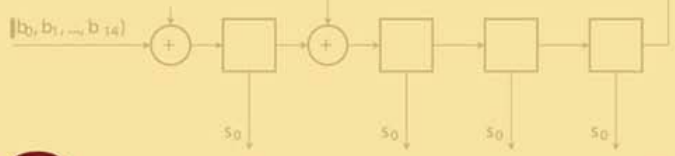
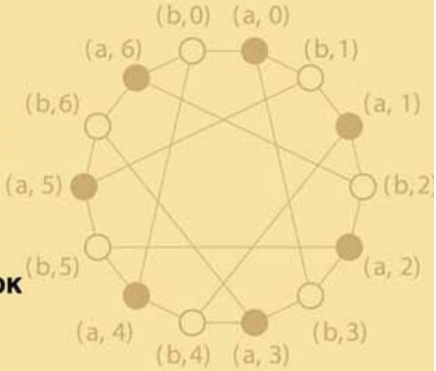
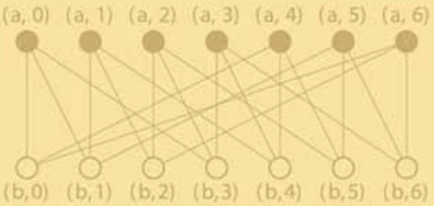
Б. Д. Кудряшов



Основы теории кодирования

- Потенциальная эффективность кодирования
- Алгебраические коды
- Коды над решетками. Сверточные коды
- Каскадные коды, кодированная модуляция, турбокоды
- Коды с малой плотностью проверок на четность





bhv

Б. Д. Кудряшов

Основы теории кодирования

Допущено Учебно-методическим объединением вузов Российской Федерации
по университетскому политехническому образованию
в качестве учебного пособия для студентов высших учебных заведений,
обучающихся по направлению подготовки бакалавра 09.03.02
«Информационные системы и технологии» (230400)

Санкт-Петербург
«БХВ-Петербург»
2016

УДК 519.72
ББК 32.811.4
К88

Кудряшов Б. Д.

К88 Основы теории кодирования: учеб. пособие. — СПб.: БХВ-Петербург, 2016. — 400 с.: ил. — (Учебная литература для вузов)

ISBN 978-5-9775-3527-4

В учебное пособие, ориентированное на семестровый курс лекций, включены классические разделы теории кодирования: линейные коды, основы построения и декодирования алгебраических кодов. Рассказывается о представлении кодов решетками, о декодировании по максимуму правдоподобия. Приведены основы теории сверточных кодов, введение в каскадные коды, модуляционные коды и турбо-коды. Отдельная глава посвящена низкоплотностным кодам, находящим все более широкое применение в телекоммуникационных стандартах. Все необходимые математические сведения приведены в виде приложений к главам учебного пособия. В книге много численных примеров, детальных алгоритмов, примеров программ MATLAB.

Для студентов технических вузов, обучающихся по информационным специальностям, аспирантов и инженеров

УДК 519.72
ББК 32.811.4

Рецензенты:

Е .Т. Мирончиков — д-р техн. наук, проф. Петербургского государственного унивеситета путей сообщения;

Ф. И. Соловьева — д-р физ.-мат. наук, проф. Новосибирского государственного университета.

Подписано в печать 31.08.15.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 32,25.
Тираж 700 экз. Заказ №
"БХВ-Петербург", 191036, Санкт-Петербург, Гончарная ул., 20.

Первая Академическая типография "Наука"
199034, Санкт-Петербург, 9 линия, 12/28.

ISBN 978-5-9775-3527-4

Оглавление

Предисловие

Теорию кодирования можно рассматривать как раздел теории информации, посвященный проблеме достижения пределов скорости передачи информации, предугаданных Клодом Шенноном. В связи с тем, что научное сообщество готовится отметить в 2016 году столетний юбилей Шеннона, в 2013 году от лица ученых и инженеров, работающих в области теории информации и ее приложений, президент IEEE M. S. Coleman, президент Bell Labs G. Rittenhouse и президент MIT R. Reif выступили с инициативой выпустить в США почтовую марку, посвященную этому юбилею. В их обращении к руководству Почтовой службы США говорится о значении теории Шеннона для нашего времени:

> *We are writing to propose a stamp to honor the centenary of* **Clode Elwood Shannon, father of the information age**.
> *Information systems like smart phones, MP3 players and World Wide Web are changing our lives. They store our memories, connect our communities, and carry out entertainment. They facilitate healthcare delivery and provide the backbone upon our financial, industrial and governmental institutions are built.*
> **Much as Newton discovered the laws of gravity and motion, Shannon discovered the laws of information.** *By uncovering the simple laws at the heart of such overwhelmingly complex problems as communication, computing, memory and cryptography, Shannon unlocked the door that led us to the information age.*

Эта высокая оценка значимости открытий Шеннона относится в большой степени к теории кодирования, как к конструктивной части теории информации. Невероятно высокая надежность современной мобильной связи и запоминающих устройств, их миниатюрность, низкое энергопотребление — все это достигается применением эффективных кодов, исправляющих ошибки.

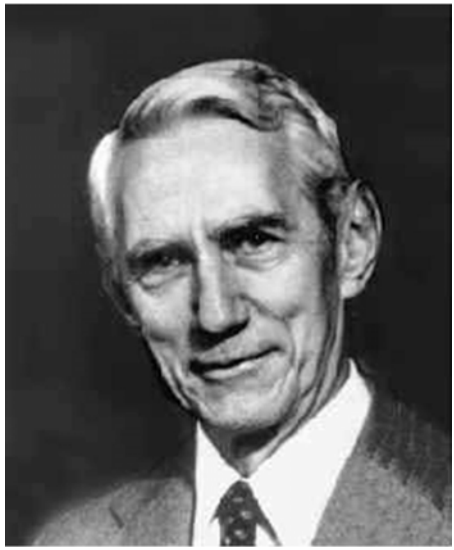


Рис. 1. Клод Шеннон

По мере появления новых задач и новых технологий для решения этих задач содержание теории кодирования быстро изменялось, особенно в последние 20 лет. С другой стороны, время, отводимое на изучение предмета, все время уменьшается, поскольку появляются на свет новые важные объекты для освоения студентами информационных специальностей. В результате относительно недавно изданные и переизданные классические монографии и учебники по теории кодирования объемом свыше 1000 страниц (см., например, [26]), тяжелы для студентов и аспирантов (в том числе и в буквальном смысле) и при этом не успевают отражать важные обновления в теории и практике кодирования.

Поэтому задача данной книги — сконцентрировать фундаментальные практические важные результаты современной теории кодирования в объеме семестрового курса. При этом предполагается, что большая часть прикладных вопросов изучается на практических занятиях, при выполнении лабораторных работ и самостоятельно при подготовке проектов. Поэтому в учебник включены задачи, решение которых требует использования компьютера, а также примеры программ, которые будут полезны при решении этих задач и при реализации алгоритмов кодирования и декодирования.

Особенность теории кодирования — разнообразие используемых математических методов. «Классические» методы кодирования основывались на дискретной математике, на теории конечных полей. Параллельно развивались вероятностные методы. В 1990-е годы произошла «турбо-революция», после которой выросло значение алгоритмических методов анализа графов. Соответственно, предполагается, что читатель умеет программировать, знаком с основами теории вероятности, комбинаторики и линейной алгебры, знает терминологию теории графов, а также фундаментальные понятия высшей алгебры. Хотя почти все используемые термины дополнительно поясняются, этих напоминаний может оказаться недостаточно и тогда придется воспользоваться учебниками по соответствующим разделам математики.

Одна из трудностей при отборе материала — избежать перекоса в сторону тематики, которая наиболее близка и интересна автору. В полной мере избежать такого перекоса не удалось. Представлению кодов решетчатыми диаграммами, теории сверточных кодов и некоторым другим вопросам уделено, возможно, несколько больше внимания, чем следовало. С другой стороны, некоторые важные вопросы теории и практики кодирования не попали в учебник. Обзор литературы, по которой можно восполнить пробелы, приведен в параграфе 1.2.

Автору учебника посчастливилось учиться теории информации и кодированию у В. Д. Колесника, Г. Ш. Полтырева, Е. Т. Мирончикова. Пониманию предмета помогли практическая работа с Г. Тененгольцем в США, с компанией Samsung, совместные научные исследования с Р. Йоханнессоном и его аспирантами в университете г. Лунд (Швеция), сотрудничество с В. Сидоренко и М. Боссертом

в университете г. Ульм (Германия). Всем своим учителям, коллегам и партнерам автор выражает свою глубокую и искреннюю признательность.

Автор признателен активным читателям и беспощадным критикам учебника — студентам факультета компьютерных технологий Санкт-Петербургского университета информационных технологий, механики и оптики за конструктивные замечания и десятки обнаруженных неточностей.

Особенную благодарность автор выражает своей жене, коллеге и соавтору Бочаровой И. Е. В материалы разделов, посвященных построению сверточных кодов и кодов с малой плотностью проверок на четность, включены некоторые результаты наших совместных работ.

1. Введение

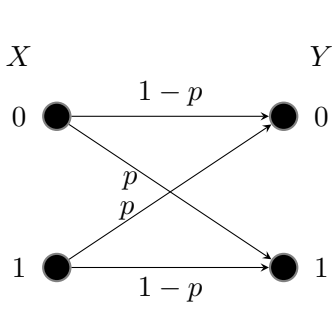
В 1948 году, когда, благодаря опубликованию работы Клода Шеннона [103], появилась на свет теория информации, большая часть информации передавалась в аналоговом виде. Сегодня ситуация изменилась на противоположную: по каналам связи сегодня вместо человеческой речи или музыки передаются нули и единицы. Более того, почти все сферы повседневной жизни так или иначе зависят от скорости и надежности передачи информации.

В этой главе мы воспользуемся фундаментальными теоремами Шеннона для того, чтобы указать теоретические пределы эффективности помехоустойчивого кодирования и сравнить их с эффективностью кодов, используемых на практике. В дальнейшем, близость к теоретическим пределам будет служить для нас критерием для сравнения классов кодов.

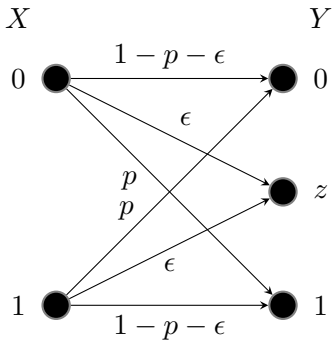
1.1. Постановка задачи помехоустойчивого кодирования

Сначала поясним общую идею защиты информации от ошибок. Предположим, что в нашем распоряжении имеется канал, на вход которого можно подавать двоичные символы (рис. 1.1, *а*).

Такой канал называют двоичным симметричным каналом (ДСК), вероятность p_0 называют переходной вероятностью или вероятностью ошибки одного символа. Пусть, скажем, $p_0 = 10^{-3}$ (реальные каналы зачастую много хуже). Такая надежность передачи информации не может считаться удовлетворительной. Как сделать ее выше?



а) ДСК



б) ДСтК

Рис. 1.1. Примеры каналов: *а* — двоичный симметричный канал (ДСК); *б* — двоичный канал со стираниями (ДСтК)

Передатчик может, например, повторять передачу каждого бита 3 раза, а приемник принимать решение по большинству.

Упражнение 1.1. Подсчитайте вероятность ошибки при трехкратном повторении каждого бита передаваемой информации при переходной вероятности ДСК равной $p_0 = 10^{-3}$.

Хотя вероятность ошибки стала меньше, такое решение нельзя признать удачным, поскольку скорость передачи данных уменьшилась в 3 раза. Это означает, что ресурсы системы (полоса частот, количество линий связи и т.п.) для передачи того же объема информации должны быть тоже увеличены в 3 раза. Или, например, оператор мобильной связи вынужден был бы сократить количество абонентов в 3 раза. Нельзя ли придумать более экономное решение?

Упражнение 1.2. Объединим биты передаваемой информации в пары. Множество «расширенных сообщений» состоит теперь из 4 различных двоичных комбинаций. Сопоставим парам бит последовательности длины 5, как это показано в табл. 1.1. Забегая вперед, список передаваемых последовательностей назовем *кодом*, сами последовательности — *кодовыми словами*, а отображение передаваемых данных на кодовые слова — *кодированием*. Убедитесь в том, что любые однократные ошибки при таком кодировании не

опасны. Какова вероятность ошибок при использовании данного кода в условиях предыдущего упражнения? Сравните характеристики систем связи, основанных на кодах из упражнений 1.1 и 1.2.

Таблица 1.1. Код для упражнения 1.2

Сообщения	Кодовые слова
00	00000
01	10110
10	01011
11	11101

Упражнение 1.3. Предположим, что приемник использует коды из упражнений 1.1 и 1.2 следующим образом: если принятая последовательность — кодовое слово, то соответствующее сообщение выдается получателю. В противном случае получателю выдается только сигнал, свидетельствующий о том, что принятая информация ненадежна (сигнал обнаружения ошибок). Для того же канала, что и в предыдущих упражнениях, подсчитайте вероятность того, что получателю будет выдана неверная информация, и вероятность выдачи сигнала обнаружения ошибок.

Знания, полученные при выполнении упражнений, позволяют заключить, что, выбирая различные хитроумные правила кодирования, можно получать различные обменные соотношения между эффективностью использования канала и обеспечиваемой надежностью.

В общем случае кодирование заключается во внесении избыточности в передаваемое сообщение. Вместо некоторого числа k бит по каналу передается количество сигналов, достаточное для передачи n бит, $n > k$. Отношение $R = k/n$ называют *скоростью кода*. То, что из общего числа 2^n возможных сигналов для передачи информации используется лишь 2^k, позволяет, как мы видели выше при выполнении упражнений, исправлять либо обнаруживать ошибки. Задача декодера — выбрать из числа кодовых слов одно, предположитель-

но — то, что передавалось. *Вероятностью ошибки* мы называем вероятность того, что решение, вынесенное декодером, неверно.

Интуиция подсказывает, что последовательности (кодовые слова), используемые для передачи сообщений, должны как можно сильнее отличаться друг от друга, иными словами, находиться на как можно большем расстоянии. Понятие расстояния Хэмминга и его влияние на характеристики помехоустойчивости мы обсудим ниже на примерах (см. задачи в конце параграфа).

Постановка задачи построения кодов как задачи достижения наибольшего минимального расстояния между кодовыми словами — не единственная возможная, но, по крайней мере, до последнего времени она была основной и позволила достичь внушительных результатов во многих приложениях. В середине 90-х годов прошлого столетия огромное количество теоретических исследований, а затем и прикладных, было посвящено турбо-кодам [47]. Позже, в конце 1990-х — начале 2000-х годов выяснилось, что предложенный Галлагером 30 лет назад класс кодов с малой плотностью проверок на четность (МППЧ-кодов) [76, 5] во многих отношениях лучше турбо-кодов. И турбо-коды, и МППЧ-коды характеризуются тем, что их минимальное расстояние зачастую невелико, но не оно определяет помехоустойчивость кода. Подробнее мы обсудим эту проблему при изучении соответствующих классов кодов.

Применительно к ДСК, точное утверждение о возможном соотношении между скоростью кода и достижимой надежностью, заимствованное из теории информации, — следующее.

Для ДСК с переходной вероятностью p_0 пропускная способность канала вычисляется по формуле

$$C = 1 - h(p_0), \tag{1.1}$$

где

$$h(x) = -x \log_2 x - (1-x) \log_2 (1-x) \quad — \tag{1.2}$$

энтропия двоичного ансамбля. При скорости передачи в битах на символ канала R, меньшей величины пропускной способности C, может быть обеспечена сколь угодно малая вероятность ошибки декодирования за счет увеличения длины используемых кодов (и, соответственно, увеличения сложности кодирования и декодирования). При скорости больше C надежная передача невозможна.

Например, при $p_0 = 10^{-3}$ имеем $C = 0,9886$, т.е. для обеспечения сколь угодно высокой надежности достаточно внести меньше 2% избыточных данных.

Сразу заметим, что даже после изучения курса теории кодирования мы не сумеем указать рецепт достижения такой эффективности кодирования. Теоремы Шеннона доказаны неконструктивными методами и не дают ответа на вопрос о том, как добиться предельно достижимой помехоустойчивости.

В реальной системе связи на вход канала поступают не нули и единицы, а сигналы, выбираемые в соответствии с физической природой канала (электрические импульсы, электромагнитные волны, акустические колебания и т.п.). В этой более общей ситуации вход канала $x(t)$ и выход канала $y(t)$ могут быть связаны, например, соотношениями вида

$$y(t) = x(t) + n(t),$$

где случайный процесс $n(t)$ представляет собой шум в канале связи. Такой канал называют *непрерывным* (до этого мы рассматривали дискретный канал) и, если шум $n(t)$ не зависит от передаваемых сигналов (от $x(t)$), то шум называют *аддитивным*.

На передаваемые сигналы накладываются ограничения в виде мощности сигнала P и занимаемой полосы частот W. Энергия одного сигнала длительности T равна $E = PT$. Чтобы оценить предельную эффективность кодирования, нужно еще условиться о модели шума. Предположим, что $n(t)$ — стационарный случайный процесс, одномерное распределение которого — гауссово с нулевым математическим ожиданием и с дисперсией $N_0/2$. Спектральная плотность мощности предполагается равномерной равной $N_0/2$ (вт/Гц) в диапазоне частот $[-W, W]$. Такой шум мы называем *аддитивным белым гауссовским шумом (АБГШ)* и соответствующий канал — *каналом с АБГШ*.

В этих условиях пропускная способность канала, измеренная в битах, передаваемых в единицу времени, вычисляется по формуле

$$C_t = W \log_2 \left(1 + \frac{P}{WN_0}\right) \quad \text{бит/с}. \tag{1.3}$$

Индексом t мы подчеркиваем, что в данном случае (в отличие от (1.1)) пропускная способность измеряется в битах на единицу времени.

Соответствующая теорема Шеннона утверждает, что при скорости передачи информации ниже C_t можно добиться сколь угодно малой вероятности ошибки, а выше этой скорости надежная передача невозможна.

Например, при полосе частот 3400 Гц и отношении сигнал/шум на 1 Гц $P/(WN_0) = 100$ (это приблизительные параметры реальной телефонной линии) находим $C_t \approx 22000$ бит/с.

При отсутствии ограничений на полосу из (1.3), переходя к пределу при $W \to \infty$, получаем совсем простую формулу

$$C_t = \frac{P}{N_0 \ln 2} \quad \text{бит/с}. \tag{1.4}$$

Сопоставим между собой характеристики системы связи без кодирования и системы с таким кодированием, которое по своим параметрам близко к теоретическим пределам (1.3) и (1.4). Особенно удобно это сделать на примере системы без ограничений на полосу.

Мы всегда можем ввести в систему кодирование, не нарушая ограничения на мощность, передавая вместо каждого сигнала длительности T секунд более короткие сигналы меньшей энергии. Коротких сигналов больше, и полученную избыточность можно использовать для применения кодов при сохранении постоянной величины битовой скорости передачи информации R_t, измеренной в битах/с (при этом расширяется занимаемая сигналом полоса частот, но мы временно игнорируем это обстоятельство).

За время T при затратах энергии PT будет передано R_tT бит информации. Введем характеристику

$$E_b = \frac{P}{R_t}, \tag{1.5}$$

отражающую затраты энергии на бит передаваемой информации.

Величину E_b/N_0 называют *отношением сигнал/шум на бит*. В инженерной практике энергетические характеристики принято измерять в децибелах (дБ): для числа A его значение в децибелах равно $10 \lg A$ дБ.

Согласно (1.4), надежная передача возможна при

$$R_t < C_t = \frac{P}{N_0 \ln 2} = \frac{E_b R_t}{N_0 \ln 2}$$

или

$$\frac{E_b}{N_0} > \ln 2 = -1.59 \quad \text{дБ}.$$

Итак, из теоремы кодирования для канала с АБГШ следует: надежная передача информации возможна только тогда, когда отношение сигнал/шум на бит не меньше величины -1.59 дБ.

Рассмотрим передачу информации двоичными сигналами без кодирования. При использовании для передачи информации по каналу с гауссовским шумом двоичных противоположных сигналов при передаче символов 0 и 1 на выходе демодулятора наблюдаются случайные величины с плотностями распределения

$$f(y|0) = \frac{1}{\sqrt{\pi N_0}} e^{-\frac{(y+\sqrt{E})^2}{N_0}} \text{ и } f(y|1) = \frac{1}{\sqrt{\pi N_0}} e^{-\frac{(y-\sqrt{E})^2}{N_0}} \tag{1.6}$$

соответственно. Эти плотности показаны на рис. 1.2.

