоппы могут быть оценены как $k \ge n - m \deg G(x)$ и $d \ge \deg G(x) + 1$.

Ограничимся далее случаем двоичных кодов. Пусть ненулевые символы некоторого кодового слова расположены в позициях $l_1, \ldots, l_w$. Введем многочлен $f(x) = \prod_{i=1}^{w}(x - a_{l_i})$. Тогда

$$0 \equiv \sum_{i=1}^{n} \frac{c_i}{x-a_i} = \sum_{i=1}^{w} \frac{1}{x-a_{l_i}} = \frac{f'(x)}{f(x)} \bmod G(x),$$

где $f'(x)$ — формальная производная многочлена $f(x)$ по x. В силу того, что многочлены $f(x)$ и $G(x)$ взаимно просты, это означает, что $G(x)|f'(x)$. В поле характеристики 2 $f'(x)$ содержит только четные степени, а значит, является полным квадратом, т.е. $f'(x) = u^2(x)$. Пусть $Q(x)$ — полный квадрат наименьшей степени, делящийся на $G(x)$. Тогда $Q(x)|f'(x)$. Следовательно, использование $Q(x)$ вместо $G(x)$ в (6.11) никак не повлияет на множество задаваемых кодовых слов. Таким образом, можно улучшить оценку минимального расстояния до $d \ge \deg Q(x) + 1$. Очевидно, что если исходный многочлен Гоппы свободен от квадратов, то $Q(x) = G^2(x)$ и $d \ge 2 \deg G(x) + 1$. Такие коды называются *сепарабельными кодами Гоппы*.

Для декодирования кодов Гоппы могут использоваться любые алгоритмы, пригодные для декодирования обобщенных кодов Рида-

Соломона. В случае сепарабельных кодов Гоппы при этом следует использовать проверочную матрицу, соответствующую многочлену $Q(x) = G^2(x)$, так как она содержит вдвое большее число строк, чем матрица, соответствующая $G(x)$, и позволяет получить число компонентов вектора синдрома, достаточное для исправления $\deg G(x)$ ошибок в метрике Хэмминга.

Можно показать [8], что существуют коды Гоппы, лежащие на границе Варшамова-Гилберта.

6.4.3. Криптосистема Мак-Элиса

Существование большого числа альтернантных кодов и, в частности, кодов Гоппы, делает возможным использование их в криптографии.

Рассмотрим сценарий передачи сообщения от стороны А к стороне Б по открытым каналам связи. Для защиты от несанкционированного доступа необходимо зашифровать это сообщение некоторым образом. Симметричные криптосистемы, использующий один и тот же ключ для шифрования и дешифрования данных, не позволяют решить эту задачу, так как их использование фактически сводит ее к проблеме безопасной передачи ключа шифрования. Использование несимметричных криптосистем, известных также как системы шифрования с открытым ключом, позволяет преодолеть эту трудность. Эти системы предполагают использование общеизвестного публичного ключа P для шифрования сообщения u. Расшифровать его может только владелец секретного ключа S, который не должен разглашаться. Криптосистема может быть описана в виде функций шифрования $E(u, P)$ и дешифрования $D(C, S)$ таких, что $D(E(u, P), S) = M$. Безопасность данной схемы достигается в том случае, если зная C и P, вычислительно крайне сложно решить уравнение $C = E(u, P)$ относительно u (или, тем более, восстановить S).

Так как задача декодирования произвольного линейного кода в метрике Хэмминга является NP-полной, она является привлекательной для использования при построении криптосистем с открытым ключом. Предположим, что задан некоторый $(n, k, d \geq 2t+1)$ код с порождающей матрицей $G = (I|A)$, для которого известен эффективный алгоритм декодирования с исправлением t ошибок. Построим публичный ключ как $P = VGF$, где V — произвольная $k \times k$ обратимая матрица, F — перестановочная матрица. Шифрование сообщения u, представляющего собой вектор длины k, можно произвести как

$$C = E(u, P) = uP + e,$$

где e — произвольный (случайный) вектор веса t. Секретный ключ состоит из матриц F^{-1} и V^{-1}, а также данных, необходимых для использования эффективного алгоритма декодирования. Процедура дешифрования сводится к вычислению $C' = CF^{-1} = MVG + eF^{-1}$ и декодированию этого вектора. Заметим, что $\mathrm{wt}(eF^{-1}) = t$, т.е. модифицированный вектор ошибки может быть однозначно найден с помощью декодера. После его нахождения можно извлечь вектор uV как первые k символов $C' - eF^{-1}$, и воспользоваться V^{-1} для нахождения u.

Для того, чтобы восстановить сообщение, не зная секретного ключа, необходимо решить задачу декодирования по критерию минимального расстояния Хэмминга. Для этого можно воспользоваться методом декодирования по информационным совокупностям, описанным в разд. 3.1. Поиск информационной совокупности, свободной от ошибок, может быть существенно упрощен, если удалось перехватить криптограммы $y_1 = uP + e_1$ и $y_2 = uP + e_2$, соответствующие одному и тому же сообщению u. В этом случае ненулевые элементы вектора $y = y_1 + y_2 = e_1 + e_2$ расположены на тех позициях, в которых e_1 или e_2 отличны от нуля. Вместе с тем, небольшое число нулевых элементов вектора y может соответствовать случаю наличия ошибок одновременно в соответствующих символах e_1 и e_2. Таким образом, можно существенно сократить число перебираемых информационных совокупностей.

Восстановление секретного ключа по публичному в случае кодов Гоппы требует перебора всех возможных многочленов $G(x)$ и векторов локаторов $(a_1, \ldots, a_n)$. До настоящего времени найти эффективные алгоритмы решения этой задачи не удалось. Однако аналогичные структурные атаки против вариантов криптосистемы Мак-Элиса, основанной на других кодах, оказались вполне успешными.

Существенным недостатком криптосистемы Мак-Элиса является необходимость использования достаточно длинных (например, $n = 2048, k = 1751, d = 55$ [17]) кодов, что приводит к большому размеру публичного ключа.

Упражнения

1. Постройте порождающий многочлен $(10, 5, 6)$ кода Рида-Соломона над $\mathbb{F}_{11}$.
2. Докажите, что коды Рида-Соломона в смысле определения 6.1 действительно имеют минимальное расстояние $n - k + 1$.
3. Опишите код Рида-Соломона с порождающим многочленом $g(x) = (x - \beta^b)(x - \beta^{b+1}) \cdots (x - \beta^{b+n-k-1}))$ в терминах определения 6.2.

4. Покажите, что циклический $(2^m - 1, k, d)$ код Рида-Соломона над $\mathbb{F}_{2^m}$ содержит подкод, являющийся кодом БЧХ с конструктивным расстоянием d.
5. Найдите все кодовые слова $(5, 2, 4)$ кода Рида-Соломона над $\mathbb{F}_5$ с вектором локаторов $(0, 1, 2, 3, 4)$, отличающиеся от вектора $(1, 2, 0, 0, 0)$ не более чем в двух позициях.

7. Коды Рида-Маллера

7.1. Конструкция кодов

7.1.1. Построение с помощью полиномов Жегалкина

Определение 7.1. *Кодом Рида-Маллера* $RM(r,m)$ порядка r и длины 2^m называется множество векторов, полученных путем вычисления значений многочленов

$$a(V_1,\ldots,V_m)=\sum_{i_1+\ldots+i_m\le r} a_{i_1,\ldots,i_m}V_1^{i_1}\cdots V_m^{i_m}, i_1,\ldots,i_m\in\{0,1\}$$

во всех точках $(v_1,\ldots,v_m)\in\mathbb{F}_2^m$.

Полиномы, лежащие в основе этой конструкции, носят название *многочленов Жегалкина*. Ясно, что коды Рида-Маллера являются линейными и имеют размерность $k=\sum_{i=0}^{r}C_m^i$. Кроме того, коды Рида-Маллера являются вложенными, т.е. $RM(0,m)\subset RM(1,m)\subset\ldots\subset RM(m,m)$. Первый код в этой цепочке является кодом $(2^m,1,2^m)$ с повторениями, в то время как последний — тривиальным $(2^m,2^m,1)$ кодом.

Пример 7.1. Порождающие матрицы кодов Рида-Маллера длины 8 образованы следующими векторами, полученными путем подстановки $V_i=v_i$ в соответствующие одночлены:

$a_{(0,0,0)}$	1	1	1	1	1	1	1	1	1
$a_{(1,0,0)}$	V_1	0	1	0	1	0	1	0	1
$a_{(0,1,0)}$	V_2	0	0	1	1	0	0	1	1
$a_{(0,0,1)}$	V_3	0	0	0	0	1	1	1	1
$a_{(1,1,0)}$	V_1V_2	0	0	0	1	0	0	0	1
$a_{(0,1,1)}$	V_2V_3	0	0	0	0	0	0	1	1
$a_{(1,0,1)}$	V_1V_3	0	0	0	0	0	1	0	1
$a_{(1,1,1)}$	$V_1V_2V_3$	0	0	0	0	0	0	0	1

(7.1)

В дальнейшем будем помечать столбцы порождающей матрицы и символы кодовых слов соответствующими векторами $(v_1,\ldots,v_m)\in GF(2)^m$.

Теорема 7.1. *Кодом, дуальным к* $RM(r,m)$*, является* $RM(m-r-1,m)$.

Доказательство. Степень любого полинома Жегалкина, соответствующего кодовому слову $RM(r,m)$, не превосходит r. Степень любого полинома Жегалкина, соответствующего кодовому слову $RM(m-r-1,m)$,

не превосходит $m-r-1$. Скалярное произведение $c = c' \cdot c'', c' \in RM(r,m), c'' \in RM(m-r-1,m)$ может быть представлено как вектор значений многочлена Жегалкина, равного произведению многочленов, соответствующих c' и c'', во всех точках $\mathbb{F}_2^m$. Степень такого многочлена не превосходит $m-1$. Каждый одночлен $\prod_{j=1}^{m} V_j^{i_j}, i_j \in \{0,1\}$ такой, что некоторые $i_j = 0$, принимает ненулевое значение в четном числе позиций. Эти позиции соответствуют различным значениям тех переменных, для которых $i_j = 0$. Следовательно, c имеет четный вес, т.е. c' и c'' всегда ортогональны. □

7.1.2. Построение с помощью конструкции Плоткина

Коды Рида-Маллера могут быть также получены с помощью конструкции Плоткина $(C_1, C_1 + C_2)$ (см. разд. 12.2.3). Действительно, всякий полином Жегалкина может быть представлен как $a(V_1, \ldots, V_{m+1}) = b(V_1, \ldots, V_m) + V_{m+1}c(V_1, \ldots, V_m)$. Упорядочим лексикографически все двоичные векторы $(v_1, \ldots, v_{m+1})$ длины $m+1$ вначале по переменной v_{m+1}, затем по v_m и т.д. Тогда при вычислении значений $a(v_1, \ldots, v_{m+1})$ последовательность $b(v_1, \ldots, v_m)$ повторится два раза, а вклад второго слагаемого на первой половине точек окажется равен нулю. Таким образом, если $G(r,m)$ — порождающая матрица кода $RM(r,m)$, то $G(r+1, m+1) = \begin{pmatrix} G(r+1,m) & G(r+1,m) \\ 0 & G(r,m) \end{pmatrix}$.

Теорема 7.2. *Минимальное расстояние кода $RM(r,m)$ равно $D(r,m) = 2^{m-r}$.*

Доказательство. Для $r = 0$ и $r = m$ это утверждение очевидно. Предположим, что оно верно для всех $r, m : m \leq m'$ и $0 \leq r \leq m$. Тогда в силу теоремы 12.1, $D(r, m'+1) = \min(2D(r,m'), D(r-1,m')) = 2^{m'+1-r}$. □

7.2. Декодирование

7.2.1. Жесткое декодирование

Рассмотрим декодирование кодов Рида-Маллера в метрике Хэмминга. Принятый сигнал представим как

$$y_{v_1 \ldots v_m} = a(v_1, \ldots, v_m) + e_{v_1 \ldots v_m}, \deg a(v_1, \ldots, v_m) \leq r, v_i, e_{v_1 \ldots v_m} \in \mathbb{F}_2 \,. \tag{7.2}$$

Будем искать многочлен $\hat{a}(v_1, \ldots, v_m)$, значения которого отличались бы от величин $y_{v_1 \ldots v_m}$ не более чем в $\frac{D(r,m)-1}{2}$ позициях.

Свойство вложенности кодов Рида-Маллера позволяет организовать их поэтапное декодирование. Будем считать, что информационные символы задаются коэффициентами $a_{i_1,\ldots,i_m}$ полинома Жегалкина, зашумленный вектор значений которого поступил на вход декодера.

Вначале попытаемся найти коэффициенты при членах, имеющих степень r. Если бы ошибок не было, их можно было бы вычислить как

$$a_{i_1\ldots,i_m} = \sum_{v\in X_j(i_1,\ldots,i_m)} y_v, \tag{7.3}$$

где суммирование производится по модулю 2 и $X_j(i_1,\ldots,i_m)$ — множество из $2^{wt((i_1,\ldots,i_m))}$ булевских векторов, принимающих все возможные значения в позициях $k : i_k = 1$, и с фиксированными значениями, определяемыми номером j, в других позициях. Таких тождеств можно выписать 2^{m-r} штук. Для того, чтобы показать справедливость (7.3), заметим, что

$$\begin{aligned}\sum_{v\in X_j(i_1,\ldots,i_m)} y_v &= \sum_{v\in X_j(i_1,\ldots,i_m)}\ \sum_{\beta:\mathrm{wt}(\beta)\le r} a_\beta v_1^{\beta_1}\cdots v_m^{\beta_m} = \\ &= \sum_{\beta:\mathrm{wt}(\beta)\le r} \underbrace{\left(\sum_{v\in X_j(i_1,\ldots,i_m)} a_\beta v_1^{\beta_1}\cdots v_m^{\beta_m}\right)}_{N(X_j(i_1,\ldots,i_m),\beta)} \mathrm{mod}2.\end{aligned}$$

Величина $N(X_j(i_1,\ldots,i_m),\beta)$ равна числу элементов в множестве $X_j(i_1,\ldots,i_m)$ (т.е. столбцов порождающей матрицы), имеющих 1 во всех позициях, помеченных единицами в векторе β. При вычислении $N(X_j(i_1,\ldots,i_m),\beta)$ перебираются векторы v, принимающие все возможные значения в позициях $k : i_k = 1$. При $\beta = (i_1,\ldots,i_m)$ только один из них содержит единицы на всех ненулевых позициях β. Следовательно, $N(X_j(i_1,\ldots,i_m),(i_1,\ldots,i_m)) = 1$. Если $\beta \neq (i_1,\ldots,i_m)$, число таких элементов четное и $N(X_j(i_1,\ldots,i_m),\beta) \equiv 0 \bmod 2$.

Однако в реальности величины y_v могут быть искажены ошибками. В этом случае можно воспользоваться тем, что тождества (7.3), соответствующие различным j, используют непересекающиеся наборы символов y_v, т.е. одна ошибка в принятой последовательности оказывает влияние ровно на одну вычисленную таким образом оценку. Это дает возможность построить 2^{m-r} независимых оценок для $a_{i_1\ldots,i_m}$ и выбрать в качестве окончательного решения значение $\hat{a}_{i_1\ldots,i_m}$, встречающееся чаще всего среди вычисленных оценок (метод голосования). Ес-

ли имело место менее 2^{m-r-1} ошибок, это решение окажется правильным. После этого можно вычесть из принятого вектора вектор значений $\sum_{\beta:\mathrm{wt}\,\beta=r} \hat{a}_\beta v_1^{\beta_1} \cdots v_m^{\beta_m}$, где $\hat{a}_\beta$ — найденные информационные символы, и продолжить декодирование в коде $RM(r-1,m)$.

Пример 7.2. Код $RM(2,4)$ имеет порождающую матрицу

$$G = \begin{pmatrix} 1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1 \\ 0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1 \\ 0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1 \\ 0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1 \\ 0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1 \\ 0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&1 \\ 0&0&0&0&0&1&0&1&0&0&0&0&0&1&0&1 \\ 0&0&0&0&0&0&0&0&0&1&0&1&0&1&0&1 \\ 0&0&0&0&0&0&1&1&0&0&0&0&0&0&1&1 \\ 0&0&0&0&0&0&0&0&0&0&1&1&0&0&1&1 \\ 0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1 \end{pmatrix} \begin{matrix} a_{0000} \\ a_{1000} \\ a_{0100} \\ a_{0010} \\ a_{0001} \\ a_{1100} \\ a_{1010} \\ a_{1001} \\ a_{0110} \\ a_{0101} \\ a_{0011} \end{matrix}$$

Этот код способен исправить одну ошибку. Рассмотрим декодирование вектора $y = (1,1,1,1,1,1,1,1,0,0,0,0,1,1,0,1)$ в этом коде. Выпишем тождества для коэффициентов при членах второго порядка полинома Жегалкина[1], соответствующих правильным кодовым словам:

$$\begin{array}{llll}
a_{1100} = y_0 + y_1 + y_2 + y_3 & = y_4 + y_5 + y_6 + y_7 & = y_8 + y_9 + y_{10} + y_{11} & = y_{12} + y_{13} + y_{14} + y_{15} \\
a_{1010} = y_0 + y_1 + y_4 + y_5 & = y_2 + y_3 + y_6 + y_7 & = y_8 + y_9 + y_{12} + y_{13} & = y_{10} + y_{11} + y_{14} + y_{15} \\
a_{1001} = y_0 + y_1 + y_8 + y_9 & = y_2 + y_3 + y_{10} + y_{11} & = y_4 + y_5 + y_{12} + y_{13} & = y_6 + y_7 + y_{14} + y_{15} \\
a_{0110} = y_0 + y_2 + y_4 + y_6 & = y_1 + y_3 + y_5 + y_7 & = y_8 + y_{10} + y_{12} + y_{14} & = y_9 + y_{11} + y_{13} + y_{15} \\
a_{0101} = y_0 + y_2 + y_8 + y_{10} & = y_1 + y_3 + y_9 + y_{11} & = y_4 + y_6 + y_{12} + y_{14} & = y_5 + y_7 + y_{13} + y_{15} \\
a_{0011} = y_0 + y_4 + y_8 + y_{12} & = y_1 + y_5 + y_9 + y_{13} & = y_2 + y_6 + y_{10} + y_{14} & = y_3 + y_7 + y_{11} + y_{15}
\end{array}$$

В декодируемом векторе присутствует ошибка, поэтому не все выражения из каждой строки дают один и тот же результат. После голосования получим $\hat{a}_{1100} = 0, \hat{a}_{1010} = 0, \hat{a}_{1001} = 0, \hat{a}_{0110} = 0, \hat{a}_{0101} = 0, \hat{a}_{0011} = 1$. Вычтем из вектора y линейную комбинацию последних шести строк порождающей матрицы с вышеуказанными коэффициентами (т.е. только одиннадцатую строку). Получим $y' = (1,1,1,1,1,1,1,1,0,0,0,0,0,0,1,0)$. Этот вектор необходимо продекодировать в коде $RM(1,4)$.

Выпишем набор тождеств для коэффициентов при членах первого порядка полинома Жегалкина:

$$\begin{array}{llllll}
a_{1000} = y'_0 + y'_1 = y'_2 + y'_3 = y'_4 + y'_5 & = y'_6 + y'_7 & = y'_8 + y'_9 & = y'_{10} + y'_{11} = y'_{12} + y'_{13} = y'_{14} + y'_{15} \\
a_{0100} = y'_0 + y'_2 = y'_1 + y'_3 = y'_4 + y'_6 & = y'_5 + y'_7 & = y'_8 + y'_{10} = y'_9 + y'_{11} & = y'_{12} + y'_{14} = y'_{13} + y'_{15} \\
a_{0010} = y'_0 + y'_4 = y'_1 + y'_5 = y'_2 + y'_6 & = y'_3 + y'_7 & = y'_8 + y'_{12} = y'_9 + y'_{13} & = y'_{10} + y'_{14} = y'_{11} + y'_{15} \\
a_{0001} = y'_0 + y'_8 = y'_1 + y'_9 = y'_2 + y'_{10} = y'_3 + y'_{11} = y'_4 + y'_{12} = y'_5 + y'_{13} & = y'_6 + y'_{14} & = y'_7 + y'_{15}
\end{array}$$

Декодируемый вектор по-прежнему содержит ошибку, поэтому снова не все тождества дают одинаковый результат. Проведя голосование, получим $\hat{a}_{1000} = 0, \hat{a}_{0100} = 0, \hat{a}_{0010} = 0, \hat{a}_{0001} = 1$. Вычитая из вектора y' пятую строку порождающей матрицы, получим $y'' = (1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1)$. Этот вектор необходимо продекодировать в коде $RM(0,4)$.

[1]Символы кодового слова нумеруются десятичными числами в интервале с 0 по 15.

Данный код является $(16, 1, 16)$-кодом с повторениями и описанная процедура декодирования сводится к выбору наиболее часто встречающегося символа декодируемого вектора. Таким образом, $\hat{a}_{0000} = 1$. Исходное кодовое слово $RM(2, 4)$ кода равно $(1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1)$.

Таким образом, жесткое декодирование вектора r в коде $RM(r, m)$ включает в себя следующие шаги:

1. Выписать тождества (7.3) для коэффициентов полинома Жегалкина при членах порядка r .
2. Для каждого коэффициента выбрать в качестве окончательного решения значение $\hat{a}_{i_1\ldots,i_m}$, встречающееся чаще всего среди оценок, вычисленных с помощью найденных тождеств.
3. Если $r = 0$, декодирование окончено. В противном случае вычислить $y' : y'_{v_1\ldots v_m} = y_{v_1\ldots v_m} - \sum_{\beta:\mathrm{wt}\,\beta=r} \hat{a}_\beta (v_1^{\beta_1} \cdots v_m^{\beta_m}) \bmod 2$ и продекодировать вектор y' в коде $RM(r-1, m)$ аналогичным образом.

Описанная процедура обеспечивает исправление всех ошибок до половины минимального расстояния кода.

7.2.2. Мягкое декодирование

Рассмотренный алгоритм жесткого декодирования может быть обобщен на случай мягкого декодирования, т.е. на тот случай, когда известны вероятности того, что символы переданного кодового слова равны 0 или 1 [28]. Эти вероятности удобно представить в виде логарифмических отношений правдоподобия[2] $L_{v_1\ldots v_m} = \ln \frac{P\{y_{v_1,\ldots,v_m}|a(v_1,\ldots,v_m)=0\}}{P\{y_{v_1,\ldots,v_m}|a(v_1,\ldots,v_m)=1\}}$. Будем предполагать, что передача осуществляется по каналу без памяти.

Оценим апостериорную вероятность того, что коэффициент порядка r полинома Жегалкина равен 1. Для этого предположим, что величины y_i, входящие в различные тождества (7.3) для рассматриваемого коэффициента $a_{i_1\ldots i_m}$, независимы. Это, конечно, неверно, так как они являются зашумленными символами одного кодового слова, а смысл операции кодирования состоит в том, чтобы сделать передаваемые по каналу символы статистически зависимыми. Тем не менее, этот подход позволяет получить приближенную оценку $P\{a_{i_1\ldots i_m} = 1|y_0^{n-1}\} = \frac{p(y_0^{n-1}|a_{i_1\ldots i_m}=1)P\{a_{i_1\ldots i_m}=1\}}{p(y_0^{n-1})} \approx \frac{P\{a_{i_1\ldots i_m}=1\}}{p(y_0,\ldots,y_n)} \prod_{j=1}^{2^{wt((i_1,\ldots,i_m))}} p(y_k, k \in X_j(i_1,\ldots,i_m)|a_{i_1\ldots i_m} = 1) =$

[2] В разд. 9.2.2 логарифмическое отношение правдоподобия было определено как $L(c_i) = \ln \frac{P\{y_i|c_i=1\}}{P\{y_i|c_i=0\}}$. Изменение определения позволяет несколько упростить преобразования.

$\frac{P\{a_{i_1\ldots im}=1\}}{p(y_0^{n-1})}\prod_{j=1}^{2^{wt((i_1,\ldots,i_m))}}\left(\frac{P\{a_{i_1\ldots im}=1|y_k,k\in X_j(i_1,\ldots,i_m)\}p(y_k,k\in X_j(i_1,\ldots,i_m))}{P\{a_{i_1\ldots im}=1\}}\right)$. Аналогичные выражения можно получить для апостериорной вероятности равенства $a_{i_1\ldots i_m}$ нулю. Предполагая, что $P\{a_{i_1\ldots i_m}=1\}=P\{a_{i_1\ldots i_m}=0\}$, получим

$$\ln\frac{P\{a_{i_1\ldots i_m}=0|y_0^{n-1}\}}{P\{a_{i_1\ldots i_m}=1|y_0^{n-1}\}}=\sum_{j=1}^{2^{wt((i_1,\ldots,i_m))}}\ln\frac{P\{a_{i_1\ldots i_m}=0|y_k,k\in X_j(i_1,\ldots,i_m)\}}{P\{a_{i_1\ldots i_m}=1|y_k,k\in X_j(i_1,\ldots,i_m)\}}. \tag{7.4}$$

Лемма 7.1. *Пусть величины x_i принимают значения 1 и 0 с вероятностями p_i и $1-p_i$ соответственно. Тогда*

$$P\left\{x_1+\ldots+x_m\equiv 0 \bmod 2\right\}=\frac{1}{2}+\frac{1}{2}\prod_{i=1}^{m}(1-2p_i).$$

Доказательство. При $m=1$ утверждение очевидно. Предположим, что оно верно для $m-1$ переменной. Тогда $P\left\{x_1+\ldots+x_m\equiv 0 \bmod 2\right\}=P\left\{x_1+\ldots+x_{m-1}=0 \bmod 2, x_m=0\right\}+P\left\{x_1+\ldots+x_{m-1}=1 \bmod 2, x_m=1\right\}=\left(\frac{1}{2}+\frac{1}{2}\prod_{i=1}^{m-1}(1-2p_i)\right)(1-p_i)+\left(\frac{1}{2}-\frac{1}{2}\prod_{i=1}^{m-1}(1-2p_i)\right)p_i$. □

Несложно заметить, что если $p_0+p_1=1$, то выполняется тождество

$$\tanh\left(\frac{1}{2}\ln\left(p_0/p_1\right)\right)=\frac{\exp(\ln(p_0/p_1))-1}{\exp(\ln(p_0/p_1))+1}=p_0-p_1=1-2p_1. \tag{7.5}$$

Следовательно,

$$\tanh\left(\frac{1}{2}\underbrace{\ln\frac{P\{a_{i_1\ldots i_m}=0|y_k,k\in X_j(i_1,\ldots,i_m)\}}{P\{a_{i_1\ldots i_m}=1|y_k,k\in X_j(i_1,\ldots,i_m)\}}}_{L_{j,(i_1,\ldots,im)}}\right)=\prod_{l\in X_j(i_1,\ldots,i_m)}\tanh\left(\frac{1}{2}L_l\right).$$

Вводя обозначения $u_l=\operatorname{sgn}(L_l), b_l=|L_l|, \phi(x)=-\ln\tanh(x/2)$ и учитывая, что $\phi(x)=\phi^{-1}(x)$, получим

$$L_{j,(i_1,\ldots,i_m)}\approx\left(\prod_{l\in X_j(i_1,\ldots,i_m)}u_l\right)\phi\left(\sum_{l\in X_j(i_1,\ldots,i_m)}\phi(b_l)\right). \tag{7.6}$$

Необходимо отметить, что в данном выражении перемножаются величины $u_l = \pm 1$, что соответствует операции "исключающее или". Непосредственное вычисление функции $\phi(x)$ в области малых и больших аргументов может привести к значительным ошибкам округления.

Как только найдены апостериорные логарифмические отношения правдоподобия коэффициентов $a_{i_1 \ldots i_m}$ при членах порядка r, можно принять решения относительно значений этих коэффициентов и вычислить соответствующую линейную комбинацию $\tilde{y}$ строк порождающей матрицы. В случае жесткого декодирования эта линейная комбинация вычиталась из принятого вектора. В данном случае принятый вектор представлен в виде вектора логарифмических отношений правдоподобия, поэтому вычитание заменяется изменением знака тех величин L_l, которым соответствуют единицы в векторе $\tilde{y}$. После этого необходимо продолжить декодирование в коде $RM(r-1, m)$.

Таким образом, мягкое декодирование кода Рида-Маллера $RM(r, m)$ может быть произведено следующим образом:

1. Выписать тождества (7.3) для коэффициентов полинома Жегалкина при членах порядка r .
2. Вычислить логарифмические отношения правдоподобия L_l символов кодового слова.
3. Для каждого коэффициента при члене порядка r полинома Жегалкина и для каждого соответствующего ему тождества j вычислить (7.6).
4. Найти апостериорное логарифмическое отношение правдоподобия для каждого коэффициента при члене порядка r полинома Жегалкина в соответствии с (7.4). Принять решение $\hat{a}_{i_1 \ldots i_m}$ относительно этих коэффициентов.
5. Вычислить линейную комбинацию y' строк порождающей матрицы, соответствующих найденным коэффициентам $\hat{a}_{i_1 \ldots i_m}$. Изменить знак у величин $L_l : y'_l = 1$.
6. Продолжить декодирование в коде $RM(r-1, m)$.

Для оценки корректирующей способности кодов Рида-Маллера можно воспользоваться объединенной верхней границей (2.12). При этом можно ограничиться первым ненулевым членом (т.е. соответствующим $i = d = 2^{m-r}$) этой суммы. При этом известно, что

$$A_{2^{m-r}} = 2^r \prod_{i=0}^{m-r-1} \frac{2^{m-i} - 1}{2^{m-r-i} - 1}.$$

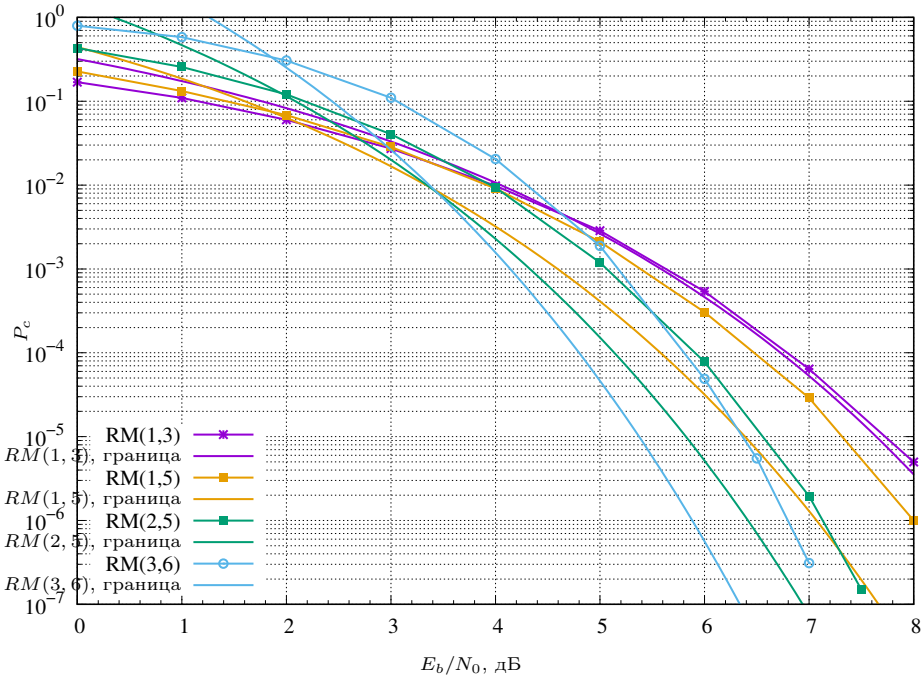


Рис. 7.1. Вероятность ошибки декодирования кодов Рида-Маллера для случая канала с АБГШ

На рис. 7.1 представлены графики зависимости вероятности ошибки декодирования некоторых кодов Рида-Маллера с помощью описанного метода, полученные с помощью имитационного моделирования, а также верхняя граница для вероятности ошибки декодирования по максимуму правдоподобия. Видно, что вероятность ошибки, достигаемая рассмотренным методом, приближается к вероятности ошибки декодирования по максимуму правдоподобия только для кода $RM(1,3)$. Для других рассмотренных кодов проигрыш составляет от 0,5 до 1 дБ.

Упражнения

1. Продекодируйте в коде Рида-Маллера $RM(2,5)$ любой вектор веса 29 с помощью мажоритарного метода.
2. Продекодируйте в коде Рида-Маллера $RM(2,5)$ с помощью декодера Плоткина (см. разд. 12.2.3) любой вектор веса 29.

8. Полярные коды

8.1. Поляризация канала

8.1.1. Основная идея

Рассмотрим преобразование $(c_0, c_1) = (u_0, u_1)\begin{pmatrix}1 & 0\\ 1 & 1\end{pmatrix}$. Предположим, что символы c_i передаются по двоичному стирающему каналу с вероятностью стирания Z, как показано на рис. 8.1. Пусть y_0, y_1 — символы, наблюдаемые на выходе канала. Используя эти символы, можно попытаться восстановить значение u_0. Действительно, если оба они не стерты, т.е. равны c_0, c_1, справедливо равенство $u_0 = c_0 + c_1 = y_0 + y_1$. Таким образом, символ u_0 не может быть восстановлен, если хотя бы один из y_i стерт, вероятность чего равна $Z_{1,0} = 2Z - Z^2 \geq Z$. Предположим теперь, что на стороне приемника известно истинное значение u_0, и рассмотрим нахождение u_1. Видно, что $u_1 = c_0 + u_0 = c_1$. Таким обарзом, символ u_1 не может быть восстановлен, если оба символа y_0, y_1 стерты. Вероятность этого равна $Z_{1,1} = Z^2 \leq Z$. Таким образом, исходный двоичный стирающий канал как бы расщепляется на два подканала, один из которых чуть лучше, а второй — чуть хуже, чем исходный канал. Это преобразование можно проделать многократно, как показано на рис. 8.2.

Видно, что в результате получились подканалы с вероятностями стирания близкими к 0 и 1. Для того, чтобы организовать надежную передачу информации, естественным решением является задействовать подканалы с низкой вероятностью стирания, а по плохим подканалам передавать некоторые предопределенные значения. Соответствующие символы u_i будем называть замороженными. Такая схема преобразования данных на стороне передатчика может быть описана как

$$c_0^{n-1} = u_0^{n-1} A_m, u_i = 0, i \in \mathcal{F}, \tag{8.1}$$

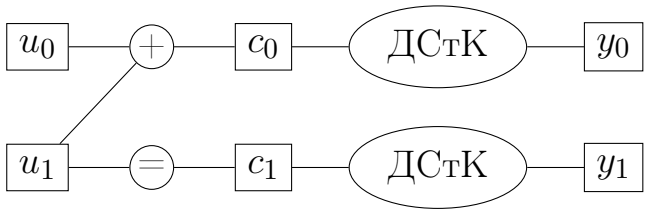


Рис. 8.1. Элементарное поляризующее преобразование

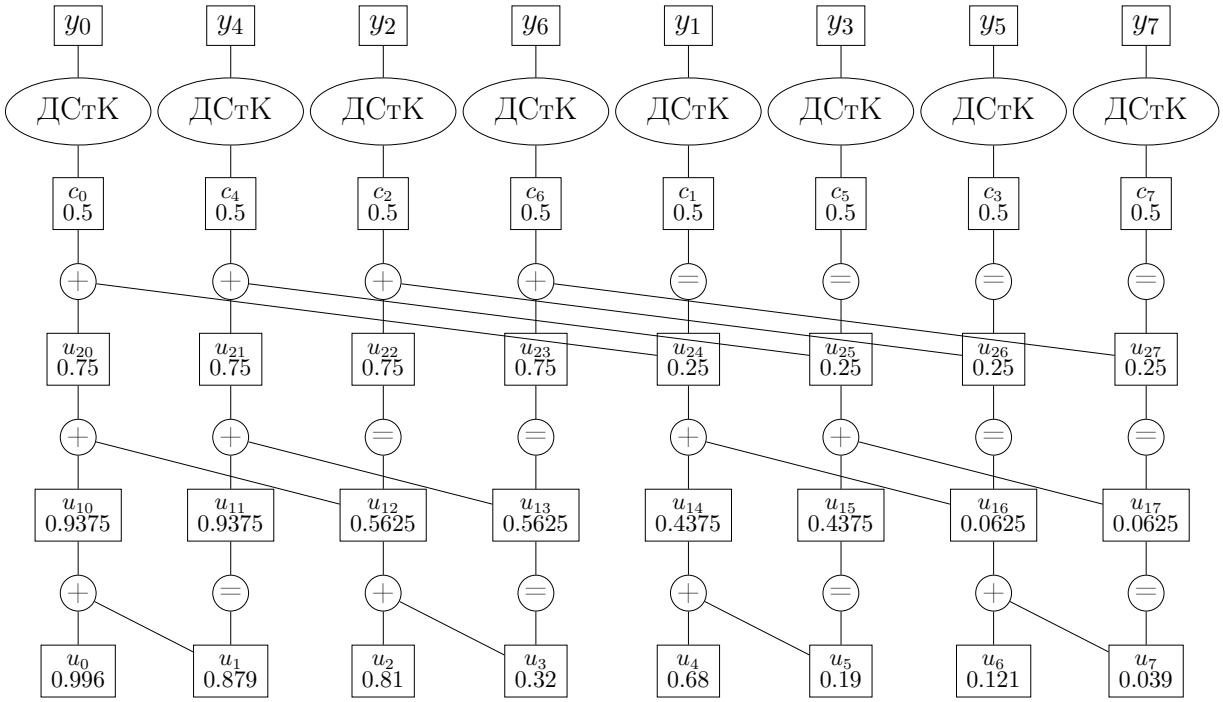


Рис. 8.2. Поляризация двоичного стирающего канала с вероятностью стирания $Z = 0.5$

где $a_i^j = (a_i, a_{i+1}, \ldots, a_j)$, $A_m = K^{\otimes m}$, $\mathcal{F}$ — множество номеров замороженных символов (множество замораживания), K — $l \times l$ матрица, называемая ядром поляризующего преобразования, и $\otimes m$ обозначает m-кратное произведение Кронекера матрицы с самой собой. Произведением Кронекера матрицы A размерности $m \times n$ и матрицы B размерности $p \times q$ называется матрица размерности $(mp) \times (nq)$ вида

$$A \otimes B = \begin{pmatrix} a_{0,0}B & a_{0,1}B & \ldots & a_{0,n-1}B \\ a_{1,0}B & a_{1,1}B & \ldots & a_{1,n-1}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m-1,0}B & a_{m-1,1}B & \ldots & a_{m-1,n-1}B \end{pmatrix}$$

В рассмотренных выше примерах использовалось ядро Арикана $K_2 = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$. Схема, задаваемая (8.1), соответствует кодированию данных линейным блоковым кодом $(n = l^m, l^m - |\mathcal{F}|)$, порождающая матрица G которого образована строками матрицы A_m, номера которых не содержатся в множестве $\mathcal{F}$. Такой код называется *полярным*.

Пример 8.1. Для $\mathcal{F} = \{0, 1, 2, 4\}$, $m = 3$ и ядра Арикана получим

$$A_3 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}, G = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix},$$

т.е. полярный код совпадает с $(8, 4, 4)$ кодом Рида-Маллера первого порядка.

8.1.2. Общий случай

Явление поляризации наблюдается не только для двоичного стирающего канала. Пусть дан некоторый симметричный по входу канал без памяти[1] с функцией переходных вероятностей[2] $W(y|x) = P_{Y|X}\{y|x\}, x \in \mathbb{X} = \{0, 1\}, y \in \mathbb{Y}$. Определим синтетические битовые подканалы с функциями переходных вероятностей

$$W_m^{(i)}(y_0^{n-1}, u_0^{i-1}|u_i) = \frac{1}{2^{n-1}} \sum_{u_{i+1}^{n-1} \in \mathbb{X}^{n-i-1}} \prod_{j=0}^{n-1} W(y_j|(u_0^{n-1}A_m)_j), u_i \in \mathbb{X}.$$

Выходным алфавитом i-ого подканала является множество $\mathbb{Y}^n \times \mathbb{X}^{i-1}$, т.е. выходом считается выход реального канала, соответствующий $n = l^m$ переданным символам, а также сведения о символах $u_j, j < i,$, переданных по подканалам $0, \ldots, i-1$. Конечно, в реальном приемнике точная информация об этих символах недоступна. Но ниже будет показано, что это не является проблемой.

Можно показать, что если матрица K обратима, и никакая перестановка ее столбцов не приводит к верхнетреугольной матрице, то параметры Бхаттачарьи битовых подканалов удовлетворяют [14, 47]

$$\forall \beta : 0 < \beta < E(K) : \liminf_{m\to\infty} P\left\{Z(W_m^{(i)}) \le 2^{-l^{m\beta}}\right\} = C(W)$$
$$\forall \beta > E(K) : \liminf_{m\to\infty} P\left\{Z(W_m^{(i)}) \ge 2^{-l^{m\beta}}\right\} = 1.$$

[1] Известны также обобщения полярных кодов на случай каналов с памятью и несимметричных каналов.

[2] Для простоты мы ограничимся здесь рассмотрением каналов с дискретным выходным алфавитом $\mathbb{Y}$. Все представленные результаты могут быть обобщены на случай непрерывного $\mathbb{Y}$

Здесь $C(W)$ — пропускная способность канала W, $E(K)$ — скорость поляризации ядра K. Кроме того,

$$\forall \delta > 0 : \lim_{m\to\infty} P\left\{C(W_m^{(i)}) \in (\delta, 1-\delta)\right\} = 0$$

$$\forall m > 0 : \sum_{i=0}^{n-1} C(W_m^{(i)}) = nC(W)$$

Вероятности в этих выражениях следует интерпретировать как долю чисел $i : 0 \le i < n$, для которых выполняется соответствующее условие. Таким образом, при увеличении числа слоев m поляризующего преобразования почти все битовые подканалы сходятся или к почти бесшумным, или почти полностью зашумленным, причем доля почти бесшумных подканалов равна пропускной способности исходного канала W.

Скорость поляризации ядра может быть вычислена как

$$E(K) = \frac{1}{l}\sum_{i=0}^{l} \log_l D_i, \tag{8.2}$$

где $D_i = d_H(K_i, \langle K_{i+1}, \ldots, K_{l-1}\rangle)$ — частичные расстояния ядра K, K_i — строка i матрицы K. i-ое частичное расстояние равно минимальному расстоянию Хемминга от K_i до всевозможных линейных комбинаций нижележащих строк ядра. В частности, для ядра Арикана $D_0 = 1, D_1 = 2$ и $E(K_2) = 0.5$.

Рекурсивная структура матрицы A_m позволяет выполнить умножение на нее, т.е. произвести кодирование полярного кода, со сложностью $O(n \log n)$. Например, в случае ядра Арикана видно, что $A_m = \begin{pmatrix} A_{m-1} & 0 \\ A_{m-1} & A_{m-1} \end{pmatrix}$. Следовательно, вычисление $(c', c'') = (u', u'')A_m$ сводится к нахождению $v' = u'A_{m-1}$, $v'' = u''A_{m-1}$ и $c' = v' + v'', c'' = v''$. Сложность такого алгоритма равна $T(n) = 2T(n/2) + n/2 = \frac{n}{2}\log_2 n$.

8.2. Декодирование полярных кодов

8.2.1. Алгоритм последовательного исключения

Декодирование полярных кодов может быть выполнено с помощью *алгоритма последовательного исключения*, который последовательно для $i = 0, 1, \ldots, l^m - 1$ формирует оценки

$$\widehat{u}_i = \begin{cases} 0 & i \in \mathcal{F} \\ \arg\max_{u_i} W_m^{(i)}(\widehat{u}_0^{i-1}, u_i | y_0^{n-1}) & i \notin \mathcal{F}\,. \end{cases}$$

Здесь $W_m^{(i)}(u_0^i|y_0^{n-1}) = \frac{W(y_0^{n-1},u_0^{i-1}|u_i)}{2W(y_0^{n-1})}$.

Если оценки $\widehat{u}_0^{i-1}$ являются правильными, то вычисляемые при этом вероятности в точности равны соответствующим переходным вероятностям битовых подканалов. Если хотя бы одно из решений $\hat{u}_i$ оказывается неправильным, то происходит ошибка декодирования. В этом случае уже не имеет значения то, что на последующих фазах i вероятности вычисляются неправильно.

Необходимо отметить, что оценки $\hat{u}_i$, формируемые алгоритмом последовательного исключения, не являются оптимальными, так как не учитывают условия замораживания $u_j = 0, j > i, j \in \mathcal{F}$. Тем не менее, полярные коды достаточно большой длины при декодировании методом последовательного исключения позволяют обеспечить сколь угодно малую вероятность ошибки на любой скорости, меньшей $C(W)$, т.е. достигают пропускной способности канала [14, 47].

В силу рекурсивной структуры матрицы A_m можно заметить, что

$$W_m^{(li+s)}(u_0^{li+s}|y_0^{n-1}) = \sum_{u_{li+s+1}^{li+l-1}} \prod_{j=0}^{l-1} W_{m-1}^{(i)}\left((u_{lt}^{l(t+1)-1}K), 0 \le t \le i|y_{j,l}^{n-1}\right), \tag{8.3}$$

где $W_0^{(0)}(x|y) = P_{X|Y}(x|y)$, $y_{j,l}^{n-1} = (y_j, y_{j+l}, \ldots, y_{j+n-l})$, $0 \le s < l$, $0 \le i < \frac{n}{l}$. Вычисление значений этого выражения носит название обработки или маргинализации ядра. Непосредственное вычисление значений (8.3) требует $O(2^l l)$ операций. Более эффективные алгоритмы рассмотрены в [75, 73]. Для практически наиболее важного случая ядра Арикана вычисление этого выражения подробно рассмотрено в разделе 8.4.

Можно заметить, что при $s > 0$ выражение (8.3) включает в себя те же самые значения $W_{m-1}^{(i)}$, что и при $s = 0$. Эти значения могут быть вычислены однократно и затем использованы для различных s. При этом к (8.3) на слое m придется обратиться n раз, на слое $m-1$ эта формула будет задействована для вычисления l блоков по n/l вероятностей, и т.д. Таким образом, общее число операций обработки ядра, выполняемых алгоритмом последовательного исключения, составляет $n \log_l n$.

Вероятность ошибки декодирования методом последовательного исключения может быть оценена как

$$P_{SC} \leq \sum_{i \notin \mathcal{F}} Z(W_m^{(i)}). \tag{8.4}$$

8.2.2. Списочный алгоритм Тала-Варди

Как было отмечено выше, алгоритм последовательного исключения при формировании решений относительно информационных символов $\hat{u}_i, i \notin \mathcal{F}$, не учитывает ограничения, наложенные на символы $u_j, j > i$. Из-за этого его корректирующая способность на кодах с практически значимыми длинами оказывается неудовлетворительной. Она может быть существенно улучшена путем рассмотрения в декодере не более L векторов $\hat{u}_0^{i-1}$, называемых путями. На фазе i для каждого из путей строятся его возможные продолжения $\hat{u}_0^i$. Если $i \in \mathcal{F}$, значения $\hat{u}_i$ определяются однозначно и число путей не меняется. В противном случае оно удваивается. Из получившихся путей выбираются L с наибольшими значениями вероятностей $W_m^{(i)}(\hat{u}_0^i | y_0^{n-1})$. После получения путей длины n из них выбирается тот путь, который имеет наибольшую вероятность $W_m^{(n-1)}(\hat{u}_0^{n-1} | y_0^{n-1})$, или удовлетворяет некоторым дополнительным условиям. Реализация этого алгоритма требует использования структур данных, которые позволяли бы переиспользовать промежуточные результаты вычислений для различных путей. Эти структуры описаны в [66]. Их использование позволяет реализовать декодирование со сложностью $O(Ln \log n)$. Уже при небольших значениях L данный подход позволяет реализовать декодирование почти по максимуму правдоподобия.

При недостаточном размере списка L декодер Тала-Варди может допускать ошибки, обусловленные удалением правильного пути на одной из фаз.

8.2.3. Последовательное декодирование

Существенным недостатком алгоритма Тала-Варди является то, что он строит L кодовых слов, из которых не более чем одно может являться правильным, т.е. большая часть вычислений выполняется впустую. Для преодоления этой проблемы может использоваться *последовательное декодирование* [83]. Ключевым элементом последовательного декодера является приоритетная очередь[3] (ПО), хранящая пары $(M, v_0^{\phi-1})$,

[3] В литературе ее иногда называют стеком.

где $M = M(v_0^{\phi-1}, y_0^{n-1})$ — вес пути $v_0^{\phi-1}$, и предоставляющая эффективные алгоритмы для следующих операций [41]:

- поместить пару в ПО;
- извлечь пару $(M, v_0^{\phi-1})$ (или только $v_0^{\phi-1}$) с наибольшим значением M;
- удалить заданную пару из ПО.

Будем считать, что ПО способна хранить не более Θ элементов. В контексте полярных кодов стековый алгоритм декодирования работает следующим образом:

1. Поместить в ПО путь нулевой длины с весом 0. Пусть $t_0^{n-1} = 0$. В дальнейшем вектор t_0^{n-1} используется для числа проходов декодера через различные фазы.
2. Извлечь из ПО путь $v_0^{\phi-1}$ с наибольшим весом. Пусть $t_\phi \leftarrow t_\phi + 1$.
3. Если $\phi = n$, возвратить кодовое слово $v_0^{n-1}A_m$ и завершить работу.
4. Если число допустимых потомков узла $v_0^{\phi-1}$ превышает количество свободных ячеек в ПО, удалить из нее элемент с наименьшим весом.
5. Вычислить веса $M(v_0^{\phi}, y_0^{n-1})$ допустимых потомков v_0^{ϕ} извлеченного узла и поместить их в ПО.
6. Если $t_\phi \geq L$, удалить из ПО все узлы $v_0^{j-1}, j \leq \phi$.
7. Перейти к шагу 2.

В дальнейшем под итерацией будет пониматься один проход этого алгоритма по шагам 2–7.

С точки зрения корректирующей способности параметр L имеет тот же смысл, что и размер списка в списочном алгоритме Тала-Варди, т.к. это значение является верхней границей для числа путей t_ϕ, рассматриваемых декодером на каждй фазе ϕ. Шаг 6 гарантирует, что алгоритм завершит свою работу за не более чем Ln итераций. Это является также верхней границей для числа элементов, которые могут быть помещены в ПО. Однако эксперименты показывают, что алгоритм может вполне успешно работать и с намного меньшим размером Θ ПО. Шаг 4 гарантирует, что ПО никогда не переполнится.

Существует множество различных подходов к определению весовой функции для последовательного декодирования. В общем случае она должна обеспечивать осмысленное сравнение путей $v_0^{\phi-1}$ различных длин ϕ. В [71] было предложено определить вес пути как

$$M(v_0^{\phi-1}) = R(v_0^{\phi-1}, y_0^{n-1}) - \mathbf{M}_{Y_0^{n-1}}\left[R(u_0^{\phi-1}, Y_0^{n-1}\right], \qquad (8.5)$$

где

$$R(v_0^{\phi-1}, y_0^{n-1}) = \max_{v_\phi^{n-1}} W_m^{(n-1)}(v_0^{n-1}|y_0^{n-1}),$$

матожидание в (8.5) берется по случайным величинам, соответствующим выходу канала, и u_0^{n-1} — вектор, использованный кодером.

В лучшем случае сложность такого алгоритма составляет $O(n \log n)$, в худшем случае — $O(Ln \log n)$. Было [71] показано, что такой подход позволяет значительно снизить среднюю сложность по сравнению со списочным декодером Тала-Варди при незначительном ухудшении корректирующей способности.

Недостатком этого метода является увеличение задержки декодирования, которая к тому же оказывается случайной величиной.

8.3. Усовершенствованные конструкции

Существенным недостатком полярных кодов является их низкое минимальное расстояние. В частности, для кодов с ядром Арикана оно составляет $d = O(\sqrt{n})$. В результате даже при декодировании по максимуму правдоподобия полярные коды уступают многим известным кодам МППЧ и турбо-кодам.

8.3.1. Полярные коды с CRC

Простым способом улучшить дистантные свойства полярных кодов является добавление к данным перед кодированием их полярным кодом некоторой контрольной суммы. В качестве таковой можно использовать проверочные символы циклического кода, обнаруживающего ошибки (cyclic redundancy check). Приемник, использующий списочный декодер Тала-Варди, должен исключить из полученного списка кодовых слов те слова, которые не удовлетворяют соответствующим проверочным соотношениям [66]. Получаемые таким образом коды имеют корректирующую способность, сопоставимую с известными LDPC кодами.

Идея этого подхода основана на том наблюдении, что полярный код содержит небольшое число ненулевых кодовых слов малого веса d. Большая их часть не удовлетворяет проверочным соотношениям CRC. При удачном выборе параметров все такие слова исключаются этими дополнительными проверочными соотношениями, что приводит к коду с большим минимальным расстоянием.

8.3.2. Полярные подкоды

Динамически замороженные символы

Конструкцию полярных кодов можно обобщить, положив замороженные символы равными не нулю, а каким-то линейным комбинациям предшествующих символов, т.е.

$$u_{j_i} = \sum_{s=0}^{j_i-1} u_s V_{i,s}, 0 \le i < n-k, j_i \in \mathcal{F}, \tag{8.6}$$

где V — $(n-k) \times n$ *матрица ограничений*, причем j_i — номер столбца, содержащего последний ненулевой элемент i-ой строки. Предполагается, что все j_i различны. Символы с ненулевой правой частью в (8.6) называются динамически замороженными. Полярные коды с динамически замороженными символами могут быть продекодированы с помощью алгоритма последовательного исключения и его аналогов, описанных выше. Единственное отличие состоит в процедуре обработки замороженных символов, т.е.

$$\widehat{u}_i = \begin{cases} \sum_{s=0}^{j_i-1} \hat{u}_s V_{t_i,s} & i \in \mathcal{F} \\ \arg\max_{u_i} W_m^{(i)}(\widehat{u}_0^{i-1}, u_i | y_0^{n-1}) & i \notin \mathcal{F}, \end{cases}$$

где t_i — номер строки, последний ненулевой элемент которой содержится в столбце $i \in \mathcal{F}$.

Рассмотрим $(n = l^m, k, d)$ код $\mathcal{C}$ с проверочной матрицей H. Пусть $A_m = K^{\otimes m}$ — матрица $n \times n$ поляризующего преобразования. Т.к. A_m обратима, любой вектор длины n может быть получен как результат $c_0^{n-1} = u_0^{n-1} A_m$ применения поляризующего преобразования к подходящему вектору u_0^{n-1}. Изучим ограничения, которые должны быть наложены на u_0^{n-1}, чтобы результат его применения принадлежал коду $\mathcal{C}$.

Эти ограничения задаются уравнением $u_0^{n-1} A_m H^T = 0$. Путем элементарных преобразований строк можно получить матрицу ограничений $V = QHA_m^T$, где Q — такая обратимая матрица, что последние ненулевые элементы всех строк V расположены в различных столбцах, т.е. значения $j_i = \max\{t | V_{i,t} \neq 0\}, 0 \le i < n-k$, различны. Это дает возможность декодировать произвольные линейные блоковые коды с использованием методов декодирования, предложенных изначально для полярных кодов [74]. Однако в общем случае корректирующая способность при этом оказывается неудовлетворительной, т.к. размороженными оказываются многочисленные символы, передаваемые по ненадежным подканалам. Можно показать, что расширенные примитивные коды БЧХ в узком смысле

в меньшей степени подвержены недостатку. На малых длинах они могут быть весьма эффективно продекодированы с помощью вышеописанного последовательного алгоритма.

Необходимо понимать, что перестановка столбцов проверочной матрицы H (т.е. переход к эквивалентному коду) может существенно изменить матрицу ограничений V, в т.ч. множество замораживания. В [74] описана структура перестановок столбцов проверочной матрицы кодов БЧХ, обеспечивающая структуру множества замораживания с достаточно малой вероятностью ошибки декодирования методом последовательного исключения. Более конкретно, в случае поляризующего преобразования на основе ядра Арикана проверочная матрица расширенного примитивного $(n = 2^m, k, d)$ кода БЧХ в узком смысле должна быть представлена в виде

$$H = \begin{pmatrix} x_0^0 & x_1^0 & \dots & x_{n-1}^0 \\ x_0^1 & x_1^1 & \dots & x_{n-1}^1 \\ \vdots & \vdots & \ddots & \vdots \\ x_0^{d-2} & x_1^{d-2} & \dots & x_{n-1}^{d-2} \end{pmatrix},$$

где $n = 2^m$, $x_i = \sum_{j=0}^{m-1} i_j \beta_j$, $(\beta_0, \dots, \beta_{m-1})$ — некоторый базис поля $GF(2^m)$ и $i = \sum_{j=0}^{m-1} i_j 2^j$ — разложение числа i по степеням 2. Для получения матрицы ограничений необходимо представить элементы этой матрицы в виде двоичных векторов-столбцов, домножить получившуюся матрицу на A_m^T и исключить линейно зависимые строки.

Пример 8.2. Рассмотрим расширенный примитивный $(16, 7, 6)$ код БЧХ в узком смысле. Его проверочная матрица имеет вид

$$H = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ 0 & 1 & 4 & 9 & 3 & 2 & 7 & 6 & 12 & 13 & 8 & 9 & 27 & 14 & 11 & 10 \\ 0 & 1 & 8 & 27 & 12 & 10 & 1 & 1 & 18 & 15 & 15 & 12 & 8 & 10 & 8 & 12 \\ 0 & 1 & 3 & 2 & 9 & 4 & 6 & 7 & 27 & 14 & 12 & 13 & 10 & 11 & 9 & 8 \end{pmatrix}$$

Здесь элементы поля $GF(2^4)$ представлены в виде целых чисел, которые получаются подстановкой числа 2 вместо примитивного элемента $\alpha : \alpha^4 + \alpha + 1 = 0$. В частности, число 3 следует интерпретировать как $\alpha + 1$. Представив эту матрицу в двоичном виде, домножив на A_4^T, где $A_4 = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}^{\otimes 4}$, и исключив линейно зависимые строки,

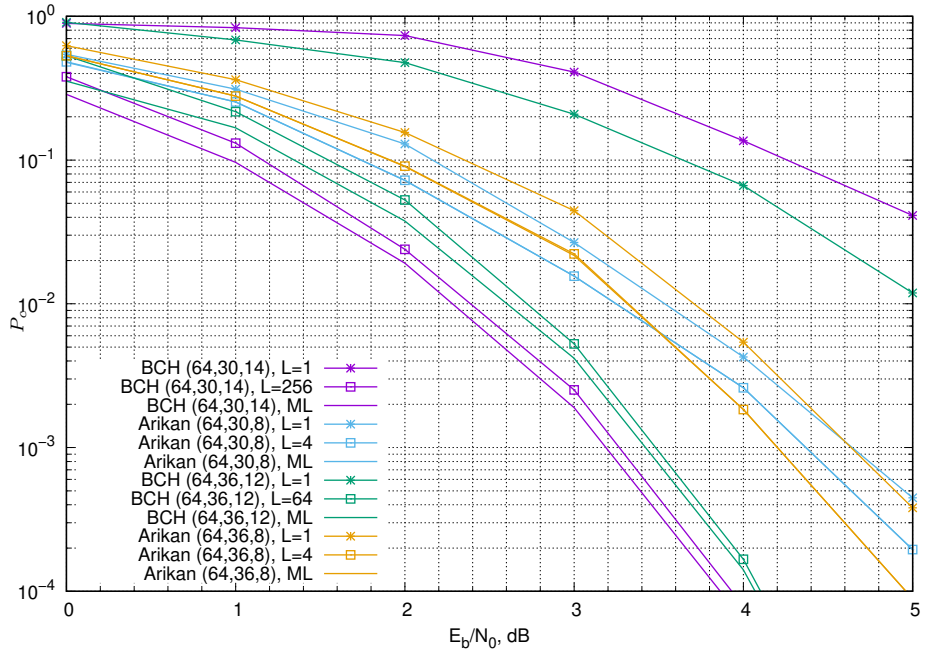


Рис. 8.3. Декодирование полярных кодов и кодов БЧХ списочным алгоритмом Тала-Варди

получим

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

Таким образом, замороженными оказались символы из множества $\mathcal{F} = \{12, 10, 9, 8, 6, 4, 2, 1, 0\}$.

На рис. 8.3 представлены результаты моделирования для полярных кодов и кодов БЧХ длины 64. Использовалось поляризующее преобразование с ядром Арикана Видно, что при размере списка 1 (т.е. в случае алгоритма последовательного исключения) полярные коды обеспечивают наименьшую вероятность ошибки. Декодирование полярных кодов по максимуму правдоподобия (ML) обеспечивается уже при $L = 4$. Но если дополнительно увеличить размер списка, декодер Тала-Варди может обеспечить декодирование кодов БЧХ почти по максимуму правдоподобия. При этом их корректирующая способность в силу большего мини-

мального расстояния оказывается намного лучше, чем у полярных кодов Арикана.

Полярные подкоды в узком смысле

Можно целенаправленно построить матрицу ограничений V, которая задавала бы код с достаточно большим минимальным расстоянием и при этом избегала бы размораживания ненадежных символов. *Полярный* (n, k, d) *подкод в узком смысле* задается матрицей ограничений $V = \begin{pmatrix} V' \\ V'' \end{pmatrix}$, где V' — матрица ограничений некоторого (n, K, d) кода C', называемого далее протокодом, V'' — матрица, содержащая $K - k$ различных строк веса 1, причем единицы в ее строках расположены в столбцах, соответствующих наименее надежным незамороженным символам[4] кода C'. В качестве протокодов C' в [74] предложено использовать расширенные примитивные коды БЧХ в узком смысле.

Необходимо понимать, что при использовании алгоритма последовательного исключения вероятность ошибки декодирования полярных подкодов не может быть меньше по сравнению с полярным кодов с такими же параметрами, построенным для того же канала. Однако при использовании списочного и последовательного алгоритмов декодирования полярные подкоды обеспечивают значительное улучшение корректирующей способности.

Пример 8.3. Рассмотрим построение $(16, 6, 6)$ полярного подкода кода БЧХ, представленного в примере 8.2. В случае двоичного стирающего канала с вероятностью стирания 0.5 параметры Бхаттачарьи (вероятности стирания) битовых подканалов поляризующего преобразования Арикана равны $Z_{16} = (\underline{0.999}, \underline{0.992}, \underline{0.985}, 0.77, \underline{0.96}, \underline{0.65}, 0.53, 0.1, \underline{0.9}, \underline{0.47}, \underline{0.35}, 3.7 \cdot 10^{-2}, \underline{0.23}, 1.5 \cdot 10^{-2}, 7.8 \cdot 10^{-3}, 1.5 \cdot 10^{-5})$. Здесь подчеркнуты значения, соответствующие замороженным символам. Видно, что символ u_3 не заморожен и соответствует наибольшему значению параметра Бхаттачарьи. Таким образом, для получения полярного подкода необходимо наложить дополнительное условие $u_3 = 0$.

Полярные подкоды в широком смысле

Полярный (n, k) *подкод в широком смысле* задается матрицей ограничений $V = \begin{pmatrix} \overline{V} \\ \tilde{V} \end{pmatrix}$, где $\overline{V}$ — матрица ограничений классического (n, K) полярного кода $\overline{C}$ с множеством замораживания $\overline{\mathcal{F}}$ и $\tilde{V}$ — $(K - k) \times n$ матрица ограничений, обеспечивающая исключение из $\overline{C}$ кодовых слов малого веса. Можно показать, что в случае ядра Арикана и некото-

[4]Под надежностью символа здесь понимается надежность соответствующего подканала.

рых других ядер [13] все ненулевые кодовые слова минимального веса $c_0^{n-1} = u_0^{n-1} A_m : \mathrm{wt}(c_0^{n-1}) = d$ имеют по крайней мере один символ $u_i = 1$, причем вес соответствующей строки матрицы A_m также равен d. Для исключения таких слов необходимо наложить линейные ограничения, которые включали бы *все* (или почти все) такие символы u_i. Эти ограничения динамического замораживания должны быть наложены на символы с *наименьшими возможными номерами*, чтобы списочный/последовательный декодер могли их учесть на ранних этапах декодирования, снижая таким образом вероятность потери правильного пути. В связи с этим в [70] было предложено налагать ограничения динамического замораживания на $t \leq K - k$ символов $u_i, i \notin \overline{\mathcal{F}}$, с *наибольшими* возможными значениями i, такими что вес соответствующих строк матрицы A_m равен d. При отсутствии достаточного числа таких строк могут быть взяты строки большего веса. Собственно коэффициенты в уравнениях (8.6) динамического замораживания могут быть выбраны как двоичные равновероятные (псевдо)случайные значения. Соответствующие замороженные символы называются динамически замороженными символами типа А.

Кроме того, динамически замороженными могут быть выбраны $s = K - k - t$ наименее надежных символов, не входящих в $\overline{\mathcal{F}}$ и не использованных в качестве динамически замороженных символов типа А. Соответствующие коэффициенты ограничений динамического замораживания также могут быть выбраны (псевдо)случайным образом. Таким символы называются динамически замороженными символами типа Б. Их использование позволяет уменьшить среднюю вероятность неправильных путей в списочном декодере Тала-Варди, что снижает вероятность потери декодером правильного пути. Данная конструкция положена в основу процедуры кодирования в стандарте мобильной связи 5G.

Пример 8.4. Рассмотрим построение $(16, 6)$ полярного подкода в широком смысле с ядром Арикана. Пусть $\overline{\mathcal{F}} = \{0, 1, 2, 4, 8\}$. Незамороженные символы соответствуют строкам матрицы A_4 веса не менее 4. Пусть $t = 4$. Тогда в качестве динамически замороженных символов типа А следует выбрать символы с номерами $12, 10, 9, 6$. Используя значения параметров Бхаттачарьи, приведенные в примере 8.3, получим, что в качестве символа типа Б следует выбрать u_3. В результате множество замораживания для полученного кода совпадает с таковым для полярного подкода в узком смысле, приведенного в примере 8.3. Однако коэффициенты в ограничениях динамического замораживания в данном случае могут быть выбраны псевдослучайным образом.

Хотя минимальное расстояние получаемых таким образом кодов может быть хуже, чем в случае полярных подкодов в узком смысле, на практике полярные подкоды в широком смысле демонстрируют лучшую корректирующую способность, по крайней мере в области малых отношений сигнал/шум. Более того, существует возможность целенаправленно подбирать коэффициенты в ограничениях динамического замораживания с целью явной минимизации числа кодовых слов малого веса в получаемом таким образом коде [72].

8.4. Полярные коды с ядром Арикана

Наибольшее изученным классом полярных кодов являются коды с ядром Арикана. Рассмотрим более подробно методы их декодирования и построения.