Поскольку в отсутствие кодирования каждый сигнал несет один бит информации, энергия сигнала равна $E_s = E_b$. Приемник максимального правдоподобия (см. задачу 2 в конце данной главы) при $y < 0$ выносит решение о том, что передавался 0, и решение в пользу 1 в противном случае. Вероятность ошибки в одном бите не зависит от передаваемого символа и (при передаче нуля) может быть вычислена как площадь заштрихованной области на рис. 1.2. Иными словами, вероятность ошибки равна

$$P_b = \int_0^\infty f(y|0)dy = \int_{-\infty}^0 f(y|1)dy = Q\left(\sqrt{\frac{2E_b}{N_0}}\right), \tag{1.7}$$

где использовано обозначение

$$Q(x) = 1 - \frac{1}{\sqrt{2\pi}} \int_{-\infty}^x e^{-\frac{t^2}{2}} dt. \tag{1.8}$$

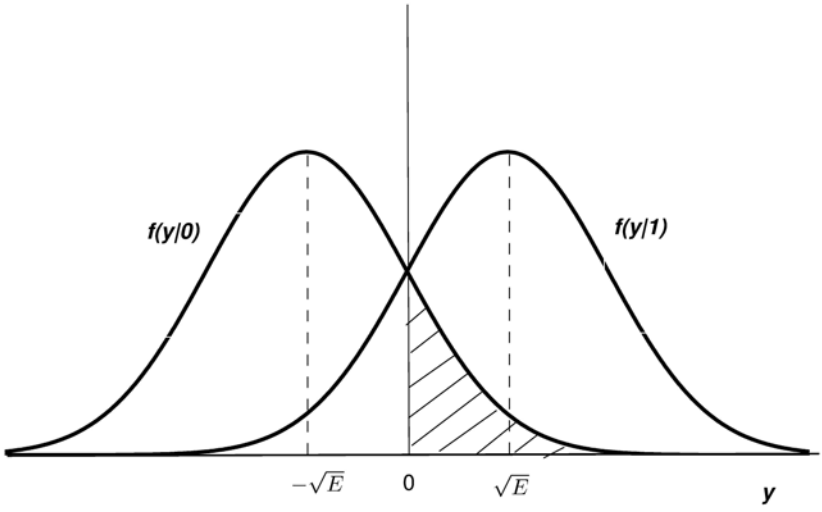


Рис. 1.2. Условные распределения значений на выходе канала при передаче символов 0 и 1 противоположными сигналами

График зависимости вероятности ошибки от отношения сигнал/шум при передаче без кодирования приведен на рис. 1.3. На этом же рисунке показана асимптота Шеннона $E_b/N_0 = -1.59$ дБ. Из графиков, в частности, можно заметить, что для достижения вероятности ошибки 10^{-6} требуется $E_b/N_0 = 10.53$ дБ на бит. Следовательно, при такой вероятности ошибки потенциально достижимый *энергетический выигрыш кодирования* составляет $1.59 + 10.53 = 12.06$ дБ, или в 16 раз! Цитируя известное высказывание Блэйхута, можно сказать, что улучшать каналы связи — это выбрасывать деньги на ветер. Лучше просто применить кодирование!

Энергетический выигрыш кодирования одного метода передачи информации по отношению к другому при заданном уровне вероятности ошибки измеряется как разница выраженных в децибелах требуемых отношений сигнал/шум в канале для этих методов. Когда говорят просто об энергетическом выигрыше, имеется в виду выигрыш по отношению к передаче без кодирования.

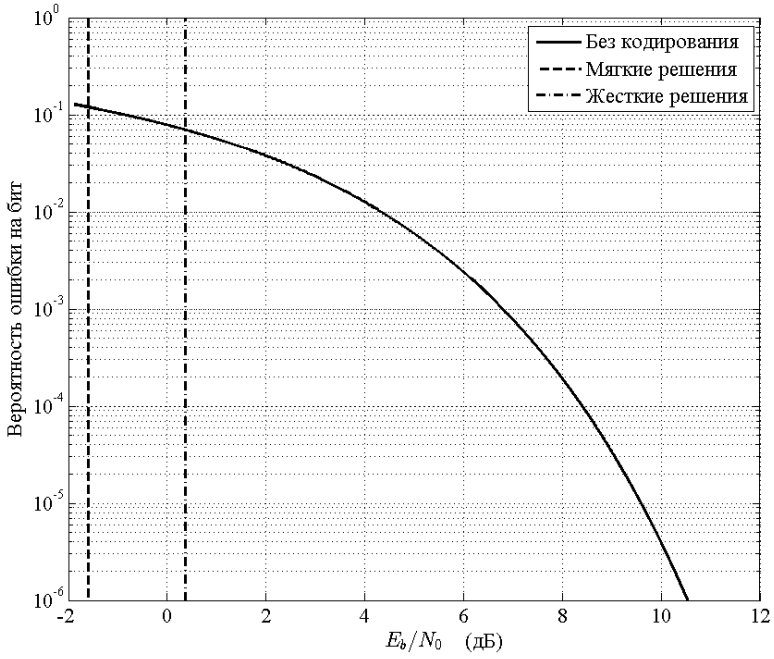


Рис. 1.3. Характеристики некодированной передачи и пределы Шеннона для декодирования с жесткими и мягкими решениями

К сожалению, если принять во внимание некоторые важные практические ограничения, то передача при $E_b/N_0 = -1.59$ дБ на бит окажется невозможной. Например, предположим, что информация передается двоичными сигналами, и на приемной стороне сначала выносится решение (0 или 1) о каждом сигнале, а уже затем выполняется декодирование, т.е. выбор кодового слова, ближайшего к принятой из канала двоичной последовательности. Такой метод обработки сигналов называют *декодированием с жесткими решениями.* Это предположение возвращает нас к показанной на рис. 1.1 модели ДСК.

Обозначим через p_0 вероятность ошибочного решения о сигнале энергии E_s. Максимально достижимая скорость кодирования R_c в битах на символ канала не превышает пропускной способности ДСК

с переходной вероятностью p_0, т.е., согласно (1.1), (1.7) и (1.8),

$$R_c < 1 - h(p_0) = 1 - h\left(Q(\sqrt{2E_s/N_0})\right).$$

Величину E_s/N_0 называют *отношением сигнал/шум на сигнал.*

Принятое выше условие о бесконечной полосе означает по сути, что при заданной скорости передачи в битах в секунду мы укорачиваем сигналы, уменьшаем энергию E_s каждого двоичного сигнала и при этом имеем $p_0 \to 1/2$. Раскладывая двоичную энтропию $h(p_0)$ по степеням $(1/2 - p_0)$ в окрестности точки $p_0 = 1/2$, получаем

$$R_c < 2\left(\frac{1}{2} - p_0\right)^2 \log_2 e$$

(можно использовать разложение $\ln(1+x) = x - x^2/2 + x^3/3 - x^4/4 + \ldots$).

Для функции $Q(x)$ при малых значениях аргумента имеет место приближенное равенство

$$Q(x) \approx \frac{1}{2} - \frac{x}{\sqrt{2\pi}}.$$

В итоге получаем

$$R_c < \frac{2\log_2 e}{\pi}\frac{E_s}{N_0}.$$

Поскольку R_c представляет собой количество бит, доставляемое сигналом энергии E_s, затраты энергии на бит равны $E_b = E_s/R_c$. Для отношения сигнал/шум на бит E_b/N_0 имеем следующее ограничение на минимальную его величину, при которой возможна надежная передача информации

$$\frac{E_b}{N_0} = \frac{E_s}{R_c N_0} > \frac{\pi}{2}\ln 2 = 1.09 = 0.37 \quad \text{дБ}.$$

Эти подсчеты показывают, что потери, связанные с принятием жестких решений на выходе демодулятора, составляют приблизительно 2 дБ. Асимптота, соответствующая декодированию с жесткими решениями, показана на рис. 1.3.

Вернемся к *декодированию с мягкими решениями*, т.е. к такой системе связи, в которой при выборе наилучшего кодового слова учитывается надежность полученных от демодулятора символов. Примем теперь во внимание ограничение на полосу, точнее говоря, эффективность использования предоставленной для передачи информации полосы частот.

Спектральной эффективностью кодирования называется величина

$$\beta = \frac{R_t}{W} \quad \frac{\text{бит}}{\text{с} \cdot \text{Гц}}. \tag{1.9}$$

Поясним смысл этого параметра системы связи.

Ограничение на полосу W можно интерпретировать как ограничение на минимальную допустимую длительность сигнала T, поскольку W обратно пропорциональна T. При использовании корректирующих кодов в передаваемую последовательность вносится избыточность. Обозначим через R_c скорость кода, т.е. долю информационных символов в передаваемой последовательности. Для того чтобы сравнивать эффективность системы с кодированием и без кодирования, мы должны приравнять их скорости передачи информации, измеренные в битах в секунду. Мы можем это сделать, пропорционально уменьшив длительность сигналов и, тем самым, расширив полосу. В системе с кодированием длительность сигнала должна быть уменьшена до R_cT, соответственно, полоса частот пропорционально увеличивается до W/R_c. Если спектральная эффективность (число передаваемых бит на один герц полосы) без кодирования была равной β, она уменьшается при введении кодов до величины βR_c. Таким образом, спектральная эффективность кодирования адекватно отражает плату за использование кодов — увеличение ширины полосы частот. Поэтому правильно сравнивать различные способы кодирования при фиксированной спектральной эффективности.

Из формулы (1.3) для пропускной способности гауссовского канала с ограниченной полосой с учетом (1.5) имеем

$$\frac{R_t}{W} < \log_2\left(1 + \frac{R_t}{W}\frac{E_b}{N_0}\right) \quad \text{бит/с}.$$

Отсюда находим границу снизу на отношение сигнал/шум при заданной спектральной эффективности

$$\frac{E_b}{N_0} > \frac{2^\beta - 1}{\beta}.$$

Наиболее наглядными и практически интересными получаются результаты в случае, когда рассматривается передача сигналов со «скоростью Найквиста», т.е. при длительности сигналов $T = 1/(2W)$. Это минимально возможная длительность ортогональных сигналов в заданной полосе W. При этом скорость передачи в битах в секунду становится равной

$$R_t = \frac{R_c}{T} = 2WR_c$$

и соответствующая ей спектральная эффективность равна

$$\beta = 2R_c. \tag{1.10}$$

Получаем еще одну границу снизу на отношение сигнал/шум при заданной скорости используемого корректирующего кода

$$\frac{E_b}{N_0} > \frac{2^{2R_c} - 1}{2R_c}.$$

Вернемся к рис. 1.3. На графике кривая вероятности ошибки без кодирования пересекается с асимптотами Шеннона. Из этого можно сделать неправильный вывод о том, что при некоторой вероятности ошибки кодирование оказывается неэффективным. В действительности, опираясь на теорию кодирования при заданном критерии качества, можно уточнить оценки Шеннона для малых отношений сигнал/шум на бит с учетом того обстоятельства, что допустима некоторая вероятность ошибки. Эта поправка к пределу Шеннона впервые была использована Мак-Элисом [24]. Результаты подсчетов с учетом такого уточнения представлены на рис. 1.4.

В реальных системах скорость $R_c \leq 1/2$ используется в относительно широкополосных каналах связи: некоторых каналах мобильной связи, спутниковых системах, каналах дальней космической связи. Напротив, скорость $R_c > 1/2$ соответствует узкополосным

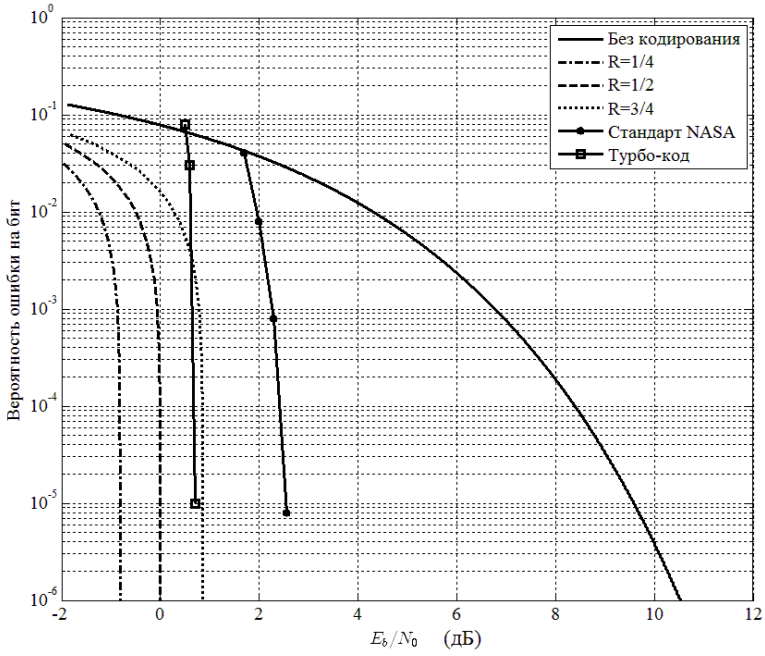


Рис. 1.4. Характеристики некодированной передачи, стандарта NASA, турбо-кодов и пределы Шеннона для скоростей кодов 1/4, 1/2 и 3/4

каналам, например, каналам телефонной связи. Примеры границ Шеннона для скоростей $R_c = 1/4$, $1/2$ и $3/4$ вместе с кривой для некодированной передачи показаны на рис. 1.4. Кроме того, показаны характеристики нескольких вариантов построения реальных систем связи со скоростью кодов близкой к $1/2$.

Кривая, помеченная как «Стандарт NASA», отражает характеристики каскадного кода (см. параграфы 10.2, 10.4) на основе кода Рида–Соломона (255, 223) (глава 6) и сверточного кода со скоростью $R = 1/2$ (глава 8). Результаты моделирования заимствованы из [115]. Этот каскадный код использован для связи с космическим кораблем «Галилео», запущенным в 1989 году к Юпитеру. Кривая «Турбо-код» отражает характеристики турбо-кода (см. параграф

10.3) длиной 65 536 со скоростью 1/2 из работы [47]. Этот код слишком сложен для практического применения, но МППЧ-код (глава 11) примерно такой же длины и примерно с такими же характеристиками помехоустойчивости уже применяется в современном международном стандарте спутникового цифрового телевидения DVB-S2 [119].

Мы видим, что реально достижимый эффект от кодирования составляет примерно 9–10 дБ. Из рисунка напрашивается также неправильный вывод о том, что характеристики реальных систем уже достаточно близки к теоретическому пределу и поиск новых кодов уже неинтересен. Действительно, развитие теории кодирования и технологии реализации сложных электронных схем позволили в последние 10–15 лет совершить большой шаг вперед. Однако нерешенных задач остается очень много. В частности, коды, характеристики которых показаны на рис. 1.4, имеют чрезвычайно большую длину и высокую сложность реализации кодирования и декодирования. Их применение, например, в мобильных устройствах, невозможно. Поиск простых и эффективных методов защиты от ошибок остается актуальной задачей.

Более того, физические ресурсы для передачи данных, например, полоса доступных для радиосвязи частот, ограничены, и все острее стоит проблема их наиболее эффективного использования. Вернемся к критерию спектральной эффективности. Согласно (1.10), спектральная эффективность равна удвоенной скорости кода. Отсюда следует, что при двоичном кодировании $\beta \leq 2$, что означает, что на один герц полосы можно передать не более 2 бит в секунду. Как добиться более высокой спектральной эффективности?

Ответ почти очевиден: нужно использовать недвоичные коды. Но для передачи недвоичных кодов потребуются недвоичные сигналы. Вместо уже привычных нулей и единиц, которым сопоставлялись положительные и отрицательные импульсы, приходится использовать наборы сигналов, отличающихся между собой значениями амплитуд, частот и фаз. Над такими наборами сигналов тоже можно строить коды. Прорыв в теории таких кодов над сигналами произошел в 1982 году благодаря Унгербоеку [108], построившему целый класс кодов, которые называют *сигнально-кодовыми*

конструкциями. Ориентиром при построении таких кодов служит уже не расстояние Хэмминга, а расстояние Евклида между кодовыми словами, символы которых — вещественные числа. Задача построения кодов с высокой спектральной эффективностью будет кратко рассмотрена в параграфе 10.4. Отметим, что «зазор» между достигнутыми на сегодня результатами и предельно достижимыми характеристиками (пропускной способностью) тем больше, чем больше значение требуемой спектральной эффективности β.

1.2. Обзор кодов для защиты информации от ошибок

В последующих главах мы рассмотрим довольно много классов кодов и алгоритмов кодирования и декодирования. Все ли эти коды важны? Нельзя ли ограничиться изучением самых «лучших» кодов?

Многообразие изучаемых кодов связано, во-первых, с многообразием задач передачи информации, и, во-вторых, с тем, что теория кодов динамично развивается, причем иногда почти забытые идеи и методы неожиданно снова обретают актуальность.

Фундамент теории кодирования был заложен К. Шэнноном. Перевод его основных работ опубликован в сборнике [19]. Уже первые работы Шэннона указывают на кодирование как на способ обеспечения надежной передачи информации. В 1950-е появились первые конструкции кодов (например, коды Хэмминга, Голея, Рида–Маллера), сформировалось понятие линейных кодов, разделение кодов на блоковые и сверточные. 1960-е годы были очень плодотворными для теории кодирования. Для блоковых кодов появились на свет алгебраические конструкции (коды БЧХ и Рида–Соломона), допускающие декодирование с малой сложностью. Были разработаны эффективные методы декодирования сверточных кодов (последовательное декодирование, алгоритм Витерби). Помимо этого, были изобретены каскадные коды Форни, коды Галлагера с малой плотностью проверок на четность.

В 1970-80-е годы были разработаны сигнально-кодовые конструкции, коды Гоппы и др. Начиная с 1990-х годов, внимание боль-

шого числа исследователей привлекли коды, для которых эффективно итеративное декодирование — турбо-коды и коды с малой плотностью проверок на четность. Сегодня именно эти коды — лидеры как по числу научных публикаций, так и по применению во вновь разрабатываемых системах связи.

Среди достижений последнего времени нужно указать на полярные коды Арикана [30].

Основные результаты теории кодирования ориентированы на применение кодов либо в двоичном симметричном канале, либо в канале с аддитивным гауссовским шумом. Эти модели далеко не всегда адекватно описывают реальные условия применения кодов. Поэтому параллельно с развитием общей теории кодирования велись поиски способов адаптации корректирующих кодов к специфическим средам распространения или хранения данных, накладывающим ограничения на допустимые коды, либо требующим изменения критериев выбора кодов. Наиболее характерные примеры — магнитные носители, лазерные компакт-диски, оптоволоконные и квантовые каналы, каналы электрических сетей, многоантенные системы мобильной связи (multiple input multiple output (MIMO)) и др.

Даже поверхностная библиография по теории кодирования включала бы тысячи работ. В табл. 1.2 приведены основные разделы теории кодирования и указаны ссылки на источники, в основном монографии и учебники, по которым можно ознакомиться с терминологией и основными понятиями по каждому направлению. Список литературы не претендует на полноту. Подбирались материалы, по возможности адресованные студентам, на русском языке или переведенные на русский, написанные широко известными специалистами по соответствующим разделам.

Таблица 1.2. Литература по основным классам корректирующих кодов

Раздел теории кодирования	Глава или параграф учебника	Литература
Общая теория кодирования (построение кодов, границы)		
Линейные блоковые коды	2,3	[12, 8]
Коды Рида–Маллера	—	[10, 15]
Нелинейные блоковые коды	—	[14, 10, 15]
Коды для криптографии	—	[15]
Каскадные конструкции	10.1,10.2	[18, 11, 23]
Алгоритмы декодирования линейных блоковых кодов		
Мажоритарное декодирование	—	[9, 12]
Алгоритмы для каналов с мягкими решениями	4	[11, 17, 26, 27]
Циклические коды, декодирование, обобщения		
Общая теория	5,6	[12, 2]
Методы декодирования	7	[2, 17]
Альтернантные коды, коды Гоппы, алгебро-геометрические коды	—	[10, 8]
Декодирование по алгоритму Гурусвами–Судана	—	[8, 27]
Сверточные коды и их декодирование		
Последовательное декодирование	—	[4, 6, 25]
Алгебраическая структура	9	[25]
Характеристики и таблицы	8	[25, 7, 26, 3]
Коды для итеративного декодирования		
Теория и построение кодов	10.3,11	[5, 16, 28]
Сверточные низкоплотностные коды	—	[35]
Полярные коды	—	[30]
Коды для специальных условий применения		
Модуляционные коды	10.4	[22, 11]
Коды для магнитной записи	—	[8]
Квантовые коды	—	[60]
Коды для MIMO	—	[21, 27]
Коды для сетей связи	—	[89, 85]

В данном учебнике акцент сделан на тех методах кодирования, которые нашли отражение в сегодняшней практике защиты информации от ошибок. Как видно из табл. 1.2, вне рамок рассмотрения оказались некоторые важные разделы теории кодирования, как классические, так и появившиеся относительно недавно. Таким образом, изучение данного пособия — не больше чем первый шаг в освоении увлекательной и актуальной области знаний.