8.4.1. Декодирование

Точные выражения

В случае ядра Арикана выражение (8.3) может быть переписано как

$$W_\lambda^{(2i)}(u_0^{2i}|y_0^{2^\lambda-1}) = \sum_{u_{2i+1}=0}^{1} W_{\lambda-1}^{(i)}(u_{0,e}^{2i} + u_{0,o}^{2i+1}|y_{0,e}^{2^\lambda-1})W_{\lambda-1}^{(i)}(u_{0,o}^{2i+1}|y_{0,o}^{2^\lambda-1})$$

$$W_\lambda^{(2i+1)}(u_0^{2i+1}|y_0^{2^\lambda-1}) = W_{\lambda-1}^{(i)}(u_{0,e}^{2i} + u_{0,o}^{2i+1}|y_{0,e}^{2^\lambda-1})W_{\lambda-1}^{(i)}(u_{0,o}^{2i+1}|y_{0,o}^{2^\lambda-1}),$$

где $0 < \lambda \leq m$, $a_{0,e}^j$ и $a_{0,o}^j$ обозначают, соответственно, подвектора a_0^j с четными и нечетными индексами. Удобно ввести логарифмические отношения правдоподобия $L_\lambda^{(i)}(y_0^{2^\lambda-1}, u_0^{i-1}) = \ln \frac{W_\lambda^{(i)}(u_0^{i-1},0|y_0^{2^\lambda-1})}{W_\lambda^{(i)}(u_0^{i-1},1|y_0^{2^\lambda-1})}$. Видно, что

$$\begin{aligned} L_\lambda^{(2i+1)}(y_0^{2^\lambda-1}, u_0^{2i}) &= \\ &= \log \frac{W_{\lambda-1}^{(i)}(u_{0,e}^{2i-1} + u_{0,o}^{2i-1}, u_{2i} + 0|y_{0,e}^{2^\lambda-1})W_{\lambda-1}^{(i)}(u_{0,o}^{2i-1}, 0|y_{0,o}^{2^\lambda-1})}{W_{\lambda-1}^{(i)}(u_{0,e}^{2i-1} + u_{0,o}^{2i-1}, u_{2i} + 1|y_{0,e}^{2^\lambda-1})W_{\lambda-1}^{(i)}(u_{0,o}^{2i-1}, 1|y_{0,o}^{2^\lambda-1})} = \\ &= (-1)^{u_{2i}} L_{\lambda-1}(y_{0,e}^{2^\lambda-1}, u_{0,e}^{2i-1} + u_{0,o}^{2i-1}) + L_{\lambda-1}(y_{0,o}^{2^\lambda-1}, u_{0,o}^{2i-1}). \quad (8.7) \end{aligned}$$

Преобразования, аналогичные тем, которые были использованы в п. 7.2.2, приводят к

$$L_{\lambda}^{(2i)}(y_0^{2^\lambda-1}, u_0^{2i-1}) = 2\tanh^{-1}\left(\tanh\left(\frac{1}{2}L_{\lambda-1}^{(i)}(y_{0,e}^{2^\lambda-1}, u_{0,e}^{2i-1} + u_{0,o}^{2i-1})\right)\cdot \right.$$
$$\left.\tanh\left(\frac{1}{2}L_{\lambda-1}^{(i)}(y_{0,e}^{2^\lambda-1}, u_{0,e}^{2i-1} + u_{0,o}^{2i-1}\right)\right) \tag{8.8}$$

Декодер минимум-сумма

Описанные выше вычисления с использованием вероятностей или нелинейных функций подвержены численным проблемам и сложны в реализации. В связи с этим получил распространение следующий приближенный подход, использование которого приводит к незначительному ухудшению корректирующей способности.

Пусть

$$R_{\lambda}^{(i-1)}(v_0^{i-1}, y_0^{2^\lambda} - 1) = \max_{v_i^{2^\lambda-1}} W_\lambda(v_0^{n-1}|y_0^{2^\lambda-1}).$$

Определим модифицированные логарифмические отношения правдоподобия

$$S_{\lambda}^{(i)}(v_0^{i-1}|y_0^{2^\lambda-1}) = R_{\lambda}^{(i)}(v_0^{i-1}, 0, y_0^{2^\lambda-1}) - R_{\lambda}^{(i)}(v_0^{i-1}, 1, y_0^{2^\lambda-1}). \tag{8.9}$$

Можно заметить, что

$$R_{\lambda}^{(i)}(v_0^{i}, y_0^{2^\lambda-1}) = R_{\lambda}^{(\phi-1)}(v_0^{i-1}, y_0^{2^\lambda-1}) + \tau(S_{\lambda}^{(i)}(v_0^{i-1}|y_0^{2^\lambda-1}), v_i), \tag{8.10}$$

где функция штрафа определена как

$$\tau(S, v) = \begin{cases} 0, & \text{если } \operatorname{sgn}(S) = (-1)^v \\ -|S|, & \text{иначе.} \end{cases}$$

Видно, что

$$R_{\lambda}^{(2i)}(v_0^{2i}, y_0^{N-1}) = \max_{v_{2i+1}}\left(R_{\lambda-1}^{(i)}\left(v_{0,e}^{2i+1} \oplus v_{0,o}^{2i+1}, y_0^{\frac{N}{2}-1}\right) + R_{\lambda-1}^{(i)}\left(v_{0,o}^{2i+1}, y_{\frac{N}{2}}^{N-1}\right)\right),$$
$$R_{\lambda}^{(2i+1)}(v_0^{2i+1}|y_0^{N-1}) = R_{\lambda-1}^{(i)}\left(v_{0,e}^{2i+1} \oplus v_{0,o}^{2i+1}, y_0^{\frac{N}{2}-1}\right) + R_{\lambda-1}^{(i)}\left(v_{0,o}^{2i+1}, y_{\frac{N}{2}}^{N-1}\right),$$

где $N = 2^\lambda$, $0 < \lambda \leq m$, и начальные значения для рекурсии задаются $R_0^{(0)}(b, y_j) = \log W_0^{(0)}\{b|y_j\}$, $b \in \{0, 1\}$. Из (8.9) получим

$$\begin{aligned} S_\lambda^{(2i)}(v_0^{2i-1}|y_0^{2^\lambda-1}) =& \max(J(0)+K(0), J(1)+K(1)) - \\ & \max(J(1)+K(0), J(0)+K(1)) \\ =& \max(J(0)-J(1)+K(0)-K(1), 0) - \\ & \max(K(0)-K(1), J(0)-J(1)) \\ S_\lambda^{(2i+1)}(v_0^{2i}|y_0^{2^\lambda-1}) =& J(v_{2i}) + K(0) - J(v_{2i}+1) - K(1) \end{aligned}$$

где $J(c) = R_{\lambda-1}^{(i)}((v_{0,e}^{2i-1} \oplus v_{0,o}^{2i-1}).c|y_0^{2^{\lambda-1}-1})$, $K(c) = R_{\lambda-1}^{(i)}(v_{0,o}^{2i-1}.c|y_{2^{\lambda-1}}^{2^\lambda-1})$. Заметим, что

$$J(0) - J(1) = a = S_{\lambda-1}^{(i)}(v_{0,e}^{2i-1} \oplus v_{0,o}^{2i-1}|y_0^{2^{\lambda-1}-1})$$

и

$$K(0) - K(1) = b = S_{\lambda-1}^{(i)}(v_{0,o}^{2i-1}|y_{2^{\lambda-1}}^{2^\lambda-1})$$

Рассматривая всевозможные комбинации знаков величин a, b, получим

$$S_\lambda^{(2i)}(v_0^{2i-1}|y_0^{2^\lambda-1}) = Q(a,b) = \operatorname{sgn}(a)\operatorname{sgn}(b)\min(|a|,|b|), \tag{8.11}$$

$$S_\lambda^{(2i+1)}(v_0^{2i}|y_0^{2^\lambda-1}) = P(v_{2i}, a, b) = (-1)^{v_{2i}}a + b. \tag{8.12}$$

Начальные значения для этой рекурсии задаются $S_0^{(0)}(y_i) = \log\frac{W\{0|y_i\}}{W\{1|y_i\}}$. Величины $R(v_0^i, y_0^{n-1}) = R_m^{(i)}(v_0^i, y_0^{2^n-1})$ могут использоваться для выбора наилучших путей в описанном выше списочном алгоритме Тала-Варди. Эти же величины используются и в последовательном алгоритме декодирования.

Выражение (8.11) можно интерпретировать как нахождение наименее надежного из символов, комбинируемых отдельно взятым ядром Арикана. Выражение (8.12) схоже с правилом разнесенного приема, т.е. комбинированием нескольких независимых оценок одного и того же символа.

8.4.2. Оценка надежности подканалов

Напомним, что классическая конструкция полярных кодов предполагает построение множества замораживания $\mathcal{F}$, включающего в себя номера ненадежных подканалов поляризующего преобразования. Надежность подканалов может быть охарактеризована их параметрами Бхаттачарьи, вероятностями ошибки на бит или пропускными способностями.

Ниже представлены некоторые методы расчета этих величин. Отметим, что при использовании всех описанных ниже методов сложность расчета надежности подканалов составляет $O(n)$ операций.

Параметры Бхаттачарьи

Ариканом доказано, что параметры Бхаттачарьи $Z_{m,i} = Z(W_m^{(i)})$ битовых подканалов удовлетворяют

$$Z_{m,2i} \leq 2Z_{m-1,i} - Z_{m-1,i}^2 \tag{8.13}$$

$$Z_{m,2i+1} = Z_{m-1,i}^2, \tag{8.14}$$

причем равенство в (8.13) имеет место только в случае двоичного стирающего канала. Эти выражения могут быть использованы для построения полярных кодов, оптимальных для двоичного стирающего канала. Однако для иных каналов эти выражения крайне неточны.

Оценки для ухудшенных и улучшенных каналов

Главной сложностью на пути точного расчета параметров битовых подканлов является то, что мощность их выходных алфавитов $\mathbb{Y}^{2^\lambda} \times \{0,1\}^i, 0 \leq i < 2^\lambda$, растет дважды экспоненциально с числом слоев λ поляризующего преобразования. В связи с этим в [65] было предложено для каждого такого подканала рассмотреть его "ухудшенную" и "улучшенную" версии с выходным алфавитом фиксированной мощности μ. Соответствующие функции переходных вероятностей могут быть построены со сложностью $O(\mu^2 \log \mu)$, что позволяет получить верхние и нижние оценки для пропускной способности, параметра Бхаттачарьи и вероятности ошибки битовых подканалов.

Гауссовская аппроксимация

Полярные коды являются линейными. При использовании симметричного по входу канала вероятность ошибки декодирования линейного кода не зависит от передаваемых кодовых слов. В связи с этим для целей анализа можно ограничиться случаем нулевого кодового слова. Можно предположить, что логарифмические отношения правдоподобия, задаваемые (8.7)–(8.8), распределены по нормальному закону, причем их дисперсия равна удвоенному матожиданию. Это предположение справедливо для $L_0^{(0)}(y)$ в случае канала с аддитивным белым гауссовским шумом. В общем случае распределение $L_\lambda^{(i)}(\mathbf{0}, y_0^{2^\lambda - 1})$ весьма далеко от нормального. Но несмотря на это, такая аппроксимация дает удивительно точные результаты.

С учетом указанных допущений, получим, что матожидания $\mathcal{L}_\lambda^{(i)} = \mathbf{M}_{Y_0^{2^\lambda-1}}\left[L_\lambda^{(i)}(\mathbf{0}, Y_0^{2^\lambda-1})\right], 0 \le i < 2^{\lambda-1}$ логарифмических отношений правдоподобия равны

$$\mathcal{L}_\lambda^{(2i)} = \Xi\left(\mathcal{L}_{\lambda-1}^{(i)}\right) = \phi^{-1}\left(1 - \left(1 - \phi\left(\mathcal{L}_{\lambda-1}^{(i)}\right)\right)^2\right) \tag{8.15}$$

$$\mathcal{L}_\lambda^{(2i+1)} = 2\mathcal{L}_{\lambda-1}^{(i)}, \tag{8.16}$$

где $\mathbf{M}\left[L_0^{(0)}\right] = \mathcal{L}_0^{(0)} = 2/\sigma^2$ и

$$\phi(x) = \begin{cases} 1 - \frac{1}{\sqrt{4\pi x}} \int_{-\infty}^{\infty} \tanh\frac{u}{2} e^{-\frac{(u-x)^2}{4x}} du, & x > 0 \\ 1, & x = 0. \end{cases}$$

Можно заметить, что функция $\Xi(x)$ является достаточно гладкой. С помощью метода наименьших квадратов была найдена аппроксимация

$$\Xi(x) \approx \begin{cases} 0.98611x - 2.31515 & x > 12 \\ x(9.0047 \cdot 10^{-3}x + 0.76943) - 0.95068, & 3.5 < x \le 12 \\ x(0.062883x + 0.36784) - 0.16267 & 1 < x \le 3.5 \\ x(0.22024x + 0.06448) & \text{иначе.} \end{cases} \tag{8.17}$$

Вероятность ошибки в каждом подканале может быть вычислена как

$$P_i \approx Q\left(\sqrt{\mathcal{L}_m^{(i)}/2}\right), 0 \le i \le 2^m - 1. \tag{8.18}$$

Распределение ЛОПП для декодера минимум-сумма

Можно показать, что при передаче нулевого кодового слова по каналу без памяти модифицированные логарифмические отношения правдоподобия $S_\lambda^{(i)}(0, Y_0^{2^\lambda-1})$, задаваемые (8.11)–(8.12), имеют функции распределения $F_\lambda^{(i)}(x)$, задаваемые

$$F_\lambda^{(2i)}(x) = \begin{cases} 2F_{\lambda-1}^{(i)}(x)(1 - F_{\lambda-1}^{(i)}(-x)), & x < 0 \\ 2F_{\lambda-1}^{(i)}(x) - (F_{\lambda-1}^{(i)}(-x))^2 - (F_{\lambda-1}^{(i)}(x))^2, & x \ge 0 \end{cases} \tag{8.19}$$

$$F_\lambda^{(2i+1)}(x) = \int_{-\infty}^{\infty} F_{\lambda-1}^{(i)}(x-y) dF_{\lambda-1}^{(i)}(y), \tag{8.20}$$

где $F_0^{(0)}(x)$ — функция распределения ЛОПП на выходе канала [44]. Использование этих формул требует дискретизации множества значений ЛОПП, причем требуемое число уровней дискретизации растет с длиной кода.

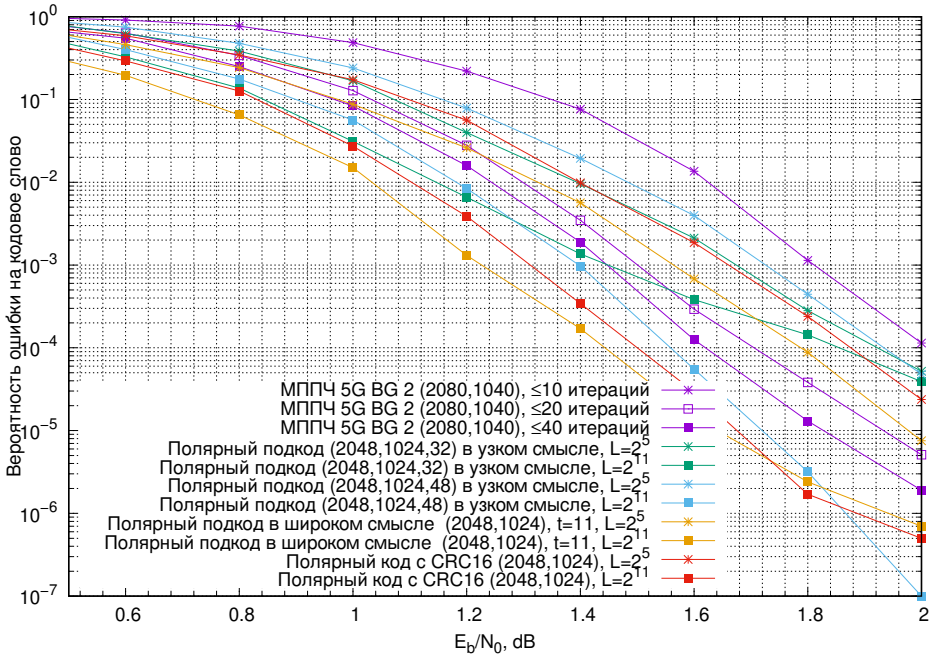
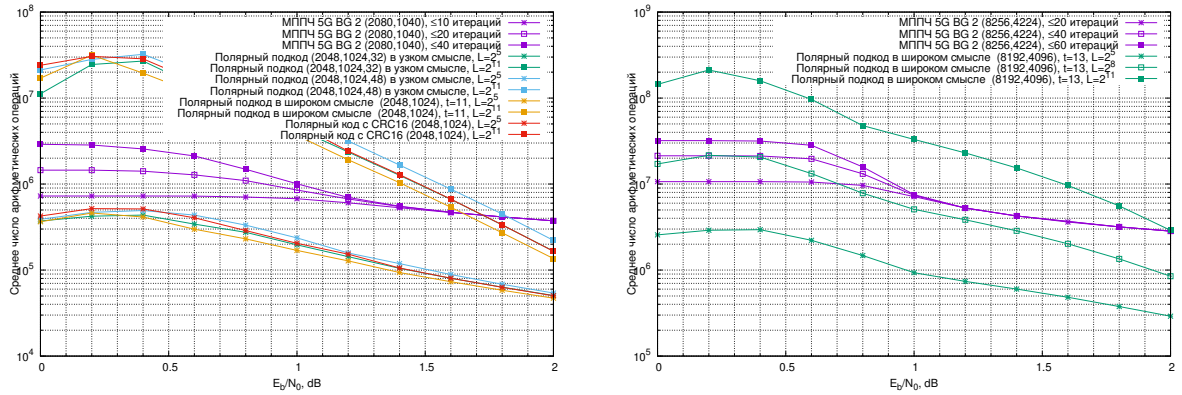


Рис. 8.4. Сравнение полярных и МППЧ кодов

8.4.3. Сравнение кодов

На рис. 8.4 представлены результаты моделирования для полярных подкодов и кодов с малой плотностью проверок на четность (МППЧ, LDPC) из стандарта мобильной связи 5G со скоростью 1/2 и длинами около 2048. Видно, что на малых отношениях сигнал/шум полярный подкод в широком смысле обеспечивает лучшую корректирующую способность по сравнению с полярным кодом с CRC и полярным подкодом в узком смысле. Последний при $d = 32, L = 2048$ демонстрирует эффект “полки”, состоящий в значительном замедлении скорости убывания вероятности ошибки. Однако при $L = 2048$ “полка” проявляется уже у полярного подкода в широком смысле ввиду его недостаточного минимального расстояния. При $L = 2048, E_b/N_0 > 1.8$ дБ наилучшую корректирующую способность обеспечивает полярный подкод в узком смысле с $d = 48$. При $L = 32$ полярный подкод в широком смысле показывает примерно такую же вероятность ошибки, как и код МППЧ, декодер которого выполняет не более 20 итераций. Дальнейшее увеличение числа итераций (т.е. сложности) в декодере кода МППЧ приводит к крайне незначительному улучшению корректирующей способности. При этом увеличение размера списка L в случае полярного подкода приводит к значительному снижению вероятности ошибки декодирования. Результаты для некоторых других длин представлены на рис. 10.8.



(a) (b)

Рис. 8.5. Сложность декодирования полярных и МППЧ кодов:
a — $n \approx 2048$; b — $n \approx 8192$

Видно, что на малых длинах полярные подкоды демонстрируют значительно лучшую корректирующую способность по сравнению с кодами МППЧ, однако на больших длинах на больших отношениях сигнал/шум коды МППЧ имеют значительное преимущество. При достаточно большом размере списка списочный и последовательный декодеры полярных кодов обеспечивают декодирование почти по максимуму правдоподобия. При недостаточном минимальном расстоянии возможно возникновение эффекта “полки”, который может быть устранен увеличением конструктивного расстояния полярного подкода в узком смысле или параметра t полярного подкода в широком смысле. Это, однако, может потребовать увеличения L на стороне декодера, что влечет за собой увеличение сложности и объема памяти декодера.

На рис. 8.5, a представлена средняя сложность последовательного декодирования полярных подкодов [71], а также средняя сложность алгоритма распространения доверия декодирования кода МППЧ. Эти результаты были получены для тех же кодов, что и на рис. 8.4. Видно, что на больших отношениях сигнал/шум полярные подкоды имеют меньшую сложность декодирования и лучшую корректирующую способность по сравнению с кодом МППЧ. Как видно из рис. 8.5, b и 10.8, b, при увеличении длины кода при сопоставимой корректирующей способности декодирование кодов МППЧ оказывается заметно проще по сравнению с полярными подкодами.

Необходимо также учитывать, что задержка последовательного декодирования значительно превышает таковую для случая алгоритма рас-

пространения доверия. Несколько меньшую задержку имеет списочный алгоритм Тала-Варди, однако его сложность равна $O(Ln\log_2 n)$. В настоящее время практически приемлемыми считаются значения $L \leq 32$.

Упражнения

1. Постройте полярный код $(32, 16)$ для двоичного стирающего канала с вероятностью стирания 0.5.
2. Продекодируйте методом последовательного исключения в построенном выше коде произвольный двоичный вектор веса 5, считая, что он получен на выходе двоичного симметричного канала.
3. Докажите формулы (8.19)–(8.20).

Часть III

Коды на графах

9. Сверточные коды

9.1. Конструкция кодов

9.1.1. Основные понятия

Сверточные коды нашли широкое применение в телекоммуникационных системах благодаря простоте своей реализации. Однако в настоящее время они постепенно вытесняются другими методами помехоустойчивого кодирования, обладающими большей корректирующей способностью.

Древовидным (n_0, k_0)*-кодом* называется отображение на себя множества полубесконечных последовательностей элементов из $\mathbb{F}_q$ такое, что если для любого M первые Mk_0 элементов двух полубесконечных последовательностей совпадают, то первые Mn_0 элементов отображений этих последовательностей также совпадают [1, 10]. Кодер древовидного кода может хранить в своей оперативной памяти W последних информационных символов. Эта величина, т.е. объем памяти кодера, называется *длиной кодового ограничения*. Частные случаи древовидных кодов характеризуются наличием различных комбинаций следующих свойств:

1. Конечность длины кодового ограничения.
2. Постоянство во времени. Если две различные входные последовательности совпадают во всем, за исключением некоторого временного сдвига на целое число кадров, то соответствующие им выходные последовательности также совпадают во всем, за исключением временного сдвига на то же число кадров.
3. Линейность. Если $C(d_1)$ и $C(d_2)$ — кодовые последовательности, соответствующие информационным последовательностям d_1 и d_2, то $\forall \alpha, \beta \in \mathbb{F}_q : C(\alpha d_1 + \beta d_2) = \alpha C(d_1) + \beta C(d_2)$.
4. Систематичность. Систематическим древовидным кодом называется код, в котором каждый кадр информационной последовательности составляет первые k_0 символов первого из тех кадров кодовой последовательности, на которые влияет данный информационный кадр.

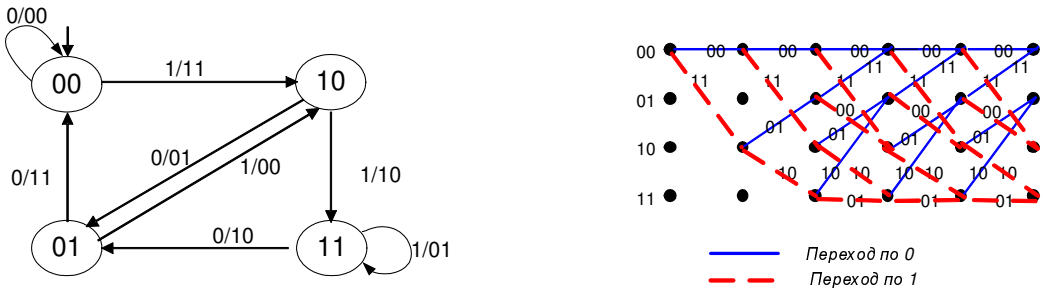
Определение 9.1. Линейный постоянный во времени древовидный (n_0, k_0)-код с конечной длиной кодового ограничения называется *сверточным кодом*.

(a) (b)

Рис. 9.1. Кодеры сверточного кода:
a — несистематический; b — рекурсивный систематический

Сверточный код может быть порожден с помощью линейных регистров сдвига (фильтров с конечным импульсным откликом), как это показано на рис. 9.1, a.

Кодер сверточного кода является конечным автоматом и может быть описан с помощью *графа переходов*, пример которого приведен на рис. 9.2, a. Заметим, что состояния помечены векторами символов, хранящихся в ячейке регистра сдвига, т.е. поданных на вход сверточного кодера за последние m тактов. Однако при реализации декодера намного удобнее использовать решетчатую диаграмму, которая представляет собой временную развертку диаграммы переходов. Пример ее приведен на рис. 9.2, b. Ребра, соединенные с узлом решетки и находящиеся слева от него, называются входящими, а находящиеся справа — исходящими. Для каждого ребра решетки указан кадр, появляющийся на выходе кодера при срабатывании соответствующего перехода. Узлы решетки будем нумеровать парами (s, i), где s — состояние кодера и i — номер кадра. При рассмотрении сверточных кодов конечной длины решетка должна быть усечена по временной оси. При этом, если кодер после обработки информационной последовательности переводится в нулевое состояние, из решетки должны быть удалены все ребра, ведущие в недостижимые состояния (сравните рис. 9.4 и 9.2, b). Таким образом, всякое кодовое слово сверточного кода представляет собой путь в решетке, начинающийся в нулевом состоянии в нулевой момент времени и оканчивающийся также в нулевом состоянии в момент времени, соответствующий окончанию обработки кодируемой последовательности.



(a) (b)

Рис. 9.2. Описание сверточного кода с $G(D) = (D^2 + D + 1, D^2 + 1)$: a — граф переходов; b — решетка

9.1.2. Порождающая матрица

В общем случае операцию кодирования, реализуемую регистром сдвига, можно описать как $c(D) = u(D)G(D)$, где $u(D)$ — вектор размерности k_0, элементами которого являются формальные степенные ряды (при практической реализации — многочлены), коэффициентами которых являются кодируемые символы из $\mathbb{F}_q$, $G(D)$ — $k_0 \times n_0$ порождающая матрица, содержащая передаточные функции (многочлены) фильтров. При этом длина кодового ограничения определяется как[1]

$$W = \sum_{j=1}^{k_0} \max_{1 \le i \le n_0} \deg G_{ji}(D).$$

В реальных системах связи данные передаются в виде блоков конечной длины, т.е. сверточный код рассматривается как линейный блоковый. Для упрощения декодирования и снижения вероятности ошибки целесообразно обеспечить приведение кодера в нулевое состояние после окончания передачи блока данных[2]. Для этого в несистематический кодер необходимо подать $k_0 \max \deg_{G_{ij}(D)}$ нулевых битов, а в рекурсивный систематический кодер — столько же битов, равных выходу цепи обратной связи кодера. В этом случае выход фильтра с конечным импульсным откликом может быть представлен как линейная свертка входной после-

[1]Некоторыми авторами она определяется как максимальное число символов кодовой последовательности, которые могут измениться при изменении одного символа информационной последовательности, что эквивалентно $W = 1 + \max_{i,j} \deg G_{ji}(D)$.

[2]Иногда это сделать невозможно в силу каких-либо внешних ограничений. Все описываемые далее алгоритмы применимы и в этом случае после тривиальных модификаций.

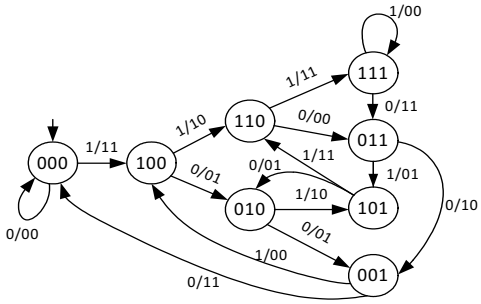


Рис. 9.3. Диаграмма переходов катастрофического кодера

довательности и импульсного отклика (передаточной функции) фильтра, что и объясняет название кодов.

Также как и в случае линейных блоковых кодов, существует множество различных порождающих матриц одного и того же сверточного кода. Они связаны друг с другом соотношением $G'(D) = T(D)G(D)$, где $T(D)$ — обратимая над полем рациональных функций $\mathbb{F}_q[[D]]$ матрица. В частности, порождающая матрица может быть приведена к виду $G'(D) = (I|A(D))$, где I — единичная матрица и $A(D)$ — матрица рациональных функций. Такое представление позволяет построить систематический кодер сверточного кода. Его реализация требует использования фильтров с бесконечным импульсным откликом, как показано на рис. 9.1, b ($G'(D) = (1, \frac{D^2+1}{D^2+D+1})$). Иногда такие коды называют *рекурсивными систематическими сверточными кодами*. Такое название не вполне корректно, так как точно такое же множество кодовых последовательностей может быть получено и с помощью несистематического кодера.

Пример 9.1. Рассмотрим сверточный код с порождающей матрицей $G(D) = (D^2 + D + 1, D^2 + 1)$. Домножим ее слева на обратимую матрицу $T(D) = (D + 1)$. Это приведет к $G'(D) = (D^3 + 1, D^3 + D^2 + D + 1)$. Видно, что длина кодового ограничения увеличилась до 3. Однако если на вход такого кодера подать полубесконечную последовательность, задаваемую рядом $u(D) = \sum_{i=0}^{\infty} D^i = \frac{1}{1+D}$, на выходе кодера появится лишь последовательность конечного веса, которую можно описать как (D^2+D+1, D^2+1). Отметим, что эта же последовательность появилась бы на выходе кодера исходного кода при подаче на вход последовательности $u'(D) = 1 + 0D + \ldots$. На рис. 9.3 представлена диаграмма переходов модифицированного кодера. Видно, что около состояния “111” присутствует цикл (петля), характеризующийся нулевым весом выходной последовательности при ненулевом весе входной последовательности.

Поведение кодера, при котором некоторые входные последовательности бесконечного веса отображаются на кодовые последовательности конечного веса, является крайне нежелательным с практической точки

зрения, так как при передаче по зашумленному каналу единственная ошибка декодера в оценке последовательности переходов может привести к бесконечному числу ошибок в оценке информационной последовательности. Поэтому такие коды (правильнее сказать, кодеры или порождающие матрицы) носят название *катастрофических*. Они в некотором смысле противоположны рекурсивным систематическим кодерам, которые отображают информационные последовательности конечного веса на кодовые последовательности бесконечного веса. Такое поведение, напротив, оказывается полезным с практической точки зрения (см. главу 13).

Для того, чтобы понять природу катастрофических кодов, введем понятие *правой обратной матрицы* $G^{-1}(D) : G(D)G^{-1}(D) = I$. Правая обратная матрица существует, если исходная матрица $G(D)$ является матрицей полного ранга. Отметим, что правая обратная матрица совпадает с обратной, если $G(D)$ является квадратной обратимой матрицей. В общем случае $G^{-1}(D)$ является прямоугольной матрицей.

Если $c(D)$ соответствует кодовой последовательности сверточного кода с порождающей матрицей $G(D)$, то в отсутствии ошибок информационная последовательность может быть восстановлена как $u(D) = c(D)G^{-1}(D) = u(D)G(D)G^{-1}(D)$. Отсюда видно, что $c(D)$ может иметь конечный вес, а $u(D)$ — бесконечный, только если $G^{-1}(D)$ содержит рациональные элементы. Таким образом, доказана

Теорема 9.1. *Порождающая матрица $G(D)$ не является катастрофической тогда и только тогда, когда ее правая обратная $G^{-1}(D)$ содержит только многочленные элементы.*

Для $(2, 1)$-сверточных кодов это условие сводится к требованию взаимной простоты порождающих многочленов. Как показывает пример 9.1, неудачное преобразование $\tilde{G}(D) = T(D)G(D)$ некатастрофического кодера может привести к катастрофическому кодеру. Очевидно, что $(T(D)G(D))^{-1} = G^{-1}(D)T^{-1}(D)$. Если $G(D)$ соответствует некатастрофическому кодеру и $T(D)$ — обратимая квадратная матрица, то для того, чтобы исключить появление рациональных элементов в $\tilde{G}^{-1}(D)$, достаточно исключить их наличие в $T^{-1}(D)$. В соответствии с правилом Крамера, это достигается, если $\det T(D) \in \mathbb{F}_q \setminus \{0\}$. Поэтому сверточные кодеры, соответствующие порождающим матрицам $\tilde{G}(D)$ и $G(D)$, называются эквивалентными, если $\tilde{G}(D) = T(D)G(D)$ и $\det T(D) = 1$.

9.1.3. Весовые свойства сверточных кодов

Для оценки корректирующей способности сверточных кодов необходимо оценить число кодовых последовательностей различного веса. Будем считать, что всякая кодовая последовательность соответствует последовательности переходов кодера, начинающейся и заканчивающейся в нулевом состоянии.

Каждый переход между состояниями кодера сопровождается порождением n_0 кодовых символов. Всякое состояние i кодера можно охарактеризовать многочленом $X^{(i)} = X^{(i)}(D) = \sum_j x_{ij} D^j$, где x_{ij} равно числу путей с выходной последовательностью веса j, ведущих в это состояние из начального состояния. Так как нас интересуют пути, начинающиеся и заканчивающиеся в нулевом состоянии кодера, необходимо расщепить это нулевое состояние на два подсостояния, одно из которых (с номером 0) будет начальным (т.е. возможны переходы из него, но не в него), а другое (с номером q^W, где W — длина кодового ограничения) — конечным (т.е. возможны переходы только в него). Петля, соответствующая переходу из начального состояния в него же, должна быть отброшена. Это позволяет исключить из рассмотрения как последовательности, начинающие отличаться от нулевой не в начальном кадре, так и последовательности, образованные конкатенацией нескольких кодовых последовательностей. Имеет место равенство

$$X^{(i)} = \sum_{k \in S_i} X^{(k)} D^{w_{ki}}, i = 1..q^W, \tag{9.1}$$

где S_i — множество состояний, из которых возможен переход в состояние i, и w_{ki} — вес последовательности, порождаемой при переходе из состояния k в состояние i. Система уравнений (9.1) может быть разрешена в поле рациональных функций с помощью стандартных методов линейной алгебры. При этом удобно выразить ее решение через переменную $X^{(0)}$. Передаточная функция сверточного кода определяется как $T(D) = \frac{X^{(q^W)}}{X^{(0)}}$. Коэффициенты ее разложения в ряд Тейлора в окрестности нуля задают количество кодовых последовательностей различного веса. Степень младшего члена этого ряда называется *свободным расстоянием* $d_{\text{св}}$ сверточного кода.

Пример 9.2. Рассмотрим код с диаграммой переходов, представленной на рис. 9.2, а. Имеет место система уравнений

$$\begin{aligned} X^{(10)} &= D^2X^{(00)} + D^0X^{(01)} \\ X^{(11)} &= DX^{(10)} + DX^{(11)} \\ X^{(01)} &= DX^{(10)} + DX^{(11)} \\ X^{(100)} &= D^2X^{(01)}. \end{aligned}$$

Для удобства номера состояний здесь приведены в двоичной системе счисления. Состояние 00, обозначенное как начальное на исходной диаграмме переходов, было расщеплено на два состояния, одно из которых (100) является конечным (поглощающим). Заметим, что переход из состояния 00 в него же, имевший место на исходной диаграмме, здесь не учтен. Решая эту систему методом Гаусса, получим $X^{(100)} = \frac{D^5}{1-2D}X^{(00)}$, т.е. $T(D) = \frac{D^5}{1-2D} = D^5 + 2D^6 + 4D^7 + 8D^8 + \ldots$. Таким образом, существует единственная последовательность переходов, дающая кодовую последовательность веса 5, две последовательности, дающие кодовые последовательности веса 6 и т.д. Визуальный анализ решетки на рис. 9.2, b подтверждает этот вывод.

Однако для анализа вероятности ошибки декодирования требуется более детальная информация о коде. Помимо сведений о количестве кодовых последовательностей различного веса, необходимо также знать веса соответствующих им информационных последовательностей. Для этого перепишем (9.1) как

$$X^{(i)} = \sum_{k \in S_i} X^{(k)} D^{w_{ki}} N^{v_{ki}}, i = 1..q^W, \tag{9.2}$$

где v_{ki} — вес информационной последовательности, вызывающей переход из состояния k в состояние i. Проделав далее те же действия, получим ряд $T(N, D)$, коэффициенты которого задают количество кодовых последовательностей различного веса, порождаемых информационными последовательностями различного веса. Заметим, что исходная передаточная функция получается как $T(D) = T(1, D)$.

Пример 9.3. Продолжим рассмотрение предыдущего примера. Модифицированная система уравнений имеет вид

$$\begin{aligned} X^{(10)} &= ND^2X^{(00)} + ND^0X^{(01)} \\ X^{(11)} &= NDX^{(10)} + NDX^{(11)} \\ X^{(01)} &= DX^{(10)} + DX^{(11)} \\ X^{(100)} &= D^2X^{(01)}. \end{aligned}$$

Передаточная функция имеет вид $T(N, D) = \frac{D^5N}{1-2DN} = ND^5 + 2N^2D^6 + 4N^3D^7 + 8N^4D^8 + \ldots$. Визуальный анализ решетки показывает, что кодовая последовательность веса 5 действительно получается при подаче на вход кодера единственного ненулевого бита.

Таблица 9.1

Порождающие многочлены сверточных кодов со скоростью 1/2

Кодовое ограничение W	$g_1(D)$	$g_2(D)$	$d_{\text{св}}$
2	5	7	5
3	15	17	6
4	23	35	7
5	53	75	8
6	133	171	10
7	247	371	10
8	561	753	12
9	1167	1545	12
10	2335	3661	14
11	4335	5723	15
12	10533	17661	16
13	21675	27123	16

В противоположность кодам БЧХ, для которых существуют простые методы построения, обеспечивающие гарантированное минимальное расстояние, практически все известные хорошие сверточные коды были найдены переборными методами на ЭВМ. В табл. 9.1 представлены порождающие многочлены некоторых сверточных кодов со скоростью 1/2 [10]. Многочлены представлены в виде восьмеричной битовой маски, т.е., например, 15 соответствует $1 + D^2 + D^3$.

9.2. Методы декодирования

При рассмотрении методов декодирования ограничимся случаем кодов со скоростью $R = 1/n_0$, хотя все рассматриваемые ниже методы применимы и в более общем случае. При описании алгоритмов декодирования будем обозначать как $x_i(s', s), 0 \leq i < n_0$, i-ый кодовый символ, порождаемый кодером при переходе из состояния s' в s.

9.2.1. Алгоритм Витерби

Рассмотрим задачу декодирования сверточных кодов. Она может быть сформулирована как поиск кодовой последовательности (или соответствующей ей информационной последовательности), являющейся наиболее правдоподобной для принятой последовательности сигналов [1, 10].

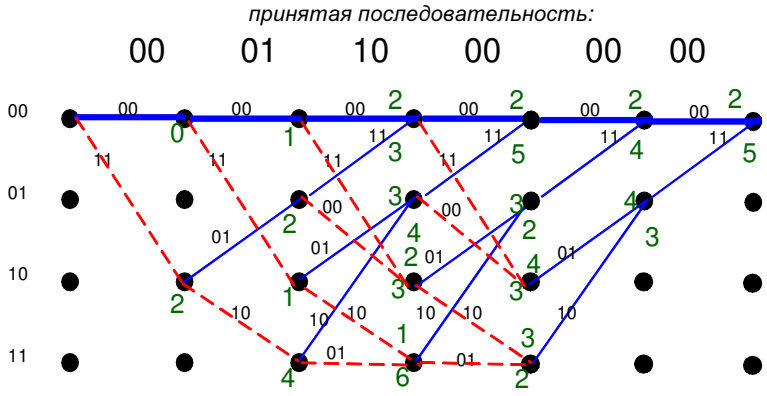


Рис. 9.4. Пример работы алгоритма Витерби в метрике Хэмминга

Предположим, что принятый сигнал представим как

$$y_{in_0+j} = c_{in_0+j} + e_{in_0+j}, 0 \le j < n_0, 0 \le i < L,$$

где c_i — переданный сигнал, e_i — ошибка. Эта модель применима как в случае двоичного симметричного канала (тогда $y_i, c_i, e_i \in GF(2)$) с переходной вероятностью $p < 1/2$, так и в случае аддитивного Гауссовского канала с двоичной амплитудной модуляцией (в этом случае $c_i \in \left\{-\sqrt{E_s}, \sqrt{E_s}\right\}, e_i \sim \mathcal{N}(0, \sigma^2)$). Декодирование по максимуму правдоподобия состоит в поиске кодового слова, ближайшего к принятой последовательности в метрике Хэмминга или Евклида. Эту задачу можно переформулировать как поиск в решетке пути с минимальной метрикой, начинающегося в нулевом состоянии в начальный момент времени, заканчивающегося в нем же в конечный момент времени L. Каждый путь можно также охарактеризовать последовательностью переходов между состояниями кодера $S = (s_0, s_1, \ldots, s_L)$. Под метрикой пути понимается расстояние между кодовым словом, соответствующим данному пути, и принятой последовательностью, т.е.

$$Q_L(S) = \sum_{i=0}^{t-1} d(i, s_i, s_{i+1}),$$

где L — число кадров, образующих кодовое слово, $d(\cdot)$ — используемая функция расстояния. и $d(i, s', s) = \sum_{j=0}^{n_0-1} d(r_{in_0+j}, x_j(s', s))$ — метрика i-го ребра решетки на данном пути, соединяющего узлы (s', i) и $(s, i+1)$. Видно, что $Q_{t+1}(S) = Q_t(S) + d(t+1, s_t, s_{t+1})$

В решетке существует множество путей, причем эти пути могут расходиться и сходиться. Ясно, что если имеются два пути $(c'|c)$ и $(c''|c)$

с общим хвостом, оптимальным может быть только тот из этих путей, который имеет меньшую метрику первой части. Это дает возможность воспользоваться принципом динамического программирования, который лежит в основе *алгоритма Витерби*. Под метрикой Q_{st} узла (s,t) решетки будем понимать наименьшую из накопленных за t переходов метрик путей, начинающихся в узле $(0,0)$ и входящих в узел (s,t). Все пути, кроме имеющего минимальную метрику, не могут быть решениями задачи декодирования и потому должны быть отброшены. Оставшийся путь называется выжившим. Таким образом, в каждый узел решетки входит не более чем один выживший путь. Этот выживший путь может далее разветвляться на пути, ведущие в другие состояния. Декодирование заканчивается вычислением метрики Q_{0L} конечного состояния $(0,L)$. Решением задачи декодирования является единственный выживший путь, входящий в это состояние. Заметим, что на каждом шаге этого алгоритма хранится информация о 2^W путях, из которых $2^{(W-1)}$ объявляются выжившими. Для хранения выживших путей каждому узлу (s,t) решетки необходимо приписать число s', указывающее на предыдущее состояние в выжившем пути. Однако длина хранимых выживших путей постоянно возрастает, что может быть неудобно при реализации. Поэтому, как правило, хранят информацию только о PW последних переходах на каждом из путей, где P равно 5 или 6. Величина PW носит название *ограничения длины при декодировании*. При этом на i-ом шаге алгоритма Витерби, кроме PW начальных, принимается окончательное решение относительно символов кодового слова, образующих кадр $i - PW$. В качестве таковых должны быть выбраны символы пути с наименьшей текущей накопленной метрикой.

Таким образом, алгоритм Витерби включает в себя следующие шаги:

1. Пусть $j = 0, Q_{s,0} = \begin{cases} 0, s = 0 \\ \infty, s \neq 0 \end{cases}$.
2. Для каждого состояния s вычислить $Q_{s,j+1} = \min_{s':(s',s)\in T} \left(Q_{s',j} + d(j,s',s)\right)$, где T — множество возможных переходов. Объявить путь $(0,\ldots,s',s)$, на котором достигнут минимум, выжившим. При наличии нескольких путей с одинаковой минимальной текущей накопленной метрикой выбрать любой из них.
3. $j = j + 1$. Если $j < L$, перейти к шагу 2.

4. Возвратить символы, соответствующие выжившему пути, заканчивающемуся в нулевом состоянии.

Пример его работы приведен на рис. 9.4. Заметим, что для реализации алгоритма достаточно наличия таблицы переходов кодера T и массива объемом $2^W PW$ для хранения выживших путей.

В случае метрики Евклида заметим, что $d(r_i, c_i) = r_i^2 - 2r_i c_i + c_i^2$. Для двоичного кода $c_i \in \{-\sqrt{E_s}, \sqrt{E_s}\}$ и $c_i^2 = E_s$. Следовательно, метрики узлов равны $\sum_i \left(r_i^2 - 2r_i c_i + c_i^2\right)$. Для всех состояний величины $\sum_i \left(r_i^2 + c_i^2\right)$ одинаковы, а потому могут быть исключены из рассмотрения. Следовательно, метрику ребра можно определить как корреляцию $d_{j,s,s'} = \sum_{k=1}^{n_0} r_{jn_0+k} c_{jn_0+k}$, где $c_{jn_0+k} = x_k(s, s')$ — символы, порождаемые кодером при переходе из состояния s в s', а минимизацию на втором шаге алгоритма заменить максимизацией. Этот подход применим и в тех случаях, когда вместо величин r_i на вход декодера подаются логарифмические отношения правдоподобия для отдельных символов кодового слова.

9.2.2. Алгоритм Бала-Коке-Елинека-Равива

Рассмотрим другой подход к декодированию, состоящий в поиске наиболее правдоподобного (вероятного) значения каждого информационного бита[3] $u_i, 0 \le i < L$, для заданной последовательности принятых сигналов $y_0^{Ln_0-1} = (u_0, \ldots, y_{Ln_0-1})$ [55, 59]. *алгоритм Бала-Коке-Елинека-Равива* (БКЕР) был первоначально предложен для декодирования линейных блоковых кодов, которые могут быть описаны с помощью переменной во времени решетчатой диаграммы. Усеченные кодовые последовательности сверточных кодов фиксированной длины образуют линейный блоковый код с регулярной решеткой (см. рис. 9.2, b). Это значительно упрощает реализацию алгоритма БКЕР, который в настоящее время чаще всего используется для декодирования именно сверточных кодов.

Также как и в случае алгоритма Витерби, будем предполагать, что передача осуществлялась по каналу без памяти. Вычислим логарифмическое отношение правдоподобия $L_k = \ln \frac{P\{u_k=1|y_0^{Ln_0-1}\}}{P\{u_k=0|y_0^{Ln_0-1}\}}$ для каждого информационного бита. Исходными данными для этой задачи являют-

[3]Здесь под информационными битами понимаются в том числе и служебные биты, используемые для приведения кодера в нулевое состояние после передачи полезных данных.

ся логарифмические отношения правдоподобия символов кодового слова $L(c_i) = \ln \frac{p(y_i|c_i=1)}{p(y_i|c_i=0)}$.

Искомая величина может быть найдена как

$$L_k = \ln \frac{P\{u_k = 1|y_0^{Ln_0-1}\}}{P\{u_k = 0|y_0^{Ln_0-1}\}} = \ln \frac{\sum\limits_{(s',s)\in S_1} p(s_k = s', s_{k+1} = s, y_0^{Ln_0-1})/p(y_0^{Ln_0-1})}{\sum\limits_{(s',s)\in S_0} p(s_k = s', s_{k+1} = s, y_0^{Ln_0-1})/p(y_0^{Ln_0-1})},$$

где S_1 и S_0 — множества пар состояний, переход между которыми осуществляется при подаче информационного бита, равного соответственно 1 и 0, $p(y_0^{Ln_0-1})$ — совместная плотность распределения принятых сигналов, $p(s_k = s', s_{k+1} = s, y_0^{Ln_0-1})$ — совместная плотность распределения принятых сигналов и состояний кодера на шагах k и $k+1$. Так как поведение кодера при обработке k-го информационного бита определяется только его состоянием s' на предыдущем шаге и канал не имеет памяти, эту плотность можно записать как $p(s_k = s', s_{k+1} = s, y_0^{Ln_0-1}) = \underbrace{p(s_k = s', y_0^{kn_0-1})}_{\alpha_k(s')} \underbrace{p(s_{k+1} = s, y_{kn_0}^{(k+1)n_0-1}|s_k = s')}_{\gamma_{k+1}(s',s)} \underbrace{p(y_{(k+1)n_0}^{Ln_0-1}|s_{k+1} = s)}_{\beta_{k+1}(s)}$. Из формулы Байеса вытекает, что

$$\alpha_k(s) = \sum_{\tilde{s}\in S} \alpha_{k-1}(\tilde{s})\gamma_k(\tilde{s}, s)$$

и

$$\beta_k(\tilde{s}) = \sum_{s\in S} \gamma_{k+1}(\tilde{s}, s)\beta_{k+1}(s),$$

где S — множество состояний кодера. Таким образом, совместные плотности распределения состояний кодера и фрагментов принятой последовательности рекурсивно выражаются через аналогичные плотности на предыдущем и последующем шагах соответственно. Непосредственное вычисление этих величин приводит к значительным ошибкам округления, поэтому приходится ввести вспомогательные величины $\alpha'_k(s) = \frac{\alpha_k(s)}{p(y_0^{kn_0-1})}$ и $\beta'_k(s) = \frac{\beta_k(s)}{p(y_{kn_0}^{Ln_0-1}|y_0^{kn_0-1})}$. Поделив $p(s_k = s', s_{k+1} = s, y_0^{Ln_0-1})$ на $\frac{p(y_0^{Ln_0-1})}{p(y_{kn_0}^{(k+1)n_0-1})} = p(y_0^{(k+1)n_0-1}|y_{kn_0}^{(k+1)n_0-1})p(y_{(k+1)n_0}^{Ln_0-1}|y_0^{(k+1)n_0-1}) = p(y_0^{kn_0-1})p(y_{(k+1)n_0}^{Ln_0-1}|y_0^{(k+1)n_0-1})$, получим $p(s_k = s', s_{k+1} =$

$s|y_0^{Ln_0-1})p(y_{kn_0}^{(k+1)n_0-1}) = \alpha'_k(s')\gamma_{k+1}(s',s)\beta'_{k+1}(s)$. При этом

$$L_k = \ln \frac{\sum_{(s',s)\in S_1} \alpha'_k(s')\gamma_{k+1}(s',s)\beta'_{k+1}(s)}{\sum_{(s',s)\in S_0} \alpha'_k(s')\gamma_{k+1}(s',s)\beta'_{k+1}(s)}. \tag{9.3}$$

Учитывая, что $p(y_0^{kn_0-1}) = \sum_{s\in S} \alpha_k(s)$, получим $\alpha'_k(s) = \frac{\alpha_k(s)}{\sum_{s'\in S}\alpha_k(s')} = \frac{\sum_{\tilde{s}\in S}\alpha_{k-1}(\tilde{s})\gamma_k(\tilde{s},s)}{\sum_{s'\in S}\sum_{\tilde{s}\in S}\alpha_{k-1}(\tilde{s})\gamma_k(\tilde{s},s')}$. Поделив числитель и знаменатель этого выражения на $p(y_0^{(k-1)n_0-1})$, получим

$$\alpha'_k(s) = \frac{\sum_{\tilde{s}\in S}\alpha'_{k-1}(\tilde{s})\gamma_k(\tilde{s},s)}{\sum_{s'\in S}\sum_{\tilde{s}\in S}\alpha'_{k-1}(\tilde{s})\gamma_k(\tilde{s},s')}. \tag{9.4}$$

Начальными условиями для этой рекуррентной формулы являются $\alpha'_0(s) = \alpha_0(s) = \begin{cases} 1, & s=0 \\ 0, & s\neq 0 \end{cases}$. Далее, $p(y_{kn_0}^{Ln_0-1}|y_0^{kn_0-1}) = p(y_{kn_0}^{Ln_0-1}|y_0^{(k+1)n_0-1})\frac{p(y_0^{(k+1)n_0-1})}{p(y_0^{kn_0-1})} = \frac{p(y_{(k+1)n_0}^{Ln_0-1}|y_0^{(k+1)n_0-1})}{p(y_0^{kn_0-1})}p(y_0^{(k+1)n_0-1}) = \frac{p(y_{(k+1)n_0}^{Ln_0-1}|y_0^{(k+1)n_0-1})}{p(y_0^{kn_0-1})}\sum_{s\in S}\alpha_{k+1}(s) = p(y_{(k+1)n_0}^{Ln_0-1}|y_0^{(k+1)n_0-1})\sum_{s\in S}\sum_{s'\in S}\alpha'_k(s')\gamma_{k+1}(s',s)$. Отсюда вытекает, что

$$\beta'_k(\tilde{s}) = \frac{\sum_{s\in S}\gamma_{k+1}(\tilde{s},s)\beta'_{k+1}(s)}{\sum_{s\in S}\sum_{s'\in S}\alpha'_k(s')\gamma_{k+1}(s',s)}. \tag{9.5}$$

Начальными условиями для этой рекуррентной формулы являются $\beta'_L(s) = \beta_L(s) = \begin{cases} 1, & s=0 \\ 0, & s\neq 0 \end{cases}$, где L — число кадров в принятой последовательности. Это условие было получено в предположении, что после обработки всех информационных битов кодер был переведен в нулевое состояние, т.е. $L = L_0 + \max\deg G_{ij}(D)$, где L_0 — истинное число информационных символов. Осталось заметить, что $\gamma_{k+1}(s',s) = p(s_{k+1}=s, y_{kn_0}^{(k+1)n_0-1}|s_k=s') = P\{s_{k+1}=s|s_k=s'\}p(y_{kn_0}^{(k+1)n_0-1}|s_k = s', s_{k+1}=s) = P\{u_k = \widehat{x}(s',s)\}p(y_{kn_0}^{(k+1)n_0-1}|s_k=s', s_{k+1}=s)$. Вероятность $P\{s_{k+1}=s|s_k=s'\} = P\{u_k=\widehat{x}(s',s)\}$ представляет собой априорную вероятность перехода из состояния s' в состояние s при кодировании k-го информационного бита, т.е. вероятность того, что этот бит равен конкретному значению $\widehat{x}(s',s)$. Если рассматриваемый декодер используется как часть итеративного декодера каскадного кода (например, турбо-кода, рассматриваемого ниже), эта вероятность может быть

найдена из апостериорных логарифмических отношений правдоподобия $L_k^{(e)}$, вычисленных другим декодером, как $P\{u_k = 1\} = \frac{\exp(L_k^{(e)})}{1+\exp(L_k^{(e)})}$, откуда следует, что

$$P\{u_k = a\} = \frac{\exp(L_k^{(e)}/2)}{1 + \exp(L_k^{(e)})} \exp\left((2a-1)\frac{L_k^{(e)}}{2}\right).$$

В противном случае все символы можно считать равновероятными, что эквивалентно $L_k^{(e)} = 0$. Условная плотность распределения $p(y_{kn_0}^{(k+1)n_0-1} | s_k = s', s_{k+1} = s)$ определяется символами $x_j(s', s), 0 \le j < n_0$, порождаемыми кодером при переходе из состояния s' в s. В силу предположения об отсутствии памяти у канала имеем $p(y_{kn_0}^{(k+1)n_0-1} | s_k = s', s_{k+1} = s) = \prod_{j=0}^{n_0-1} p_{Y|X}(y_{kn_0+j} | x_j(s', s))$. Если исходные данные для декодера представлены в форме логарифмических отношений правдоподобия, $p_{Y|X}(y_{kn_0+j} | x_j(s', s)) \sim \exp\left(\frac{2x_j(s',s)-1}{2} L(c_{kn_0+j})\right)$. Учитывая, что величины $\gamma_{k+1}(s', s)$ фигурируют как в числителе, так и в знаменателе выражений (9.3),(9.4), (9.5), коэффициенты пропорциональности в выражениях для $\gamma(s', s)$ можно отбросить, получив

$$\gamma_{k+1}(s', s) = \exp\left(\frac{(2\widehat{x}(s', s) - 1)L_k^{(e)} + \sum_{j=0}^{n_0-1}(2x_j(s', s) - 1)L(c_{n_0k+j})}{2}\right). \tag{9.6}$$

Таким образом, алгоритм БКЕР для кодов со скоростью $1/n_0$ включает в себя следующие шаги:

1. Нахождение логарифмических отношений правдоподобия отдельных символов кодового слова $L(c_i), i = 0..Ln_0 - 1$.
2. Вычисление величин $\gamma_k(s', s)$ согласно (9.6).
3. Вычисление $\alpha'_k(s)$ (прямая рекурсия) согласно (9.4).
4. Вычисление $\beta'_k(s)$ (обратная рекурсия) согласно (9.5).
5. Вычисление апостериорных логарифмических отношений правдоподобия $L_k, 0 \le k < L$, информационных битов согласно (9.3).
6. Принятие решения относительно каждого из информационных битов.

9.2.3. Анализ вероятности ошибки

Попытаемся оценить вероятность ошибки декодирования сверточного кода с помощью алгоритма Витерби [10]. Ясно, что она определяется расстоянием между различными кодовыми последовательностями,

которое в свою очередь зависит от диаграммы переходов кодера. Так как сверточный код является линейным, достаточно оценить только количество ненулевых кодовых последовательностей различного веса. При этом ограничимся рассмотрением только таких последовательностей, которые отличаются от нулевой кодовой последовательности в первом же кадре. Кроме того, будем предполагать, что код имеет бесконечную длину. В большинстве практических случаев длина передаваемых блоков данных достаточно велика, что позволяет пренебречь эффектами, связанными с конечностью длины кода.

Сведения о весовой структуре сверточного кода могут быть использованы для нахождения вероятности ошибочного декодирования. Ограничимся случаем двоичных кодов. Предположим, что через аддитивный гауссовский канал передавалась кодовая последовательность из всех нулей и декодер Витерби использует корреляционную метрику. Ошибка декодирования происходит в том случае, если при слиянии ненулевого и нулевого путей, происходящем в некотором кадре B, оказывается, что метрика Q_1 ненулевого пути превосходит метрику Q_0 нулевого пути[4]. Таким образом, необходимо найти вероятность $P\{Q_1 - Q_0 > 0\} = P\{\sum_{i=0}^{L-1}\sum_{j=0}^{n_0-1} r_{in_0+j}(c_{in_0+j}+1) > 0\}$, где c_{in_0+j} — кодовые символы, соответствующие ненулевой последовательности. Заметим, что $c_{in_0+j} + 1 = \begin{cases} 0, & c_{in_0+j} = -1 \\ 2, & c_{in_0+j} = 1 \end{cases}$. Таким образом, достаточно рассмотреть вероятность $P\{\sum_{l\in U} r_l > 0\}$, где U — множество позиций ненулевых символов рассматриваемого ошибочного пути. Принятые сигналы r_j являются независимыми Гауссовскими случайными величинами с матожиданием $-\sqrt{E_s}$ и дисперсией $N_0/2$. Следовательно, эта вероятность равна $P\{Q_1 - Q_0 > 0\} = P_2(d) = Q\left(\sqrt{\frac{2E_s}{N_0}d}\right) = Q\left(\sqrt{2\frac{E_b}{N_0}Rd}\right)$, где R — скорость кода и $d = |U|$. Однако имеется много путей с различными весами d, сливающихся с нулевым. Ошибка произойдет в случае, если метрика Q_i любого из этих путей окажется больше метрики нулевого пути Q_0. К сожалению, кодовые последовательности, соответствующие этим путям, могут иметь ненулевые символы на одних и тех же позициях. В результате случайные величины Q_i оказываются зависимыми. Поэтому удается выписать лишь верхнюю границу для вероятности ошибки в

[4]Так как сверточный код имеет бесконечную длину и повторяющуюся структуру, за одно декодирование может произойти несколько ошибок. Поэтому здесь рассматривается вероятность ошибочного декодирования только первого кадра.

первом кадре

$$P \leq \sum_{d=d_{\text{св}}}^{\infty} a_d Q\left(\sqrt{2\frac{E_b}{N_0}Rd}\right), \tag{9.7}$$

где a_d — коэффициенты ряда $T(D) = \sum_d a_d D^d$. С практической точки зрения больший интерес представляет вероятность ошибки декодирования на информационный бит. Каждый из ошибочных путей характеризуется весом w соответствующей ему информационной последовательности. Вероятность ошибки, соответствующая пути с весом кодовой последовательности d, может быть найдена как $\frac{w}{k_0}P_2(d)$. Передаточная функция $T(N, D)$ может быть представлена как $T(N, D) = \sum_{w,d} a_{wd} N^w D^d$, где a_{wd} — количество кодовых последовательностей веса d, порождаемых информационными последовательностями веса w. Для получения верхней границы вероятности ошибки на бит необходимо просуммировать величины $\frac{w}{k_0}P_2(d)$ по всем возможным путям, задаваемым параметрами w и d. Заметим, что $t(D) = \left.\frac{\partial T(N,D)}{\partial N}\right|_{N=1} = \sum_d D^d \left(\sum_w a_{wd} w\right) = \sum_d b_d D^d$. Таким образом,

$$P_b \leq \frac{1}{k_0} \sum_{d=d_{\text{св}}}^{\infty} b_d Q\left(\sqrt{2\frac{E_b}{N_0}Rd}\right). \tag{9.8}$$

Вероятность ошибки на кодовое слово конечной длины, полученное путем усечения кодовой последовательности сверточного кода, пропорциональна P_b и длине слова. Это делает сверточные коды малопригодными для передачи пакетов данных большой длины.

На рис. 9.5 представлены графики зависимости вероятности ошибки декодирования различных несистематических сверточных кодов со скоростью 1/2 в канале с АБГШ с помощью алгоритма Витерби, полученные с помощью статистического моделирования, а также соответствующие теоретические оценки. Порождающие многочлены кодов указаны в восьмеричном виде. Заметим, что при малых отношениях сигнал/шум граница (9.8) практически бесполезна. Это связано с тем, что в этой области вероятность одновременного появления нескольких событий $C_i > C_0$ достаточно велика, что не было должным образом учтено при анализе. Но при отношении сигнал/шум более 4 дБ результаты анализа практически не отличаются от результатов моделирования. При вероятностях ошибки менее 10^{-7} получение сколько-нибудь достоверных результатов методом статистического моделирования требует чрезмерно больших затрат вычислительного времени, что делает выражение (9.8) единствен-

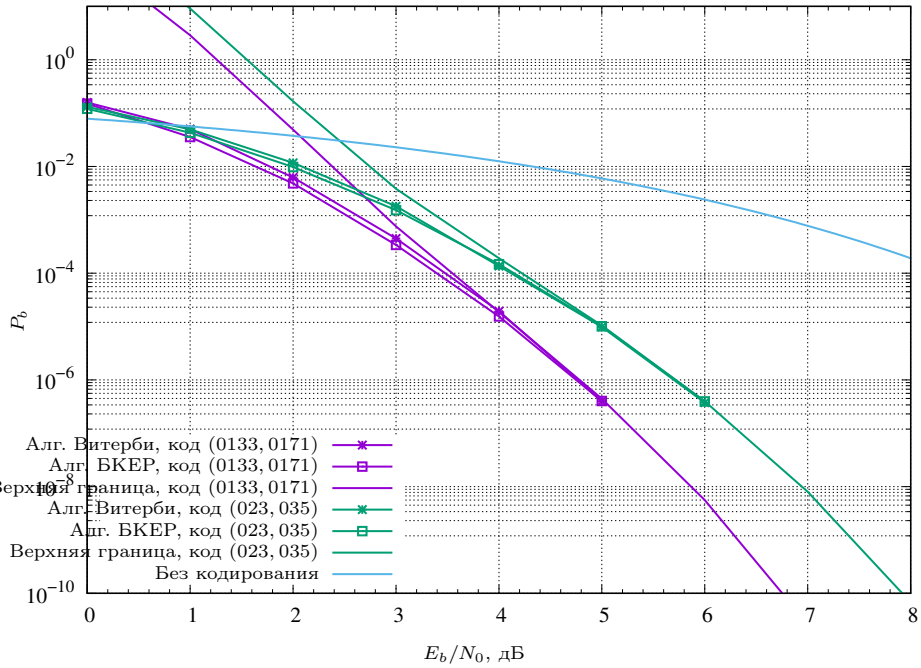


Рис. 9.5. Вероятность ошибки декодирования сверточных кодов в аддитивном Гауссовском канале

ным средством сравнения различных кодов. Кроме того, видно, что при малых отношениях сигнал/шум алгоритм БКЕР обеспечивает несколько меньшую вероятность ошибки по сравнению с алгоритмом Витерби. Это связано с тем, что алгоритм БКЕР производит побитовое декодирование, т.е. пытается найти наиболее вероятное значение каждого бита, в то время как алгоритм Витерби пытается найти наиболее вероятную их последовательность. При малых отношениях сигнал/шум часто оказывается, что есть несколько путей в решетке с близкими метриками. Алгоритм Витерби выбирает ровно один из них, и такой выбор зачастую оказывается ошибочным. Алгоритм БКЕР имеет возможность возвращать последовательности, соответствующие участкам различных путей в решетке, что и обеспечивает небольшой выигрыш по вероятности ошибки на бит.

Важной особенностью сверточных кодов является то, что ошибки декодера, как правило, приводят к возникновению нескольких близко расположенных неправильно оцененных информационных символов. Длина таких пакетов ошибок пропорциональна минимальному свободному расстоянию кода. Это необходимо учитывать при построении каскадных конструкций, основанных на сверточных кодах.

Упражнения

1. Изобразите кодер для сверточного кода с порождающей матрицей $G = \begin{pmatrix} 1 + x^2 + x^3 & 1 + x + x^3 \end{pmatrix}$, а также его рекурсивного систематического варианта. Постройте соответствующие решеточные диаграммы.
2. Найдите передаточную функцию для вышеуказанного кода.
3. Покажите, что обратимые операции над строками порождающей матрицы сверточного кода не влияют на его минимальное свободное расстояние.

10. Коды с малой плотностью проверок на четность

Коды с малой плотностью проверок на четность (МППЧ) были предложены в 1963 г. Галлагером [36]. Однако недостаточные возможности микроэлектроники не позволили в то время оценить их практические преимущества, и они были забыты более чем на 30 лет. В 1999 г. они были повторно открыты Мак-Кеем [50], который обнаружил, что эти коды позволяют приблизиться к пределу Шеннона, после чего они начали активно внедряться в самые разнообразные системы хранения и передачи информации.

10.1. Основные понятия

Кодом с малой плотностью проверок на четность называется линейный блоковый код с разреженной проверочной матрицей. Понятие разреженной матрицы во многом является расплывчатым. С практической точки зрения, к таковым можно отнести матрицы, доля ненулевых элементов которых не превышает 10%. С точки зрения анализа ансамблей кодов, матрицы, принадлежащие некоторому семейству, можно отнести к категории разреженных, если доля ненулевых элементов в них при длине кода, стремящейся к бесконечности, стремится к нулю. Необходимо понимать, что свойство разреженности относится только к одной из проверочных матриц кода; обратимыми линейными преобразованиями она всегда может быть приведена к плотному виду.

Как и в случае турбо-кодов, принадлежность кода к числу низкоплотностных определяется в основном применимостью алгоритмов декодирования, описанных в разд. 10.2. Как будет показано ниже, с увеличением плотности проверочной матрицы предположения, лежащие в их основе, становятся все менее близкими к реальности, что приводит к снижению точности промежуточных вычислений и понижению корректирующей способности.

Ограничимся рассмотрением двоичных кодов МППЧ. $m \times n$ проверочную матрицу H кода МППЧ удобно представлять в виде *графа Таннера*. Он представляет собой двудольный[1] неориентированный граф,

[1]Двудольным графом называется граф, множество вершин которого разбито на два непересекающихся подмножества, причем вершины каждого из подмножеств друг с другом не соединены ребрами.

имеющий n символьных и m проверочных узлов (вершин). i-ый проверочный узел соединен ребром с j-ым символьным узлом тогда и только тогда, когда $H_{ij} = 1$. *Степенью узла* называется число ребер, входящих в него. Очевидно, что степени символьных и проверочных узлов равны числу ненулевых элементов в соответствующих столбцах и строках проверочной матрицы. Граф Таннера (и соответствующий код, хотя это не совсем корректно, так как код обладает множеством различных проверочных матриц, соответствующих различным графам Таннера) называется (d_v, d_c)- *регулярным*, если степени всех символьных и проверочных узлов равны d_v и d_c соответственно. В более общем случае *нерегулярный* граф может быть охарактеризован *распределениями степеней символьных и проверочных узлов*

$$\lambda(x) = \sum_{i \geq 1} \lambda_i x^{i-1}, \rho(x) = \sum_{j \geq 1} \rho_j x^{j-1}.$$

Здесь λ_i (ρ_i) — доля ребер в графе Таннера, соединенных с символьными (проверочными) узлами степени i. Очевидно, что $\lambda(1) = \rho(1) = 1$. Необходимо отметить, что эти распределения не позволяют однозначно задать граф Таннера.