Выводы

- Потенциально достижимая скорость передачи информации в канале связи в битах в секунду определяется ограничениями на мощность сигнала и полосу частот. Последнее ограничение может быть пересчитано в минимальную допустимую скорость кодирования в битах на сигнал.

- Применение кодов, исправляющих ошибки, обеспечивает большой энергетический выигрыш по сравнению с передачей без кодирования. Для обеспечения высокой эффективности использования ресурсов канала коды должны быть длинными и допускать декодирование с мягкими решениями.

- Важной характеристикой кода является его минимальное расстояние. Задача построения хороших кодов может быть сформулирована как задача построения кодов с наибольшим возможным минимальным расстоянием при заданной скорости.

- Минимальное расстояние — важный параметр кода, но не только оно принимается во внимание при выборе способа защиты от ошибок. С учетом ограничений на сложность реализации кодера и декодера при решении многих практических задач лучшими оказываются коды с относительно небольшим минимальным расстоянием, но зато допускающие простое декодирование.

Задачи

1. Для последовательностей, символы которых выбираются из алфавита $A = \{0, \ldots, q-1\}$, расстоянием Хэмминга называется число несовпадающих символов. Докажите, что расстояние Хэмминга удовлетворяет аксиомам метрики, т.е. что оно неотрицательно, симметрично и для него имеет место неравенство треугольника.

2. Пусть $\boldsymbol{x}, \boldsymbol{y} \in A^n = \{0, \ldots, q-1\}^n$ — соответственно входная и выходная последовательности канала связи, заданного условными вероятностями $p(\boldsymbol{y}|\boldsymbol{x})$. Предположим, что канал стационарен и является каналом без памяти, т.е.

$$p(\boldsymbol{y}|\boldsymbol{x}) = \prod_{i=1}^{n} p(y_i|x_i),$$

и переходные вероятности $p(y_i = x|x_i = y)$ не зависят от i.

Декодированием по максимуму апостериорной вероятности (МАВ) кода $C = \{\boldsymbol{c}_m\}, m = 1, \ldots, M$ над алфавитом A называют принятие решения в пользу слова с номером

$$\hat{m} = \arg\max_m p(\boldsymbol{c}_m|\boldsymbol{y}).$$

Декодированием по максимуму правдоподобия (МП) кода $C = \{\boldsymbol{c}_m\}, m = 1, \ldots, M$ над алфавитом A называют принятие решения в пользу слова с номером

$$\hat{m} = \arg\max_m p(\boldsymbol{y}|\boldsymbol{c}_m).$$

Если в правилах МП и МАВ одинаковое значение максимума целевой функции достигается для множества из нескольких значений m, то производится случайный выбор решения из этого множества. Каким должно быть распределение вероятностей на кодовых словах, чтобы декодирование по МП и декодирование по МАВ были эквивалентными независимо от переходных вероятностей канала?

3. Докажите, что декодирование по МАВ минимизирует среднюю по всем сообщениям вероятность ошибки декодирования $P_e = \sum\limits_m p(\boldsymbol{c}_m) \Pr(\hat{m} \neq m | \boldsymbol{x} = \boldsymbol{c}_m)$.

   Подсказка: запишите среднюю вероятность правильного декодирования в виде

$$P_c = \sum_m \sum_{\boldsymbol{y} \in R_m} p(\boldsymbol{y}|\boldsymbol{c}_m) p(\boldsymbol{c}_m),$$

   где R_m представляет собой множество тех $\boldsymbol{y}$, по которым принимается решение в пользу $\boldsymbol{c}_m$.

4. Предположим, что входные и выходные символы стационарного канала без памяти выбираются из алфавита $X = Y = \{0, \ldots, q-1\}$ и на этом алфавите определена операция сложения по модулю q. Такой канал называют симметричным, если

$$p(y|x) = \begin{cases} \frac{p_0}{q-1}, & x \neq y \\ 1 - p_0, & x = y \end{cases}.$$

   При каких условиях декодирование по минимуму расстояния Хэмминга (МРХ) совпадает с декодированием по МП?

5. Рассмотрим код $C = \{c_m\}, m = 1, \ldots, M$ над алфавитом A и предположим, что минимальное из попарных расстояний Хэмминга между последовательностями кода равно d. Эту величину называют *минимальным расстоянием кода.* Покажите, что такой код исправляет любые комбинации ошибок кратности не более

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

   где скобки $\lfloor \cdot \rfloor$ обозначают округление числа вниз до ближайшего целого, и обнаруживает любые комбинации ошибок кратности не более $d - 1$.

6. Для канала с двоичным входом и гауссовским шумом, описываемого переходными вероятностями (1.6), напишите МАТЛАБ-программу построения зависимости вероятности

ошибки от отношения сигнал/шум в канале связи (кривая 1 на рис. 1.3 и 1.4).

7. Для канала со стираниями, показанного на рис. 1.1, зависимость пропускной способности от параметров канала p и ε выражается формулой

$$C = (1-\varepsilon)\left(1 - h\left(\frac{p}{1-\varepsilon}\right)\right).$$

Предположим, что этот канал получен из канала с гауссовским шумом принятием решения о стирании всякий раз, когда модуль значения на выходе канала меньше порога Δ. Для фиксированного значения отношения сигнал/шум E_s/N_0 постройте график зависимости пропускной способности от величины Δ. (Замечание: для получения содержательных результатов отношение E_s/N_0 нужно выбрать таким, чтобы при Δ пропускная способность C была заметно меньше 1).

8. Сформулируйте условия, при выполнении которых декодирование по МП в канале со стираниями (ДСтК) сводится к декодированию по минимуму расстояния Хэмминга на нестертых позициях. Сколько стираний исправляет код с минимальным расстоянием d? Сколько ошибок исправляет код с минимальным расстоянием d при наличии s стираний?

9. Предположим, что код с минимальным расстоянием d используется в ДСК в режиме исправления ошибок кратности до $t = \lfloor (d-1)/2 \rfloor$ (для некоторых кодов возможно только такое декодирование, поскольку декодирование по МП намного сложнее). Запишите формулу для вероятности ошибки при таком декодировании. Запишите также формулу для вероятности ошибочного декодирования в ДСтК при исправлении t ошибок и s стираний. Какие комбинации ошибок исправляет декодер МП и не исправляет декодер, рассматриваемый в данной задаче?

10. Код Голея имеет длину $n = 23$, число информационных символов $k = 12$, минимальное расстояние $d = 7$. Считая, что этот

код используется в ДСК в режиме исправления ошибок, постройте зависимость энергетического выигрыша кодирования от характеристик канала связи, т.е. переходной вероятности ДСК и отношения сигнал/шум на бит в исходном канале с гауссовским шумом.

11. Рассмотрим два гипотетических кода: код длины n, исправляющий t ошибок, и код длины $2n$, исправляющий $2t$ ошибок. Какой из кодов лучше? Подтвердите ваши соображения численными примерами.

Приложение. Биномиальное и полиномиальное распределения

Начнем с того, что рассмотрим множество последовательностей вида $\boldsymbol{x} = (x_1, \ldots, x_n)$, в которых элемент x_i может принимать одно из M_i значений, $i = 1, \ldots, n$. Очевидно, число различных $\boldsymbol{x}$ равно

$$|\{\boldsymbol{x} = (x_1, ..., x_n) : x_i \in \{0, ..., M_i - 1\}\}| = \prod_{i=1}^{n} M_i. \qquad (1.11)$$

В частности, число различных M-ичных последовательностей длины n равно M^n.

Предположим, что все компоненты $\boldsymbol{x} = (x_1, \ldots, x_n)$ выбираются из одного и того же множества объема M, но «без возвращения», т.е. один и тот же элемент не может повториться в $\boldsymbol{x}$ больше одного раза. Число таких последовательностей представляет собой число *размещений* из M элементов по n и равно

$$A_M^n = M(M-1) \times \ldots \times (M-n+1) = \frac{M!}{(M-n)!}. \qquad (1.12)$$

Эта формула следует из (1.11). В частном случае, когда $M = n$, речь идет о всевозможных *перестановках* из M элементов. Число различных перестановок P_M равно

$$P_M = M! \qquad (1.13)$$

Итак, существует A_M^n различных слов длины n из неповторяющихся букв алфавита объема M. При этом подразумевается, что порядок следования букв в словах существенен, т.е. слова, отличающиеся порядком букв, различны. Подсчитаем теперь, сколько можно построить различных неупорядоченных подмножеств объема n из алфавита объема M. Поскольку $P_n = n!$ различных последовательностей образуют теперь одно подмножество, число различных подмножеств равно

$$\begin{aligned} C_M^n &= \binom{M}{n} = \frac{A_M^n}{P_n} = \frac{M(M-1)\times\ldots\times(M-n+1)}{n!} = \\ &= \frac{M!}{n!(M-n)!}. \end{aligned} \tag{1.14}$$

Это число называют числом *сочетаний* из M элементов по n. Здесь мы привели два обозначения для числа сочетаний. Первое, C_M^n, распространено в российской математической литературе. Второе, $\binom{M}{n}$ используют в международных математических и технических изданиях. Мы будем придерживаться второго обозначения. Формула (1.14) не имеет смысла при $M, n \le 0$ и при $M < n$. Для всех этих случаев мы формально определим значения числа сочетаний равными нулю, т.е.

$$\binom{M}{n} = 0, \text{если } M \le 0, \text{или } n \le 0, \text{или } M < n. \tag{1.15}$$

Числа $\binom{M}{n}$ называют биномиальными коэффициентами в связи с использованием в известной формуле бинома Ньютона

$$(a+b)^n = \sum_{i=0}^{n} \binom{n}{i} a^i b^{n-i}.$$

Подсчитаем число двоичных последовательностей длины n, содержащих заданное число τ_1 единиц и $\tau_0 = n - \tau_1$ нулей. Заметим, что номера позиций единиц образуют неупорядоченное подмножество множества чисел $\{1, \ldots, n\}$. Из (1.14) следует, что искомое число последовательностей равно

$$N(\tau_0, \tau_1) = \binom{n}{\tau_1} = \frac{n}{\tau_0!\tau_1!}. \tag{1.16}$$

Обобщим эту формулу на множество последовательностей над произвольным алфавитом $X = \{0, \ldots, M-1\}$. Напомним, что композицией последовательности x мы называем вектор $\boldsymbol{\tau}(\boldsymbol{x}) = (\tau_0(\boldsymbol{x}), \ldots, \tau_{M-1}(\boldsymbol{x}))$, в котором $\tau_i(\boldsymbol{x})$ обозначает число элементов $x_t = i$ в последовательности $\boldsymbol{x} = (x_1, \ldots, x_n)$. Ближайшая цель состоит в нахождении числа последовательностей $\boldsymbol{x}$ с заданной композицией $\boldsymbol{\tau} = (\tau_0, \ldots, \tau_{M-1})$.

Начнем с того, что положим $M = 3$ и найдем число последовательностей длины n с композицией $\boldsymbol{\tau} = (\tau_0, \tau_1, \tau_2)$, $n = \tau_0 + \tau_1 + \tau_2$. Для этого зафиксируем некоторое расположение нулей (это можно сделать $\binom{n}{\tau_0}$ способами) и подсчитаем число различных расположений τ_1 единиц на оставшихся $n - \tau_0$ позициях. Это число, как мы знаем, равно $\binom{n-\tau_0}{\tau_1}$. Поскольку каждому расположению нулей соответствует такое количество расположений единиц, общее число последовательностей равно

$$N(\boldsymbol{\tau}) = \binom{n}{\tau_0}\binom{n-\tau_0}{\tau_1} = \frac{n!}{\tau_0!(n-\tau_0)!}\frac{(n-\tau_0)!}{\tau_1!(n-\tau_0-\tau_1)!} = \frac{n!}{\tau_0!\tau_1!\tau_2!}.$$

Аналогично для алфавита произвольного объема M легко вывести формулу

$$N(\boldsymbol{\tau}) = \frac{n!}{\tau_0! \ldots \tau_{M-1}!}. \tag{1.17}$$

Эта формула является естественным обобщением формулы для биномиальных коэффициентов. Числа $N(\boldsymbol{\tau})$ называют полиномиальными или мультиномиальными коэффициентами. Также естественно обобщается формула бинома Ньютона на случай возведения в степень произвольного числа слагаемых:

$$(a_0 + \ldots + a_{M-1})^n = \sum_{\tau:\tau_0+\ldots+\tau_{M-1}=n} N(\boldsymbol{\tau}) \prod_{i=0}^{M-1} a_i^{\tau_i}.$$

Полученные формулы достаточно просты для подсчетов с помощью компьютера, но неудобны для анализа. Ниже мы получим асимптотические формулы, достаточно точные при больших длинах последовательностей n. Все вычисления основаны на формуле Стирлинга

$$\sqrt{2\pi n} n^n e^{-n} \exp\left\{\frac{1}{12n+1}\right\} < n! < \sqrt{2\pi n} n^n e^{-n} \exp\left\{\frac{1}{12n}\right\}. \tag{1.18}$$

Ее подстановка в (1.17) дает

$$N(\boldsymbol{\tau}) \;<\; (2\pi n)^{-\frac{M-1}{2}} 2^{n\log_2 n - \sum\limits_i \tau_i \log_2 \tau_i} \left(\prod_i \frac{n}{\tau_i}\right)^{1/2} \times$$

$$\times \;\exp\left\{\frac{1}{12n} - \sum_i \frac{1}{12\tau_i + 1}\right\}. \qquad (1.19)$$

Из этой формулы получаем верхнюю оценку для логарифма числа последовательностей с заданной композицией

$$\log_2 N(\boldsymbol{\tau}) < nH(\hat{\boldsymbol{p}}) - \frac{M-1}{2}\log_2(2\pi n) - \frac{1}{2}\sum_i \log_2 \hat{p}_i, \qquad (1.20)$$

где в (1.18)–(1.20) индекс i пробегает те значения, для которых τ_i положительно, через $\hat{p}_i$ обозначены вычисленные по $\boldsymbol{x}$ оценки вероятностей букв, т.е. $\hat{p}_i = \tau_i/n$; через $H(\hat{\boldsymbol{p}})$ обозначена энтропия ансамбля X, вычисленная по распределению вероятностей $\hat{\boldsymbol{p}} = (\hat{p}_0, \ldots, \hat{p}_{M-1})$. Помимо (1.20) можно также записать менее точную, но более компактную оценку

$$\log_2 N(\boldsymbol{\tau}) < nH(\hat{\boldsymbol{p}}) - \frac{M-1}{2}\log_2(2\pi) + \frac{1}{2}\log_2 \frac{n}{n-M+1}. \qquad (1.21)$$

Рассмотрим теперь последовательность из n опытов, исходы которых — числа множества $X = \{0, \ldots, M-1\}$. Предположим, что опыты независимы, и вероятности исходов $\{p_0, \ldots, p_{M-1}\}$ не меняются от опыта к опыту. Такую последовательность опытов иногда называют схемой Бернулли. Некоторые авторы используют термин «схема Бернулли» только для опытов с двумя исходами ($M = 2$), «удача» и «неудача». Пусть $\boldsymbol{x} = (x_1, \ldots, x_n)$ — некоторая реализация в данной схеме, тогда, в силу независимости исходов опытов,

$$p(\boldsymbol{x}) = \prod_{i=1}^{n} p(x_i).$$

В данном произведении сомножители принимают не более M различных значений. Перегруппировав сомножители, получим

$$p(\boldsymbol{x}) = \prod_{j=0}^{M-1} p_j^{\tau_j(\boldsymbol{x})},$$

где $\boldsymbol{\tau}(\boldsymbol{x}) = (\tau_0(\boldsymbol{x}), \ldots, \tau_{M-1}(\boldsymbol{x}))$ — композиция последовательности $\boldsymbol{x}$. Таким образом, вероятность конкретной последовательности в схеме Бернулли полностью определяется ее композицией и одинакова для последовательностей с одинаковой композицией. Для того, чтобы подсчитать вероятность заданной композиции $\boldsymbol{\tau}$, нужно просуммировать вероятности последовательностей с такой композиции. Получаем из (1.16)

$$p(\boldsymbol{\tau}) = N(\boldsymbol{\tau}) \prod_{j=0}^{M-1} p_j^{\tau_j} = \frac{n!}{\prod\limits_{j=0}^{M-1} \tau_j!} \prod_{j=0}^{M-1} p_j^{\tau_j}. \tag{1.22}$$

Это распределение вероятностей называют полиномиальным или мультиномиальным. Частным случаем формулы (1.22) является формула биномиального распределения. Вероятность k успехов в серии из n опытов, при которых вероятность успеха в одном опыте равна p, равна

$$p_n(k) = \binom{n}{k} p^k (1-p)^{n-k}.$$

Например, при подсчете вероятности ошибки декодирования для кода, исправляющего t ошибок, используют формулу

$$P_e = \sum_{i=t+1}^{n} \binom{n}{i} p^i (1-p)^{n-i}. \tag{1.23}$$

Другой пример: вычисление вероятности ошибки декодирования кода с минимальным расстоянием d при исправлении всех комбинаций t ошибок и s стираний при условии, что $2t + s + 1 \leq d$. Обозначив вероятности ошибки в канале через p и стирания через ε, из (1.22) получаем формулу

$$P_e = \sum_{(t,s):2t+s+1>d} \frac{n!}{t!s!(n-t-s)!} p^t \varepsilon^s (1-p-\varepsilon)^{n-t-\varepsilon}. \tag{1.24}$$

С помощью формулы Стирлинга для больших n и t можно вместо (1.23) получить приближенные формулы:

$$\sum_{i=j}^{n} \binom{n}{i} p^i (1-p)^{n-i} \;\geq\; \binom{n}{j} p^j (1-p)^{n-j}; \tag{1.25}$$

$$\sum_{i=j}^{n} \binom{n}{i} p^i (1-p)^{n-i} \;\leq\; \frac{j(1-p)}{j(1-p)-(n-j)p} \binom{n}{j} p^j (1-p)^{n-j}. \tag{1.26}$$

Для подсчета биномиальных коэффициентов можно воспользоваться одним из приближений для полиномиальных распределений (1.19, 1.20), подставив $M = 2$, но более точный результат даст формула

$$\sqrt{\frac{n}{8j(n-j)}} 2^{nh(j/n)} \leq \binom{n}{j} \leq \sqrt{\frac{n}{2\pi j(n-j)}} 2^{nh(j/n)}, \tag{1.27}$$

где $h(x) = -x\log_2(x) - (1-x)\log_2(1-x)$. Из приведенных выше формул непосредственно следуют предельные соотношения, которые пригодятся при анализе поведения кодов и методов декодирования с ростом длины кодовых слов:

$$\lim_{n\to\infty} \frac{1}{n} \log_2 \binom{n}{n\delta} \;=\; h(\delta); \tag{1.28}$$

$$\lim_{n\to\infty} \frac{1}{n} \log_2 \sum_{i=0}^{n\delta} \binom{n}{i} \;=\; h(\delta); \tag{1.29}$$

$$\lim_{n\to\infty} \frac{1}{n} \log_2 \frac{n!}{(np_0)!\ldots(np_{M-1})!} \;=\; H(\boldsymbol{p}); \tag{1.30}$$

$$\lim_{n\to\infty} \frac{1}{n} \log_2 \sum_{i=n\beta+1}^{n} \binom{n}{i} p^i (1-p)^{n-i} \;=\; T(p,\beta), \tag{1.31}$$

где $\delta \in (0,1)$, $\boldsymbol{p} = (p_0, ..., p_{M-1})$ — вектор вероятностей, $p < 1/2$, $\beta \in (p,1)$,

$$T(p,\beta) = p\log_2 \frac{p}{\beta} + (1-p)\log_2 \frac{1-p}{1-\beta}.$$

2. Линейные коды

Мы знаем из предыдущей главы, что коды — это наборы кодовых слов. Чтобы быть эффективными, коды должны быть длинными, обладать большим минимальным расстоянием при заданной скорости, иметь разумную сложность кодирования и декодирования.

Даже для не очень длинных кодов списки кодовых слов слишком велики, чтобы можно было сохранять их в памяти кодера и декодера. Наш первый шаг к сокращению сложности описания кодов — построение линейных кодов, т.е. кодов, слова которых образуют линейное пространство. Для описания линейного кода достаточно задать базис пространства, а кодирование сводится к умножению на матрицу. К сожалению, в этом разделе мы не получим ответ на вопрос, как декодировать линейные коды с приемлемой для практики сложностью. Задача декодирования линейных кодов общего вида почти так же сложна, как и задача декодирования кода, заданного списком кодовых слов. Чтобы упростить декодирование, нам придется дополнительно сузить класс кодов. Собственно, этому вопросу будут посвящены все последующие разделы курса.