Иногда бывает удобно выразить эти распределения в терминах узлов. Пусть E — число ребер в графе Таннера. Тогда число символьных и проверочных узлов степени i равно $V_i = \frac{\lambda_i E}{i}$ и $C_i = \frac{\rho_i E}{i}$, соответственно. Отсюда доля символьных и проверочных узлов степени i равна, соответственно,

$$\Lambda_i = \frac{\lambda_i E/i}{\sum_{j \geq 1} \lambda_j E/j} = \frac{\lambda_i/i}{\int_0^1 \lambda(x)dx}$$

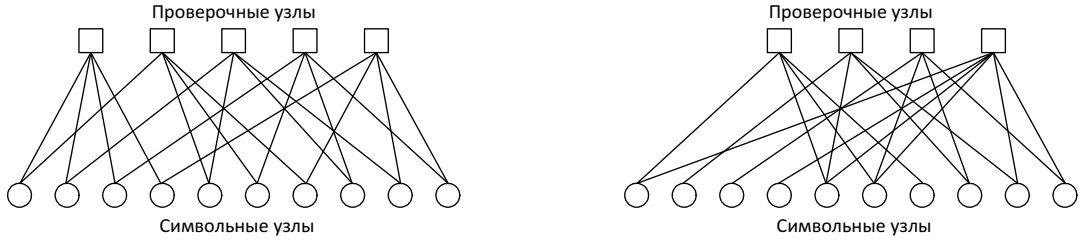
и

$$P_i = \frac{\rho_i E/i}{\sum_{j \geq 1} \rho_j E/j} = \frac{\rho_i/i}{\int_0^1 \rho(x)dx}.$$

Отсюда можно выразить относительную размерность проверочной матрицы как

$$\frac{m}{n} = \frac{\sum_{i \geq 1} C_i}{\sum_{j \geq 1} V_j} = \frac{\int_0^1 \rho(x)dx}{\int_0^1 \lambda(x)dx}. \tag{10.1}$$

При этом скорость кода может быть больше $1 - m/n$, если проверочная матрица содержит линейно зависимые строки.



(a) (b)

Рис. 10.1. Графы Таннера:
a — Регулярный; b — Нерегулярный

Пример 10.1. Рассмотрим проверочную матрицу

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

Соответствующий ей регулярный $(2, 4)$-граф Таннера представлен на рис. 10.1, a. Складывая вторую строку с пятой и исключая первую строку, получим другую проверочную матрицу этого же кода. Соответствующий ей нерегулярный граф представлен на рис. 10.1, b. Его распределения степеней имеют вид $\lambda(x) = \frac{4}{18}x^0 + \frac{8}{18}x^1 + \frac{6}{18}x^2$ и $\rho(x) = \frac{12}{18}x^3 + \frac{6}{18}x^5$.

10.2. Декодирование

10.2.1. Алгоритм инвертирования битов

Декодирование кодов МППЧ в метрике Хэмминга может быть выполнено с помощью следующего простейшего алгоритма. Для каждого символьного узла графа Таннера рассмотрим соединенные с ним проверочные узлы. Проверим, выполняются ли соответствующие им проверочные соотношения. Если число невыполненных проверочных соотношений превышает некоторый порог (например, половину от их числа), предположим, что соответствующий символ принятой последовательности искажен и попытаемся исправить ошибку, проинвертировав его. Будем продолжать эти действия до тех пор, пока не будут выполнены все проверочные соотношения, или не будет превышено максимальное число итераций.

В основе этого алгоритма лежит предположение о том, что проверочные узлы соединены с достаточно малым числом символьных узлов

(т.е. проверочная матрица разрежена), так что *почти* на каждое проверочное соотношение оказывает влияние не более чем одна ошибка. В этом случае каждая операция коррекции, как правило, уменьшает число ошибок в принятой последовательности и декодер сходится.

10.2.2. Алгоритм распространения доверия

Прежде чем рассматривать алгоритм декодирования для произвольного канала без памяти, рассмотрим процесс восстановления кодового слова кода МППЧ, переданного по двоичному стирающему каналу. Проверочная матрица задает набор соотношений вида $\sum_{i\in\mathcal{V}_j} c_i = 0, 1 \leq j \leq m$, где c_i — символы кодового слова. В принятой последовательности некоторые из них могут быть стерты, т.е. неизвестны. В силу разреженной структуры проверочной матрицы с большой вероятностью в достаточно большом числе этих соотношений окажется не больше чем один неизвестный символ. Этот символ может быть выражен из них через остальные известные символы, после чего некоторые из таких уравнений, в которых было несколько неизвестных, могут превратиться в уравнения с одним неизвестным, давая таким образом возможность найти значения еще нескольких стертых символов. Этот процесс можно интерпретировать как передачу сообщений от символьных узлов, соответствующих нестертым или восстановленным символам, к проверочным, которые вычисляют значения стертых символов и пересылают их по назначению.

Описанный алгоритм декодирования представляет собой процедуру решения системы линейных уравнений методом подстановок, из которого исключен этап ее преобразования к подходящему (верхнетреугольному) виду. В связи с этим некоторые конфигурации стираний, которые могут быть исправлены декодером (например, основанным на методе Гаусса решения систем линейных уравнений), реализующим критерий максимума правдоподобия, данным алгоритмом исправлены быть не могут. Это компенсируется исключительной простотой реализации.

Пример 10.2. Система проверочных соотношений, соответствующая матрице, приведенной в примере 10.1, имеет вид

$$\begin{cases} c_1 + c_2 + c_3 + c_4 &= 0 \\ c_1 + c_5 + c_6 + c_7 &= 0 \\ c_2 + c_5 + c_8 + c_9 &= 0 \\ c_3 + c_6 + c_8 + c_{10} &= 0 \\ c_4 + c_7 + c_9 + c_{10} &= 0 \end{cases}$$

Предположим, что стертыми оказались символы c_2, c_4, c_5, c_6. Из четвертого уравнения можно восстановить $c_6 = c_3 + c_8 + c_{10}$, а из пятого — $c_4 = c_7 + c_9 + c_{10}$. После

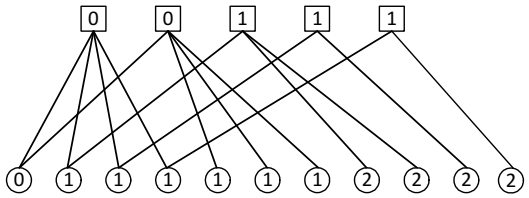


Рис. 10.2. Дерево Таннера

этого появляется возможность выразить из второго уравнения $c_5 = c_1 + c_6 + c_7$ и из первого $c_1 = c_2 + c_3 + c_4$.

Рассмотрим более общий случай произвольного канала без памяти. Попытаемся оценить вероятность того, что i-ый символ кодового слова равен 1 или 0 при условии того, что наблюдается принятая последовательность $y_1^n = (y_1, \ldots, y_n)$ и выполнены все проверки на четность. Для того, чтобы упростить задачу, будем считать, что граф Таннера представляет собой дерево и i-ый символьный узел является его корнем (см. рис. 10.2). Таким образом, множество символьных узлов разбивается на несколько уровней в зависимости от их расстояния в дереве от корня, который считается узлом нулевого уровня.

Введем обозначения $\mathbf{C}_i = \{j | H_{ji} = 1\}, 1 \leq i \leq n$, и $\mathbf{V}_j = \{i | H_{ji} = 1\}, 1 \leq j \leq m$. Найдем вероятность $q_i(b)$ того, что символ c_i, расположенный на нулевом уровне в дереве Таннера, равен $b \in \{0, 1\}$, при условии того, что выполнены все проверки на четность. Пусть C_j — событие, означающее выполнение j-ой проверки на четность. Пусть $T(i)$ — множество проверочных узлов, находящихся в поддереве графа Таннера, корнем которого является символьный узел i. Тогда

$$
\begin{aligned}
q_i(b) &= P\left\{c_i = b | y_1^n, \cap_{j \in T(i)} C_j\right\} = \frac{P\left\{c_i = b, y_1^n, \cap_{j \in T(i)} C_j\right\}}{P\left\{y_1^n, \cap_{j \in T(i)} C_j\right\}} = \\
&= P\left\{c_i = b | y_1^n\right\} \frac{P\left\{\cap_{j \in T(i)} \overline{C}_j | c_i = b, y_1^n\right\}}{P\left\{\cap_{j \in T(i)} C_j | y_1^n\right\}} = P\left\{c_i = b | y_i\right\} \cdot \\
&\cdot \quad P\left\{\cap_{j \in \mathbf{C}_i} C_j | c_i = b, y_1^n, \cap_{j \in T(i) \setminus \mathbf{C}_i} C_j\right\} \frac{P\left\{\cap_{j \in T(i) \setminus \mathbf{C}_i} C_j | c_i = b, y_1^n\right\}}{P\left\{\cap_{j \in T(i)} C_j | y_1^n\right\}} =
\end{aligned}
$$

$$= P\{c_i = b|y_i\} P\left\{\bigcap_{j\in\mathbf{C}_i} C_j | c_i = b, y_1^n, \bigcap_{j\in T(i)\setminus\mathbf{C}_i} C_j\right\} \underbrace{\frac{P\left\{\cap_{j\in T(i)\setminus\mathbf{C}_i} C_j | y_1^n\right\}}{P\left\{\cap_{j\in T(i)} C_j | y_1^n\right\}}}_{A} =$$

$$= AP\{c_i = b|y_i\} \prod_{j\in\mathbf{C}_i} \underbrace{P\left\{\sum_{i'\in\mathbf{V}_j} c_{i'} = 0 | c_i = b, y_1^n, \cap_{j'\in T(i)\setminus\mathbf{C}_i} C_{j'}\right\}}_{r_{ji}(b)}. \qquad (10.2)$$

Предпоследний переход возможен в силу того, что никакие другие проверки на четность, кроме учтенных в $\overline{C}_i$, не содержат символ c_i. Нормировочная константа A может быть найдена из условия $q_i(0) + q_i(1) = 1$. Последний переход возможен в силу того, что проверки на четность, входящие в $\mathbf{C}_i$, не имеют никаких общих символов, кроме c_i, вследствие чего события их выполнения являются независимыми.

С учетом леммы 7.1, вероятность

$$r_{ji}(b) = P\left\{\sum_{i'\in\mathbf{V}_j\setminus\{i\}} c_{i'} = b | y_1^n, \cap_{j'\in T(i)\setminus\mathbf{C}_i} C_{j'}\right\}$$

может быть найдена как

$$r_{ji}(1) = 1 - r_{ji}(0) \qquad (10.3)$$

$$r_{ji}(0) = \frac{1}{2}\left(1 + \prod_{i'\in\mathbf{V}_j\setminus\{i\}} \left(1 - 2P\left\{c_{i'} = 1 | y_1^n, \cap_{j'\in T(i)\setminus\mathbf{C}_i} C_{j'}\right\}\right)\right) =$$

$$= \frac{1}{2}\left(1 + \prod_{i'\in\mathbf{V}_j\setminus\{i\}} \left(1 - 2P\left\{c_{i'} = 1 | y_1^n, \cap_{j'\in T(i')\setminus\{j\}} C_{j'}\right\}\right)\right). \qquad (10.4)$$

Последний переход возможен в силу того, что здесь из рассмотрения исключаются все проверочные соотношения, в которых участвует символ i. Поэтому далее можно рассматривать только те проверочные узлы, которые находятся в поддереве графа Таннера с корнем в узле i'. Преобразования, аналогичные (10.2), приводят к

$$q_{ij}(b) = P\left\{c_i = b | y_1^n, \cap_{j'\in T(i)\setminus\{j\}} C_{j'}\right\} =$$

$$= A'P\{c_i = b|y_i\} \prod_{j'\in\mathbf{C}_i\setminus\{j\}} r_{j'i}(b). \qquad (10.5)$$

Таким образом, получены рекуррентные уравнения, которые позволяют выразить $q_i(b)$ для узла i, являющегося корнем дерева Таннера, через $q_{i'j}(b) = A'P\left\{c_{i'} = b|y'_i\right\}$, где i' — символьные узлы, являющиеся его листьями. К сожалению, можно показать, что коды с проверочной матрицей, граф Таннера которой имеет вид дерева, имеют минимальное расстояние не более 2 при скорости $R \geq 1/2$ или могут быть получены из таковых путем повторения некоторых символов при $R < 1/2$ [29]. Ясно, что они не могут обеспечить сколь-нибудь приемлемую корректирующую способность. Тем не менее, описанный метод может быть использован и в случае графов Таннера, содержащих циклы. В этом случае события выполнения различных проверочных соотношений оказываются зависимыми, что делает представленные выше формулы для $q_i(b), r_{ji}(b), q_{ij}(b)$ приближенными. Однако, практика показывает, что при обхвате (т.е. длине кратчайшего цикла) графа Таннера не менее 6, связанные с этим погрешности пренебрежимо малы.

Непосредственное использование приведенных выше выражений сопряжено с необходимостью перемножения и вычитания величин, близких к 1, что приводит к значительным ошибкам округления. Для того, чтобы избежать их, введем логарифмические отношения правдоподобия $L_i^{(v)} = \ln\frac{q_i(0)}{q_i(1)}, L_{ij}^{(v)} = \ln\frac{q_{ij}(0)}{q_{ij}(1)}, L_{ji}^{(c)} = \ln\frac{r_{ji}(0)}{r_{ji}(1)}$. Будем считать, что исходные данные для декодера также представлены в виде логарифмических отношений правдоподобия[2] $L_i = \ln\frac{P\{c_i=0|y_i\}}{P\{c_i=1|y_i\}}$. Заметим, что имеет место тождество

$$\tanh\left(\frac{1}{2}\ln\frac{1-p}{p}\right) = \frac{e^{\ln((1-p)/p)-1}}{e^{\ln((1-p)/p)+1}} = 1-2p.$$

Переписав (10.4) как

$$1-2r_{ji}(1) = \prod_{i'\in\mathbf{V}_j\setminus\{i\}}(1-2q_{i'j}(1)),$$

получим, что

$$\tanh\left(\frac{1}{2}L_{ji}^{(c)}\right) = \prod_{i'\in\mathbf{V}_j\setminus\{i\}}\tanh\left(\frac{1}{2}L_{i'j}^{(v)}\right).$$

[2]Ранее они были определены как $L_i = \ln\frac{P\{c_i=1|y_i\}}{P\{c_i=0|y_i\}}$. Изменение обозначений позволяет несколько упростить полученные выражения.

Пусть $L_{i'j}^{(v)} = a_{i'j} b_{i'j}$, где $b_{i'j} = |L_{i'j}^{(v)}|, a_{i'j} = \operatorname{sgn}(L_{i'j}^{(v)})$. Учитывая, что гиперболический тангенс является нечетной функцией, получим

$$L_{ji}^{(c)} = 2 \prod_{i' \in \mathbf{V}_j \setminus \{i\}} a_{i'j} \tanh^{-1} \prod_{i' \in \mathbf{V}_j \setminus \{i\}} \tanh\left(\frac{1}{2} L_{i'j}^{(v)}\right).$$

Таким образом, *алгоритм распространения доверия* может быть записан как

$$L_i^{(v)} = L_i + \sum_{j \in \mathbf{C}_i} L_{ji}^{(c)} \tag{10.6}$$

$$L_{ij}^{(v)} = L_i + \sum_{j' \in \mathbf{C}_i \setminus \{j\}} L_{j'i}^{(c)} \tag{10.7}$$

$$L_{ji}^{(c)} = \left(\prod_{i' \in \mathbf{V}_j \setminus \{i\}} a_{i'j} \right) \phi \left(\sum_{i' \in \mathbf{V}_j \setminus \{i\}} \phi(b_{i'j}) \right), \tag{10.8}$$

где $\phi(x) = -\ln \tanh(x/2)$. Несложно показать, что $\phi(\phi(x)) = x$, что и было использовано при получении (10.8).

Эти тождества могут быть использованы для декодирования следующим образом:

1. Положить $L_{ji}^{(c)} = 0$.
2. Вычислить $L_i^{(v)}$ в соответствии с (10.6).
3. Построить вектор $\widehat{c} = (\widehat{c}_1, \ldots, \widehat{c}_n)$ оценок символов кодового слова, где $\widehat{c}_i = 1 \Leftrightarrow L_i^{(v)} < 0$. Если $H\widehat{c}^T = 0$, завершить декодирование и возвратить кодовое слово $\widehat{c}$.
4. Вычислить $L_{ij}^{(v)}$ в соответствии с (10.7).
5. Вычислить $L_{ji}^{(c)}$ в соответствии с (10.8).
6. Перейти к шагу 2.

Перемножение величин $a_{i'j}$ фактически сводится к подсчету четности числа их отрицательных значений, т.е. операции “исключающее или”. Таким образом, наиболее трудоемкой частью этого метода оказывается вычисление функции $\phi(x)$, график которой представлен на рис. 10.3. При непосредственном ее вычислении возможно возникновение значительных ошибок округления в области больших и малых x. Для избежания этой проблемы может быть целесообразно построить подходящие аппроксимации, точность вычисления которых с помощью машинной арифметики может оказаться выше, чем в случае исходной функции. Другим способом избежания этих проблем и снижения сложности

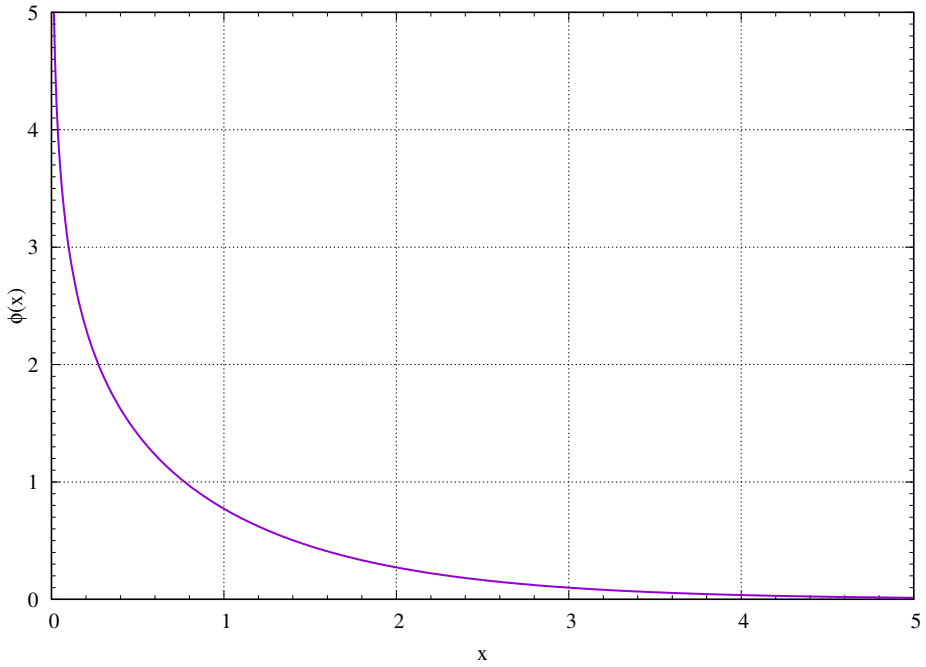


Рис. 10.3. Функция $\phi(x)$

является использование заранее вычисленной таблицы ее значений. Еще одним методом снижения сложности вычислений является аппроксимация суммы в (10.4) наибольшим из слагаемых. Так как функция $\phi(x)$ является монотонно убывающей, это приводит к выражению

$$L_{ji}^{(c)} \approx \left(\prod_{i' \in \mathbf{V}_j \setminus \{i\}} a_{i'j} \right) \min_{i' \in \mathbf{V}_j \setminus \{i\}} b_{i'j}.$$

Интерпретируя $b_{i'j}$ как меру надежности символа i' с учетом уже обработанных проверок на четность, получим, что $L_{ji}^{(c)}$ представляет собой оценку i-ого символа, выраженную из j-ого проверочного соотношения, причем ее надежность определяется надежностью наихудшего из прочих символов, участвующих в этом соотношении. Исходная формула (10.8) фактически представляет собой более точное выражение этой идеи, лежащей в основе алгоритма распространения доверия. Существуют и другие методы ускорения алгоритма распространения доверия [60].

10.2.3. Эволюция плотностей

При анализе корректирующей способности ограничимся случаем симметричных по выходу каналов, т.е. будем предполагать, что $p_{Y|X}(y|1) = p_{Y|X}(-y|-1)$. В этом случае корректирующая способность линейного кода и алгоритма распространения доверия не зависят от то-

го, какое из кодовых слов передавалось. Для удобства предположим, что передавалось нулевое кодовое слово. В этом случае величины L_i имеют одно и то же распределение, которое может быть легко вычислено. Например, в случае аддитивного гауссовского канала с дисперсией шума σ^2 и использования модулятором отображения $0 \to 1, 1 \to -1$ из (1.2) вытекает, что $L_i = \frac{2y_i}{\sigma^2}$, т.е. $L_i \sim \mathcal{N}(2/\sigma^2, 4/\sigma^2)$. Будем обозначать соответствующую плотность распределения как $p_L(x)$. Преобразования (10.7)–(10.8) являются детерминированными. Это позволяет рекурсивно вычислить плотности распределения величин $L_{ij}^{(v)}, L_{ji}^{(c)}$ на каждой итерации алгоритма распространения доверия. Ошибка декодирования имеет место, если среди сообщений, циркулирующих по сети, задаваемой графом Таннера, присутствуют отрицательные величины. Вероятность такого события можно оценить, проинтегрировав соответствующую плотность распределения от $-\infty$ до 0.

Предположим, что длина кода стремится к бесконечности, и входящие сообщения каждого узла являются независимыми случайными величинами. Дальнейший анализ будет проводиться не для конкретного графа Таннера, а для ансамбля графов с заданными распределениями степеней символьных и проверочных узлов, т.е. его результаты должны рассматриваться как средние по всем соответствующим кодам.

Известно, что плотность распределения суммы независимых случайных величин X и Y может быть выражена через их плотности распределения $p_X(x)$ и $p_Y(y)$ как их свертка, т.е.

$$p_{X+Y}(z) = p_X \otimes p_Y = \int_{-\infty}^{\infty} p_X(z-y) p_Y(y) dy.$$

Предполагая, что все сообщения, входящие в символьный узел степени d на итерации l, имеют одну и ту же плотность распределения $p_{C_{l-1}}(x)$, получим, что плотность распределения случайных величин $L_{ij}^{(v)}$ равна

$$p_{V_l}^{(d)} = p_L \otimes p_{C_{l-1}}^{\otimes(d-1)},$$

где $p_{C_{l-1}}^{\otimes(d-1)}$ обозначает свертку $d-1$ функций $p_{C_{l-1}}(x)$ с собой. Так как в графе Таннера присутствуют символьные узлы различных степеней, необходимо произвести усреднение по ним. Величины $L_{ij}^{(v)}$ параметризуются фактически номером ребра (однозначно задается парой (i,j)), поэтому справедливо равенство

$$p_{V_l} = p_L \otimes \sum_{i \geq 1} \lambda_i p_{C_{l-1}}^{\otimes(i-1)} = p_L \otimes \lambda(p_{C_{l-1}}).$$

Здесь под $\lambda(p_{C_{l-1}})$ понимается результат "подстановки" плотности распределения $p_{C_{l-1}}$ в многочлен, задающий распределение степеней символьных узлов, причем операция возведения в степень i заменяется (i)-кратной сверткой.

Перепишем (10.8) как $L_{ji}^{(c)} = \Gamma^{-1}\left(\sum_{i'\in\mathbf{V}_j\setminus\{i\}}\Gamma(L_{i'j}^{(v)})\right)$, где $\Gamma(L_{i'j}^{(v)}) = (\mathbf{1}_{L_{i'j}^{(v)}<0}, \phi(|L_{i'j}^{(v)}|))$, $\mathbf{1}_X$ — индикаторная функция, принимающая значение 1 при выполнении условия X и 0 в противном случае. Суммирование выполняется покомпонентно, причем сложение первых (двоичных) компонент выполняется по модулю 2. Таким образом, функция распределения $\Gamma(L_{i'j}^{(v)})$ имеет вид $G(s,x) = \mathbf{1}_{s=0}P\left\{L_{i'j}^{(v)} \geq 0, \phi(|L_{i'j}^{(v)}|) < x\right\} + \mathbf{1}_{s=1}P\left\{L_{i'j}^{(v)} < 0, \phi(-L_{i'j}^{(v)}) < x\right\} = \mathbf{1}_{s=0}P\left\{L_{i'j}^{(v)} > \phi(x)\right\} + \mathbf{1}_{s=1}P\left\{L_{i'j}^{(v)} < -\phi(x)\right\}$. Дифференцируя ее по x, получим плотность распределения $g(s,x) = \mathbf{1}_{s=0}g_0(x)+\mathbf{1}_{s=1}g_1(x)$. Сумма двумерных случайных величин $\Gamma(G)$ и $\Gamma(F)$ с плотностями распределения $g(s,x)$ и $f(s,x)$ такого вида имеет плотность распределения

$$\begin{aligned} h(s,x) &= g(s,x)\otimes f(s,x) = \mathbf{1}_{s=0}h_0(x) + \mathbf{1}_{s=1}h_1(x) = \\ &= \mathbf{1}_{s=0}(g_0(x)\otimes f_0(x) + g_1(x)\otimes f_1(x)) + \\ &\quad +\mathbf{1}_{s=1}(g_1(x)\otimes f_0(x) + g_0(x)\otimes f_1(x)), \end{aligned}$$

т.е. также представляется как свертка. Зная ее, можно восстановить плотность распределения одномерной случайной величины $\Gamma^{-1}(\Gamma(G) + \Gamma(F))$. Злоупотребляя обозначениями, будем обозначать операции преобразования плотностей распределения, соответствующие применению отображения $\Gamma(x)$, как $\Gamma(p)$. Таким образом, плотность распределения случайных величин $L_{ij}^{(c)}$ может быть найдена как

$$p_{C_l} = \Gamma^{-1}\left(\sum_{i\geq 1}\rho_i(\Gamma(p_{V_l}))^{\otimes(i-1)}\right) = \Gamma^{-1}\rho(\Gamma(p_{V_l})).$$

Таким образом, плотность распределения сообщений на итерации l равна

$$p_{V_l} = p_L \otimes \lambda(\Gamma^{-1}\rho(\Gamma(p_{V_(l-1)}))). \tag{10.9}$$

Вероятность наличия в сети ошибочных сообщений на итерации l равна

$$P_e^{(l)} = \int_{-\infty}^{0} p_{V_l}(x)dx. \tag{10.10}$$

Можно показать [57], что $P_e^{(l)} \to_{l\to\infty} = 0$ тогда и только тогда, когда последовательность плотностей распределения p_{V_l} сходится к $\delta(x-\infty)$, где $\delta(x)$ — дельта-функция Дирака. Под сходимостью последовательности $f_l(x)$ к предельной функции $f(x)$ здесь понимается, что для всех x последовательность значений функции распределения $F_l(x) = \int_{-\infty}^{x} f_l(y)dy$ сходится к $f(x)$.

Кроме того, при достаточно слабых ограничениях имеют место следующие факты:

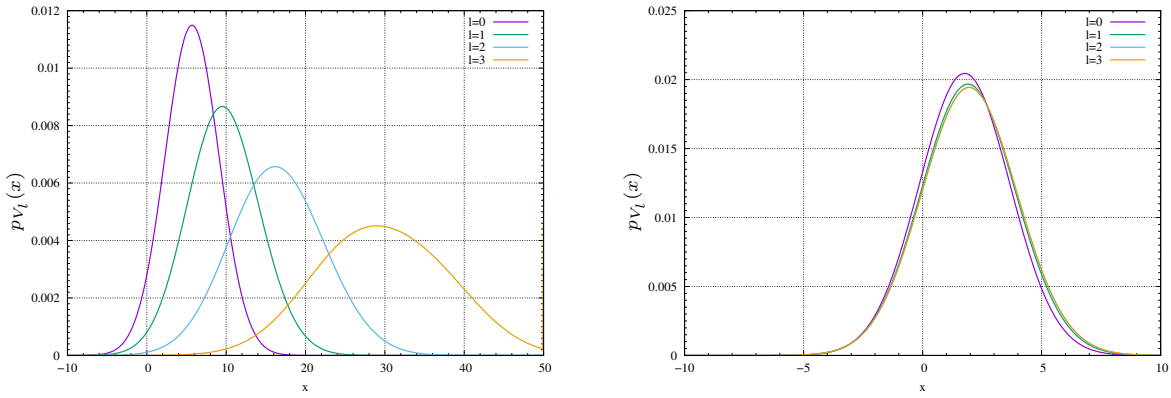
- Если $\lambda'(0)\rho'(1) > e^r$, то существует такое $\xi = \xi(\lambda(x), \rho(x), p_L) > 0$, что для всех $l \in N, P_e^{(l)} > \xi$.
- Если $\lambda'(0)\rho'(1) < e^r$, то существует такое $\xi = \xi(\lambda(x), \rho(x), p_L) > 0$, что если для некоторого $l \in \mathbb{N}\, P_e^{(l)} \leq \xi$, то при выполнении дальнейших итераций (т.е. при $l' \to \infty$) $P_e^{(l')}$ стремится к нулю.
- Последовательность плотностей распределения (10.9) сходится к некоторой неподвижной точке, которая зависит исключительно от начального распределения p_L.

Здесь

$$r = -\ln\left(\int_{-\infty}^{\infty} p_L(x)e^{-x/2}dx\right).$$

Эта величина является некоторой метрикой качества канала. Этот результат показывает, что если характеристики рассматриваемого ансамбля недостаточно хороши для канала с заданным качеством, то вероятность ошибки декодирования всегда ограничена снизу некоторой положительной величиной. В противном случае, если в начале процесса декодирования удалось "добиться некоторого прогресса" в исправлении ошибок, то при последующих итерациях все ошибки будут исправлены.

На рис. 10.4 представлены примеры эволюции плотностей распределения сообщений для двух рассмотренных случаев. Видно, что при $\sigma > \sigma_0$ плотность распределения сообщений практически не меняется (рис. 10.4, b). В противном случае их среднее значение быстро возрастает. Одновременно с ним возрастает и дисперсия, поэтому сходимость последовательности плотностей распределения к δ-функции из графика неочевидна. Однако из графика видно, что с увеличением l возрастает и наименьшее значение x, при превышении которого *плотность распределения* $p_{V_l}(x)$ становится существенно больше нуля. За счет этого последовательность значений *функции распределения* в любой точке $x < \infty$ действительно сходится к нулю, а при $x = \infty$ — к бесконечности.



(a) (b)

Рис. 10.4. Эволюция плотностей распределения:
a — $\sigma < \sigma_0$; b — $\sigma > \sigma_0$

Успешность декодирования определяется тем, соответствует ли неподвижная точка эволюции плотностей распределению с положительной или нулевой вероятностью ошибки. Предположим, что качество канала (т.е. плотность распределения p_L) однозначно описывается одним параметром σ (например, в случае аддитивного гауссовского канала таковым параметром является среднеквадратическое отклонение шума). Тогда существует некоторое пороговое значение σ_0 такое, что при $\sigma < \sigma_0$ вероятность ошибки стремится к нулю с ростом числа итераций, и ограничена снизу некоторой положительной величиной в противном случае. Это значение называется *порогом итеративного декодирования*. Более точно, для всех $\epsilon > 0$ и $\sigma < \sigma_0$ существуют $n = n(\epsilon, \sigma)$ и $l = l(\epsilon, \sigma)$ такие, что почти для всех[3] кодов длины не менее n, которым соответствуют графы Таннера из ансамбля $(\lambda(x), \rho(x))$, вероятность ошибки декодирования после выполнения l итераций алгоритма распространения доверия не превосходит ϵ. С другой стороны, при $\sigma > \sigma_0$ вероятность ошибки всех кодов данного ансамбля ограничена снизу некоторой величиной, зависящей от σ и не зависящей от числа итераций. Это утверждение весьма похоже на прямую и обратную теоремы кодирования.

Необходимо понимать, что данный анализ был выполнен в предположении о независимости сообщений, циркулирующих по сети, задаваемой графом Таннера. Из этих утверждений не вытекает, что конкретный код из заданного ансамбля сможет обеспечить сколь угодно малую ве-

[3] Доля кодов, для которых это не выполняется, убывает к нулю экспоненциально быстро с ростом n.

роятность ошибки при заданном качестве канала $\sigma < \sigma_0$. Тем не менее, эти результаты могут быть использованы при построении хороших кодов МППЧ. Для этого заметим, что выражения (10.9)–(10.10) позволяют оценить вероятность ошибки на l-ой итерации. Если она оказалась достаточно малой, можно считать, что в дальнейшем она сойдется к нулю. Таким образом, существует численный алгоритм, позволяющий проверить, достигается ли для заданного качества канала сколь угодно малая вероятность ошибки при использовании ансамбля $(\lambda(x), \rho(x))$. Воспользовавшись методом дихотомии, можно найти собственно порог σ_0, однозначно связанный с распределениями $(\lambda(x), \rho(x))$. Это дает возможность произвести его максимизацию по параметрам этих распределений, при которой необходимо учесть условия $\lambda(1) = \rho(1) = 1, R \geq 1 - \frac{\int_0^1 \rho(x)dx}{\int_0^1 \lambda(x)dx}$, где R — требуемая скорость кода. Данная оптимизационная задача имеет множество локальных минимумов, но использование механизмов глобальной максимизации (например, метода отжига) позволяет получить достаточно хорошие результаты. Подробно вопросы эффективной реализации данной оптимизационной процедуры рассмотрены в работах [57, 54].

10.3. Методы построения кодов

В данном разделе представлено краткое описание некоторых конструкций кодов МППЧ. К настоящему времени их общее число составляет несколько десятков и продолжает расти, поэтому сколь-нибудь полный обзор этой области теории кодирования может составить предмет отдельной книги (см., например, [60]).

10.3.1. Конструкция Галлагера

В оригинальной работе Галлагера [36] был исследован класс кодов МППЧ с проверочной матрицей вида $H = \begin{pmatrix} H_1 \\ H_2 \\ \vdots \\ H_g \end{pmatrix}$, где H_1 — $m \times n$ матрица, содержащая единицы в позициях $ir \ldots (i+1)r - 1$ в каждой строке $i, 0 \leq i \leq m - 1$, а прочие матрицы получаются из нее путем перестановки столбцов. Им было показано, что при $g \geq 3$ минимальное расстояние таких кодов может быть достаточно хорошим. Однако эти коды являются регулярными и требуют подбора перестановок столбцов, не приводящих к возникновению циклов длины 4 в графе Таннера.