2.1. Арифметика пространства двоичных последовательностей

Над двоичными символами можно выполнять операции сложения и умножения. Мы рассматриваем числа 0 и 1 как элементы кольца вычетов по модулю 2 и как группу по умножению. Иными словами, множество чисел $\{0,1\}$ образует простейшее *поле* из двух элементов F_2. Ниже приведены таблицы сложения и умножения (табл.

2.1). Аналогично определяется арифметика поля вычетов по модулю произвольного простого q. Это поле мы обозначаем через F_q.

Таблица 2.1. Таблицы сложения и умножения в поле F_2

$+$	0	1
0	0	1
1	1	0

$\times$	0	1
0	0	0
1	0	1

Упражнение 2.1. Постройте таблицы сложения и умножения для $q = 3$.

От вычислений над скалярными величинами перейдем к изучению действий над последовательностями (векторами). Для произвольного множества A мы через A^n обозначаем множество последовательностей длины n, составленных из элементов множества A.

Определим сумму векторов $\boldsymbol{x} = (x_1, \ldots, x_n)$ и $\boldsymbol{y} = (y_1, \ldots, y_n)$ как покомпонентную сумму $\boldsymbol{x} + \boldsymbol{y} = (x_1 + y_1, \ldots, x_n + y_n)$. Умножение вектора $\boldsymbol{x}$ на скалярную величину $a \in F_2$ определим как $a\boldsymbol{x} = (ax_1, \ldots, ax_n)$.

Множество векторов, замкнутое относительно операций сложения и умножения на скаляр, называется *линейным векторным пространством.*

Пример 2.1. Примеры двоичных линейных пространств:

А) $\{00, 01, 10, 11\}$,

Б) $\{000, 001, 010, 011, 100, 101, 110, 111\}$,

В) $\{000, 011, 101, 110\}$,

Г) $\{000, 111\}$.

Множество F_2^n всех двоичных последовательностей длины n, очевидно, является линейным пространством. В частности, пространства примеров А и Б совпадают с F_2^2 и F_2^3. Такие пространства

содержат в общем случае 2^n векторов. Пространство примера В содержится в F_2^3, но не совпадает с ним, т.е. является его *линейным подпространством.*

Заметим, что каждое из трех пространств примера 2.1 содержит нулевой вектор. Этим свойством обладает любое линейное пространство (почему?).

Ключевым понятием в описании линейных пространств является понятие базиса.

Базисом линейного пространства называется наибольшее возможное множество линейно независимых векторов пространства. Число базисных векторов называется *размерностью пространства.*

Пример 2.2. Для линейных пространств примера 2.1 имеем:

А) Размерность равна 2, пример базиса: $\{01, 10\}$.

Б) Размерность равна 3, примеры базисов: $\{001, 010, 100\}, \{110, 011, 111\}$.

В) Размерность равна 2, пример базиса: $\{011, 101\}$.

Г) Размерность равна 1, базис: $\{111\}$.

Основные свойства базиса и порожденного им пространства:

- Любой вектор пространства единственным образом может быть представлен в виде линейной комбинации базисных векторов.
- Число элементов векторного пространства размерности k равно q^k.

Доказательство свойств оставляем читателю в качестве упражнения.

Определение 2.1. *Линейным q-ичным кодом (n, k)-кодом C называется любое k-мерное подпространство пространства F_q^n всевозможных векторов длины n.*

Скоростью линейного (n,k)*-кода* называется отношение $R = k/n$. В двоичном случае ($q = 2$) скорость измеряется в битах на символ канала.

Мы уже замечали, что хорошим будет код, слова которого «далеки» друг от друга в смысле расстояния Хэмминга, которое для последовательностей $\boldsymbol{x}$, $\boldsymbol{y}$ длины n определяется как

$$d(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{n} d(x_i, y_i),$$

где

$$d(x,y) = \begin{cases} 0, & x = y; \\ 1, & x \neq y. \end{cases}$$

Расстояние Хэмминга удовлетворяет аксиомам расстояния:

1. $d(\boldsymbol{x},\boldsymbol{y}) = 0$ если и только если $\boldsymbol{x} = \boldsymbol{y}$
2. $d(\boldsymbol{x},\boldsymbol{y}) = d(\boldsymbol{y},\boldsymbol{x})$ (симметричность)
3. $d(\boldsymbol{x},\boldsymbol{y}) \leq d(\boldsymbol{x},\boldsymbol{z}) + d(\boldsymbol{z},\boldsymbol{y})$ (неравенство треугольника)

Таким образом, имеем метрическое пространство.

Минимальным расстоянием кода C называется наименьшее из попарных расстояний между кодовыми словами:

$$d = \min_{\boldsymbol{x},\boldsymbol{y}\in C, \boldsymbol{x}\neq\boldsymbol{y}} d(\boldsymbol{x},\boldsymbol{y}). \tag{2.1}$$

Важность этой характеристики кода обусловлена следующим фактом.

Теорема 2.1. *Код с минимальным расстоянием* d *исправляет любые комбинации ошибок кратности* $t \leq \lfloor (d-1)/2 \rfloor$

Доказательство. Рассмотрим декодирование по минимуму расстояния Хэмминга. Если произошло t ошибок, то расстояние от переданного кодового слова до принятой последовательности равно t. Нужно доказать, что не найдется другого слова на расстоянии t или меньше. Для этого достаточно воспользоваться неравенством треугольника и определением минимального расстояния. □

Для линейного кода формула (2.1) может быть упрощена:

Теорема 2.2.

$$d = \min_{\boldsymbol{x}\in C, \boldsymbol{x}\neq 0} d(\boldsymbol{x}, \mathbf{0}) = \min_{\boldsymbol{x}\in C, \boldsymbol{x}\neq 0} w(\boldsymbol{x}), \tag{2.2}$$

где $w(\boldsymbol{x}) = d(\boldsymbol{x}, \mathbf{0})$ — число ненулевых элементов в последовательности $\boldsymbol{x}$ или вес вектора $\boldsymbol{x}$ в метрике Хэмминга.

Упражнение 2.2. Докажите теорему 2.2.

Упражнение 2.3. Найдите скорости и минимальные расстояния кодов примера 2.1. Убедитесь в том, что использование теоремы 2.2 упрощает поиск минимального расстояния. Во сколько раз?

После введения понятия базиса стало понятнее, как можно сэкономить память, предназначенную для хранения информации о коде: достаточно помнить не все пространство, а только его базис, а конкретные кодовые слова порождать в процессе кодирования. Это означает, что, например, для (1000,500)-кода вместо 2^{500} кодовых слов достаточно хранить 500 базисных слов длины 1000. Для этого достаточно 500 Кбит памяти. Таким объемом памяти никого сегодня не удивишь. На самом деле, и эта цифра завышена. Например, можно использовать циклические коды, в которых слова являются циклическим сдвигом друг друга. Тогда можно будет хранить всего одно слово, т.е. всего 1000 бит. И даже эта величина может быть уменьшена.

Декодирование линейного кода, как и кода общего вида, может быть выполнено путем сравнения каждого кодового слова с принятой из канала последовательностью и выбора ближайшего к ней в смысле некоторой метрики (например, метрики Хэмминга). На первый взгляд, сложность декодирования линейного кода намного меньше, чем кода общего вида. Ведь для перебора по множеству кодовых слов теперь нет необходимости хранить полный список кодовых слов. Однако если речь идет о переборе по 2^{500} кодовым последовательностям, то сложность обработки отдельной последовательности уже не имеет значения. Нужно думать о методах сокращения перебора. В следующих параграфах мы сделаем самые первые шаги в направлении упрощения декодирования двоичных кодов.

Еще один важный вопрос: не потеряем ли мы в потенциальной эффективности кодирования, ограничивая себя рассмотрением только линейных кодов? Можно доказать, что по крайней мере для симметричных каналов и при достаточно больших длинах кодов характеристики линейных кодов (в среднем по множеству кодов) такие же, как и характеристики кодов общего вида. Подробнее об этом можно почитать в учебниках по теории информации [6, 13].

2.2. Порождающая и проверочная матрицы

Определение 2.2. *Порождающей матрицей линейного (n, k)-кода называется матрица размера $k \times n$, строки которой — его базисные векторы.*

Например, матрицы

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

являются порождающими для кодов В) и Г) примера 2.1.

Мы знаем, что кодовые слова — линейные комбинации базисных векторов, т.е. строк матрицы G. Это означает, что слова могут быть получены умножением вектора на матрицу. Сообщение записывается в виде вектора $\boldsymbol{m} = (m_1, \ldots, m_k)$, и соответствующее сообщению кодовое слово вычисляется по формуле

$$\boldsymbol{c} = \boldsymbol{m}G.$$

Тем самым, вектор из k бит превращается в последовательность из n двоичных кодовых символов, передаваемых по каналу или записываемых в память запоминающего устройства.

Сделаем первые шаги к пониманию того, как можно упростить декодирование линейного кода.

Предположим, что для некоторого двоичного вектора $\boldsymbol{h} = (h_1, \ldots, h_n)$ все кодовые слова (n, k)-кода $C = \{\boldsymbol{c}_i\}, i = 0, \ldots, 2^k - 1$ удовлетворяют тождеству

$$(\boldsymbol{c}_i, \boldsymbol{h}) = 0, \quad i = 0, \ldots, 2^k - 1, \tag{2.3}$$

в котором запись $(\boldsymbol{x}, \boldsymbol{y})$ обозначает скалярное произведение векторов $\boldsymbol{x}$ и $\boldsymbol{y}$.

Про такой вектор $\boldsymbol{h}$ мы скажем, что он ортогонален коду, и назовем его *проверкой* по отношению к коду. Найдя такой вектор, мы могли бы проверять с помощью тождества (2.3), является ли принятая из канала последовательность кодовым словом.

Заметим теперь, что (2.3) справедливо для всех кодовых слов, если оно справедливо для базисных векторов, т.е. если

$$G\boldsymbol{h}^{\mathrm{T}} = 0, \tag{2.4}$$

где верхний индекс T обозначает транспонирование.

Чем больше таких «проверок» мы найдем, тем, по-видимому, больше ошибок сумеем обнаружить и исправить.

Упражнение 2.4. Докажите, что проверки образуют линейное пространство.

Это упражнение очень простое. Пространство проверок назовем пространством, ортогональным линейному коду, или *проверочным пространством*.

Упражнение 2.5. Докажите, что в порождающей матрице (n, k)-кода найдутся k линейно независимых столбцов.

Здесь нужно вспомнить, что такое ранг матрицы, и воспользоваться тем, что ранги матрицы, вычисленные по строкам и по столбцам, равны.

Упражнение 2.6. Найдите размерность линейного пространства проверок.

Решение. Чтобы справиться с этим непростым упражнением, воспользуемся утверждением упражнения 2.5. Найдем и зафиксируем список номеров k линейно независимых столбцов матрицы G и назовем этот набор индексов *информационной совокупностью*.

Совокупность индексов остальных позиций назовем *проверочной совокупностью*. Сами позиции и записанные на них символы будем называть соответственно информационными и проверочными. Чуть позже смысл этих названий станет яснее.

Выберем произвольно и зафиксируем значения вектора $\boldsymbol{h}$ на $r = n - k$ позициях проверочной совокупности. Какими должны быть значения на позициях информационной совокупности, чтобы выполнилось (2.4)?

Просуммируем в каждом уравнении слагаемые, соответствующие проверочной совокупности, обозначим сумму $\boldsymbol{h}^*$ и перенесем сумму в правую часть. Получится система линейных уравнений. Столбцы матрицы G, соответствующие информационной совокупности, образуют базис в пространстве столбцов, и, значит, вектор $-\boldsymbol{h}^*$ может быть единственным образом записан в виде их линейной комбинации.

Мы получили взаимно-однозначное соответствие между значениями компонент вектора $\boldsymbol{h}$ на информационных и проверочных позициях. Поскольку значения вектора $\boldsymbol{h}$ на проверочных позициях можно выбрать 2^{n-k} способами (q^{n-k} способами для q-ичного кода), в проверочном пространстве будет 2^{n-k} (соответственно, q^{n-k}) элементов, и размерность пространства будет равна $r = n - k$. □

Упражнение 2.7. На примере порождающей матрицы

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

убедитесь в правильности приведенных выше рассуждений.

Результат упражнения 2.6 сформулируем в виде теоремы.

Теорема 2.3. *Размерность проверочного пространства линейного (n, k)-кода равна $r = n - k$.*

Число r называется *избыточностью* кода.

Базис проверочного пространства запишем в виде матрицы

$$H = \begin{pmatrix} h_{11} & h_{12} & \cdots & h_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ h_{r1} & h_{r2} & \cdots & h_{rn} \end{pmatrix},$$

называемой *проверочной матрицей* кода.

Проверочная и порождающая матрицы связаны соотношением

$$GH^{\mathrm{T}} = 0. \tag{2.5}$$

Из этого соотношения мы видим, что для любого кодового слова $\boldsymbol{c} \in C$ имеет место

$$\boldsymbol{c}H^{\mathrm{T}} = 0. \tag{2.6}$$

Это тождество можно использовать как критерий принадлежности произвольной последовательности коду, т.е. для обнаружения ошибок.

Зная G, можно найти H. Для того чтобы понять, как это сделать, заметим, что один тот же код можно задать разными порождающими матрицами, выбирая по-разному базис пространства. Больше того, заменив в G любую строку на любую линейную комбинацию этой строки с другими строками, мы получаем новую порождающую матрицу того же кода.

Перестановка столбцов матрицы G, вообще говоря, приводит к другому коду, но этот другой код по своим характеристикам не отличается от исходного. Коды, различающиеся только нумерацией позиций, называются *эквивалентными*.

Понятно, что для каждого кода найдется такой эквивалентный ему код, что первые k позиций кодовых слов образуют информационную совокупность, иными словами, первые k столбцов порождающей матрицы образуют невырожденную подматрицу размера $k \times k$. Заменяя строки линейными комбинациями строк (метод Гаусса), полученную матрицу можно привести к виду

$$G = \begin{pmatrix} I_k & P \end{pmatrix}, \tag{2.7}$$

где I_k — единичная матрица порядка k, а P — некоторая матрица размера $k \times r$.

Матрица вида (2.7) называется порождающей матрицей, приведенной к *систематическому виду*, а соответствующий код называется *систематическим*. Кодирование для систематического кода проще, чем для кода общего вида:

$$\boldsymbol{c} = \boldsymbol{m}G = \begin{pmatrix} \boldsymbol{m} & \boldsymbol{m}P \end{pmatrix}, \tag{2.8}$$

т.е. в кодовом слове первые k позиций — просто копия информационной последовательности $\boldsymbol{m}$, а остальные $r = n - k$ (проверочных) позиций получаются умножением информационного вектора

на матрицу размера $k \times r$, что иногда существенно меньше, чем $n \times r$. Соответственно, и информация о систематическом коде занимает существенно меньше памяти, чем информация о линейном коде общего вида.

Замечание. Формально выражения «систематический код» и «несистематический код» неправильные, поскольку один и тот же код (совокупность кодовых слов) может быть описан как систематической, так и несистематической порождающей матрицей. В этом смысле любой код одновременно и систематический, и несистематический. Тем не менее, мы не будем отступать от привычной устоявшейся терминологии.

Для двоичного систематического кода с порождающей матрицей в форме (2.7) проверочная матрица может быть вычислена по формуле

$$H = \begin{pmatrix} P^{\mathrm{T}} & I_r \end{pmatrix}. \tag{2.9}$$

Упражнение 2.8. Проверьте (2.9). Подсказка: для этого нужно подставить (2.9) и (2.7) в (2.5).

Упражнение 2.9. Обобщите (2.9) на случай кода над произвольным полем F_q.

Как найти проверочную матрицу для несистематического кода?

Нужно привести матрицу к систематическому виду и воспользоваться (2.8). Если первые k столбцов порождающей матрицы образуют невырожденную подматрицу (первые k позиций образуют информационную совокупность), то для приведения к систематической форме достаточно таких операций как перестановка строк и замена строк линейными комбинациями строк. Если нет — нужно будет сначала найти информационную совокупность и перенумеровать позиции так, чтобы первые позиции стали информационными. После того, как найдена проверочная матрица эквивалентного систематического кода, нужно восстановить исходный порядок следования символов кодового слова.

Упражнение 2.10. Сформулируйте алгоритм нахождения порождающей матрицы по проверочной.

Упражнение 2.11. Объясните, почему любой набор из номеров k линейно-независимых столбцов называется информационной совокупностью.

Упражнение 2.12. Постройте порождающие и проверочные матрицы для кодов из примера 2.1.

Подведем итоги этого основополагающего параграфа.

- Линейный (n, k)-код может быть задан любой из двух матриц: порождающей G размера $k \times n$ либо проверочной H размера $r \times n$. По G легко найти H приведением кода к систематической форме. Аналогично по H находится G.
- Матрица G используется при кодировании (формула (2.8)), матрицей H можно воспользоваться при декодировании, по меньшей мере, для обнаружения ошибок. Выполнение тождества (2.6) свидетельствует о том, что данная последовательность принадлежит коду.

2.3. Вычисление минимального расстояния по проверочной матрице

Формула (2.2) нахождения минимального расстояния в терминах порождающей матрицы записывается как

$$d = \min_{\boldsymbol{m} \neq 0} w\left(\boldsymbol{m}G\right). \tag{2.10}$$

Вычисление по этой формуле требует перебора по $2^k - 1$ ненулевым словам кода. Возвращаясь к примеру (1000,500)-кода, легко понять, почему на практике мы часто пользуемся кодами, минимальное расстояние которых никому не известно.

Казалось бы, для кода (1000,900) проблема еще сложнее. Однако заметим, что при $r < k$, т.е. когда скорость кода больше $1/2$, проверочная матрица имеет меньший размер, чем порождающая. Хотелось бы воспользоваться этим обстоятельством для упрощения поиска минимального расстояния кода.

Согласно формуле (2.6), кодовому слову $\boldsymbol{c}$ веса $w(\boldsymbol{c})$ соответствует некоторый набор из $w(\boldsymbol{c})$ столбцов проверочной матрицы H, сумма которых — нулевой вектор. Это означает, что соответствующие столбцы линейно зависимы. Следовательно, для нахождения минимального расстояния кода нужно найти минимальный набор линейно зависимых столбцов проверочной матрицы. Приходим к следующей теореме.

Теорема 2.4. *Минимальное расстояние линейного (n, k)-кода равно d в том и только в том случае, когда любые $d-1$ столбцов проверочной матрицы линейно независимы и существует набор из d линейно зависимых столбцов.*

Подумав над тем, сколько всего может быть линейно независимых столбцов в матрице с r строками, получаем следующую интересную границу на минимальное расстояние.

Теорема 2.5. *Граница Синглтона: минимальное расстояние линейного (n, k)-кода удовлетворяет неравенству $d \leq r+1 = n-k+1$.*

Доказательство. Ранг проверочной матрицы равен r, следовательно $r+1$ столбцов всегда линейно зависимы. Граница Синглтона следует из теоремы 2.4.

Еще одно доказательство прямо следует из представления порождающей матрицы в систематической форме (2.7) и теоремы 2.2. Минимальный вес строк порождающей матрицей, очевидно, является оценкой сверху на минимальное расстояние кода. □

Упражнение 2.13. Докажите, что:

А) если в H нет нулевых столбцов, то минимальное расстояние кода не меньше 2;

Б) если в H все столбцы различны и ненулевые, то минимальное расстояние кода не меньше 3;

В) если одна из строк проверочной матрицы H двоичного кода не содержит нулей, то минимальное расстояние кода четно.

Упражнение 2.14. Найдите минимальные расстояния кодов примера 2.1.

Дуальным к данному коду называется код, порождающая матрица которого является проверочной матрицей данного кода.

Упражнение 2.15. Найдите минимальные расстояния кодов, дуальных к кодам примера 2.1.

2.4. Примеры кодов

Сформулированные в упражнениях свойства линейных кодов позволяют построить несколько хороших кодов.

Пример 2.3. Код с порождающей матрицей

$$G = (I_n)$$

представляет собой (n, n)-код со скоростью 1 и минимальным расстоянием $d = 1$. Поскольку число проверочных символов $r = n - k = 0$, проверочная матрица неопределена.

Пример 2.4. $(n, n-1)$-код с проверочной матрицей

$$H = \begin{pmatrix} 1 & 1 & \cdots & 1 \end{pmatrix}$$

(вектор из n единиц) имеет минимальное расстояние $d = 2$. Его порождающая матрица имеет вид

$$G = \begin{pmatrix} 1 & 0 & \cdots & 0 & 1 \\ 0 & 1 & \cdots & 0 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 1 \end{pmatrix}.$$

Это весьма экономный код. Он позволяет ценой одного избыточного символа обнаруживать любые однократные ошибки и любые другие комбинации ошибок нечетного веса. Его называют *кодом с проверкой на четность*.

Пример 2.5. Пусть

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

Этот код — двойственный к коду предыдущего примера. Очевидно, в таком коде всего два слова, т.е. имеем $(n, 1)$-код. Его проверочная матрица

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

Расстояние кода $d = n$, он исправляет комбинации ошибок кратности до половины длины кодового слова.

Пример 2.6. (Коды Хэмминга). Построим код, исправляющий любые однократные ошибки. Для этого нужно составить проверочную матрицу из различных столбцов. При числе проверочных символов r длина кода равна $n = 2^r - 1$, а число информационных символов равно $k = 2^r - r - 1$. Такие $(2^r - 1, 2^r - r - 1)$-коды называются *кодами Хэмминга.* Таким образом, имеем бесконечную последовательность кодов с расстоянием 3. В качестве примера рассмотрим случай $r = 3$, т.е.

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

Для кода Хэмминга легко указать процедуру исправления ошибок. Предположим, что передаваемое сообщение имеет вид $\boldsymbol{m} = (1011)$. Ему соответствует кодовое слово $\boldsymbol{c} = \boldsymbol{m}G = (1011010)$.

Предположим, что при передаче этой последовательности произошла одна ошибка и принятая последовательность имеет вид

$$\boldsymbol{x} = \boldsymbol{c} + \boldsymbol{e} = (1011010) + (0100000) = (1111010),$$

где через $\boldsymbol{e}$ обозначен вектор ошибки.

Декодер начинает с проверки того, является ли принятая последовательность кодовым словом. По формуле (2.6) находим

$$\boldsymbol{x}H^{\mathrm{T}} = (010).$$

Полученная последовательность оказалась двоичной записью номера ошибочной позиции. Легко убедиться в том, что это верно для любой одиночной ошибки. Попробуйте объяснить, почему это свойство имеет место.

Отметим, что двоичные коды Хэмминга оптимальны в том смысле, что не существует кодов (даже нелинейных) с бóльшим числом слов с расстоянием 3 при такой же длине.

Упражнение 2.16. Получите из кода Хэмминга код с минимальным расстоянием $d = 4$.

Пример 2.7. (Расширенные коды Хэмминга). Решение последнего упражнения состоит в том, чтобы дописать к проверочной матрице строку из всех единиц — проверку на четность. Расстояние нового кода будет не меньше 3 и будет четным, значит, оно равно 4. Кроме того, перед дописыванием проверки на четность можно приписать к H столбец из всех нулей, что не повлияет на расстояние. Итоговый код имеет в общем случае параметры $n = 2^{r-1}$, $k = n - r = 2^{r-1} - r$. При $r = 4$ проверочная матрица имеет вид

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

Отметим, что расширенные двоичные коды Хэмминга также оптимальны в том смысле, что не существует кодов с бóльшим числом слов с расстоянием 4 при такой же длине кодовых слов.

Пример 2.8. (Коды дуальные к кодам Хэмминга). Начнем с того, что запишем проверочную матрицу кода Хэмминга (15,11)

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

Обращает на себя внимание то, что в строках матрицы H одинаковое количество единиц. В данном случае 8. По построению матрицы этим свойством обладает проверочная матрица любого кода Хэмминга, т.е. в общем случае каждая строка содержит 2^{r-1} единиц. Более того, этим свойством обладает и любая нетривиальная линейная комбинация строк. Действительно, если бы единиц в некоторой линейной комбинации $\boldsymbol{v}$ было бы больше или меньше, то заменой одной из строк на такую линейную комбинацию мы получили бы новую матрицу, которая уже не содержала бы всех ненулевых двоичных последовательностей длины r в качестве столбцов. Как следствие, появились бы одинаковые столбцы, расстояние кода было бы меньше 3.

Итак, мы убедились в том, что любая линейная комбинация строк проверочной матрицы кода Хэмминга, а значит, строк порождающей матрицы дуального кода, имеет вес $d = 2^{r-1}$. Таким образом, код имеет большое минимальное расстояние (примерно половина длины кода) и обладает тем свойством, что для всех пар кодовых слов расстояния одинаковы. Такую конструкцию называют *симплексом*, а соответствующий код — *симплексным кодом*.

Пример 2.9. (Коды дуальные к расширенным кодам Хэмминга). Запишем проверочную матрицу расширенного кода Хэмминга (16,11):

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

Выбрав эту матрицу в качестве порождающей матрицы, получим код, в котором из $2^r - 1 = 31$ ненулевых кодовых слов кода 30 слов имеют вес $d = n/2 = 2^{r-2} = 8$ и одно слово — вес $n = 2^{r-1} = 16$.

Код, дуальный к расширенному $(2^{r-1}, 2^{r-1} - r)$-коду Хэмминга с расстоянием 4 представляет собой $(2^{r-1}, r)$-код с расстоянием $d = n/2 = 2^{r-2}$, который является кодом Рида–Маллера первого порядка.

Построение кодов с расстоянием 5 и больше — заметно более трудная задача. Теория конечных полей Галуа позволяет построить такие коды (циклические коды) и даже указать простые методы декодирования. Этим вопросам посвящены соответствующие разделы курса. Однако сужение класса линейных кодов к циклическим кодам ведет к некоторым потерям в достижимой величине минимального расстояния.

Поиск оптимальных по расстоянию кодов — увлекательная задача, привлекающая внимание большого числа математиков. Таблицы лучших найденных кодов можно найти в интернете [77]. Небольшая выписка из этой таблицы приведена на рис. 3.2 и 3.3. Заметим, что наименьшая длина, для которой до сих пор (до 2014 года) неизвестны параметры оптимальных кодов, равна 32. Неизвестно, например, может ли расстояние (32,14)-кода быть равным 9 (известен только (32,14)-код с $d=8$), а расстояние (32,18)-кода равным 7 (известен (32,18)-код с $d=6$).

2.5. Синдромное декодирование

Если скорость линейного (n,k)-кода $R = k/n > 1/2$, то размерность $r = n - k$ дуального пространства меньше размерности k самого кода. Попробуем воспользоваться этим фактом для упрощения декодирования в ДСК по минимуму расстояния Хэмминга. Задача декодера — найти кодовое слово, отличающееся от принятой двоичной последовательности в наименьшем числе позиций.

Пусть G и H обозначают порождающую и проверочную матрицу (n,k)-кода и $\boldsymbol{y}$ — двоичную последовательность, подлежащую декодированию. Если передавалось кодовое слово $\boldsymbol{c} \in C$, то $\boldsymbol{y}$ можно представить в виде

$$\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e},$$

где вектор $\boldsymbol{e} = (e_1, \ldots, e_n)$ называют *вектором ошибок*.

Сколько различных комбинаций ошибок может исправить код? Казалось бы, хороший код исправляет много комбинаций ошибок, а плохой — мало. На самом деле число исправляемых комбинаций ошибок не зависит от кода и равно $2^{n-k} = 2^r$.

Действительно, в силу линейности кода объем множества последовательностей $\boldsymbol{y}$, декодируемых в слово $\boldsymbol{c}$, одинаков для всех слов $\boldsymbol{c} \in C$. (Точнее, решающие области можно выбрать конгруэнтными, т.е. совпадающими с точностью до сдвига в хэмминговом пространстве.) Этот объем можно подсчитать, поделив общее число 2^n последовательностей длины n на число 2^k слов кода. Обозначим через $E = \{\boldsymbol{e}\}$ множество исправляемых комбинаций ошибок, $|E| = 2^r$. Таким образом, хороший код отличается от плохого тем, что для хорошего кода множество исправляемых комбинаций по своей форме близко к хэмминговой сфере, т.е. состоит из комбинаций ошибок меньшего веса, что соответствует более вероятным комбинациям ошибок. В частности, для хорошего кода выше величина гарантированной кратности исправляемых ошибок.

При декодировании по МП (в данном случае, по минимуму расстояния, см. задачу 4 к главе 1) мы должны обеспечить исправление всех 2^r комбинаций ошибок множества E.

Поскольку для кодового слова выполняется равенство

$$\boldsymbol{c}H^T = 0, \tag{2.11}$$

тривиальное решение задачи декодирования со сложностью порядка 2^r состоит в том, чтобы перебрать по множеству 2^r комбинаций ошибок из E, всякий раз прибавляя предполагаемый вектор ошибок к $\boldsymbol{y}$ и вычисляя произведение $(\boldsymbol{y}+\boldsymbol{e})H^T$. Вектор $\boldsymbol{c} = \boldsymbol{y}+\boldsymbol{e}$, для которого будет получен нулевой результат, выбирается в качестве результата декодирования.

Очевидный недостаток метода состоит в том, что он предполагает хранение множества E и умножение на проверочную матрицу для каждой последовательности множества E. Усовершенствуем этот способ декодирования.

Для этого рассмотрим произведение

$$\boldsymbol{s} = \boldsymbol{y}H^{\mathrm{T}}, \tag{2.12}$$

называемое *синдромом* вектора $\boldsymbol{y}$. Неравенство синдрома нулевому вектору указывает на наличие ошибок в принятой последовательности $\boldsymbol{y}$.

Поскольку

$$\boldsymbol{s} = \boldsymbol{y}H^{\mathrm{T}} = (\boldsymbol{c}+\boldsymbol{e})H^{\mathrm{T}} = \boldsymbol{e}H^{\mathrm{T}}, \tag{2.13}$$

имеет место взаимнооднозначное соответствие между 2^r различными синдромами и 2^r исправляемыми комбинациями ошибок.

Это означает, что умножение на матрицу достаточно выполнить один раз — при вычислении синдрома $\boldsymbol{s}$. В памяти декодера должна храниться таблица соответствия между синдромами и векторами ошибок. Эта таблица для каждого синдрома $\boldsymbol{s}$ хранит вектор наименьшего веса $\boldsymbol{e}$, для которого $\boldsymbol{s} = \boldsymbol{e}H^{\mathrm{T}}$.

Декодер после вычисления $\boldsymbol{s} = \boldsymbol{y}H^{\mathrm{T}}$ по таблице находит соответствующий вектор ошибок $\boldsymbol{e}$, и результатом декодирования служит кодовое слово $\boldsymbol{c} = \boldsymbol{y} + \boldsymbol{e}$. Процедура декодирования пояснена алгоритмом 2.1.

Алгоритм 2.1. Алгоритм синдромного декодирования

Input: Выход канала $\boldsymbol{y}$;
Output: Информационные символы ;
Вычислить синдром $\boldsymbol{s} = \boldsymbol{e}H^{\mathrm{T}}$;
Извлечь из ячейки памяти с адресом $\boldsymbol{s}$ вектор ошибок $\boldsymbol{e}$;
Вычислить оценку кодового слова $\hat{\boldsymbol{c}} = \boldsymbol{y} + \boldsymbol{e}$. ;
Вывод: Информационные символы, соответствующие кодовому слову $\hat{\boldsymbol{c}}$

Упражнение 2.17. Постройте таблицу для синдромного декодирования кода с порождающей матрицей

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

Итак, по сути, мы рассмотрели два способа декодирования в канале с жесткими решениями (ДСК) со сложностью порядка 2^r. В обоих случаях требуется память объема пропорционального 2^r. Табличное декодирование выглядит предпочтительнее с вычислительной точки зрения.

Существуют более изощренные методы декодирования, которые, правда, эффективны не для всех линейных кодов, а при выполнении некоторых ограничений. Идеи этих методов сформулированы в задачах в конце главы.

В завершение заметим, что в терминах теории групп (см. Приложение) векторное пространство образует абелеву группу относительно операции сложения векторов. Линейный код является подгруппой этой группы. Множество последовательностей $\boldsymbol{y}$, имеющих одинаковый синдром, образует смежный класс с некоторым вектором $\boldsymbol{e} \in E$ в качестве образующего элемента. Этот же элемент является вектором наименьшего веса среди всех элементов смежного класса. Его называют также *лидером* смежного класса.

2.6. Радиус покрытия и декодирование по минимуму расстояния Хэмминга [1]

Мы уже знаем два способа декодирования для произвольного линейного кода:

- перебор по $L = 2^k$ кодовым словам;
- перебор по $L = 2^{n-k}$ векторам ошибок.

Удобно характеризовать алгоритмы этого класса асимптотическим показателем экспоненты сложности

$$\kappa = \lim_{n\to\infty} \frac{\log L}{n}.$$

Описанные выше алгоритмы приводят нас к оценке

$$\kappa(R) \le \min\{R, 1-R\} \tag{2.14}$$

для любого линейного кода со скоростью R.

Замечание. Понятия сложности и экспоненты сложности введены неформально. Следовало дать формальное описание вычислителя, на котором выполняются операции, принять во внимание использование памяти и т.д. Точные формулировки можно найти, например, в [44].

[1] Приведенные в данном параграфе сведения не используются в дальнейшем. Кроме того, в нем используются некоторые сведения (граница Варшамова–Гилберта) из главы 3.

Задача состоит в том, чтобы уменьшить показатель экспоненты сложности, по возможности, без потери в надежности принятия решения.

Отметим, что декодирование в канале с жесткими решениями, даже если это декодирование по максимуму правдоподобия, имеет весьма ограниченную практическую значимость. Оно намного хуже декодирования с мягкими решениями, которое будет изучаться ниже, и ненамного проще. Тем не менее, идеи, лежащие в основе этих алгоритмов, интересны, поучительны и заслуживают краткого рассмотрения. Кроме того, их применение может выходить за рамки помехоустойчивого кодирования. Например, они могут быть использованы в криптографии при анализе шифров, основанных на корректирующих кодах.

2.6.1. Радиус покрытия

Минимальное расстояние кода определяет число гарантированно исправляемых кодом ошибок t, в то время как некоторые комбинации ошибок гораздо более высокой кратности, чем t, могут быть исправлены декодером максимального правдоподобия (декодером по минимуму расстояния Хэмминга). Радиус покрытия — наибольшее число ошибок, которые может исправить код.

Для произвольного множества $A \subset F_2^n$ последовательностей длины n введем обозначение для расстояния между последовательностью $\boldsymbol{x} \in F_2^n$ и A:

$$d(A, \boldsymbol{x}) = \min_{\boldsymbol{a} \in A} d(\boldsymbol{x}, \boldsymbol{a}).$$

Радиусом покрытия кода C называется величина

$$\rho = \max_{\boldsymbol{x}} d(\boldsymbol{x}, C). \tag{2.15}$$

Иными словами, для линейного (n, k)-кода ρ представляет собой наибольший вес среди весов всех 2^{n-k} последовательностей, декодируемых в нулевое кодовое слово.

Одно из практических применений радиуса покрытия связано с кодированием двоичного источника с заданным уровнем ошибки

в метрике Хэмминга (более точно, с вероятностной мерой искажения). Если радиус покрытия равен ρ, то максимальная ошибка приближения любой последовательности $\boldsymbol{x}$ словами кода C не превысит ρ/n.

Задачу построения кодов с заданной кратностью гарантированно исправляемых ошибок t называют еще задачей упаковки пространства шарами радиуса t с центрами, задаваемыми словами кода. Построение кодов с заданным радиусом покрытия ρ — задача покрытия пространства шарами радиуса ρ. В первом случае мы строим упаковку с максимальным числом шаров (кодовых слов), во втором — строим покрытие минимальным числом кодовых слов. Коды, удовлетворяющие границе плотной сферической упаковки (см. следующую главу), являются оптимальным решением обеих задач. Список таких кодов невелик. Среди нетривиальных двоичных кодов это только коды Хэмминга и Голея.

Сформулируем некоторые свойства радиуса покрытия.

Свойство 2.1. *Декодирование в ДСК с исправлением всех ошибок кратности $\nu \leq \rho$ включительно эквивалентно декодированию по минимуму расстояния Хэмминга.*

Свойство 2.2. *Пусть $\rho(\boldsymbol{s})$ обозначает минимальный вес вектора ошибок $\boldsymbol{e}$ такого, что ему соответствует синдром $\boldsymbol{e}H^{\mathrm{T}} = \boldsymbol{s}$. Тогда*

$$\rho = \max_{\boldsymbol{s}} \rho(\boldsymbol{s}). \tag{2.16}$$

Свойство 2.3. *Радиус покрытия линейного кода не превышает числа проверочных символов кода r.*

Свойство 2.4. *Если радиус покрытия линейного кода $\rho \geq d$, то размерность кода k может быть увеличена без уменьшения минимального расстояния кода.*

Свойство 2.5. *Если (n, k)-код имеет наибольшую возможную размерность k для заданных длины n и минимального расстояния d, то имеет место неравенство $\rho \leq d$.*

Упражнение 2.18. Докажите сформулированные свойства радиуса покрытия.

Вычисление радиуса покрытия для произвольного линейного кода — существенно более сложная задача, чем задача вычисления минимального расстояния кода.

Упражнение 2.19. Предложите алгоритм нахождения радиуса покрытия линейного кода. Как можно использовать свойство 2.2 для поиска радиуса покрытия высокоскоростных кодов?

Как часто бывает в теории кодирования, при том, что характеристики кодов вычислить очень трудно, найти средние характеристики случайных кодов можно достаточно просто. Рассмотрим, например, ансамбль случайных линейных кодов, задаваемых проверочными матрицами, элементы которых — случайные равновероятные независимые двоичные символы. Если удастся установить, что средняя по этому ансамблю величина радиуса покрытия равна ρ, это будет означать, что найдется по меньшей мере один код с таким радиусом покрытия. Более того, из закона больших чисел можно будет заключить, что при большой длине кода почти все случайно выбираемые коды будут иметь такой радиус.

Имеет место следующее утверждение:

Теорема 2.6. *При достаточно большой длине (n, k)-кода со скоростью $R = k/n$ для любого $\epsilon > 0$ в ансамбле равновероятных случайных кодов существуют коды с радиусом покрытия ρ таким, что*

$$h\left(\frac{\rho}{n}\right) \leq 1 - R + \epsilon. \tag{2.17}$$

Иными словами, существуют коды, радиус покрытия которых удовлетворяет границе Варшамова–Гилберта (см. теорему 3.4).

Доказательство можно найти в [20] для произвольных кодов и в [64] для линейных кодов. Еще одно (более простое) доказательство приведено в [44].

2.6.2. Декодирование по соседям нулевого слова

Этот остроумный алгоритм декодирования предложен в [87].

Рассмотрим (n, k)-код C. Обозначим через $R(\boldsymbol{c})$ решающую область кодового слова $c \in C$ в метрике Хэмминга, а через $D(\boldsymbol{c})$

ее *ближайшую окрестность*, т.е. множество всех таких последовательностей $\boldsymbol{x} \in F_2^n \setminus R(\boldsymbol{c})$, из которых инверсией одного из символов получается последовательность $\boldsymbol{x}' \in R(\boldsymbol{c})$.

Множество слов $Z \subseteq \{C \setminus \mathbf{0}\}$ называется *множеством соседей нулевого слова*, если это множество наименьшей мощности такое, что

$$D(\mathbf{0}) \subseteq \bigcup_{\boldsymbol{c} \in Z} R(\boldsymbol{c}).$$

Иными словами, решающие области элементов множества соседей нулевого слова граничат с решающей областью нулевого слова и полностью покрывают ближайшую окрестность нулевого слова. Отметим, что множество соседей нуля не обязательно единственно, но мощность этого множества $N_{\mathrm{Z}} = |Z|$ для заданного кода определена однозначно.

Теорема 2.7. *Для любого* $\boldsymbol{x} \notin R(\mathbf{0})$ *найдется* $\boldsymbol{c} \in Z$ *такой, что*

$$w(\boldsymbol{x} + \boldsymbol{c}) < w(\boldsymbol{x}).$$

Доказательство Будем по одному обнулять любые ненулевые символы $\boldsymbol{x}$, пока не получим $\boldsymbol{x}' \in D(\mathbf{0}) \bigcap R(\boldsymbol{c}')$ для некоторого $\boldsymbol{c}' \in Z$. Из неравенства треугольника имеем

$$\begin{aligned} d(\boldsymbol{c}, \boldsymbol{x}) &\leq d(\boldsymbol{x}, \boldsymbol{x}') + d(\boldsymbol{x}', \boldsymbol{c}') < \\ &< d(\boldsymbol{x}, \boldsymbol{x}') + d(\boldsymbol{x}', \mathbf{0}) = w(\boldsymbol{x}), \end{aligned}$$

откуда следует утверждение теоремы. □

Из теоремы следует алгоритм декодирования перебором по множеству соседей нулевого слова. Для последовательности $\boldsymbol{x}$ на выходе канала перебором по множеству соседей находим $\boldsymbol{c}' \in Z$, который, будучи сложен с $\boldsymbol{x}$, даст последовательность $\boldsymbol{x}'$ меньшего веса. Повторяя процедуру для новой последовательности, после не больше чем $w(\boldsymbol{x})$ итераций придем к последовательности $\boldsymbol{x}'$, вес которой не может быть уменьшен добавлением соседей. Эта последовательность принадлежит $R(\mathbf{0})$. Суммируя соседей $\boldsymbol{c}'$, использованных на последовательных итерациях, получаем оценку $\hat{\boldsymbol{c}}$ переданного кодового слова. Псевдокод декодирования по множеству соседей представлен в виде алгоритма 2.2.