10.3.2. Прогрессивное наращивание ребер

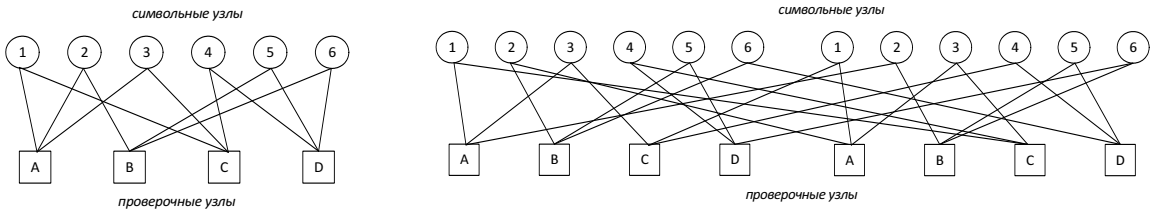
Одним из простейших способов построения кодов с малой плотностью проверок на четность с произвольным распределением степеней символьных узлов является итеративное наращивание ребер в графе Таннера [38]. Данный метод предполагает, что все проверочные узлы имеют почти одинаковую степень[4], и итеративно строит граф Таннера, содержащий заданное число символьных узлов заданных степеней. При этом накладывается требование максимизации обхвата графа. Первоначально граф не содержит ни одного ребра. Все символьные узлы обрабатываются в порядке возрастания их *планируемой* степени. На каждом шаге алгоритма текущий символьный узел соединяется ребром с проверочным узлом, который выбирается таким образом, чтобы максимизировать длину кратчайшего из создаваемых при этом циклов. Неопределенность при этом разрешается в пользу проверочного узла с наименьшей текущей степенью. После того, как степень текущего символьного узла достигнет требуемой, осуществляется переход к следующему символьному узлу.

Основным преимуществом этого метода является возможность построения графов Таннера с любым распределением степеней символьных узлов и достаточно большим обхватом. Впрочем, практика показывает, что увеличение обхвата графа свыше 6 не оказывает значимого влияния на качество декодирования. Главным недостатком описанной конструкции является отсутствие какой-либо структуры у получаемых таким образом кодов, что значительно осложняет аппаратную реализацию кодера и декодера, а также достаточно небольшое их минимальное расстояние.

10.3.3. Протографы

Хорошие нерегулярные структурированные коды МППЧ могут быть получены путем построения некоторого протографа с небольшим числом узлов и последующего его “масштабирования” до требуемых параметров. Процедура масштабирования сводится к созданию Q копий этого протографа и “переключению” ребер, соединяющих узлы с одинаковыми номерами, между этими копиями. В терминах проверочной матрицы эта процедура сводится к замене единиц в проверочной матрице, задающей протограф, различными $Q \times Q$ перестановочными матрица-

[4]Большинство распределений $\rho(x)$, получаемых как результат оптимизационной процедуры, описанной в разд. 10.2.3, имеют вид $\rho(x) = x^d$, поэтому данное ограничение не является существенным.



(a) (b)

Рис. 10.5. Построение графа Таннера из протографа:
a — Протограф; b — Результат масштабирования

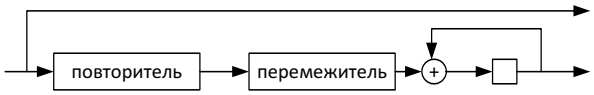


Рис. 10.6. Кодер повторительно-накопительного кода

ми и нулей — нулевыми матрицами той же размерности. Как правило, при этом используются перестановочные матрицы циклического сдвига на l_{ij} позиций, причем параметры l_{ij} масштабируются пропорционально Q [39].

Пример 10.3. На рис. 10.5 представлен пример протографа, соответствующего матрице $H = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$, и полученного из него графа Таннера, соответствующего матрице $H' = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix}$.

10.3.4. Повторительно-накопительные коды

Повторительно-накопительные коды [27, 43], устройство кодера для которых представлено на рис. 10.6, относятся к числу кодов МППЧ с хорошим минимальным расстоянием. При кодировании каждый символ данных повторяется несколько раз (для регулярных кодов — фиксированное, для нерегулярных — переменное число раз в соответствии с заданным распределением), полученные символы переставляются и пропускаются через устройство-интегратор, которое суммирует символ

со значением, полученным на предыдущем шаге. К полученным таким образом кодовым символам дописываются информационные. Таким образом, порождающая матрица этих кодов имеет вид $G = (RPA|I)$, где R — матрица, описывающая повторитель, P — перестановочная матрица, $A = \begin{pmatrix} 1 & 1 & \dots & 1 \\ 0 & 1 & \dots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{pmatrix}$ — матрица, соответствующая накопителю. Отсюда можно получить выражение для проверочной матрицы $H = \left(\begin{array}{cccccc|c} 1 & 0 & 0 & \dots & 0 & 0 & \\ 1 & 1 & 0 & \dots & 0 & 0 & \\ 0 & 1 & 1 & \dots & 0 & 0 & P^T R^T \\ \vdots & \ddots & \ddots & & \vdots & \vdots & \\ 0 & 0 & 0 & \dots & 1 & 1 & \end{array}\right)$, которая, очевидно, является разреженной. Распределение степеней символьных узлов соответствующего графа Таннера определяется матрицей R, т.е. повторителем.

10.3.5. Конструкция на базе кодов Рида-Соломона

Двоичные высокоскоростные регулярные коды МППЧ с большим минимальным расстоянием могут быть построены на базе $(N, 2, N-1)$ кодов Рида-Соломона над некоторым полем $\mathbb{F}_q$ [23]. Очевидно, что любые два кодовых слова кода Рида-Соломона с такими параметрами имеют не более чем один общий символ. Его кодовыми словами являются векторы, полученные путем вычисления значений многочленов вида $f(x) = f_1 x - f_0$ в различных элементах $\alpha \in \mathbb{F}_q \setminus \{0\}$. Пронумеруем все элементы $\tau \in \mathbb{F}_q$ натуральными числами и сопоставим каждому из них вектор e_τ длины q, содержащий единицу в позиции τ и нули в прочих позициях. Выберем M различных значений f_1 и для каждого из них выпишем все q кодовых слов $(f(\alpha_1), \dots, f(\alpha_n))$, соответствующих различным f_0, после чего заменим их элементы векторами $e_{f(\alpha_i)}$ и запишем их в виде $(Mq) \times (Nq)$ матрицы H. Очевидно, что при фиксированном f_1 значения $f_1\alpha_i - f_0$ пробегают вместе с f_0 все возможные элементы $\mathbb{F}_q$, поэтому матрица H получится блочно-перестановочной. Соответствующий ей граф Таннера свободен от циклов длины 4, так как если допустить их существование, то это означало бы наличие двух кодовых слов кода Рида-Соломона, совпадающих в двух позициях.

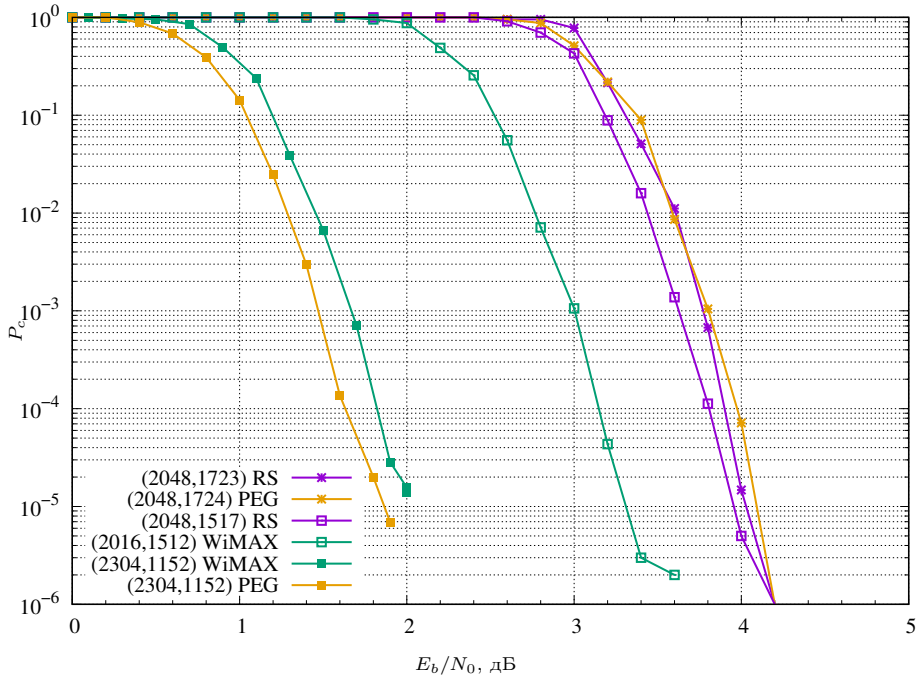


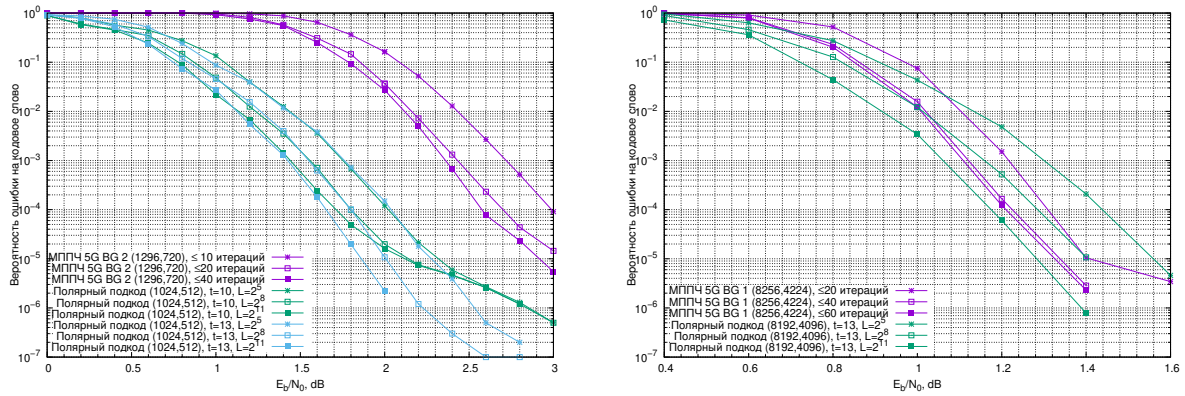
Рис. 10.7. Вероятность ошибки декодирования некоторых кодов МППЧ

В обширной литературе по МППЧ кодам можно встретить описание многих похожих конструкций. В работе [67] они были проанализированы с единых позиций.

10.3.6. Сравнительный анализ

На рис. 10.7 представлены результаты статистического моделирования для некоторых кодов МППЧ, полученных с помощью вышеописанных конструкций. Регулярные коды $(2048, 1723)$ и $(2048, 1517)$ были получены с помощью конструкции, основанной на кодах Рида-Соломона с параметрами $q = 64, N = 32, M = 6$ и $M = 14$ соответственно. Кривые с меткой WiMAX соответствуют нерегулярным кодам, построенным на основе протографов в соответствии со спецификацией IEEE 802.16e [39]. Кроме того, были рассмотрены регулярный $(2048, 1724)$ и нерегулярный $(2304, 1152)$ коды, построенные методом прогрессивного наращивания рёбер (PEG). Во всех случаях нерегулярные коды строились с использованием оптимизированных распределений степеней символьных узлов.

Из представленных результатов видно, что для незначительного снижения скорости регулярных кодов из семейства RS требуется значительное увеличение параметра M, т.е. числа строк в проверочной мат-



(a) (b)

Рис. 10.8. Вероятность ошибки декодирования некоторых кодов МППЧ: a — Длина ≈ 1024; b — Длина ≈ 8192

рице[5], которое определяет сложность декодирования. При этом, однако, корректирующая способность увеличивается незначительно. Намного лучшие результаты демонстрирует нерегулярный $(2016, 1512)$ код, построенный на основе протографа. Для кодов со скоростью $1/2$ случайный PEG-код демонстрирует заметно большую корректирующую способность по сравнению с кодом WiMAX, построенном с помощью протографовой конструкции, при вероятности ошибки более 10^{-5}, однако при дальнейшем увеличении отношения сигнал/шум скорость убывания вероятности ошибки декодирования этого кода резко падает, что может быть объяснено слишком малым его минимальным расстоянием. Аналогичное поведение демонстрирует и PEG код $(2048, 1724)$.

На рис. 10.8 представлены результаты статистического моделирования для кодов МППЧ из стандарта 5G. Эти коды строятся на основе протографов, причем для коротких и длинных кодов стандартом предусмотрены различные протографы. Для сравнения приведены результаты для полярных подкодов в узком смысле с близкими параметрами.

Видно, что на длине около 1024 полярные подкоды обеспечивают значительно лучшую корректирующую способность по сравнению с кодом МППЧ. Однако с увеличением длины корректирующая способность кодов МППЧ улучшается значительно быстрее по сравнению с полярным подкодами. Увеличение максимального числа итераций сверх 20 не приводит к существенному снижению вероятности ошибки, хотя даже при 60 итерациях алгоритм распространения доверия еще не реализует декоди-

[5]Следует отметить, что многие из них являются линейно зависимыми.

рование по максимуму правдоподобия рассмотренного кода. Код МППЧ с $n = 8256$, построенный с использованием иного базового графа, требует большего числа итераций для достижения близкой к предельной корректирующей способности по сравнению с коротким кодом. Для получения сопоставимой корректирующей способности полярный код требует значительного увеличения размера списка L в списочном/последовательном декодере. Стоит также отметить, что, как видно из рис. 10.8, b, среднее число итераций при декодировании кода МППЧ почти не зависит от максимального числа итераций при $E_b/N_0 \geq 1$ дБ. Это означает, что вероятность того, что декодеру потребуется большое число итераций, весьма мала. В то же время, средняя сложность декодирования полярного подкода существенно зависит от L, что свидетельствует о высокой вероятности того, что последовательному декодеру потребуется много итераций. Эти особенности декодеров должны учитываться при организации буферизации обрабатываемых данных в приемнике.

Упражнения

1. Пусть H — проверочная матрица, состоящая из всех возможных строк веса 2. Каково минимальное расстояние кода, определяемого ею?
2. Пусть дан многочлен $h(x) = 1 + x + x^3 + x^7$. Построим 15×15 проверочную матрицу H, строки которой представляют собой все возможные циклические сдвиги вектора его коэффициентов. Какой код определяется этой матрицей? Постройте граф Таннера, соответствующий ей, и найдите его обхват.

11. Коды для стирающего канала

11.1. Надежная доставка данных по ненадежным каналам

11.1.1. Потери в сетях с коммутацией пакетов

Большинство современных компьютерных сетей строится на базе технологий IP (Internet Protocol). Фундаментальным свойством этого протокола является отсутствие гарантий надежной доставки пакетов от отправителя к получателю. Среди причин потери пакетов можно отметить перегрузку промежуточных узлов, неисправимые (но почти всегда обнаружимые) искажения данных на физическом уровне, изменение таблиц маршрутизации, сбои оборудования, переполнение буферов операционной системы на узле-получателе и т.д. Исторически первые две причины, обусловленные слишком высокой скоростью генерации пакетов отправителем, являлись доминирующими. В связи с этим были разработаны протоколы, одновременно обеспечивающие надежную доставку данных и управление скоростью их передачи. Наиболее известным таким протоколом является TCP (Transmission Control Protocol). Принцип его работы состоит в разбиении передаваемого потока данных на последовательность сегментов, их передаче, ожидании подтверждения и, при необходимости, повторной передаче. Получатель имеет возможность подтвердить получение только непрерывного блока данных с начала до i-ого байта, даже если уже получены байты с $i + \delta'$ по $i + \delta''$. Передатчик может отправить W байт данных, не дожидаясь получения подтверждения. Если по истечении заданного интервала времени отправитель не получает подтверждения для переданных данных, все они, начиная с первого неподтвержденного, считаются утраченными и передаются заново (впрочем, этот процесс может быть остановлен своевременным получением подтверждения). Такая ситуация интерпретируется как следствие перегрузки одного из узлов сети, участвующего в передаче данных, и предпринимаются меры к быстрому снижению частоты отправки сегментов. Это является абсолютно бесполезным, если потеря пакета произошла вследствие ошибок на физическом уровне. Кроме того, для каналов с большим значением Rd, где R — максимальная скорость передачи данных по каналу и d — время прохождения пакета от отправителя к получателю, величина W, необходимая для обеспечения

непрерывной передачи, может оказаться слишком большой. Ввиду ограничений на объем памяти узла-отправителя ее приходится устанавливать многократно заниженной, что приводит к неэффективному использованию канала. В результате реальная скорость передачи данных с помощью протокола TCP зачастую оказывается многократно ниже R.

Проблема надежной доставки оказывается еще более сложной при необходимости организации мультикаста, т.е. передачи от одного источника к нескольким получателям. В этом случае пакеты, потерянные различными получателями, в большинстве случаев оказываются разными. В связи с этим при использовании протоколов с повтором передачи возникает необходимость или повторно (зачастую — многократно) произвести отправку всего блока данных (этот метод носит названия карусели данных), или организовать иерархию промежуточных узлов, каждый из которых обеспечивает надежную доставку пакетов к непосредственно связанным с ним узлам. Очевидно, что оба эти способа крайне сложны в реализации и неэффективно используют ресурсы сети.

11.1.2. Цифровой фонтан

Сеть с коммутацией пакетов можно рассматривать как 2^m-ичный стирающий канал, где m — размер пакета. Действительно, для того, чтобы обнаружить потерю пакета, достаточно лишь последовательно нумеровать их при передаче. Это позволяет воспользоваться результатами теории информации и теории кодирования для решения проблемы надежной доставки данных в условиях потерь пакетов. Однако для удобства реализации удобно считать, что каждый пакет состоит из единственного бита, предполагая, что операции, выполняемые при кодировании и декодировании, применяются одновременно ко всем битам, входящим в пакет. Тогда сеть может рассматриваться как двоичный стирающий канал.

Пусть p — вероятность потери пакета, т.е. стирания символа при передаче. Согласно (1.9), если k пакетов данных (символов) перед отправкой были закодированы оптимальным образом, для их восстановления получателю достаточно принять лишь $k(1+\epsilon)$ пакетов, причем ϵ стремится к нулю с ростом k. При этом не предполагается наличие какого-либо канала обратной связи и повторов утерянных пакетов. При размере алфавита q, стремящемся к бесконечности, этот подход может быть реализован с помощью $(n, k, n-k+1)$-кода Рида-Соломона над $\mathbb{F}_q$. Как было показано выше, любые k символов его кодового слова образуют ин-

формационную совокупность, и восстановление кодового слова сводится к задаче интерполяции. Однако с ростом размера алфавита возрастает сложность этой операции. Кроме того, величина p заранее неизвестна, что не позволяет подобрать длину кода $n \leq q$. Если q и n недостаточно велики, нестертых символов может не хватить для восстановления кодового слова; повтор ранее переданных символов не гарантирует возможности восстановления по любым k принятым пакетам.

В связи с этим возникает необходимость использования *бесскоростных* или *фонтанных* кодов. Модель цифрового фонтана предполагает, что отправитель преобразует имеющийся у него блок данных конечного размера в бесконечный поток кодовых символов, которые передаются по сети. Получатель или получатели могут воспользоваться любыми дошедшими до них пакетами (символами), рассматривая их как капли воды, которые они набирают в "подставленные в струю стаканы". Как только их "стаканы" наполнятся до краев (т.е. будет получено достаточно большое число нестертых символов), они могут воспользоваться известной процедурой декодирования и восстановить все данные. При этом желательно, чтобы процедуры кодирования и декодирования были вычислительно простыми. Фонтанные коды представляют собой множество потенциально бесконечных кодовых последовательностей. Однако при практической реализации предусматривается некоторая процедура (истечение тайм-аута, получение уведомления от принимающей стороны и т.п.), которая останавливает процесс порождения новых кодовых символов.

11.2. Конструкции кодов

11.2.1. Преобразование Луби

Рассмотрим следующий метод кодирования вектора[1] данных $u = (u_1, \ldots, u_k)$, известный как *преобразование Луби*:

1. Выбрать натуральное число d в соответствии с некоторым распределением $\rho(d)$.
2. Выбрать произвольные d символов данных, сложить их по модулю 2 и отправить получателю. Вместе с данными необходимо тем или иным способом передать номера тех символов, которые использовались на данном этапе.
3. Если это еще необходимо, перейти к шагу 1.

[1]Каждый элемент u может представлять собой вектор из l байт.

Множество полубесконечных последовательностей, которые могут быть порождены таким образом для фиксированного распределения $\rho(d)$, называется *LT-кодом*. Декодирование может быть выполнено путем решения системы линейных уравнений

$$uA = y, \tag{11.1}$$

где A — двоичная матрица такая, что $A_{ij} = 1$, если слагаемое u_i было использовано при построении j-ого принятого символа, и y — вектор, составленный из принятых символов. Декодирование является успешным, если эта система имеет единственное решение. Далеко не любая двоичная квадратная матрица является обратимой. Действительно, для того, чтобы $k \times k$ матрица была обратимой, ее первая строка не должна быть нулевой, вторая строка не должна быть нулевой и не должна совпадать с первой, i-я строка не должна быть линейной комбинацией предыдущих $i - 1$ строк. Таким образом, если считать, что все значения A_{ij} равновероятны, получим, что вероятность обратимости квадратной матрицы равна

$$P(k) = \prod_{i=1}^{k}(1 - 2^{i-k-1}) = \prod_{i=1}^{k}(1 - 2^{-i}).$$

При достаточно больших k эта величина приближенно равна 0,288. Естественно, что по мере увеличения числа столбцов (т.е. приема дополнительных кодовых символов) ранг матрицы A стремится к k, обеспечивая таким образом разрешимость системы уравнений. Однако для достижения пропускной способности двоичного стирающего канала необходимо минимизировать число избыточных символов, необходимых для гарантированного восстановления сообщения. В случае преобразования Луби минимизация должна проводиться по всем возможным распределениям $\rho(d)$. С точки зрения минимизации сложности вычислений на этапе кодирования и декодирования желательно найти такое распределение, чтобы большая часть кодовых символов имела небольшую степень.

Для того, чтобы решить эту задачу, рассмотрим субоптимальный метод решения системы (11.1), похожий на алгоритм распространения доверия, который был рассмотрен в разд. 10.2.2. Если удастся найти распределение, при котором этот алгоритм сможет восстанавливать данные по $k + \Delta$ принятым символам с небольшим запасом Δ, то декодер максимального правдоподобия (т.е. метод Гаусса решения системы (11.1)) тем более справится с этим. Впрочем, рассматриваемый далее алгоритм

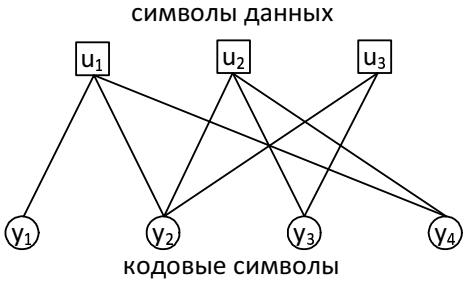


Рис. 11.1. Граф LT-кода

декодирования представляет самостоятельную ценность ввиду исключительной своей простоты.

Будем называть степенью принятого символа число слагаемых, использованных при его вычислении. Символ степени 1 ни в каких преобразованиях не нуждается, так как является символом данных, и при его получении может незамедлительно передаваться конечному потребителю. Но кроме того, его можно вычесть из других кодовых символов большей степени, при получении которых использовался этот же символ данных. В результате степень этих кодовых символов понижается, и у некоторых из них падает до единицы. Таким образом, возникает возможность восстановления все новых и новых символов данных. Но если на некотором шаге этого алгоритма не обнаруживаются кодовые символы степени 1, декодирование останавливается, хотя система уравнений (11.1) может быть разрешимой с точки зрения линейной алгебры. Будем называть волной декодирования множество кодовых символов степени 1, а сами такие символы — активными.

Пример 11.1. Рассмотрим LT-код с $k = 3$. Предположим, что получены четыре кодовых символа $y_1 = u_1, y_2 = u_1 + u_2 + u_3, y_3 = u_2 + u_3.y_4 = u_1 + u_2$. В графическом виде их взаимосвязь с символами данных можно представить так, как это показано на рис. 11.1. Первоначально волна декодирования включает только y_1. Символ u_1 может быть получен непосредственно как y_1. После его нахождения его можно вычесть из y_2 и y_4 и исключить из волны декодирования. Теперь $y_2 = u_2 + u_3, y_4 = u_2$ и волна содержит y_4. Найдя u_2, его можно вычесть из y_2 и y_3, которые заменяют его в волне декодирования. Таким образом, получены сразу две копии символа y_3.

11.2.2. Волновые распределения

Для того, чтобы выбрать оптимальное распределение $\rho(d)$, заметим, что если потери пакетов происходят независимо друг от друга, то распределения степеней принятых и переданных кодовых символов оди-

наковы. Распределение должно быть подобрано таким образом, чтобы волна декодирования не уменьшилась до нуля, так как это приведет к остановке процесса декодирования и может свидетельствовать о вырожденности матрицы A в (11.1). С другой стороны, волна не должна чрезмерно разрастаться, так как это приведет к тому, что после успешного окончания декодирования останется много бесполезных кодовых символов. Таким образом, новые кодовые символы должны входить в волну декодирования с той же скоростью, с которой символы, использованные для восстановления данных, из нее выбывают.

Попытаемся построить идеальное волновое распределение, которое удовлетворяло бы этому требованию, причем *средний* размер волны декодирования был бы равен единице. Это условие позволит минимизировать число избыточных символов. Однако если среднее случайной величины равно m, то она может принимать значения как большие, так и меньшие m. В данном случае это означает, что она может быть равной нулю. К сожалению, это происходит с вероятностью, близкой к единице, что делает идеальное распределение практически бесполезным. Однако методы, используемые при его анализе, позволят далее получить более работоспособное решение.

Будем считать одной итерацией декодера действия, необходимые для восстановления ровно одного символа данных. Тогда номер итерации t равен числу уже восстановленных символов данных. Пусть $h_t(d)$ — среднее число кодовых символов степени d на шаге t. Среднее число кодовых символов степени d, которые уменьшают свою степень до $d-1$ за одну итерацию, равно среднему числу ребер в графе (см. рис. 11.1), соединяющих узлы, соответствующие кодовым символам степени d, и еще не восстановленные символы данных, т.е. $h'_t(d) = h_t(d)d/(k-t)$. Так как необходимо обеспечить, чтобы среднее число кодовых символов степени 1 было равно 1 на каждой итерации, должны выполняться условия $h_0(1) = 1, h_1(1) = h'_0(2) = h_0(2)2/k = 1, h_2(1) = h'_1(2) = h_1(2)2/(k-1) = 1$ и т.д. Из этих соотношений можно получить *идеальное волновое распределение*

$$\rho(d) = \begin{cases} \frac{1}{k}, & d = 1 \\ \frac{1}{d(d-1)}, & 1 < d \leq d \end{cases}.$$

Убедимся в том, что оно действительно удовлетворяет вышесформулированным требованиям. Вероятность того, что кодовый символ станет активным на шаге $k - L$, когда остались непродекодированными L сим-

волов данных, равна $r(L) = \sum_{i=1}^{l} \rho(i) q(i, L)$, где $q(i, L)$ — аналогичная вероятность для кодового символа, изначально имевшего степень i. По определению активного узла $q(1, k) = 1$. При $2 \le i \le k, k-i+1 \ge L \ge 1$ $q(i, L)$ равна вероятности того, что $i-2$ символа данных, использованных кодером при порождении этого кодового символа (они соответствуют смежным узлам в графе), находятся среди $k-L-1$ ранее восстановленных символов, еще один такой символ восстанавливается на текущем шаге, и еще один остается среди невосстановленных символов данных, т.е.

$$q(i, L) = \frac{C_{k-L-1}^{i-2} C_L^1}{C_k^i} = \frac{i(i-1)L \prod_{j=0}^{i-3}(k-L-1-j)}{\prod_{j=0}^{i-1}(k-j)}.$$

Во всех остальных случаях $q(i, L) = 0$. Таким образом, $r(k) = \frac{1}{k}$, а при $L < k$ получим

$$\begin{aligned} r(l) &= \sum_{i=2}^{k} \frac{L \prod_{j=0}^{i-3}(k-L-1-j)}{\prod_{j=0}^{i-1}(k-j)} = \\ &= \frac{L}{\prod_{j=0}^{k-1}(k-j)} \sum_{i=2}^{k} \left(\prod_{j=0}^{i-3}(k-L-1-j) \prod_{j=i}^{k-1}(k-j) \right) = \\ &= \frac{L}{k!} \sum_{i=2}^{k-L+1} \frac{(k-i)!(k-L-1)!}{(k-L-i+1)!} = \\ &= \frac{L!(k-L-1)!}{k!} \sum_{j=0}^{k-L-1} C_{L-1+j}^{L-1} = \frac{L!(k-L-1)!}{k!} C_{k-1}^{L} = \frac{1}{k}. \end{aligned}$$

Таким образом, если число принятых символов равно k, на каждом шаге в среднем будет активироваться ровно $r(L)k = 1$ кодовый символ. Так как на начальном этапе среднее значение размера волны равно $\rho(1)k = 1$, в среднем ее размер останется далее равен этой величине.

Для того, чтобы с достаточно большой вероятностью $1-\delta$ размер волны декодирования не упал до нуля, необходимо создать достаточно большой запас кодовых символов степени 1. Это реализуется с помощью *устойчивого волнового распределения* $\mu(d) = \frac{\rho(d)+\tau(d)}{\beta}$, где

$$\tau(d) = \begin{cases} R/(dk), & 1 \le d \le k/R - 1 \\ R \ln(R/\delta)/k, & d = k/R \\ 0, & k/R + 1 \le d \le k, \end{cases}$$

$R = c\ln(k/\delta)\sqrt{k}$, $c \approx 1$ и $\beta = \sum_{i=1}^{k}(\rho(i) + \tau(i))$. *Теорема Луби*, доказательство которой приведено в [49], утверждает, что использование этого распределения позволяет с вероятностью $1 - \delta$ произвести восстановление данных по любому множеству $K > k$ принятых пакетов, и

$$\frac{K-k}{k} \sim \frac{\ln^2(k/\delta)}{\sqrt{k}} \xrightarrow[k\to\infty]{} 0. \tag{11.2}$$

При этом среднее число арифметических операций, требуемых для кодирования и декодирования данных, составляет $O(k\ln(k/\delta))$. Таким образом, LT-коды позволяют достичь пропускной способности двоичного стирающего канала.

11.2.3. Хищные коды

К сожалению, для сравнительно небольших значений k и малых вероятностей отказа δ избыточность LT-кодов, задаваемая (11.2), оказывается слишком большой. Для преодоления этой проблемы в работе [63] были предложены *хищные коды* (raptor codes, raptor - сокращение от Rapid Tornado, торнадо-коды являются другим классом кодов, исправляющих стирания). Основная идея этого подхода состоит в том, что если сначала закодировать данные каким-либо линейным блоковым кодом (например, БЧХ или МППЧ), способным исправлять достаточно большое число стираний, а затем подвергнуть кодовое слово этого кода преобразованию Луби, то декодер внутреннего LT-кода может отказаться от восстановления некоторого числа символов данных. Эти символы могут быть восстановлены с помощью декодера внешнего кода. Впрочем, на практике приходится использовать декодирование по максимуму правдоподобия, так как LT-декодер не способен восстановить многие исправимые конфигурации стираний. Оно сводится к решению системы уравнений $uGA = y$, где G — порождающая матрица внешнего кода и A — матрица преобразования Луби. При этом особую важность приобретает построение быстрых алгоритмов решения этой системы. Существует возможность построить реализацию алгоритма Гаусса, которая использовала бы разреженную структуру матрицы A для ускорения вычислений [45].

Часть IV

Составные коды

12. Методы комбинирования кодов

Задача построения хороших длинных кодов, вообще говоря, является достаточно нетривиальной. Ее можно решать, комбинируя и модифицируя уже построенные хорошие коды. Многие известные на сегодняшний день оптимальные конструкции получены с помощью методов, рассматриваемых в данном разделе.

12.1. Простые преобразования кодов

12.1.1. Укорочение линейных блоковых кодов

При использовании линейного (n, k, d) кода необязательно кодировать ровно k символов передаваемых данных. Если необходимо передать меньшее их количество, недостающие символы можно положить равными нулю. Как было показано в разд. 2.2.1, для всякого линейного кода можно получить порождающую матрицу (возможно, с переставленными столбцами) вида $G = (I|A)$. Если часть кодируемых символов всегда равна нулю, то при использовании такой порождающей матрицы некоторые символы кодового слова также всегда будут равны нулю. Так как они не несут никакой полезной информации, при передаче их можно пропустить. Это эквивалентно использованию кода с порождающей матрицей, полученной путем вычеркивания части строк и соответствующих им столбцов матрицы G. Ясно, что такое преобразование приводит к $(n - \delta, k - \delta, d)$ коду, где δ — количество вычеркиваемых строк.

Пример 12.1. Рассмотрим двоичный $(7, 4, 3)$ код эквивалентный коду Хэмминга, с порождающей матрицей

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}. \tag{12.1}$$

Вычеркивая последние две строки, обнаружим, что третий и четвертый столбцы полученной матрицы равны нулю и потому тоже могут быть вычеркнуты. В результате получим матрицу $G' = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 \end{pmatrix}$, порождающую $(5, 2, 3)$ код.

12.1.2. Выкалывание

Часть символов кодового слова при передаче может быть пропущена. Это приведет к уменьшению длины кода на m и минимального

расстояния не более чем на m, где m — число выкалываемых символов. В случае линейных блоковых кодов с проверочной матрицей вида $H = (A|I)$ это преобразование может быть описано как удаление m столбцов единичной подматрицы и соответствующих им (т.е. содержащих единицы в удаленных столбцах) строк. При декодировании выколотые символы должны рассматриваться как стертые.

12.1.3. Расширение

Если минимальное расстояние d двоичного линейного блокового кода нечетно, то оно может быть увеличено на единицу путем добавления к каждому кодовому слову одного символа, равного сумме всех остальных (символ проверки на четность). Действительно, в этом случае все кодовые слова веса d увеличат свой вес на единицу, т.е. минимальное расстояние кода станет равным $d+1$. Соответствующая порождающая матрица получается путем дописывания к каждой строке исходной порождающей матрицы символа проверки на четность, а проверочная матрица получается путем добавления одной строки, содержащей все единицы, и одного столбца, содержащего нули во всех позициях, кроме той, которая соответствует вновь добавленной строке.

Пример 12.2. Проделав описанные преобразования с проверочной матрицей кода Хэмминга (2.6), получим проверочную матрицу *расширенного* $(8, 4, 4)$ *кода Хэмминга*

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

12.2. Составные коды

12.2.1. Чередование кодов

Пусть даны два блоковых кода $\mathcal{C}_1$ и $\mathcal{C}_2$ с параметрами $(n_1, k_1, d_1), (n_2, k_2, d_2)$ над одним и тем же алфавитом. Их чередованием называется код $\mathcal{C} = \{(c_1, c_2)|c_1 \in \mathcal{C}_1, c_2 \in \mathcal{C}_2\}$. Параметры полученного кода имеют вид $(n_1 + n_2, k_1 + k_2, \min(d_1, d_2))$, порождающая и проверочная матрицы могут быть записаны как $G = \begin{pmatrix} G_1 & 0 \\ 0 & G_2 \end{pmatrix}$, $H = \begin{pmatrix} H_1 & 0 \\ 0 & H_2 \end{pmatrix}$.

12.2.2. Прямая сумма кодов

Пусть даны два блоковых кода $\mathcal{C}_1$ и $\mathcal{C}_2$ одинаковой длины над одним и тем же алфавитом. Их прямой суммой называется код $\mathcal{C} = \{c_1 + c_2|c_1 \in \mathcal{C}_1, c_2 \in \mathcal{C}_2\}$. В случае линейных кодов параметры получен-

ного кода имеют вид $(n, k_1 + k_2, d \leq \min(d_1, d_2))$, а порождающая матрица равна $G = \begin{pmatrix} G_1 \\ G_2 \end{pmatrix}$. Точное значение минимального расстояния и структура проверочной матрицы зависят от конкретных используемых кодов.

12.2.3. Конструкция Плоткина

Пусть даны линейные коды $\mathcal{C}_1, \mathcal{C}_2$ с параметрами (n, k_1, d_1) и (n, k_2, d_2). Построим новый код как $\mathcal{C} = \{(u, u + v) | u \in \mathcal{C}_1, v \in \mathcal{C}_2\}$. Очевидно, что он также является линейным, его длина равна $2n$, а размерность — $k_1 + k_2$. Такой метод построения известен как конструкция Плоткина или конструкция $(u|u + v)$. Порождающая матрица полученного таким образом кода имеет вид $G = \begin{pmatrix} G_1 & G_1 \\ 0 & G_2 \end{pmatrix}$.

Теорема 12.1. *Минимальное расстояние кода $\mathcal{C}$, полученного с помощью конструкции Плоткина из кодов $\mathcal{C}_1(n, k_1, d_1)$ и $\mathcal{C}_2(n, k_2, d_2)$, равно* $\min(2d_1, d_2)$.

Доказательство. Отметим, что код $\mathcal{C}$ содержит слова (u, u) и $(0, v)$, где c_i — ненулевые слова минимального веса кодов $\mathcal{C}_i, i = 1, 2$, поэтому его минимальное расстояние не может быть больше $\min(2d_1, d_2)$. Пусть c_i, c_i' — ненулевые слова кодов $\mathcal{C}_i$ такие, что $(u, u + v) \neq (u', u' + v')$. Тогда $D = d_H((u, u + v), (u', u' + v')) = d_H(u, u') + d_H(u + v, u' + v')$. Если $v = v'$, то $D = 2d_H(u, u') \geq 2d_1$. В противном случае, воспользовавшись неравенством треугольника в форме $d_H(a, c) \geq d_H(a, b) - d_H(c, b)$, получим $D \geq d_H(u, u') + d_H(u + v, u + v') - d_H(u' + v', u + v') = d_H(u, u') + d_H(v, v') - d_H(u', u) = d_H(v, v') \geq d_2$. □

Пример 12.3. Попытаемся построить двоичный $(10, 3, 5)$ код. Он является оптимальным, так как в силу границы Грайсмера $N(3, 5) \geq 5 + 3 + 2 = 10$. Воспользуемся $(7, 4, 3)$ кодом эквивалентным коду Хэмминга (см. (12.1)), укоротив его до кода $\mathcal{C}_1(6, 3, 3)$, а также кодом с повторениями $\mathcal{C}_2(6, 1, 6)$. Комбинируя их с помощью конструкции Плоткина, получим код $\mathcal{C}'(12, 4, 6)$ с порождающей матрицей $G' = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$. Выкалывая один символ (последний), получим код $\mathcal{C}'(11, 4, 5)$. Укорачивая его на один символ (первый), получим искомый код $\mathcal{C}(10, 3, 5)$ с порождающей матрицей $G = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$.

Пусть $(y_1, y_2) = (u, u+v) + (e_1, e_2)$ — вектор, соответствующий кодовому слову кода $\mathcal{C} \subset F_q^{2n}$, переданному по q-ичному симметричному каналу. Предположим, что известны алгоритмы декодирования кодов $\mathcal{C}_i$ в метрике Хэмминга, способные исправлять $\lfloor (d_i - 1)/2 \rfloor$ ошибок. Тогда декодирование кода $\mathcal{C}$ может быть выполнено следующим образом:

1. Вычислить $y_2' = y_2 - y_1 = v + e_2 - e_1$.
2. Продекодировать y_2' в коде $\mathcal{C}_2$. Пусть $\widehat{v}$ — кодовое слово, найденное при этом. Если число произошедших ошибок t не превосходит $\lfloor (d-1)/2 \rfloor$, где $d = \min(2d_1, d_2) \leq d_2$, декодер кода $\mathcal{C}_2$ сможет исправить вектор ошибки $e_2 - e_1$ в силу того, что $\mathrm{wt}_H(e_2 - e_1) \leq \mathrm{wt}_H(e_1) + \mathrm{wt}_H(e_2) = t$.
3. Вычислить $y_1' = y_2 - \widehat{v} = u_1 + e_2$ (последнее равенство верно только в том случае, если декодирование на предыдущем шаге было успешным). Продекодировать y_1 и y_1' в коде $\mathcal{C}_1$. Пусть u', u'' — найденные при этом кодовые слова. В качестве окончательного решения $\widehat{u}$ выбирается то из этих кодовых слов, для которого $d_H((\widehat{u}, \widehat{u}), (y_1, y_2 - \widehat{v}))$ является минимальным. Если число произошедших ошибок t не превосходит $\lfloor (d-1)/2 \rfloor \leq \lfloor (2d_1 - 1)/2 \rfloor$, по крайней мере одна из этих попыток декодирования будет успешной, так как если $\mathrm{wt}_H(e_1) > (d_1 - 1)/2$, то $\mathrm{wt}_H(e_2) = t - \mathrm{wt}_H(e_1) < \lfloor (2d_1 - 1)/2 \rfloor - (d_1 - 1)/2 \leq \lfloor (d_1)/2 \rfloor$.

12.3. Каскадные коды

12.3.1. Прямое произведение кодов

Пусть даны коды $\mathcal{C}_1(n_1, k_1, d_1), \mathcal{C}_2(n_2, k_2, d_2)$ над $\mathbb{F}_q$. Закодируем k_2 блоков данных размером по k_1 символов кодом $\mathcal{C}_1$ и расположим полученные кодовые слова в виде строк $k_2 \times n_1$ матрицы. Закодируем далее каждый ее столбец кодом $\mathcal{C}_2$ и выпишем все элементы матрицы в виде одного вектора. Множество $\mathcal{C}$ получаемых таким образом векторов длины $n_1 n_2$ называется прямым произведением кодов $\mathcal{C}_1$ и $\mathcal{C}_2$. Очевидно, что $\mathcal{C}$ является линейным блоковым кодом с размерностью $k_1 k_2$. Его порождающая матрица может быть получена как кронекеровское произведение[1] порождающих матриц компонентных кодов. В силу линейности не имеет значения, в каком порядке производится кодирование: вначале по строчкам, а затем по столбцам, или наоборот.

[1]Кронекеровским произведением матриц A и B называют блочную матрицу $A \times B$, полученную заменой всех элементов A_{ij} на матрицы $A_{ij}B$.

Рис. 12.1. Каскадный код

Теорема 12.2. *Минимальное расстояние прямого произведения* $\mathcal{C}$ *кодов* $\mathcal{C}_1(n_1, k_1, d_1)$ *и* $\mathcal{C}_2(n_2, k_2, d_2)$ *равно* $d_1 d_2$.

Доказательство. Рассмотрим два различных кодовых слова $\mathcal{C}$, представленных в виде $n_2 \times n_1$ матриц. Те строки, в которых они не совпадают, отличаются друг от друга не менее чем в d_1 позициях, т.е. в матрице имеется не менее чем d_1 различных столбцов. Но столбцы являются кодовыми словами кода $\mathcal{C}_2$ и различаются не менее чем в d_2 позициях, т.е. общее число несовпадающих символов не может быть меньше $d_1 d_2$.

С другой стороны, для того, чтобы построить кодовое слово веса $d_1 d_2$, можно взять кодовые слова $c^{(i)} \in \mathcal{C}_i$ веса $d_i, i = 1, 2$, и построить матрицу с элементами $c_{rs} = c_r^{(2)} c_s^{(1)}, 1 \le r \le n_2, 1 \le s \le n_1$, которая является кодовым словом кода-произведения. □

Декодирование прямого произведения кодов является достаточно нетривиальной задачей. Отметим лишь, что непосредственное применение декодеров компонентных кодов, способных исправлять $\lfloor (d_i - 1)/2 \rfloor$ ошибок в строчках и столбцах позволяет исправить лишь $\lfloor (d_1 - 1)/2 \rfloor \lfloor (d_2 - 1)/2 \rfloor$ ошибок во всем кодовом слове, что много меньше $\lfloor (d_1 d_2 - 1)/2 \rfloor$, радиуса однозначного декодирования кода-произведения.

12.3.2. Каскадные коды

Пусть даны (N, K, D) код $\mathcal{C}_1$ над $\mathbb{F}_{q^k}$ и (n, k, d) код $\mathcal{C}_2$ над $\mathbb{F}_q$. Рассмотрим кодирование блока символов $\mathbb{F}_q$ длиной Kk. Сгруппируем вначале символы в подблоки $(u_{i1}, \ldots, u_{ik}), 1 \le i \le K$, и представим их в виде элементов $u_i = \sum_{j=1}^{k} u_{ij} \beta_j$ поля $\mathbb{F}_{q^k}$, где $\beta_1, \ldots, \beta_k$ — некоторый его базис. Закодируем далее символы u_i кодом $\mathcal{C}_1$. Символы полученного кодового слова $(C_1, \ldots, C_N)$ разложим по этому же базису как $C_i = \sum_{j=1}^{k} C_{ij} \beta_j$. Блоки символов $(C_{i1}, \ldots, C_{ik})$ закодируем кодом $\mathcal{C}_2$ (см. рис. 12.1). Таким образом, будет получен (Nn, Kk, Dd) код. $\mathcal{C}_1$ и $\mathcal{C}_2$ называют, соответственно, внешним и внутренним кодами. Короткий

код $\mathcal{C}_2$ обычно не представляет труда продекодировать по максимуму правдоподобия, несколько понизив таким образом вероятность ошибки по сравнению с изначально наблюдаемой в канале. Это создает более благоприятные условия для работы декодера внешнего кода, в качестве которого обычно используют код Рида-Соломона или его аналоги, декодирование которых по максимуму правдоподобия является практически неразрешимой задачей.

Иногда в качестве внутреннего кода используется сверточный код, причем все символы C_{ij} обрабатываются как единый поток. В случае ошибки при его декодировании может появиться достаточно длинная последовательность неправильных оценок $\widehat{C}_{ij}$, которая превратится в несколько неправильных $\widehat{C}_i$. Несколько таких событий может привести к тому, что число неправильных $\widehat{C}_i$ превысит корректирующую способность внешнего кода, и произойдет ошибка его декодирования. Для борьбы с этим явлением может использоваться перемежение. Более конкретно, на этапе кодирования накапливается несколько кодовых слов внешнего кода, их символы некоторым предопределенным образом перемешиваются, после чего кодируются внутренним кодом. В этом случае ошибки, порожденные декодером внутреннего кода, оказываются рассредоточены по нескольким кодовым словам внешнего кода, что повышает вероятность успешного декодирования каждого из них.

13. Турбо-коды

13.1. Конструкция

13.1.1. Мотивация

Методы решения задачи надежной передачи информации, рассматривавшиеся во всех предшествующих главах, сводились к построению некоторого кода с определенными алгебраическими свойствами, для которого затем формулировались различные алгоритмы декодирования, использующие эти свойства. При этом ставилась задача максимизировать минимальное расстояние полученного кода, а также обеспечить исправление гарантированного числа ошибок в метрике Хэмминга. Приведенные результаты статистического моделирования свидетельствуют о том, что большинство таких алгоритмов не обеспечивает декодирование по максимуму правдоподобия, а вероятность ошибки при декодировании по максимуму правдоподобия весьма далека от оптимальной.

В 1993 году произошла революция в теории помехоустойчивого кодирования. В работе [18] был предложен противоположный подход к решению этой задачи: на основе идей, давно нашедших свое применение в теории и практике систем автоматического управления, многомерной оптимизации и статистической физике, был сформулирован алгоритм декодирования с использованием обратной связи (турбо-декодирование), а также предложены коды, которые могут быть продекодированы с помощью этого алгоритма. В отличие от рассмотренных выше строгих алгебраических методов, конструкция турбо-кодов носит во многом эвристический характер. Несмотря на это, ее использование впервые позволило приблизиться к пределу Шеннона, хотя с точки зрения классической теории кодирования эти коды оказываются весьма плохими. Тем не менее, последняя особенность ограничивает использование турбо-кодов и их аналогов в системах, требующих особо надежной помехозащиты, так как приближение к пределу Шеннона достигается только в области малых отношений сигнал/шум (до вероятности ошибки на кодовое слово порядка 10^{-5}), в то время как дальнейшее его увеличение приводит к крайне медленному снижению вероятности ошибки декодирования.

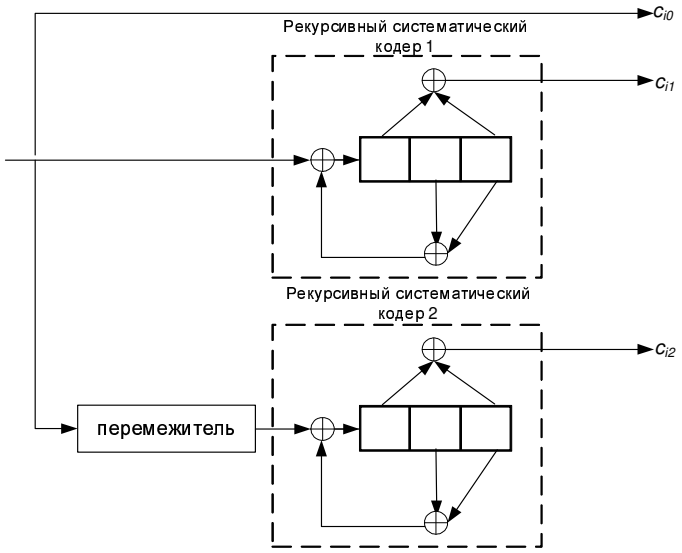


Рис. 13.1. Пример турбо-кодера

13.1.2. Кодирование

Структура классического кодера *турбо-кода* представлена на рис. 13.1. Он состоит из двух кодеров рекурсивных систематических сверточных кодов и перемежителя. Отметим, что здесь под кодером сверточного кода понимается только его часть, связанная с регистром сдвига (сравните с рис. 9.1, b), в то время как выход, дублирующий информационные символы, вынесен отдельно. Перемежитель представляет собой устройство, которое некоторым предопределенным образом переставляет символы в поданном на его вход блоке данных. Таким образом, в отличие от сверточных кодов, которые в принципе могут порождать кодовые последовательности произвольной длины, турбо-код является блоковым кодом. Приведенная на рисунке структура является примерной; допускается использование большего числа компонентных сверточных кодеров, причем их порождающие многочлены могут быть различными. Могут использоваться и (n_0, k_0)-сверточные коды с $k_0 > 1$. Очевидно, что турбо-коды являются линейными.

Как было отмечено в разд. 9.1.1, для снижения вероятности ошибки декодирования целесообразно после обработки всех кодируемых символов перевести кодер сверточного кода в нулевое состояние, что требует подачи на его вход некоторой последовательности фиктивных информационных символов. В случае рекурсивного систематического сверточного кода эта последовательность равна выходу регистра сдвига с линей-

ной обратной связью. В силу того, что информационные последовательности, обрабатываемые компонентными кодерами, входящими в состав турбо-кодера, различаются (пусть и на фиксированную перестановку), невозможно подобрать такую последовательность информационных символов, которая приводила бы в нулевое состояние оба сверточных кодера. Поэтому обычно обнуление регистра сдвига производится только для одного из них, а конечные состояния прочих полагаются при декодировании равновероятными.

Использование рекурсивных систематических сверточных кодеров обусловлено тем, что они отображают многие информационные последовательности конечного веса на кодовые последовательности бесконечного (на практике — большого) веса. При это благодаря наличию перемежителя *с большой вероятностью* информационная последовательность, которая все же была преобразована одним из кодеров в кодовую последовательность конечного и малого веса, после перестановки символов и кодирования другим сверточным кодером отобразится в последовательность большого веса.

При декодировании декодеры компонентных кодов обмениваются друг с другом информацией, представляющей собой оценки информационных символов. При этом, как было показано в разд. 9.2.3, ошибка декодирования сверточного кода приводит к возникновению пакета ошибок на выходе декодера. Таким образом, чем больше минимальное свободное расстояние компонентных кодов, тем больше неправильных "подсказок" может передать один декодер другому. В связи с этим при построении турбо-кодов используются сверточные коды с небольшой длиной кодового ограничения. При этом использование перемежителя позволяет при декодировании разнести ошибочные "подсказки", возникшие вследствие отдельно взятой ошибки декодирования, по всей длине кодового слова, снижая таким образом вероятность того, что они приведут к ошибке декодирования другого компонентного кода.

13.1.3. Построение перемежителя

Задачей перемежителя является перестановка символов в блоке данных, поданном на его вход. При этом символы, находящиеся изначально в близко расположенных позициях, должны перемещаться в максимально удаленные друг от друга позиции. Это можно формально записать как

$$0 < |i - j| < d \Rightarrow |\pi(i) - \pi(j)| \geq S, \tag{13.1}$$

где π — перестановка, реализуемая перемежителем. Помимо параметров d и S, с практической точки зрения представляют важность такие характеристики, как объем оперативной памяти и задержка, требуемые для реализации перестановки.

Простейший подход к построению перемежителя состоит в случайной генерации перестановок с отбрасыванием тех из них, которые не удовлетворяют (13.1) при заданных d и S. Недостатком такого подхода является большой объем памяти, требуемый для хранения полученной перестановки.

Табличный перемежитель исключительно прост в реализации и приводит к турбо-кодам с хорошими характеристиками. Для реализации соответствующей перестановки используется $d \times d$ таблица. $k = d^2$ информационных символов записываются в нее построчно слева направо сверху вниз, после чего считываются по столбцам, причем считывание ведется снизу вверх слева направо. Можно показать, что при этом достигается $S = d$.

Наилучшие из известных турбо-кодов были получены с помощью перемежителя, реализующего отображение $Q(i) \to Q(i+1) \bmod k, 0 \leq i < k$, где $Q(x)$ — перестановочный многочлен, т.е. такой полином, что $\forall i \exists j : i \equiv Q(j) \bmod k$. В качестве такого многочлена может быть выбран $Q(x) = l\frac{x(x+1)}{2} \bmod k$, где l — нечетное число [64].

13.2. Декодирование

13.2.1. Турбо-декодер

Основной идеей турбо-декодирования является поочередное декодирование компонентных сверточных кодов, причем результаты декодирования одного кода являются “подсказками” для другого декодера. Этот процесс повторяется несколько раз и при удачном стечении обстоятельств приводит к кодовому слову турбо-кода, являющемуся наиболее вероятным для данной принятой последовательности. Впрочем, иногда декодер сходится к некоторой последовательности, не являющейся кодовым словом, или входит в колебательный режим. Существенным недостатком описанного алгоритма турбо-декодирования является сложность обнаружения таких событий[1]. Тем не менее, в большинстве случаев этот

[1]Напомним, что описанные выше алгоритмы декодирования кодов БЧХ фиксируют отказ от декодирования в том случае, если число различных корней многочлена локаторов ошибок меньше его степени, а итеративный алгоритм декодирования кодов

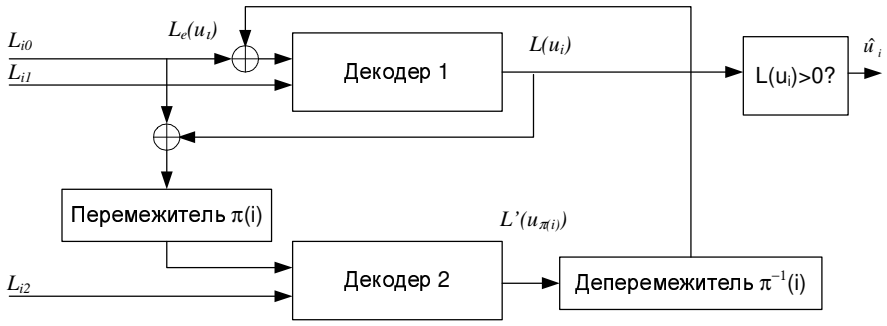


Рис. 13.2. Структура турбо-декодера

алгоритм обеспечивает декодирование почти по максимуму правдоподобия.

Структурная схема алгоритма декодирования приведена на рис. 13.2. Исходными данными декодера являются логарифмические отношения правдоподобия $L_{ij} = \ln \frac{P\{c_{ij}=1|r_{ij}\}}{P\{c_{ij}=0|r_{ij}\}}, 1 \le i \le k+W, 1 \le j \le 3$, всех символов принятой последовательности. Для определенности будем считать, что декодирование компонентных кодов осуществляется с помощью алгоритма Бала-Коке-Елинека-Равива. Напомним, что он имеет возможность (см. (9.6)) учитывать априорную информацию о вероятности того, что информационные символы u_i равны нулю или единице. Первоначально величины L_{i0}, L_{i1} обрабатываются декодером первого компонентного кода, причем $L_e(u_i)$ в (9.6) полагается равным нулю. Величины $L(u_i)$, вычисленные в соответствии с (9.3), суммируются[2] с L_{i0} и после перестановки обрабатываются декодером второго компонентного кода совместно с L_{i2}. Вычисленные им величины $L'(u_{\pi(i)})$ подвергаются обратной перестановке и рассматриваются в качестве $L_e(u_i)$. Эти действия выполняются несколько раз (обычно 5–6), после чего принимаются решения относительно каждого из информационных символов u_i.

МППЧ включает проверку на равенство нулю синдрома восстановленного кодового слова.

[2]Для упрощения описания воспользуемся тем, что все компонентные коды являются рекурсивными систематическими. Это позволяет сгруппировать $L_e(u_i)$ с соответствующим L_{i0}.

14. Кодовая модуляция

14.1. Решетчато-кодовая модуляция

14.1.1. Сигнальные множества

Для обеспечения высокой спектральной эффективности систем передачи информации необходимо использовать большие сигнальные множества $\mathbb{X}$. Если не используется помехоустойчивое кодирование, один символ M-ичного сигнального множества позволяет представить $\log_2 M$ битов полезной информации. Если время передачи одного символа остается фиксированным, это позволяет в $\log_2 M$ раз увеличить скорость передачи данных по сравнению с простейшей двоичной модуляцией ($\mathbb{X} = \left\{-\sqrt{E_s}, \sqrt{E_s}\right\}$), рассмотренной в разд. 1.2.4. Наибольшее распространение получили следующие методы цифровой модуляции:

1. *Амплитудно-импульсная модуляция* (рис. 14.1, a). Сигнальное множество имеет вид

$$\mathbb{X} = \{a_i = (2i + 1 - M)d \in \mathbb{R} \,|\, i \in \{0, 1, \ldots, M - 1\}\}.$$

Средняя энергия сигнала M-АМ равна

$$E_s = \frac{d^2}{M} \sum_{i=0}^{M-1} (2i + 1 - M)^2 = \frac{d^2}{3}(M^2 - 1).$$

Вероятность ошибки на символ при передаче данных без кодирования по аддитивному гауссовскому каналу может быть найдена как

$$P_{M\text{-АМ}} = \frac{2M - 2}{M} Q\left(\sqrt{\frac{6E_s/N_0}{M^2 - 1}}\right) = \frac{2(M - 1)}{M} Q\left(\sqrt{\frac{6\log_2 M}{M^2 - 1}\frac{E_b}{N_0}}\right)$$

2. *Фазовая модуляция* (рис. 14.1, b). Сигнальное множество имеет вид

$$\mathbb{X} = \left\{a_i = e^{\mathbf{j}\,2\pi\frac{i}{M}} \in \mathbb{C} | i \in \{0, 1, \ldots, M - 1\}\right\}.$$

Средняя энергия сигнала M-ФМ равна $E_s = 1$. Вероятность ошибки на символ при передаче данных без кодирования по аддитивному гауссовскому каналу может быть найдена как

$$P_{M\text{-ФМ}} = 1 - \int_{-\pi/M}^{\pi/M} \frac{e^{-E_s/N_0 \sin^2\Theta}}{\pi} \int_0^\infty u e^{-(u - \sqrt{E_s/N_0}\cos\Theta)^2} du d\Theta$$

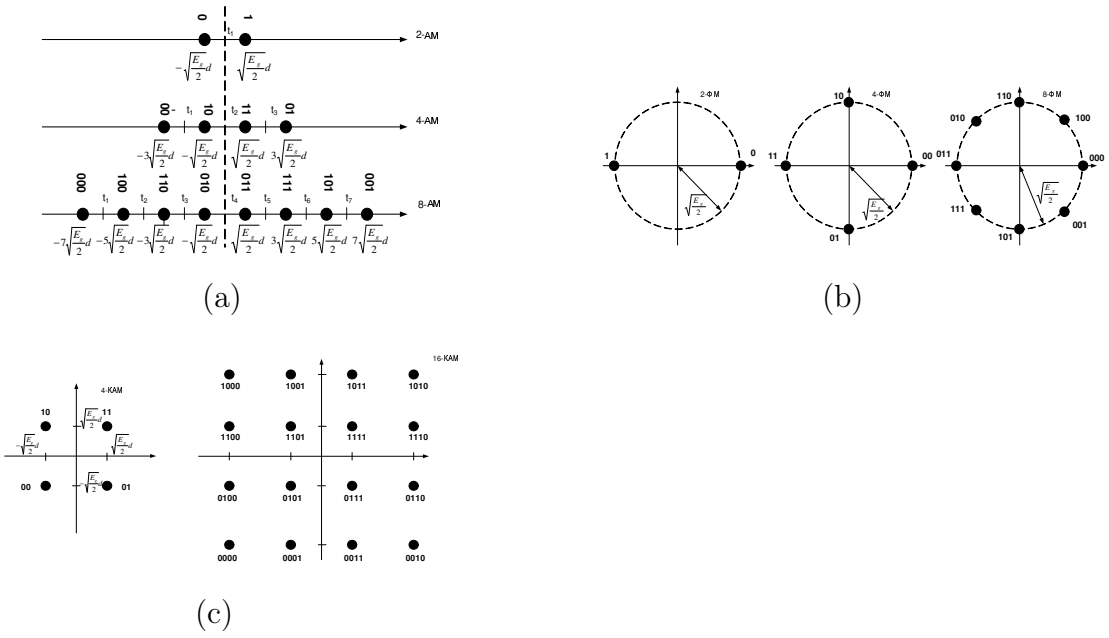


Рис. 14.1. Пространственные диаграммы сигналов различных методов модуляции:
a — амплитудно-импульсная модуляция; b — фазовая модуляция; c — квадратурно-амплитудная модуляция

3. *Квадратурно-амплитудная модуляция* (рис. 14.1, c). Если M является полным квадратом, то сигнальное множество имеет вид

$$\mathbb{X} = \left\{ a_i + a_j \, \mathbf{j} \in \mathbb{C} | a_i = (2i + 1 - \sqrt{M})d, i \in \left\{ 0, 1, \ldots, \sqrt{M} - 1 \right\} \right\}.$$

Средняя энергия сигнала M-КАМ равна $E_s = 2\frac{d^2}{3}(M^2 - 1)$. Вероятность ошибки на символ при передаче данных без кодирования по аддитивному гауссовскому каналу может быть найдена как

$$\begin{aligned} P_{M\text{-КАМ}} &= 1 - \left(1 - 2(1 - \frac{1}{\sqrt{M}}) Q \left(\sqrt{\frac{3E_s/N_0}{M-1}} \right) \right)^2 \\ &\approx 4Q \left(\sqrt{\frac{3 \log_2 M}{M-1} \frac{E_b}{N_0}} \right) \end{aligned}$$

Отметим, что вещественная и мнимая компоненты сигнала квадратурно-амплитудной модуляции фактически представляют собой сигналы амплитудно-импульсной модуляции.

Как правило, параметр d вышеприведенных сигнальных множеств выбирается таким образом, чтобы обеспечить фиксированную (обычно —

равную 1) среднюю энергию сигнала E_s. Повышения спектральной эффективности системы связи, т.е. увеличение числа битов полезной информации, передаваемых за один символьный интервал, требует использования сигнальных множеств большей размерности M. Заметим, что при фиксированном E_s увеличение M приводит к уменьшению расстояния Евклида между элементами сигнального множества. Это приводит к быстрому увеличению вероятности ошибки декодирования. Для предотвращения этого можно закодировать данные каким-либо корректирующим кодом, описанным в предыдущих главах. При этом придется обеспечить передачу дополнительных проверочных символов, что при фиксированной скорости передачи полезных данных может быть реализовано или за счет уменьшения длительности символьного интервала (т.е. времени, в течение которого осуществляется передача одного символа), или дополнительного увеличения сигнального множества. Уменьшение длительности символьного интервала приводит к снижению отношения сигнал/шум и увеличению полосы частот, занимаемых системой, что зачастую недопустимо. Увеличение сигнального множества может привести к еще большему росту вероятности ошибки, которое не всегда может быть скомпенсировано за счет использования корректирующих кодов. Таким образом, разработка методов помехоустойчивого кодирования для данного класса систем, называемых ограниченными по полосе, является нетривиальной задачей, которая может быть сформулирована как построение кодов в метрике Евклида. Все коды, рассматривавшиеся выше, изначально строились в предположении о том, что декодирование осуществляется в метрике Хэмминга, даже если это явно не было указано. Как будет показано далее, на их основе могут быть созданы коды для случая метрики Евклида. Соответствующие конструкции носят название методов *кодовой модуляции*.

14.1.2. Конструкция Унгербёка

В основе конструкции *решетчато-кодовой модуляции* (РКМ) лежит некоторый двоичный $(k+1, k)$ сверточный код и сигнальное множество $\mathbb{X}$, содержащее $M = 2^{k+1+m}$ элементов, которое разбито на 2^{k+1} подмножеств $\mathbb{X}_i$ одинакового размера. На рис. 14.2 представлена схема кодера *решетчато-кодовой модуляции*. Поток данных разбивается на блоки по $k+m$ символов. k символов каждого блока преобразуются кодером сверточного кода в $k+1$ символов. Каждый такой блок используется для того, чтобы выбрать одно из подмножеств X_i сигнального мно-

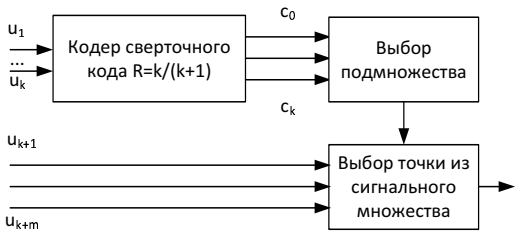


Рис. 14.2. решетчато-кодовая модуляция

жества. Конкретный передаваемый сигнал определяется оставшимися m символами данных. Множество последовательностей, которые могут быть порождены таким кодером, может быть представлено в виде путей в решетке, аналогично тому, как это было сделано в случае сверточных кодов. Структура решетки повторяет структуру решетки используемого сверточного кода, однако вместо одного ребра, соединяющего смежные ее узлы, в решетке РКМ присутствуют 2^m параллельных ребер, которые соответствуют различным элементам подмножества X_i, определяемого блоком символов, порождаемых кодером при переходе между соответствующими состояниями. Декодирование также может быть выполнено с помощью алгоритма Витерби. Напомним, что он строит набор путей в решетке, ближайших к принятой последовательности. Отбраковка малоправдоподобных путей происходит в тот момент, когда несколько путей сходятся в одном узле решетки. Очевидно, что из множества параллельных ребер, соединяющих фиксированную пару узлов решетки, достаточно рассмотреть только одно, соответствующее элементу подмножества X_i (i определяется начальным и конечным состояниями), ближайшему к принятой последовательности. При этом для минимизации вероятности ошибки желательно, чтобы расстояние Евклида между этими сигналами было как можно большим. Это необходимо обеспечить при построении разбиения $\mathbb{X} = \bigcup_{i=1}^{2^{k+1}} \mathbb{X}_i$, причем индекс i в двоичном представлении используется для выбора используемого подмножества в зависимости от того, какой блок символов был порожден сверточным кодером. Подходящее разбиение может быть построено рекурсивно. Для этого на каждом уровне рекурсии будем разбивать сигнальное множество на два подмножества так, чтобы минимальное расстояние между их элементами увеличивалось. Пример такого разбиения представлен на рис. 14.3. Как и в случае сверточных кодов, данная сигнально-кодовая конструкция

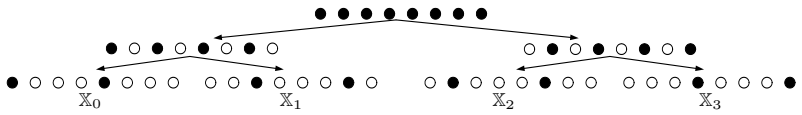


Рис. 14.3. Разбиение Унгербёка сигнального множества 8-АМ

может быть охарактеризована своим *минимальным свободным расстоянием* Евклида.

Качество различных схем кодовой модуляции может быть охарактеризовано в терминах энергетического выигрыша, который определяется как

$$\gamma_{\text{дБ}} = 10 \log_{10} \left(\frac{E_{s,c}}{E_{s,u}} \cdot d_c^2 \right),$$

где $E_{s,c}$ и d_c^2 — средняя энергия на символ и квадрат минимального свободного расстояния в системе с кодированием, и $E_{s,u}$ — средняя энергия на символ в системе без кодирования. Эта величина показывает, на сколько децибел можно уменьшить отношение сигнал/шум в системе с РКМ по сравнению с системой без кодирования, обеспечивая при этом одну и ту же достаточно малую вероятность ошибки на бит и скорость (бит/символ). Необходимо отметить, что в области сравнительно больших вероятностей ошибки корректирующая способность РКМ может оказаться хуже, чем в системах без кодирования.