Алгоритм 2.2. Алгоритм декодирования по соседям нулевого слова

Input: Выход канала $\boldsymbol{x}$;
Output: Информационные символы;
`Инициализация: Flag=1;` $\hat{\boldsymbol{c}} = \mathbf{0}$;
while Flag==1 **do**
  Flag=0;
  foreach $\boldsymbol{v} \in Z$ **do**
    if $w(\boldsymbol{x}+\boldsymbol{v}) < w(\boldsymbol{x})$ **then**
      $\hat{\boldsymbol{c}} \leftarrow \hat{\boldsymbol{c}} + \boldsymbol{v}$;
      $\boldsymbol{x} \leftarrow \boldsymbol{x} + \boldsymbol{v}$;
      Flag=1;
      break;
    end
  end
end
Вывод: Информационные символы, соответствующие кодовому слову $\hat{\boldsymbol{c}}$

Оценим асимптотическую сложность алгоритма как функцию длины кода n. Число итераций не превышает веса входной последовательности и может быть оценено сверху длиной кода. Сложность одной итерации пропорциональна числу соседей N_{Z}.

Максимальный вес вектора ошибок, декодируемого в нулевое слово, не превышает величины радиуса покрытия кода ρ, значит, вектор ошибок из ближайшей окрестности имеет вес не более $\rho+1$. Соответствующий ему сосед, в решающую область которого он попадает, удален от этого вектора на расстояние не более ρ. Отсюда выводим, что все соседи нулевого слова — слова веса не более $2\rho+1$.

Оценим сверху число соседей нулевого слова полным числом кодовых слов в n-мерном шаре $B_n(2\rho+1)$ радиуса $2\rho+1$ вокруг нулевого слова. В ансамбле случайных линейных кодов, задаваемых случайными проверочными матрицами, элементы которых — случайные двоичные числа, любая последовательность является кодовым словом с вероятностью $2^k/2^n$ (см. вывод границы Варшамова–

Гилберта, стр. 74). Поэтому можно оценить число соседей как

$$N_{\mathrm{Z}} \le 2^{k-n} \left| B_n(2\rho+1) \right|. \tag{2.18}$$

Объем шара радиуса t в n-мерном хэммингове пространстве

$$|B_n(t)| = \sum_{i=0}^{t} \binom{n}{i}.$$

Из теорем 2.7 и 3.4 следует, что почти для всех кодов из ансамбля случайных кодов радиус покрытия и минимальное расстояние кода в пределе при $n \to \infty$ удовлетворяют границе Варшамова–Гилберта

$$R = 1 - h(\delta), \quad \frac{d}{n} = \frac{\rho}{n} = \delta.$$

В итоге, из (2.18) с учетом (1.29) получаем асимптотическую оценку показателя сложности декодирования по соседям нулевого слова:

$$\kappa_{\mathrm{Z}}(R) \le \begin{cases} h(2\delta) - h(\delta), & R > 1 - h(0.25), \\ 1 - h(\delta), & R \le 1 - h(0.25). \end{cases} \tag{2.19}$$

Пороговое значение скорости $R = 1 - h(0.25) = 0.1887$ определяет минимальную скорость, начиная с которой декодирование по соседям нулевого слова эффективнее простого перебора по кодовым словам.

График функции $\kappa_{\mathrm{Z}}(R)$ показан на рис. 2.1.

Видно, что выигрыш метода декодирования по соседям нулевого слова по экспоненте сложности в сравнении с непосредственным перебором очень большой. Тем не менее, как мы увидим ниже, декодирование в ДСК может быть выполнено еще проще. Оценка сложности (2.19) интересна тем, что точно такое же поведение сложности наблюдается для декодирования кодов, представленных решетками (см. главу 4), но уже не с жесткими решениями, а с мягкими решениями в канале с АБГШ.

2.6.3. Декодирование по информационным совокупностям

Идея еще одного метода декодирования основана на том, что по значениям кодовых символов на любой информационной совокупности

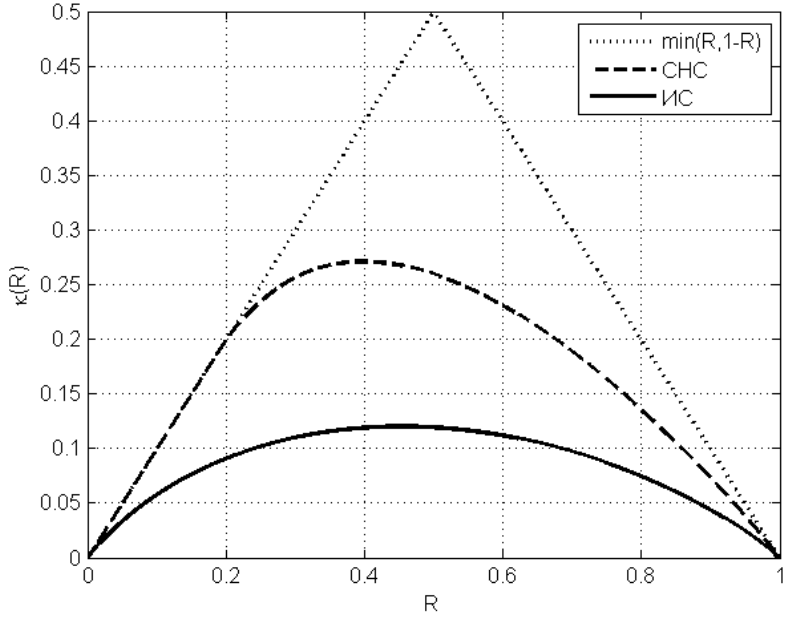


Рис. 2.1. Границы экспоненты сложности декодирования: $\min(R, 1-R)$ — декодирование перебором по кодовым словам либо синдромное декодирование; «СНС» — декодирование по соседям нулевого слова; «ИС» — декодирование по информационным совокупностям

однозначно восстанавливается кодовое слово. Согласно свойству 2.3 радиус покрытия не превышает числа проверочных символов. Поэтому, если число ошибок в канале не превышает величины радиуса покрытия кода, то, вероятно, найдется свободная от ошибок информационная совокупность, по которой можно восстановить кодовое слово. Вывод формулы асимптотической сложности декодирования по информационным совокупностям был впервые опубликован в работах [39] и [63].

Пусть $\boldsymbol{y}$ — последовательность на выходе ДСК, полученная при передаче некоторого кодового слова $\boldsymbol{c}$ двоичного (n, k)-кода. Мы будем пытаться отыскать информационную совокупность, по которой

можно восстановить кодовое слово при условии, что число ошибок не превышает радиуса покрытия кода. Хотя хороший список таких информационных совокупностей следовало бы подготовить заранее до начала декодирования, мы будем обсуждать стратегию, основанную на случайном выборе информационных совокупностей.

Конечно, не любой набор из k номеров позиций кодового слова соответствует некоторой информационной совокупности. Если мы вычислим ранг подматрицы, составленной из столбцов, номера которых указаны k выбранными позициями, то эта подматрица может быть невырожденной, в этом случае имеем информационную совокупность. Она может иметь ранг $k' < k$, и тогда только k' битов, соответствующих базисным столбцам, можно считать информационными. Найдутся в точности $2^{k-k'}$ различных кодовых слов, совпадающих с $\boldsymbol{y}$ на k позициях. Можно показать, что в среднем по ансамблю случайных кодов доля «свободных» битов $(k - k')/k$ при увеличении k стремится к нулю (см. [39, 63], а также [44]). Поэтому при анализе алгоритма мы пренебрежем дополнительными вычислениями, связанными с проверкой всех дополнительных кандидатов. Проще говоря, считаем, что любой выбор k позиций из n дает информационную совокупность.

Зафиксируем достаточно большое число N_{I}. Алгоритм декодирования состоит в том, что декодер N_{I} раз случайно выбирает информационную совокупность, всякий раз восстанавливает кодовое слово и находит расстояние до последовательности $\boldsymbol{y}$. После N_{I} попыток получателю выдается ближайшее к $\boldsymbol{y}$ кодовое слово из N_{I} найденных кандидатов.

Сложность каждой попытки довольно велика, т.к. требуется приведение порождающей матрицы к систематической форме на выбранных позициях, а затем кодирование. Тем не менее, сложность попытки растет полиномиально с длиной кода, и показатель экспоненты сложности всего алгоритма определяется допустимым максимальным числом попыток декодирования N_{I}. Псевдокод декодирования по информационным совокупностям приведен в виде алгоритма 2.3.

Вероятность ошибки декодирования по информационным совокупностям ограничена сверху суммой вероятности ошибки декодирования по максимуму правдоподобия и вероятности ошибки ал-

Алгоритм 2.3. Алгоритм декодирования по информационным совокупностям

Input: Выход канала $\boldsymbol{x}$;
Output: Информационные символы ;
`Инициализация:` $\hat{\boldsymbol{c}} = \mathbf{0}$, $d_{\text{opt}} = w(\boldsymbol{x})$;
for *i=1* **to** N_{I} **do**
  Выбрать случайный набор из k упорядоченных индексов $\boldsymbol{i} = (i_1, ..., i_k)$;
  Построить список C_0 слов, совпадающих с $\boldsymbol{x}$ на позициях из $\boldsymbol{i}$;
  foreach $\boldsymbol{c} \in C_0$ **do**
    if $d(\boldsymbol{c}, \boldsymbol{x}) < d_{\text{opt}}$ **then**
      `Обновить решение;`
      $\hat{\boldsymbol{c}} \leftarrow \boldsymbol{c}$;
      $d_{\text{opt}} \leftarrow d(\boldsymbol{c}, \boldsymbol{x})$;
    end
  end
end
Вывод: Информационные символы, соответствующие кодовому слову $\hat{\boldsymbol{c}}$

горитма — вероятности $P_e(N_{\text{I}})$ того, что все N_{I} выборов неудачны (содержат ошибки канала).

Если число ошибок в канале не превышает величины радиуса покрытия ρ, то ошибка при одной попытке происходит с вероятностью не больше

$$1 - \frac{\binom{n-\rho}{k}}{\binom{n}{k}}$$

и в результате N_{I} независимых попыток получаем оценку

$$P_e(N_{\text{I}}) \le \left(1 - \frac{\binom{n-\rho}{k}}{\binom{n}{k}}\right)^{N_{\text{I}}}.$$

Воспользуемся тождеством

$$\binom{n}{a}\binom{n-a}{b} = \binom{n}{b}\binom{n-b}{a},$$

чтобы привести оценку к более удобной форме

$$P_e(N_{\mathrm{I}}) \le \left(1 - \frac{\binom{n-k}{\rho}}{\binom{n}{\rho}}\right)^{N_{\mathrm{I}}}.$$

Заметим, что правая часть с ростом n в зависимости от выбранного числа попыток N_{I} может стремиться либо к константе $e^{-1} = \lim_{z\to\infty}(1-1/z)^z$, либо к нулю или к бесконечности. Чтобы вероятность ошибки стремилась к нулю, произведение

$$N_{\mathrm{I}} \times \frac{\binom{n-k}{\rho}}{\binom{n}{\rho}}$$

должно расти с увеличением n. Если рост будет экспоненциальным, со сколь угодно малым положительным показателем экспоненты, то дополнительная вероятность ошибки $P_e(N_{\mathrm{I}})$ будет убывать по двойному экспоненциальному закону, и ею можно будет пренебречь по сравнению с вероятностью ошибки декодирования по максимуму правдоподобия.

Таким образом, для любого $\varepsilon > 0$ граница сверху на асимптотический показатель экспоненты сложности удовлетворяет неравенству

$$\begin{aligned}
\kappa_{\mathrm{I}}(R) &\le \lim_{n\to\infty}\frac{1}{n}\log_2\frac{\binom{n}{\rho}}{\binom{n-k}{\rho}} + \epsilon = && (2.20)\\
&= h(\delta) - (1-R)h\left(\frac{\delta}{1-R}\right) + \epsilon, && (2.21)\\
R &= 1 - h(\delta), && (2.22)
\end{aligned}$$

в котором мы снова использовали обозначение $\delta = d/n = \rho/n$ для относительного расстояния кода, которое в соответствии с теоремами 2.7 и 3.4. почти для всех кодов ансамбля совпадает с относительным радиусом покрытия и удовлетворяет границе Варшамова–Гилберта. Переход от (2.20) к (2.21) основан на (1.29).

Более аккуратный выбор асимптотического поведения числа попыток позволяет доказать (2.21) при $\epsilon = 0$. График функции $\kappa_{\mathrm{I}}(R)$ показан на рис. 2.1. Мы видим, что при скорости кода $R = 1/2$

показатель экспоненты сложности примерно в 2 раза меньше, чем для декодирования по соседям нулевого слова и в 4 раза меньше показателя сложности переборного декодирования.

Не случайно именно метод декодирования по информационным совокупностям чаще других используют, например, для нахождения минимального расстояния линейных кодов.

Известны асимптотически более эффективные (но и более громоздкие в описании и анализе) алгоритмы декодирования в ДСК. Подробный обзор литературы по этой тематике приведен в [44].

Выводы

- Линейный код — линейное подпространство линейного пространства.
- Порождающая матрица кода — базис подпространства, образующего код.
- Проверочная матрица — базис ортогонального пространства.
- Любой код либо может быть представлен в систематической форме, либо эквивалентен систематическому коду.
- Преобразование порождающей матрицы в проверочную и наоборот выполняется с помощью приведения кода к систематической форме.
- Минимальное расстояние кода равно минимальному весу ненулевых кодовых слов.
- Минимальное расстояние кода равно минимальному числу линейно зависимых столбцов.
- Параметры линейного кода удовлетворяют границе Синглтона: $d \leq n - k + 1$.
- Нетрудно построить хорошие короткие линейные коды: коды с расстоянием 2, 3, 4 и дуальные к ним.

- Декодирование линейного (n,k)-кода по минимуму расстояния Хэмминга при скорости кода $R = k/n < 1/2$ может быть выполнено перебором по кодовым словам со сложностью пропорциональной 2^k. При $R = k/n > 1/2$ при помощи синдромного декодирования оно может быть выполнено со сложностью пропорциональной 2^r, $r = n - k$.

Задачи

1. Построить наилучшие в смысле минимального расстояния коды длины 6 со скоростями 1/6, 2/6, 3/6, 4/6, 5/6, 1.

2. Для наилучших известных кодов со скоростью k/n=1/2 из табл. 3.2 построить зависимости ошибки от длины кодов при декодировании с исправлением ошибок кратности до $t = \left[\frac{d-1}{2}\right]$ для ДСК с переходной вероятностью 0.1, 0.01, 0.001. Подсчитайте энергетический выигрыш кодирования для вероятности ошибки декодирования 10^{-5}.

3. По заданной проверочной (порождающей) матрице найти порождающую (проверочную), определить скорость и минимальное расстояние кода. В качестве примеров используйте матрицы из следующего списка:

а) $\begin{bmatrix}101100\\010110\\001011\end{bmatrix}$, б) $\begin{bmatrix}001011\\010110\\101100\end{bmatrix}$, в) $\begin{bmatrix}010110\\101101\end{bmatrix}$,

г) $\begin{bmatrix}010110\\101101\\110100\end{bmatrix}$, д) $\begin{bmatrix}111111\\010110\\001011\end{bmatrix}$, е) $\begin{bmatrix}001011\\111111\\101100\end{bmatrix}$,

ж) $\begin{bmatrix}010110\\111111\end{bmatrix}$, з) $\begin{bmatrix}0101100\\1011010\\1111111\end{bmatrix}$, и) $\begin{bmatrix}1110100\\0111010\\0011101\end{bmatrix}$,

к) $\begin{bmatrix}1110100\\0111010\\1111111\end{bmatrix}$.

4. Итеративные коды (коды-произведения). Рассмотрим два линейных кода с параметрами (n_1, k_1) и (n_2, k_2) и минимальными расстояниями d_1 и d_2 соответственно. Запишем $k_1 k_2$ информационных символов в виде таблицы с k_1 строками и k_2 столбцами. Допишем в каждый столбец проверочные символы, закодировав столбцы первым кодом, затем допишем в каждую из n_1 строк проверочные символы, закодировав строки вторым кодом. Докажите следующие утверждения.

   - Полученный код является линейным $(n_1 n_2, k_1 k_2)$-кодом.
   - Каждый столбец является словом первого кода, строка — второго.
   - Минимальное расстояние кода равно $d = d_1 d_2$.
   - Декодирование по строкам, а затем по столбцам гарантирует исправление не больше $d_1 d_2/4$ ошибок.

   Потенциально код способен исправлять до $t = \lfloor (d-1)/2 \rfloor = \lfloor (d_1 d_2 - 1)/2 \rfloor$ ошибок. Подумайте над тем, как улучшить декодирование, оставаясь в рамках алгоритмов, основанных на попытках декодирования составляющих кодов длин n_1 и n_2. Подсказка: при заданном минимальном расстоянии кода стираний исправляется больше, чем ошибок. Желательно часть ошибок заменить стираниями.

5. Сколько существует различных линейных (n, k)-кодов? Сколько существует различных неэквивалентных линейных (n, k)-кодов? Предложите способы поиска кодов с наилучшим минимальным расстоянием.

6. Укажите соответствие между синдромами и векторами ошибок для кода Хэмминга с произвольным числом проверочных символов.

7. Для примеров матриц из задачи 3, интерпретируемых как проверочные матрицы, постройте таблицы синдромного декодирования.

8. Большинство результатов данного раздела сформулировано для двоичных кодов. Однако это ограничение, сделанное для простоты изложения, непринципиально. Какие теоремы данного раздела и как должны быть переформулированы для случая кодов над произвольным полем F_q?

9. *Циклический код.* Под циклическим сдвигом последовательности $\boldsymbol{g} = \begin{pmatrix} g_1 & g_2 & \dots & g_{n-1} & g_n \end{pmatrix}$ на одну позицию вправо понимается последовательность $\boldsymbol{g} = \begin{pmatrix} g_n & g_1 & g_2 & \dots & g_{n-1} \end{pmatrix}$. Код, в котором циклический сдвиг любого кодового слова — тоже кодовое слово, называется *циклическим кодом* (глава 5). Постройте порождающую матрицу линейного циклического (7,4)-кода как набор из четырех циклических сдвигов последовательности $\boldsymbol{g} = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$. Определите минимальное расстояние кода. Как изменятся характеристики кода при изменении длины (последовательность $\boldsymbol{g}$ при этом дополняется нулями в нужном количестве) и (или) размерности кода? Сравните параметры получаемых кодов с параметрами лучших известных кодов.

10. Дополняя последовательность $\boldsymbol{g} = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 \end{pmatrix}$ нулями, постройте циклические коды с расстоянием 4. Сравните параметры получаемых кодов с параметрами лучших известных кодов.

11. Попробуйте построить нетривиальный циклический код с расстоянием $d = 5$ размерности $k \geq 3$ и длины $n < kd$.

12. *Сверточный код.* Выберем некоторую четную длину кода $n > 4$ и выберем в качестве первой строки порождающей матрицы (n, k)-кода дополненную нулями последовательность

    А) $\boldsymbol{g} = \begin{pmatrix} 1 & 1 & 0 & 1 \end{pmatrix}$

    Б) $\boldsymbol{g} = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$

    В) $\boldsymbol{g} = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$

    Остальные строки получите (не циклическими!) сдвигами вправо на две позиции первой строки до тех пор, пока последний (самый правый) ненулевой символ не окажется на позиции

с номером n. Определите минимальное расстояние и скорость кода. Сравните параметры получаемых кодов с параметрами лучших известных кодов.

Построенный код – частный случай усеченного сверточного кода (глава 8).

13. *Квазициклический код.* Код, в котором циклический сдвиг на фиксированное число позиций $c > 1$ любого кодового слова – тоже кодовое слово, называется *квазициклическим.* Воспользуйтесь последовательностями из предыдущей задачи для построения квазициклических кодов со скоростью $R = 1/2$. Подсчитайте минимальное расстояние кодов и сравните параметры полученных кодов с параметрами лучших известных кодов. Как изменится расстояние кода при увеличении или уменьшении длины?

Коды этого класса в русскоязычной литературе называют циклически усеченными, в то время как по-английски они называются «tail-biting»-коды. Почему?

14. Для кодов из задачи 1 найдите радиусы покрытия. Нельзя ли улучшить построенные Вами коды с точки зрения их радиуса покрытия?

15. *Декодирование методом нулевых соседей.* Постройте множество соседей нулевого слова для кодов из задачи 1. Сравните сложность декодирования по нулевым соседям со сложностью декодирования перебором по словам кода и со сложностью синдромного декодирования.

16. *Декодирование по информационным совокупностям.* Сколько информационных совокупностей имеют код Хэмминга (7,4) и расширенный код Хэмминга (8,4)? Оцените сложность декодирования по информационным совокупностям для этих кодов и сравните со сложностью других алгоритмов декодирования.

Приложение. Группы. Основные определения

Определение 2.3. *Множество $G = \{g\}$ называется группой, если на нем определена некоторая операция (обозначим ее «$*$») и выполняются следующие аксиомы.*

1. *Замкнутость: если $a, b \in G$, то $a * b \in G$.*
2. *Ассоциативность: $a * (b * c) = (a * b) * c$.*
3. *Существует единичный элемент e: $a*e = e*a = a$ для любых $a \in G$.*
4. *Для каждого элемента $a \in G$ существует обратный элемент $b \in G$ такой, что $a * b = e$.*

Если групповой операцией является сложение, то единичным элементом мы традиционно считаем 0, а обратным к a служит $-a$. Если же групповой операцией является умножение, то единичный элемент — это 1, а обратный к a записываем как a^{-1}.

Число элементов группы называется *порядком* группы.

Группа называется *коммутативной* или *абелевой*, если групповая операция коммутативна. Классический пример некоммутативной групповой операции — операция перемножения в группе невырожденных квадратных матриц. В рамках данного курса мы будем иметь дело, в основном, с коммутативными группами.

Если $H \subset G$ и H само является группой, то H называется *подгруппой*.

Для определенности рассмотрим коммутативную группу по сложению.

Определение 2.4. *Рассмотрим некоторый элемент группы $g \in G$, $g \notin H$. Тогда множество элементов вида $\{g + h, h \in H\}$ называется смежным классом группы G по подгруппе H. При этом сам элемент g называется образующим элементом смежного класса.*

Методом доказательства «от противного» нетрудно убедиться в справедливости следующего утверждения.

Теорема 2.8. *Смежные классы, соответствующие двум образующим элементам, либо совпадают, либо не пересекаются.*

Из теоремы следует, что группа единственным образом разбивается на совокупность различных смежных классов по заданной подгруппе. Все смежные классы имеют одинаковую мощность, поэтому справедлива следующая теорема.

Теорема 2.9. *Порядок подгруппы является делителем порядка группы.*

В случае абелевой группы можно определить операцию над смежными классами $B = \{b\}$ и $B' = \{b'\}$

$$B + B' = \{b\} + \{b'\} = \{b + b', b \in B, b' \in B'\}$$

(групповая операция, конечно, может быть любой, мы рассматриваем сложение только для определенности).

Множество смежных классов по подгруппе H само образует группу (с H в качестве единичного элемента). Эта группа называется *фактор-группой* группы G по H и обозначается как $G|H$.

Пример 2.10. Множество чисел $\{0, 1, 2, 3, 4, 5\}$ образует группу по сложению по модулю 6. Порядок группы равен 6. Обратными к $0, 1, \ldots, 5$ элементами являются соответственно $0, 5, 4, 3, 2, 1$. Примерами подгрупп являются $\{0, 3\}$ и $\{0, 2, 4\}$. Их порядки равны 2 и 3. Смежными классами $H = \{0, 3\}$ являются множества

$$B_0 = H = \{0, 3\},\ B_1 = \{1, 4\},\ B_2 = \{2, 5\}.$$

В этой фактор-группе можно вычислять суммы элементов. Например,

$$B_1 + B_2 = \{1 + 2, 1 + 5, 4 + 2, 4 + 5\} = \{0, 3\} = B_0.$$

3. Некоторые границы на характеристики кодов

Минимальное расстояние кодов — наиболее важная характеристика линейного кода, она чаще других параметров используется как ориентир при поиске эффективных кодовых конструкций. Для построения кодов типично используются комбинаторные, алгебраические и вероятностные методы. Желательно иметь последовательности кодов с фиксированной скоростью и линейно растущим с длиной кода минимальным расстоянием. Если это условие выполняется, можно гарантировать, что вероятность ошибки убывает экспоненциально с длиной кода, и, следовательно, может быть обеспечена сколь угодно малая вероятность ошибки декодирования. Такие коды (или классы кодов) называют *асимптотически хорошими*.

От скорости роста минимального расстояния с длиной кода зависит, насколько близко можно приблизиться к пределу Шеннона — пропускной способности канала. Однако задача построения кодов интересна и важна без привязки к конкретной модели канала. В данной главе мы рассматриваем минимальное расстояние как критерий качества кода и вычисляем для него оценки снизу и сверху.

Знание верхних и нижних границ на минимальное расстояние кодов позволяет судить о том, насколько хороша та или иная конструкция, а также избавляет от бесплодных попыток построить код с заведомо недостижимыми характеристиками.

В конце главы мы приводим верхние и нижние границы на вероятность ошибки для заданного кода — это, соответственно, границы Полтырева и Шеннона.

3.1. Граница Хэмминга

Первая граница очень простая. Она основана на том, что при упаковке пространства шарами суммарный объем всех шаров не больше общего числа точек в пространстве.

Теорема 3.1. *Для любого q-ичного кода с длиной n и минимальным расстоянием $d \geq 2t+1$ число кодовых слов M удовлетворяет неравенству*

$$M \leq \frac{q^n}{\sum\limits_{i=0}^{t} \binom{n}{i}(q-1)^i}. \tag{3.1}$$

Доказательство. Шар радиуса t с центром в точке $\boldsymbol{x}$ — это множество точек, расположенных на расстоянии не более t от $\boldsymbol{x}$. В пространстве X^n всех q-ичных последовательностей с метрикой Хэмминга объем каждого такого шара $B_t(\boldsymbol{x})$ не зависит от $\boldsymbol{x}$ и равен

$$|B_t(\boldsymbol{x})| = \sum_{i=0}^{t} \binom{n}{i}(q-1)^i.$$

Если центры шаров размещены в кодовых словах кода объема M с минимальным расстоянием $d \geq 2t+1$, то шары не пересекаются. Их суммарный объем не превышает общего числа точек в пространстве q^n, из чего следует неравенство (3.1). □

Упражнение 3.1. Сопоставьте характеристики кодов Хэмминга с границей Хэмминга.

Изучим асимптотическое поведение границы (3.1) при увеличении длины кодов.

Теорема 3.2. *Скорость $R = \log_q M/n$ любого q-ичного кода с относительным минимальным расстоянием $\delta = d/n$ при достаточно большой длине кода n удовлетворяет неравенству*

$$R \leq R_{\text{H}}(\delta) = \log_2 q - h\left(\frac{\delta}{2}\right) - \frac{\delta}{2}\log_2(q-1), \tag{3.2}$$

где $h(x) = -x\log_2 x - (1-x)\log_2(1-x)$.

Доказательство Для доказательства нужно воспользоваться формулами (1.20) или (1.21) приложения к главе 1.

В случае двоичных кодов, q=2, неравенство (3.2) принимает вид

$$R \le R_{\mathrm{H}}(\delta) = 1 - h\left(\frac{\delta}{2}\right). \tag{3.3}$$

□

График асимптотической границы Хэмминга показан на рис. 3.1.

Интересно, что выражение в правой части (3.3) очень напоминает выражение для пропускной способности ДСК $C = 1 - h(p)$. Из сравнения формул нетрудно вывести (например, с помощью неравенства Чебышева или любой другой оценки вероятности больших уклонений), что, если бы существовали коды, удовлетворяющие границе Хэмминга с равенством, то, декодируя с исправлением ошибок кратности t, можно было бы передавать информацию со скоростью сколь угодно близкой к пропускной способности. Однако, как мы увидим позже, при $n \to \infty$ и $\delta > 0$ таких кодов не существует, что не исключает, конечно, возможности приблизиться к пропускной способности канала.

Среди нетривиальных двоичных кодов равенство в границе Хэмминга имеет место только для кодов Хэмминга и кода Голея (23,12).

3.2. Граница Варшамова–Гилберта

Приведенная в предыдущем параграфе граница Хэмминга — верхняя граница на минимальное расстояние. В этом параграфе мы получим такую же простую нижнюю границу, т.е. границу, гарантирующую возможность построения кодов с некоторыми параметрами. Эта граница называется границей Варшамова–Гилберта (ВГ).

Теорема 3.3. *Если имеет место неравенство*

$$q^{n-k} > \sum_{i=0}^{d-2} \binom{n-1}{i} (q-1)^i, \tag{3.4}$$

то существует q-ичный линейный (n,k)-код с минимальным расстоянием d.

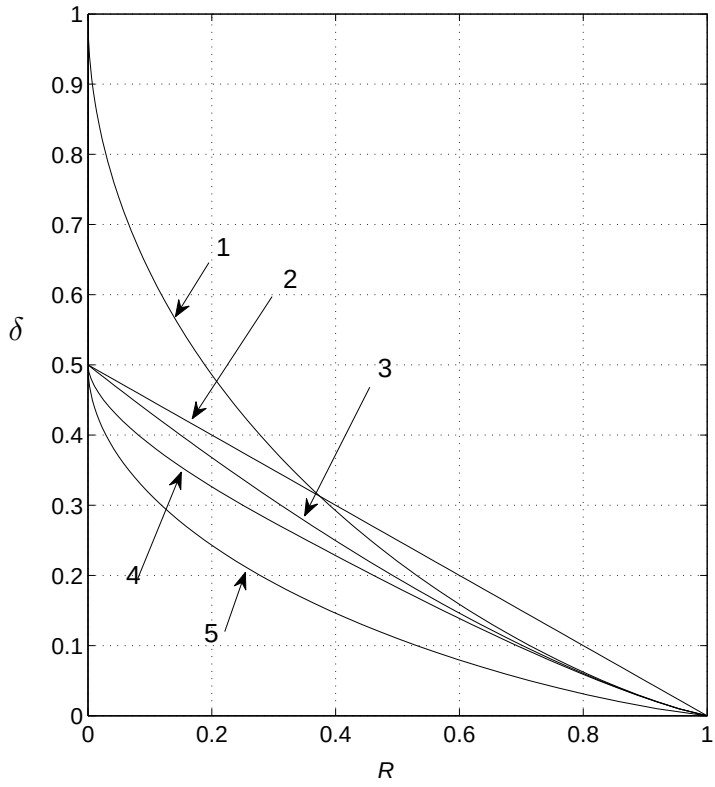


Рис. 3.1. Границы на минимальное расстояние: 1 — граница Хэмминга, 2 — граница Плоткина, 3 — граница Элайеса–Басалыго, 4 — граница МакЭлиса–Родемича–Рамсея–Велча, 5 — граница Варшамова–Гилберта

Доказательство Будем шаг за шагом строить проверочную матрицу кода с r строками и с заданным минимальным расстоянием d, т.е. матрицу, в которой любые $d-1$ столбцов линейно независимы. Первый столбец может быть любым не равным нулю. Второй столбец должен быть не кратен первому, третий не может совпадать с линейной комбинацией первых двух и т.д. На i-м шаге имеем $i-1$ столбцов длины r. Проверим, можно ли продолжить построение, добавив еще один столбец. Из общего числа 2^r возможных столб-

цов длины r нужно исключить один нулевой, $(q-1)(i-1)$ столбцов, кратных уже выбранным. Далее нужно исключить $(q-1)^2 \binom{i-1}{2}$ попарных линейных комбинаций столбцов. Продолжая, исключаем линейные комбинации вплоть до $(d-2)$ различных столбцов из $(i-1)$ выбранных. Если (3.4) верно, то код длины $n=i$ существует, но кода бóльшей длины, вообще говоря, может не существовать. $\square$

Аналогично асимптотической форме границы Хэмминга имеем асимптотическую границу ВГ.

Теорема 3.4. *При достаточно большой длине (n,k)-кода и любом $\varepsilon > 0$ найдется код с относительным минимальным расстоянием $\delta = d/n$ со скоростью $R = k/n$, для которого имеет место неравенство*

$$R \geq R_{\mathrm{VG}}(\delta) - \varepsilon,$$

где

$$R_{\mathrm{VG}}(\delta) = \log_2 q - h(\delta) - \delta \log_2(q-1). \tag{3.5}$$

Для двоичных кодов граница имеет вид

$$R_{\mathrm{VG}}(\delta) = 1 - h(\delta). \tag{3.6}$$

Эта функция показана на рис. 3.1. Из сравнения границ $R_{\mathrm{H}}(\delta)$ и $R_{\mathrm{VG}}(\delta)$ мы видим, что достижимое по границе ВГ расстояние вдвое меньше, чем оценка, вытекающая из границы Хэмминга. Если бы удалось построить длинные коды, удовлетворяющие границе ВГ, то при декодировании с исправлением $t = \lfloor (d-1)/2 \rfloor$ ошибок максимально достижимая скорость передачи со сколь угодно малой вероятностью ошибки была бы все равно ниже пропускной способности канала. Опять приходим к выводу, что декодирования с исправлением ошибок кратности, гарантируемой минимальным расстоянием кода, недостаточно для достижения пропускной способности канала. Декодирование же по МП (в случае ДСК это будет декодирование по минимуму расстояния Хэмминга) вполне может быть асимптотически эффективным.

Приведем еще одно, неконструктивное, доказательство границы ВГ. Для простоты будем рассматривать двоичные коды.

Построим проверочную матрицу (n,k)-кода H, выбирая ее элементы из множества $\{0,1\}$ независимо с одинаковыми вероятностями. Каждая такая матрица задает некоторый код, все такие коды равновероятны. С какой вероятностью код имеет минимальное расстояние, меньшее d?

Это событие эквивалентно тому, что для заданной матрицы H найдется последовательность $\boldsymbol{c} = (c_1, \ldots, c_n)$ веса $w(\boldsymbol{c}) < d$, являющаяся кодовым словом, т.е. $\boldsymbol{c}H^{\mathrm{T}} = \boldsymbol{0}$. Для этого хотя бы одна из $\binom{n}{w}$ сумм $w = w(\boldsymbol{c})$ случайных столбцов длины $(n-k)$ должна быть равна нулю. Поскольку все такие столбцы представляют собой случайные равновероятные последовательности длины $(n-k)$ (вероятность каждой последовательности равна 2^{k-n}), то вероятность нулевой суммы тоже равна 2^{k-n}, и вероятность плохого кода можно оценить как

$$\sum_{w=1}^{d-1} \Pr(\boldsymbol{c}H^{\mathrm{T}} = \boldsymbol{0} | w(\boldsymbol{c}) = w) \le 2^{-n+k} \sum_{w=1}^{d-1} \binom{n}{w}.$$

Воспользовавшись асимптотическими формулами для числа сочетаний (см. приложение к главе 1), легко убедиться, что, если скорость кода меньше $R_{\mathrm{VG}}(\delta) - \varepsilon$, то правая часть убывает с ростом n пропорционально $2^{-n\varepsilon}$. Это означает, что доля плохих кодов (с минимальным расстоянием ниже границы ВГ) стремится к нулю с увеличением длины кодов. Тем самым установлено, что доля случайных кодов с расстоянием ниже границы ВГ стремится к нулю с ростом длины кода, и, следовательно, почти все случайные коды удовлетворяют границе ВГ.

При малых n все хорошие известные коды имеют расстояние, превышающее эту границу. Означает ли это, что граница ВГ — слабая граница? Удивительно, но именно эта (асимптотическая) граница, известная с начала 50-х годов прошлого века, так и не была улучшена (для малых q, в частности, для двоичных кодов).

3.3. Граница Плоткина

При сравнении границ Хэмминга и ВГ обращает на себя внимание то, что между ними нет согласия даже в «тривиальной» точке

$R = 0$. Кажется, что чаша весов предпочтения должна склониться в сторону границы Хэмминга, поскольку код длины n с расстоянием $d = n$ существует — это код $(n, 1)$ из двух противоположных слов. Следовательно, точка $(R = 0, d/n = 1)$ кажется достижимой. Но тут кроется ошибка: с точки зрения асимптотики код $(n, 2)$ — тоже код с нулевой скоростью. Максимально достижимое относительное расстояние для нее уже заведомо меньше единицы. Для него уже граница Хэмминга недостижима.

Упражнение 3.2. Подсчитайте величину $\lim\limits_{n\to\infty}(d/n)$ для оптимального кода с $k = 2, 3, \ldots$ информационными символами. Подсказка: ответ будет приведен в следующем параграфе, посвященном границе Грайсмера.

Из этого упражнения следует, что вопрос о достижимом при $R \to 0$ минимальном расстоянии нетривиален. Ответ на этот вопрос дает граница Плоткина. В этом параграфе мы будем рассматривать для простоты только линейные двоичные коды. Нам потребуются вспомогательные леммы.

Лемма 3.5. *Сумма по всем кодовым словам (n, k)-кода значений символов на фиксированной позиции слова равна либо 0, либо 2^{k-1}. Иными словами, каждая позиция в коде либо всегда равна нулю, либо принимает значения 0 и 1 в равном числе кодовых слов.*

Доказательство. Рассмотрим, для определенности, первую позицию кода. Если в порождающей матрице G первый столбец — нулевой, то он будет нулевым во всех словах, что соответствует первой из двух возможностей в утверждении леммы.

Если первый столбец ненулевой, одна из строк G начинается с единицы. Предположим, без потери общности, что это последняя строка. Прибавив ее ко всем строкам, которые также начинаются с единицы, получим порождающую матрицу того же кода, в которой все элементы первого столбца, за исключением последнего, равны нулю.

Линейные комбинации первых $k - 1$ строк матрицы дают 2^{k-1} слов, начинающихся с нуля. Прибавление к ним последней строки дает ровно 2^{k-1} слов, начинающихся с единицы. $\square$

Лемма 3.6. *Для любого линейного кода*

$$\frac{n2^{k-1}}{2^k-1} \geq d. \tag{3.7}$$

Доказательство. В числителе левой части — оценка сверху на суммарное число единиц во всех кодовых словах. Эта оценка непосредственно следует из леммы 3.5. В знаменателе — число ненулевых кодовых слов. Таким образом, дробь слева — среднее число единиц в ненулевых словах кода. Средний вес не может быть меньше минимального, откуда следует утверждение леммы. □

Из леммы 3.6, в частности, следует, что при больших k

$$\frac{d}{n} \leq \frac{1}{2} + \frac{1}{2^{k+1}-2} \approx \frac{1}{2}. \tag{3.8}$$

Лемма 3.7. *Пусть $K(n,d)$ обозначает максимальное возможное число информационных символов кода длины n с расстоянием d. При $n > d$ имеет место неравенство*

$$K(n,d) \leq K(n-1,d) + 1. \tag{3.9}$$

Доказательство. Лемма говорит, что при заданном расстоянии увеличение длины кода на единицу позволит увеличить размерность кода не больше, чем на один бит. Достаточно доказать, что, если существует (n,k)-код с расстоянием d, то найдется $(n-1,k-1)$-код с расстоянием d.

Рассмотрим (n,k)-код и представим его порождающую матрицу G в систематической форме. Без потери общности предположим, что первый столбец начинается с единицы, а остальные символы столбца равны нулю. Вычеркнув из G первый столбец и первую строку, получим код длины $n-1$ размерности $k-1$, а расстояние кода либо не изменилось, либо увеличилось. □

Теорема 3.8. *При достаточно больших n для любого (n,k)-кода со скоростью $R = k/n$ и относительным расстоянием $\delta = d/n$ имеет место граница Плоткина*

$$R \leq R_{\mathrm{P}}(\delta) = 1 - 2\delta. \tag{3.10}$$

Доказательство. Из леммы 3.7 имеем

$$K(n,d) \le K(2d-1,d) + n - 2d + 1. \tag{3.11}$$

Из (3.8) нетрудно видеть, что, поскольку $d/(2d-1) > 1/2$, длинных кодов с положительной скоростью и относительным расстоянием $d/(2d-1)$ не существует, т.е. $\lim_{n\to\infty} K(2d-1,d)/n = 0$. Деля обе части (3.11) на n и переходя к пределу при $n \to \infty$, получаем требуемый результат. □

График границы Плоткина $R_{\mathrm{P}}(\delta)$ вместе с другими границами показан на рис. 3.1. Мы видим, что эта граница существенно сильнее границы Хэмминга для низких скоростей кодов.

3.4. Граница Грайсмера

Граница Грайсмера часто оказывается полезной для оценки возможности построения конкретного кода. Кроме того, она косвенно подсказывает, как можно строить хорошие низкоскоростные коды с большими расстояниями.

Вывод границы основан на простой лемме.