14.1.3. Хорошие решетчатые коды

Как и в случае сверточных кодов, задача подбора параметров РКМ решается переборными методами с помощью ЭВМ. В данном разделе представлены таблицы с некоторыми хорошими решетчатыми кодами. Эти таблицы предполагают использование рекурсивного систематического сверточного кода с порождающей матрицей вида

$$G = \begin{pmatrix} g_1(x)/g_0(x) & 1 & 0 & \dots & 0 \\ g_2(x)/g_0(x) & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \\ g_k(x)/g_0(x) & 0 & 0 & \dots & 0 \end{pmatrix},$$

где $g_i(x) = \sum_{l=0}^{v} g_l^i x^l$ — порождающие полиномы. Структурная схема соответствующего кодера представлена на рис. 14.4. В таблицах 14.1, 14.2 и 14.3 представлены порождающие многочлены (в восьмеричной записи) для фазовой, амплитудно-импульсной и квадратурно-амплитудной модуляции [51], а также соответствующий асимптотический энергетиче-

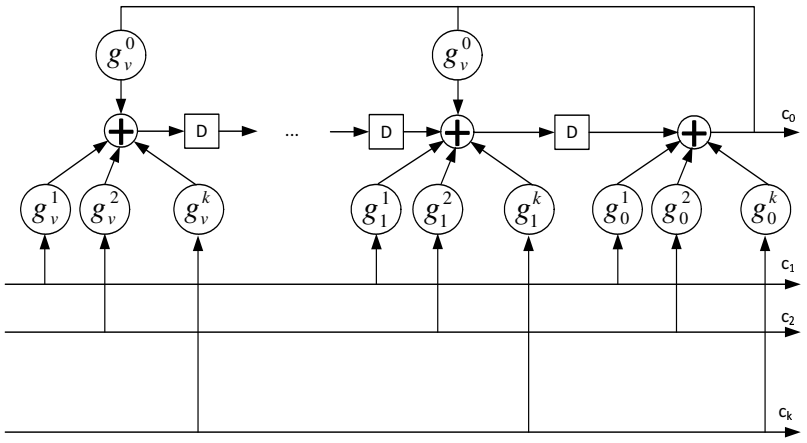


Рис. 14.4. Кодер рекурсивного систематического сверточного кода со скоростью $R = k/(k+1)$.

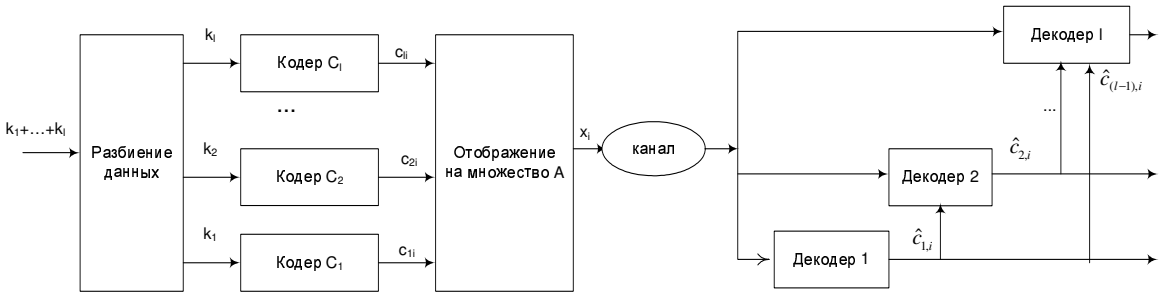


Рис. 14.5. Многоуровневое кодирование и многошаговое декодирование

ский выигрыш по сравнению с аналогичными методами модуляции без кодирования.

14.2. Многоуровневые коды

14.2.1. Основные понятия

Конструкция *многоуровневых кодов* является обобщением решетчато-кодовой модуляции и позволяет построить код длины n над некоторым алфавитом A на базе нескольких компонентных (обычно двоичных) кодов той же длины [40].

Пусть дано некоторое сигнальное множество $\mathbb{X}$, содержащее 2^l элементов. Предположим, что оно разбито на два подмножества одинаковой мощности $\mathbb{X}^{(0)}, \mathbb{X}^{(1)}$, которые в свою очередь разбиты на равномощ-

Таблица 14.1

Порождающие многочлены фазовой решетчато-кодовой модуляции

Число состояний	g_0	g_1	g_2	$\gamma_{\text{дБ}}$ 8-ФМ/4-ФМ
4	5	2	-	3.0
8	11	2	4	3.6
16	23	4	15	4.1
32	45	16	34	4.6
64	103	30	66	5.0
128	277	54	122	5.2
256	435	72	130	5.8
512	1525	462	360	5.8

Таблица 14.2

Порождающие многочлены амплитудно-импульсной решетчато-кодовой модуляции

Число состояний	g_0	g_1	$\gamma_{\text{дБ}}$ 4-АМ/2-АМ	$\gamma_{\text{дБ}}$ 8-АМ/4-АМ
4	5	2	2.55	3.31
8	13	4	3.01	3.77
16	23	4	3.42	4.18
32	45	10	4.15	4.91
64	103	24	4.47	5.23
128	235	126	5.05	5.81
256	515	362	-	5.81

ные подмножества $\mathbb{X}^{(0,0)}, \mathbb{X}^{(0,1)}, \mathbb{X}^{(1,0)}, \mathbb{X}^{(1,1)}$ и т.д.. Пусть $(c_{11}, \ldots, c_{1n})$ — результат кодирования k_1 битов данных (n, k_1, d_1) кодом $\mathcal{C}_1$. Построим последовательность $\mathbb{X}^{(c_{11})}, \ldots, \mathbb{X}^{(c_{1n})}$ множеств возможных значений для каждого из n символов формируемого кодового слова $(x_1, \ldots, x_n) \in \mathbb{X}^n$. Закодируем еще k_2 битов данных (n, k_2, d_2) кодом $\mathcal{C}_2$ и сузим множество возможных значений для каждого из символов x_i до $\mathbb{X}^{(c_{11},c_{21})}, \ldots, \mathbb{X}^{(c_{1n},c_{2n})}$. После l таких шагов кодовое слово многоуровневого кода может быть составлено из единственных элементов множеств $\mathbb{X}^{(c_{1i},c_{2i},\ldots,c_{li})}$. Разбиение сигнального множества на подмножества может выполняться различными способами; наиболее распространенным является метод Унгербёка (см. рис. 14.3).

Таблица 14.3

Порождающие многочлены квадратурно-амплитудной решетчато-кодовой модуляции

Число состояний	g_0	g_1	g_2	$\gamma_{\text{дБ}}$ 16-КАМ/ 8-ФМ	$\gamma_{\text{дБ}}$ 32-КАМ/16-КАМ	$\gamma_{\text{дБ}}$ 64-КАМ/32-КАМ
4	5	2	-	4.4	3.0	2.8
8	11	2	4	5.3	4.0	3.8
16	23	4	16	6.1	4.8	4.6
32	41	6	10	6.1	4.8	4.6
64	101	16	64	6.8	5.4	5.2
128	203	14	42	7.4	6.0	5.8
256	401	56	304	7.4	6.0	5.8
512	1001	346	510	7.4	6.0	5.8

Декодирование многоуровневых кодов может быть выполнено следующим образом. Вычислим логарифмические отношения правдоподобия для символов кодового слова первого кода как

$$L_{1i} = \ln \frac{P\{c_{1,i} = 1 | y_i\}}{P\{c_{1,i} = 0 | y_i\}} = \ln \frac{\sum_{x \in \mathbb{X}^{(1)}} P_{XY}\{x_i = x | y_i\}}{\sum_{x \in \mathbb{X}^{(0)}} P_{XY}\{x_i = x | y_i\}}$$

и передадим их декодеру первого кода. Результатом его работы является оценка соответствующего кодового слова $\widehat{c}_1 = (\widehat{c}_{11}, \ldots, \widehat{c}_{1n})$. Эта оценка позволяет снизить неопределенность относительно того, какой из символов множества $\mathbb{X}$ использовался при порождении каждого из переданных символов x_i. Это позволяет вычислить логарифмические отношения правдоподобия второго уровня как

$$L_{2i} = \ln \frac{\sum_{x \in \mathbb{X}^{(\widehat{c}_{1i},1)}} P_{XY}\{x_i = x | y_i\}}{\sum_{x \in \mathbb{X}^{(\widehat{c}_{1i},0)}} P_{XY}\{x_i = x | y_i\}}.$$

На их основе может быть найдена оценка $\widehat{c}_2$, и т.д. Данный метод носит название *алгоритма многошагового декодирования*. Структурная схема многоуровневого кодера и многошагового декодера представлена на рис. 14.5.

14.2.2. Правила выбора компонентных кодов

Выбор компонентных кодов $\mathcal{C}_i$ оказывает существенное влияние на корректирующую способность всего многоуровневого кода. В работе [79]

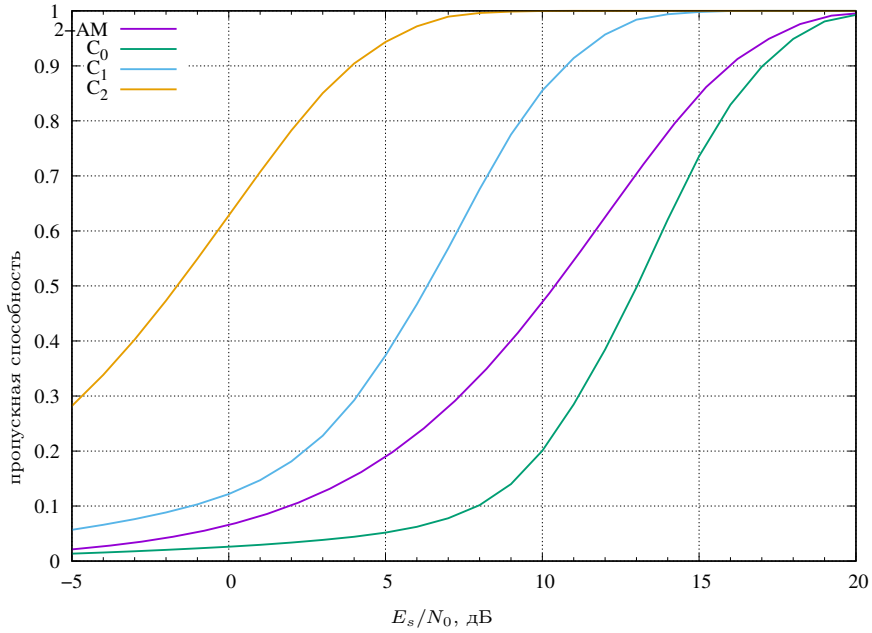


Рис. 14.6. Пропускные способности эквивалентных подканалов 8-AM

были предложены правила построения многоуровневых кодов, некоторые из которых приведены ниже.

Правило пропускных способностей

Совокупность устройства выбора подмножества сигнального множества и реального канала передачи данных может рассматриваться как некоторый эквивалентный канал, который может быть охарактеризован своей пропускной способностью. Если существует возможность использовать коды, достигающие пропускной способности канала, их скорости k_i/n должны приближаться к пропускным способностям эквивалентных подканалов. Использование этого правила приводит к многоуровневым кодам, также достигающим пропускной способности рассматриваемого канала.

Она в общем случае меньше, чем пропускная способность аналогичного канала с двоичным входом, так как один и тот же символ кодового слова кода $\mathcal{C}_i$ может передаваться с использованием различных символов из $\mathbb{X}$ в зависимости от внешних факторов (символов кодовых слов $\mathcal{C}_{i+1}, \ldots, \mathcal{C}_l$). Это создает дополнительную неопределенность, которая не способствует повышению надежности принятия решения декодером. Для того, чтобы найти пропускную способность i-ого эквивалентного канала, предположим, что декодирование $\mathcal{C}_1, \ldots, \mathcal{C}_{i-1}$ выполнено безошибочно. В этом случае статистические свойства эквивалентного канала полностью определяются значениями случайных вели-

чин $X_1, \ldots, X_{i-1}$, соответствующих символам кодовых слов $\mathcal{C}_1, \ldots, \mathcal{C}_{i-1}$, т.е[1]. $C_i = I(Y; X_i|X_1, \ldots, X_{i-1}) = I(Y; X_i, \ldots, X_l|X_1, \ldots, X_{i-1}) - I(Y; X_{i+1}, \ldots, X_l|X_1, \ldots, X_l)$. Величина $I(Y; X_i, \ldots, X_l|X_1, \ldots, X_{i-1})$ представляет собой пропускную способность, достигаемую при использовании сигнального множества $A^{(x_1,\ldots,x_{i-1})}$, где x_j — значение случайной величины X_j. Для различных x_j пропускная способность такого множества может быть разной, поэтому необходимо произвести усреднение по ним. Таким образом, пропускная способность i-ого эквивалентного канала равна $C_i = M_{x_1,\ldots,x_{i-1}}[I(Y; X_i, \ldots, X_l|x_1, \ldots, x_{i-1})] - M_{x_1,\ldots,x_i}[I(Y; X_{i+1}, \ldots, X_l|x_1, \ldots, x_i)]$. На рис. 14.6 представлены графики зависимости пропускной способности эквивалентных подканалов 8-АМ с разбиением Унгербёка. Для сравнения там же приведен график пропускной способности системы с 2-АМ с расстоянием между символами, равным расстоянию между символами в первом эквивалентном подканале. Видно, что возможность представления одного и того же символа кодового слова $\mathcal{C}_1$ приводит к значительному снижению пропускной способности.

Основным недостатком правила пропускных способностей является необходимость использования кодов $\mathcal{C}_i$, достигающих пропускной способности подканалов. До настоящего времени построить такие коды с практически приемлемыми параметрами не удалось.

Правило кодовых экспонент

Предположим, что в качестве кодов $\mathcal{C}_i$ используются случайные коды. В этом случае целесообразно подобрать их параметры таким образом, чтобы вероятность ошибки их декодирования (или экспонента случайного кодирования (1.7)) была одинаковой, т.е. скорость $R_i = k_i/n$ кода $\mathcal{C}_i$ должна быть получена как решение уравнения

$$\max_{0\le\rho\le 1} \left(E_i(\rho) - \rho R_i\right) = -\frac{1}{N}\log_2 p_e,$$

где p_e —- требуемая вероятность ошибки, $E_i(\rho) = M_{x_1,\ldots,x_{i-1}}\left[-\log_2 E_i(\rho, x_1, \ldots, x_{i-1})\right]$ и

$$E_i(\rho, x_1, \ldots, x_i) = \int_{\mathbb{Y}} \left(\sum_{x_i=0}^{1} P_X\left\{x_i\right\} p_{Y|X}^{\frac{1}{1+\rho}}(y|x_1, \ldots, x_{i-1}, x_i) \right)^{1+\rho} dy.$$

[1]Оптимизация распределения символов на входе канала здесь не производится, так как при использовании почти всех практически значимых кодов оно оказывается равномерным.

Недостатком данного правила является отсутствие гарантий того, что требуемая вероятность ошибки будет обеспечена с помощью конкретных кодов.

Правило равных вероятностей ошибки

Если дано конкретное семейство кодов $\mathbf{C}$, для которого существует возможность оценки вероятности ошибки в каждом из эквивалентных подканалов в зависимости от скорости кода, то параметры кодов $\mathcal{C}_i$ могут быть подобраны таким образом, чтобы эти вероятности были примерно одинаковы. К сожалению, эту задачу редко удается решить аналитически.

Правило сбалансированных минимальных расстояний

Каждое подмножество $A^{(x_1,\ldots,x_i)}$ сигнального множества характеризуется квадратом минимального расстояния Евклида δ_{i+1} между своими элементами. При использовании двоичного кода $\mathcal{C}_{i+1}$ с минимальным расстоянием Хэмминга d_{i+1} квадрат минимального расстояния Евклида между кодовыми словами оказывается равен $\delta_{i+1}d_{i+1}$. Параметры кодов могут быть подобраны таким образом, чтобы эти величины были примерно одинаковы для всех i. Это приводит к многоуровневым кодам с большим минимальным расстоянием, однако скорости некоторых компонентных кодов при этом часто оказываются превышающими пропускную способность эквивалентных подканалов. Это проявляется в чрезвычайно большом числе кодовых слов наименьшего веса, что приводит к слишком большой вероятности ошибки декодирования при небольших отношениях сигнал/шум. В результате многоуровневые коды, построенные по такому принципу, сильно проигрывают иным конструкциям.

Предметный указатель

Библиографический список

1. Блейхут Р. Теория и практика кодов, контролирующих ошибки. — М. : Мир, 1986.

2. ван дер Варден Б. Л. Алгебра. — М. : Наука, 1976.

3. Галлагер Р. Теория информации и надежная связь. — М. : Советское радио, 1974.

4. Кокс Д., Литтл Д., О'Ши Д. Идеалы, многообразия и алгоритмы. — М. : Мир, 2000.

5. Колесник В. Д. Кодирование при передаче и хранении информации (алгебраическая теория блоковых кодов). — М. : Высшая школа, 2009. — С. 550.

6. Кудряшов Б. Теория информации. — СПб : Питер, 2009.

7. Кудряшов Б. Основы теории кодирования. — СПб : БХВ, 2016.

8. Мак-Вильямс Ф. Д., Слоэн Н. Д. А. Теория кодов, исправляющих ошибки. — М. : Связь, 1979.

9. Питерсон У., Уэлдон Э. Коды, исправляющие ошибки. — М. : Мир, 1976.

10. Прокис Д. Цифровая связь. — М. : Радио и связь, 2000.

11. Трифонов П. В., Федоренко С. В. Метод быстрого вычисления преобразования Фурье над конечным полем // Проблемы передачи информации. — 2003. — Т. 39, № 3. — С. 3–10.

12. Федоренко С. Метод вычисления дискретного преобразования Фурье над конечным полем // Проблемы передачи инфорации. — 2006. — Т. 42, № 2. — С. 81–93.

13. Algebraic Properties of Polar Codes From a New Polynomial Formalism / Bardet M., Dragoi V., Otmani A., and Tillich J.-P. // Proceedings of IEEE International Symposium on Information Theory. — 2016.

14. Arıkan E. Channel Polarization: A Method for Constructing Capacity-Achieving Codes for Symmetric Binary-Input Memoryless Channels // IEEE Transactions on Information Theory. — 2009. — July. — Vol. 55, no. 7. — P. 3051–3073.

15. Barg A. Complexity Issues in Coding Theory // Electronic Colloquium on Computational Complexity (ECCC). — 1997. — Vol. 4,

no. 046.

16. Bellorado J., Kavcic A. Low-Complexity Soft-Decoding Algorithms for Reed-Solomon Codes–Part I: An Algebraic Soft-In Hard-Out Chase Decoder // IEEE Transactions on Information Theory. — 2010. — March. — Vol. 56, no. 3. — P. 945–959.

17. Bernstein D., Lange T., Peters C. Attacking and Defending the McEliece Cryptosystem // Post-Quantum Cryptography / Ed. by Buchmann J., Ding J. — Springer Berlin Heidelberg, 2008. — Vol. 5299 of Lecture Notes in Computer Science. — P. 31–46.

18. Berrou C., Glavieux A., Thitimajshima P. Near Shannon-limit error-correcting coding and decoding: Turbo codes // Proceedings of IEEE International Communications Conference. — 1993. — P. 1064–1070.

19. Bossert M. Channel coding for telecommunications. — Wiley, 1999.

20. Canteaut A., Chabaud F. A New Algorithm for Finding Minimum-Weight Words in a Linear Code: Application to McEliece's Cryptosystem and to Narrow-Sense BCH Codes of Length 511 // IEEE Transactions on Information Theory. — 1998. — January. — Vol. 44, no. 1. — P. 367–378.

21. Chase D. A class of algorithms for decoding block codes with channel measurement information // IEEE Transactions on Information Theory. — 1972. — January. — Vol. 18, no. 1. — P. 164–172.

22. Chen N., Yan Z. Cyclotomic FFTs with Reduced Additive Complexities Based on a Novel Common Subexpression Elimination Algorithm // IEEE Transactions on Signal Processing. — 2009. — Vol. 57, no. 3. — P. 1010–1020.

23. A Class of Low-Density Parity-Check Codes Constructed Based on Reed-Solomon Codes With Two Information Symbols / Djurdjevic I., Xu J., Abdel-Ghaffar K., and Lin S. // IEEE Communications Letters. — 2003. — July. — Vol. 7, no. 7.

24. Costa E., Fedorenko S., Trifonov P. Efficient algorithm for computing syndrome polynomial in Reed-Solomon decoder // Proceedings of 5th International ITG Conference on Source and Channel Coding (SCC). — 2004. — Vol. 181. — P. 179–183.

25. Digital Video Broadcasting (DVB); Frame structure channel coding and modulation for a second generation digital terrestrial television broadcasting system (DVB-T2). — ETSI EN 302 755 V1.1.1. —

2009.

26. Digital Watermarking and Steganography / Cox I. J., Miller M. L., Bloom J. A., Fridrich J., and Kalker T. — 2nd ed. — Morgan Kaufmann Publishers, 2008.

27. Divsalar D., Jin H., McEliece R. Coding Theorems for Turbo-Like Codes // Proceedings of the 36th Annual Allerton Conference on Communication, Control, and Computing. — 1998. — September. — P. 201–210.

28. Dumer I., Krichevskiy R. Soft-Decision Majority Decoding of Reed-Muller Codes // IEEE Transactions on Information Theory. — 2000. — January. — Vol. 46, no. 1.

29. Etzion T., Trachtenberg A., Vardy A. Which Codes Have Cycle-Free Tanner Graphs? // IEEE Transactions on Information Theory. — 1999. — September. — Vol. 45, no. 6.

30. Fedorenko S. V., Trifonov P. V. Finding roots of polynomials over finite fields // IEEE Transactions on Communications. — 2002. — Vol. 50, no. 11. — P. 1709–1711.

31. Fedorenko S. V., Trifonov P. V., Costa E. Improved hybrid algorithm for finding roots of error-locator polynomials // European Transactions on Telecommunications. — 2003. — Vol. 14, no. 5. — P. 411–416.

32. Fernandez M., Moreira J., Soriano M. Identifying Traitors Using the Koetter-Vardy Algorithm // IEEE Transactions on Information Theory. — 2011. — February. — Vol. 57, no. 2.

33. Forney G. D. Generalized minimum distance decoding // IEEE Transactions on Information Theory. — 1966. — April. — Vol. 12, no. 4. — P. 125–131.

34. Forney G. D. Trellis shaping // IEEE Transactions on Information Theory. — 1992. — March. — Vol. 38, no. 2. — P. 281–300.

35. Fossorier M. P., Lin S. Soft-Decision Decoding of Linear Block Codes Based on Ordered Statistics // IEEE Transactions on Information Theory. — 1995. — 9. — Vol. 41, no. 5. — P. 1379–1396.

36. Gallager R. Low-density Parity-Check codes : Ph. D. thesis ; MIT. — 1963.

37. Guruswami V., Sudan M. Improved Decoding of Reed-Solomon and Algebraic-Geometric Codes // IEEE Transactions on Information Theory. — 1999. — September. — Vol. 45, no. 6. — P. 1757–1767.

38. Hu X.-Y., Eleftheriou E., Arnold D.-M. Regular and Irregular Progressive Edge-Growth Tanner Graphs // IEEE Transactions on Information Theory. — 2005. — January. — Vol. 51, no. 1.

39. IEEE 802.16e. IEEE Standard for Local and metropolitan area networks. Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems. Amendment 2: Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands. — 2005.

40. Imai H., Hirakawa S. A new multilevel coding method using error correcting codes // IEEE Transactions on Information Theory. — 1977. — May. — Vol. 23, no. 3. — P. 371–377.

41. Introduction to Algorithms / Cormen T. H., Leiserson C. E., Rivest R. L., and Stein C. — 2 ed. — The MIT Press, 2001.

42. Jiang T., Vardy A. Asymptotic Improvement of the Gilbert-Varshamov Bound on the Size of Binary Codes // IEEE Transactions on Information Theory. — 2004. — August. — Vol. 50, no. 8.

43. Jin H., Khandekar A., McEliece R. Irregular repeat-accumulate codes // Proceedings of the 2nd International Symposium on Turbo Codes and Related Topics. — 2000. — September. — P. 1–8.

44. Kern D., Vorkoper S., Kuhn V. A New Code Construction for Polar Codes Using Min-Sum Density // Proceedings of International Symposium on Turbo Codes and Iterative Information Processing. — 2014. — P. 228–232.

45. Kim S., Lee S., Chung S.-Y. An Efficient Algorithm for ML Decoding of Raptor Codes over the Binary Erasure Channel // IEEE Communications Letters. — 2008. — August. — Vol. 12, no. 8.

46. Koetter R. Fast Generalized Minimum-Distance Decoding of Algebraic-Geometry and Reed-Solomon Codes // IEEE Transactions on Information Theory. — 1996. — May. — Vol. 42, no. 3.

47. Korada S. B., Sasoglu E., Urbanke R. Polar Codes: Characterization of Exponent, Bounds, and Constructions // IEEE Transactions on Information Theory. — 2010. — December. — Vol. 56, no. 12. — P. 6253–6264.

48. Lin S., Costello D. J. Error Control Coding: Fundamentals and applications. — Prentice-Hall, 1983.

49. Luby M. LT Codes // Proceedings of 43rd IEEE International Symposium on Foundations of Computer Science. — 2002. — November. — P. 271–280.

50. MacKay D. Good error correcting codes based on very sparse matrices // IEEE Transactions on Information Theory. — 1999. — March. — P. 399–431.

51. Moon T. K. Error Correction Coding. Mathematical Methods and Algorithms. — Hoboken, NJ, USA : Wiley, 2005.

52. Nielsen R. R., Hoholdt T. Decoding Reed-Solomon codes beyond half the minimum distance // Proceedings of the International Conference on Coding Theory and Cryptography. — Mexico : Springer-Verlag. — 1998. — P. 221–236.

53. Nielsen R. R., Hoholdt T. Decoding Reed-Solomon codes beyond half the minimum distance // Coding theory, Cryptography and Related Areas. — Springer-Verlag. — 2000. — P. 221–236.

54. On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit / Chung S.-Y., Forney G. D., Richardson T. J., and Urbanke R. // IEEE Communications Letters. — 2001. — February. — Vol. 5, no. 2.

55. Optimal decoding of linear codes for minimizing symbol error rate / Bahl L., Cocke J., Jelinek F., and Raviv J. // IEEE Transactions on Information Theory. — 1974. — P. 284–287.

56. Poltyrev G. Bounds on the Decoding Error Probability of Binary Linear Codes Via Their Spectra // IEEE Transactions on Information Theory. — 1994. — July. — Vol. 40, no. 4.

57. Richardson T., Shokrollahi M. A., Urbanke R. L. Design of Capacity-Approaching Irregular Low-Density Parity-Check Codes // IEEE Transactions on Information Theory. — 2001. — February. — Vol. 47, no. 2. — P. 619–637.

58. Roth R., Ruckenstein G. Efficient decoding of Reed-Solomon codes beyond half the minimum distance // IEEE Transactions on Information Theory. — 2000. — Vol. 46, no. 1. — P. 246–257.

59. Ryan W. A turbo code tutorial // Proceedings of Globecom'97. — 1997.

60. Ryan W., Lin S. Channel Codes: Classical and modern. — Cambridge University Press, 2009.

61. Sauer T. Polynomial interpolation of minimal degree and Gröbner bases // Gröbner Bases and Applications (Proceedings of the International Conference "33 Years of Gröbner Bases") / ed. by Buchberger B., Winkler F. — Cambridge University Press. — 1998. — Vol. 251 of London Mathematical Society Lecture Notes. — P. 483–494.

62. Shannon C. A Mathematical Theory of Communication // Bell System Technical Journal. — 1948. — July, October. — Vol. 27. — P. 379–423, 623–656.

63. Shokrollahi M. A. Raptor Codes // IEEE Transactions on Information Theory. — 2006. — June. — Vol. 52, no. 6.

64. Sun J., Takeshita O. Y. Interleavers for Turbo Codes Using Permutation Polynomials Over Integer Rings // IEEE Transactions on Information Theory. — 2005. — January. — Vol. 51, no. 1.

65. Tal I., Vardy A. How to Construct Polar Codes // IEEE Transactions on Information Theory. — 2013. — October. — Vol. 59, no. 10. — P. 6562–6582.

66. Tal I., Vardy A. List Decoding of Polar Codes // IEEE Transactions On Information Theory. — 2015. — May. — Vol. 61, no. 5. — P. 2213–2226.

67. A Transform Approach for Computing the Ranks of Parity-Check Matrices of Quasi-Cyclic LDPC Codes / Diao Q., Huang Q., Lin S., and Abdel-Ghaffar K. A. // Proceedings of IEEE International Symposium on Information Theory. — 2011.

68. A Trellis-Based Recursive Maximum-Likelihood Decoding Algorithm for Binary Linear Block Codes / Fujiwara T., Yamamoto H., Kasami T., and Lin S. // IEEE Transactions On Information Theory. — 1998. — March. — Vol. 44, no. 2.

69. Trifonov P. Efficient interpolation in the Guruswami-Sudan algorithm // IEEE Transactions on Information Theory. — 2010. — September. — Vol. 56, no. 9. — P. 4341–4349.

70. Trifonov P. Randomized chained polar subcodes // Proceedings of IEEE Wireless Communications and Networking Conference Workshops. — Barcelona, Spain : IEEE. — 2018. — P. 292–297.

71. Trifonov P. A score function for sequential decoding of polar codes // Proceedings of IEEE International Symposium on Information Theory. — Vail, USA. — 2018.

72. Trifonov P. Randomized Polar Subcodes with Optimized Error Coefficient // IEEE Transactions on Communications. — 2020. — November. — Vol. 68, no. 11. — P. 6714–6722.

73. Trifonov P. Recursive Trellis Processing of Large Polarization Kernels // Proceedings of IEEE International Symposium on Information Theory. — 2021.

74. Trifonov P., Miloslavskaya V. Polar subcodes // IEEE Jour-

nal on Selected Areas in Communications. — 2016. — February. — Vol. 34, no. 2. — P. 254–266.

75. Trofimiuk G., Trifonov P. Window Processing of Binary Polarization Kernels // IEEE Transactions on Communications. — 2021. — July. — Vol. 69, no. 7. — P. 4294–4305.

76. TS 26.346 V7.1.0. Technical Specification Group Services and System Aspects; Multimedia Broadcast/Multicast Service (MBMS); Protocols and codecs : Rep. / 3GPP : 2006. — September.

77. Valembois A., Fossorier M. Box and Match Techniques Applied to Soft-Decision Decoding // IEEE Transactions on Information Theory. — 2004. — 5. — Vol. 50, no. 5. — P. 796–810.

78. von zur Gathen J., Gerhard J. Modern Computer Algebra. — Cambridge University Press, 1999.

79. Wachsmann U., Fischer R. F. H., Huber J. B. Multilevel Codes: Theoretical Concepts and Practical Design Rules // IEEE Transactions on Information Theory. — 1999. — July. — Vol. 45, no. 5. — P. 1361–1391.

80. Witten I. H., Eibe F., Hall M. A. Data mining : practical machine learning tools and techniques. — 3rd ed. — Morgan Kaufmann Publishers, 2011.

81. Wu Y. New List Decoding Algorithms for Reed-Solomon and BCH Codes // IEEE Transactions on Information Theory. — 2008. — August. — Vol. 54, no. 8.

82. Wu Y. Fast Chase Decoding Algorithms and Architectures for Reed-Solomon Codes // IEEE Transactions On Information Theory. — 2012. — January. — Vol. 58, no. 1.

83. Zigangirov K. S. Some sequential decoding procedures // Problems of Information Transmission. — 1966. — Vol. 2, no. 4. — P. 1–10. — In Russian.

Приложение Задания для курсовой работы

В качестве курсовой работы предлагается выполнить программную реализацию одного из нижеперечисленных алгоритмов декодирования, соответствующий алгоритм кодирования и исследовать их поведение в некотором канале передачи данных (например, двоичном симметричном, аддитивном гауссовском) методом статистического моделирования. Полученные результаты (вероятность ошибки, сложность декодирования) необходимо сравнить с известными теоретическими оценками, приведенными в соответствующих статьях.

1. Алгоритм Витерби мягкого декодирования линейных блоковых кодов [7].
2. Алгоритм Витерби мягкого декодирования сверточных кодов [7].
3. Алгоритм рекурсивного мягкого декодирования по решеткам линейных блоковых кодов [68].
4. Алгоритм box match мягкого декодирования линейных блоковых кодов [77].
5. Алгоритм Берлекэмпа-Месси декодирования двоичных кодов БЧХ [7].
6. Алгоритм Берлекэмпа-Месси декодирования кодов Рида-Соломона [7].
7. Алгоритм Берлекэмпа-Месси декодирования двоичных кодов Гоппы [7, 8].
8. Алгоритм Сугиямы (Евклида) декодирования двоичных кодов БЧХ [8].
9. Алгоритм Сугиямы (Евклида) декодирования кодов Рида-Соломона [8].
10. Алгоритм Сугиямы (Евклида) декодирования двоичных кодов Гоппы [8].
11. Алгоритм распространения доверия декодирования LDPC кодов [7].
12. Итеративное декодирование турбо-кодов [7].
13. Алгоритм Гурусвами-Судана списочного декодирования кодов Рида-Соломона [37, 53, 58].
14. Алгоритм Ву списочного декодирования кодов Рида-Соломона [37, 53, 58, 81].
15. Списочное декодирование полярных кодов [66].

16. Последовательное декодирование полярных кодов [66, 71].

Трифонов Петр Владимирович

Основы помехоустойчивого кодирования

Учебное пособие

В авторской редакции
Редакционно-издательский отдел Университета ИТМО
Зав. РИО Н.Ф. Гусарова
Подписано к печати
Заказ №
Тираж
Отпечатано на ризографе

Редакционно-издательский отдел
Университета ИТМО
197101, Санкт-Петербург, Кронверкский пр., 49, литер А