Лемма 3.9. *Пусть $N(k,d)$ обозначает минимальную возможную длину двоичного линейного кода размерности k с минимальным расстоянием d. Тогда*

$$N(k,d) \ge d + N\left(k-1, \lceil d/2 \rceil\right), \tag{3.12}$$

где выражение $\lceil a \rceil$ обозначает округление вещественного числа a вверх до ближайшего целого.

Доказательство. Приведем порождающую матрицу произвольного кода длины $n = N(k,d)$ к виду

$$G = \left(\begin{array}{c|c} 0^{N(k,d)-d} & 1^d \\ \hline G_1 & G_2 \end{array}\right),$$

где запись a^i обозначает i-кратное повторение элемента a.

Заметим, что код, задаваемый матрицей G_1, имеет размерность $k-1$. Иначе одна из линейных комбинаций ее строк была бы нулевой, а ее продолжение (соответствующая линейная комбинация строк G_2) имело бы вид 1^d. То есть нашлось бы кодовое слово, совпадающее с первой строкой G, что означает, что исходный код не имел размерность k.

Предположим теперь, что минимальный вес одного из слов кода, заданного G_1, равен $d_1 < d/2$. Тогда вес его продолжения не меньше $d - d_1 > d/2$. Прибавив к этому слову первую строку матрицы G, мы уменьшим вес правой части до величины, меньшей $d/2$, и суммарный вес окажется меньше d. Поскольку этого не может быть, наше предположение о том, что минимальный вес кода, заданного G_1, меньше $d/2$, неверно. □

В результате $(k-1)$-кратного применения леммы приходим к следующей теореме.

Теорема 3.10.

$$N(k,d) \geq \sum_{i=0}^{k-1} \left\lceil \frac{d}{2^i} \right\rceil. \tag{3.13}$$

В задачах в конце главы приведено несколько примеров применения границы.

Подумаем над тем, возможно ли построить такие линейные коды, которые были бы оптимальными с точки зрения границы Грайсмера. Для таких кодов в (3.13) должно выполняться равенство. Точнее говоря, на каждой итерации вида (3.12) должно выполняться равенство.

Упражнение 3.3. Убедитесь в том, что симплексные коды (дуальные к кодам Хэмминга) и коды Рида–Маллера первого порядка (дуальные по отношению к расширенным кодам Хэмминга) удовлетворяют границе Грайсмера.

Итак, уже понятно, что построить очень низкоскоростные коды длины 2^{k-1} или кратной 2^k-1, удовлетворяющие границе Грайсмера, несложно. Нужно сначала выбирать все столбцы порождающей матрицы разными и затем повторить все столбцы нужное число раз.

Пусть длина кода не кратна $2^k - 1$. Это означает, что некоторые столбцы будут повторяться, некоторые — нет. Задача построения кода сводится к задаче правильного выбора столбцов, которые будут повторяться. Интуиция справедливо подсказывает, что все дополнительные столбцы должны быть различными (иначе найдется информационная последовательность, которая обнулит сразу несколько добавленных кодовых позиций). Заметим также, что первые $2^k - 1$ столбцов задают код со словами одинакового веса, значит «прибавка» минимального расстояния зависит только от веса различных линейных комбинаций дописываемых столбцов. Например, если дописанная матрица имеет неполный ранг, то минимальное расстояние итогового кода вообще не изменится.

Подробно о построении кодов, удовлетворяющих границе Грайсмера, можно почитать в главе 17 книги [10].

3.5. Другие границы

Приведем без доказательства еще две асимптотические границы, справедливые как для линейных, так и нелинейных кодов.

Граница Бассалыго–Элайеса

$$R \leq 1 - h\left(\frac{1}{2} - \frac{1}{2}\sqrt{1-2\delta}\right).$$

Граница МакЭлиса–Родемича–Рамсея–Велча (МЭРРВ)

$$R \leq \min_{u \in (0, 1-2\delta]} B(u, \delta),$$

где

$$B(u, \delta) = 1 + \mu(u^2) - \mu(u^2 + 2\delta u + 2\delta)$$

и

$$\mu(x) = h\left(\frac{1}{2} - \frac{1}{2}\sqrt{1-2x}\right).$$

Все асимптотические границы показаны на рис. 3.1 в виде функций $\delta(R)$. Сравнение графиков показывает, что в асимптотической теории кодирования вопрос о достижимой скорости кодов с заданным расстоянием не имеет окончательного ответа, несмотря на титанические усилия многих математиков. Интересно, что лучшая

верхняя граница (МЭРРВ) была опубликована в 1977 году, а нижняя граница ВГ известна с 1953 года.

Дальнейшее уточнение любой их двух границ по прошествии стольких лет кажется уже невероятным. (Отметим, что небольшое продвижение произошло в 1984 г., когда С. Г. Влэдуц, Г. Л. Кацман, М. А. Цфасман в работе [34] доказали существование кодов над алфавитом размера $q \geq 49$ с расстояниями выше границы ВГ.)

3.6. Спектр кода и оценки вероятности ошибки

В настоящее время компьютерное моделирование все больше вытесняет подсчет по формулам при анализе помехоустойчивости систем кодирования. Тем не менее, моделирование может быть неэффективным, например, при оценивании вероятностей ошибок порядка 10^{-12} и ниже. Ниже мы приводим как справочный материал лучшие известные верхние границы вероятности ошибки для линейных блоковых кодов. Эти оценки, полученные Полтыревым Г. Ш. в 1994 году [97], вполне конкурируют по точности с оценками, получаемыми моделированием.

Для получения оценок вероятности ошибки помимо расстояния кодов нужно знать *весовой спектр линейного кода*, т.е. набор чисел S_0, S_1, ..., S_n, где S_w обозначает число кодовых слов веса w.

Завершает параграф нижняя оценка вероятности ошибки декодирования блокового кода, полученная Шенноном в 1959 году [102, 19].

3.6.1. Граница вероятности ошибки через спектр кода для ДСК

В силу линейности кода вероятность ошибки не зависит от передаваемого слова, и поэтому граница выводится для нулевого слова. Граница для ДСК выводится очень просто, она основана на разделении векторов ошибок на «близкие» и «далекие». К вероятностям ошибок, связанных с векторами из этих групп, применяются разные способы оценивания. Как далекие рассматриваются векторы

веса больше порога W. Конкретное значение порога мы выберем позже.

Пусть P_e обозначает вероятность ошибки, а $P_e(W)$ — ее оценку, вычисленную при фиксированном значении параметра W. Применительно к ДСК с переходной вероятностью p получаем неравенство

$$P_e \le P_e(W) = \sum_{w=d}^{n} P(\mathcal{E}, w, \nu < W) + \Pr(\nu \ge W), \tag{3.14}$$

в котором $P(\mathcal{E}, w, \nu < W)$ — совместная вероятность двух событий:

- одно из слов веса w ближе к принятому вектору, чем нулевое слово;
- число ошибок в канале ν меньше W.

Для уточнения оценки (3.14) заметим, что близкие векторы приводят к ошибкам только в пользу слов веса

$$w \le 2(W-1).$$

Среди $l \le W-1$ ошибок η ошибок приходится на w ненулевых позиций, а остальные $l-\eta$ расположены на $n-w$ нулевых позициях слова. Заметим, что $l \in \{t_w, ..., W-1\}$, $t_w = \lceil (w+1)/2 \rceil$, $\eta \in \{t_w, ..., \min\{l, w\}\}$, т.е. диапазон значений η зависит от w. Напомним, что $\lceil a \rceil$ обозначает округление a вверх до ближайшего целого.

Обозначим $\eta_w = \min\{\eta, w\}$. Оценивая вероятность объединения событий суммой вероятностей событий, получаем

$$\begin{aligned} P(\mathcal{E}, w, \nu < W) \quad &\le \quad S_w \sum_{l=t_w}^{W-1} \sum_{\eta=t_w}^{l} \binom{w}{\eta_w} p^{\eta_w} (1-p)^{w-\eta_w} \times \\ &\times \quad \Pr\left((l-\eta_w) \text{ ошибок на } (n-w) \text{ нулевых позициях}\right) \\ = \quad & S_w \sum_{l=t_w}^{W-1} \sum_{\eta=t_w}^{l} \binom{w}{\eta_w} p^{\eta_w} (1-p)^{w-\eta_w} \binom{n-w}{l-\eta_w} p^{l-\eta_w} (1-p)^{n-w-l+\eta_w} \\ = \quad & S_w \sum_{l=t_w}^{W-1} \sum_{\eta=t_w}^{l} \binom{w}{\eta_w} \binom{n-w}{l-\eta_w} p^{l} (1-p)^{n-l}, \end{aligned} \tag{3.15}$$

где $\{S_w\}$ — спектр весов кодовых слов кода.

Неравенство (3.14) принимает вид

$$P_e \leq \sum_{w=d}^{2(W-1)} S_w \sum_{l=t_w}^{W-1} \sum_{\eta=t_w}^{l} \binom{w}{\eta_w} \binom{n-w}{l-\eta_w} p^l (1-p)^{n-l} + \\ + \sum_{l=W}^{n} \binom{n}{l} p^l (1-p)^{n-l}, \tag{3.16}$$

аналогичный лемме 1 [97]. Отличия — в уточнении диапазонов индексов суммирования.

Поменяем порядок суммирования в (3.16)

$$P_e \leq \sum_{l=t_d}^{W-1} \sum_{w=2l-1}^{2(W-1)} S_w \sum_{\eta=t_w}^{l} \binom{w}{\eta_w} \binom{n-w}{l-\eta_w} p^l (1-p)^{n-l} + \\ + \sum_{l=W}^{n} \binom{n}{l} p^l (1-p)^{n-l}. \tag{3.17}$$

Первое слагаемое, соответствующее близким векторам, принимает вид

$$\sum_{l=t_d}^{W-1} B(l,W) p^l (1-p)^{n-l},$$

где

$$B(l,W) = \sum_{w=2l-1}^{2(W-1)} S_w \sum_{\eta=t_w}^{l} \binom{w}{\eta_w} \binom{n-w}{l-\eta_w}.$$

Из (3.17) заключаем, что порог W определяет множитель перед $p^l(1-p)^{n-l}$, равный либо $B(l,W)$, либо $\binom{n}{l}$. Для малых W коэффициент $B(l,W)$ меньше $\binom{n}{l}$, но с ростом l ситуация меняется на противоположную. Оптимальным будет выбор W, при котором

$$B(l = W+1, W) \geq \binom{n}{W}.$$

Небольшое уточнение оценки Полтырева можно получить, если учесть, что при $\eta = w/2$ ошибка происходит с вероятностью $1/2$.

Обозначим

$$\alpha(w,\eta) = \begin{cases} 1/2, & w = 2\eta \\ 1 & w \neq 2\eta \end{cases}.$$

Окончательная форма оценки

$$P_e \leq \sum_{w=d}^{2(W-1)} S_w \sum_{l=t_w}^{W-1} \sum_{\eta=t_w}^{l} \alpha(w,\eta) \binom{w}{\eta_w} \binom{n-w}{l-\eta_w} p^l (1-p)^{n-l} + \\ + \sum_{l=W}^{n} \binom{n}{l} p^l (1-p)^{n-l}, \tag{3.18}$$

где W — наименьшее значение такое, что

$$\sum_{w=2l-1}^{2W} S_w \sum_{\eta=t_w}^{W} \alpha(w,\eta) \binom{w}{\eta_w} \binom{n-w}{W-\eta_w} \geq \binom{n}{W}.$$

Заметим, что параметр W не зависит от канала, он определяется только свойствами кода: его длиной и спектром.

3.6.2. Граница вероятности ошибки для гауссовского канала

Перейдем к каналу с гауссовским шумом. Считаем, что двоичные кодовые символы передаются противоположными сигналами, энергию сигналов принимаем равной единице. Дисперсия величин y, вычисляемых демодулятором, равна $\sigma^2 = N_0/2$.

Обозначим через

$$f(x) = \frac{1}{2\pi} e^{-\frac{x^2}{2}} \tag{3.19}$$

стандартную гауссовскую плотность распределения вероятностей. Если математические ожидания y при передаче нуля и единицы равны соответственно -1 и $+1$, то распределения вероятностей на выходе демодулятора имеют вид

$$f(y|0) = f\left(\frac{y+1}{\sigma}\right); \quad f(y|1) = f\left(\frac{y-1}{\sigma}\right).$$

Оценка Полтырева (сам автор назвал ее тангенциально-сферической границей) выражается через некоторые специальные

функции. Это не создает проблем при выполнении вычислений, поскольку для этих функций имеются подпрограммы в среде МАТЛАБ.

Гамма-функция определяется как

$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} dt. \tag{3.20}$$

При целых значениях аргумента $\Gamma(n) = (n-1)!$. Полезны также следующие свойства гамма-функции

$$\Gamma(z+1) = z\Gamma(z); \quad \Gamma\left(\frac{1}{2}\right) = \sqrt{\pi}.$$

Естественные обобщения гамма-функции — неполная гамма-функция

$$\Gamma(z,x) = \int_x^\infty t^{z-1} e^{-t} dt,$$

нижняя неполная гамма-функция

$$\gamma(z,x) = \int_0^x t^{z-1} e^{-t} dt$$

и регуляризированная гамма-функция

$$P(z,x) = \frac{\gamma(z,x)}{\Gamma(z)}.$$

Очевидно, $\Gamma(z,x) + \gamma(z,x) = \Gamma(z)$.

Через гамма-функции выражается функция так называемого χ^2_n-распределения, описывающая распределение вероятностей суммы квадратов n независимых гауссовских случайных величин с нулевым математическим ожиданием и единичной дисперсией.

Плотность χ^2_n-распределения для величины x с n степенями свободы имеет вид

$$f(x,n) = \frac{x^{n/2-1} e^{-x/2}}{2^{n/2}\Gamma(n/2)}.$$

Для кумулятивной функции χ^2_n-распределения имеем выражение

$$\chi^2_n(x) = \frac{\gamma(n/2, x/2)}{\Gamma(n/2)} = P(n/2, x/2).$$

С учетом введенных обозначений оценка вероятности ошибки имеет вид

$$P_e \leq \int_{-\infty}^{\sqrt{n}} f\left(\frac{x}{\sigma}\right) \left\{ \sum_{w \leq w_0} \left\{ S_w \int_{\beta_w(x)}^{r_x} f\left(\frac{y}{\sigma}\right) \chi^2_{n-2}\left(\frac{r_x^2 - y^2}{\sigma^2}\right) dy \right\} + \right.$$
$$\left. + \quad 1 - \chi^2_{n-1}\left(\frac{r_x^2}{\sigma^2}\right) \right\} dx + Q\left(\frac{\sqrt{n}}{\sigma}\right). \tag{3.21}$$

В этой формуле $f(\cdot)$ — гауссовская плотность (3.19), функция $Q(\cdot)$ определена соотношением (1.8),

$$\begin{aligned} w_0 &= \left\lfloor \frac{r_0^2 n}{r_0^2 + n} \right\rfloor; \\ r_x &= r_0\left(1 - \frac{x}{\sqrt{n}}\right); \\ \mu_w(r) &= \frac{1}{r}\sqrt{\frac{w}{1 - w/n}}; \\ \beta_w(x) &= \left(1 - \frac{x}{\sqrt{n}}\right)\sqrt{\frac{w}{1 - w/n}}. \end{aligned}$$

Параметр r_0 представляет собой решение относительно r уравнения

$$\sum_{w:\mu_w(r)<1} S_w \int_0^{\arccos \mu_w(r)} \sin^{n-3}\phi \quad d\phi = \sqrt{\pi}\frac{\Gamma\left(\frac{n-2}{2}\right)}{\Gamma\left(\frac{n-1}{2}\right)}. \tag{3.22}$$

На первый взгляд, формулы слишком сложны, чтобы быть полезными для практических расчетов. Кажется, что при современных вычислительных возможностях проще получить оценки вероятности ошибки с помощью моделирования. Относительно компактная МАТЛАБ-программа 3.1 для вычисления границы Полтырева поможет преодолеть это заблуждение.

Программа 3.1. Подсчет границы Полтырева вероятности ошибки декодирования по максимуму правдоподобия

```
function PE=tsb_upperbound
% Верхняя граница Полтырева на вероятность ошибки
```

```
% PE при известном спектре SP
%
% ПРИМЕР ПРИМЕНЕНИЯ (на примере кода Хэмминга(7,4))
% Глобальные переменные
% global n SP sigma;
%
% Параметры кода
% k=4, n=7;
% SP=[1 0 0 4 3 0 0 0]
%
% Пересчет отношения сигнал-шум на бит в СКО шума
% SNRb=1;
% SNR=10^(SNRb/10);
% SNR=SNR*k/n;
% sigma=sqrt(1/2/SNR);
%
% Вызов функции
% PE=tsb_upperbound;

% Основная функция
global n sigma r0 SP;
% Уравнение на пороговые величины r0, w0
r=sqrt(n);
r0 = fzero(@tsb_eq,r);
w0=floor(r0.^2*n./(r0.^2+n));
% Границы интегрирования
T=10*sigma;
rx=  @(x) r0*(1-x/sqrt(n));
% Первое слагаемое
PE=1-normcdf(sqrt(n)/sigma);
% Суммирования по спектру кода
for w=find(SP(2:w0+1))
    beta = @(x) (1-x/sqrt(n))*sqrt((w)/(1-w/n));
    % Двумерный интеграл в пределах
    %(-T,sqrt(n)) по x  и (beta,rx) по y
    PE = PE+SP(w+1)* ...
    quad2d(@fun1,-T,sqrt(n),beta,rx,'AbsTol',10^-8);
end;
% Последнее слагаемое
PE=PE+quad(@fun2, -T, sqrt(n), 10^-8);
%-------------------------------------------------%
% Подынтегральная функция для двумерного
% интегрирования
function z=fun1(x,y)
global r0 n sigma;
```

```
rx=r0*(1-x/sqrt(n));
z= normpdf(y,0,sigma).*normpdf(x,0,sigma).*...
chi2cdf((rx.^2-y.^2)/sigma^2,(n-2)*ones(size(y)));
%-------------------------------------------------%
% Подынтегральная функция для одномерного
% интегрирования
function z=fun2(x)
global r0 n sigma;

rx=r0*(1-x/sqrt(n));
z=(1-chi2cdf(rx.^2/sigma^2,n-1)).*normpdf(x,0,sigma);
%-------------------------------------------------%
% Функция для уравнения относительно r0
function y=tsb_eq(r)
global n SP;
sinf = @(x) sin(x).^(n-3);
r=abs(r);
y=-sqrt(pi)*gamma((n-2)/2)/gamma((n-1)/2);
for w=find(SP(2:n))
    mu=sqrt(w/(1-w/n))/r;
    if mu>=1, break; end;
    y=y+SP(w+1)*quad(sinf,0,acos(mu));
end;
```

3.6.3. Нижняя граница Шеннона

Сравнение с шенноновским пределом отношения сигнал/шум, подсчитанным по формуле для пропускной способности, не всегда продуктивно. При ограниченной задержке декодирования пропускная способность канала заведомо недостижима. Нижние границы на вероятность ошибки, подсчитанные с учетом длины кода, позволяют гораздо более точно оценить, насколько близко построенный код к наилучшему возможному результату при заданной длине.

Несмотря на большие усилия, предпринятые для получения хороших нижних границ, одной из наиболее точных до сих пор остается полученная в 1959 году граница Шенона [102]. Детальный обзор работ по верхним и нижним оценкам вероятности ошибки приведен в [81].

Границу Шеннона называют также границей плотной сферической упаковки. Ее идея состоит в том, что кодовые слова кода длины n со скоростью R представляются в виде точек на поверхности

евклидовой сферы радиуса $n\sqrt{E_s}$, где E_s — энергия сигнала. Решающим областям соответствуют конусы с общей вершиной в начале координат.

Рассмотрим сферу, вписанную в основание одного такого конуса, и обозначим через соответствующий θ телесный полуугол, $\theta \in [0, \pi]$. Поскольку такие конусы, соответствующие разным кодовым словам, не пересекаются, максимальное число кодовых слов можно оценить как отношение площади, вырезанной на поверхности сферы полууглом всего n-мерного пространства

$$\Omega_n(\pi) = \frac{2\pi^{n/2}}{\Gamma(n/2)}$$

к площади, вырезанной одним конусом

$$\Omega_n(\theta) = \frac{2\pi^{\frac{m-1}{2}}}{\Gamma(\frac{n-1}{2})} \int_0^\theta (\sin\phi)^{n-2} d\phi.$$

Здесь $\Gamma(x)$ — гамма-функция (3.20). Приравнивая отношение площадей числу кодовых слов $M = 2^{nR}$, находим минимальное возможное значение параметра θ как решение уравнения

$$\frac{\Omega_n(\theta)}{\Omega_N(\pi)} = 2^{-nR}. \tag{3.23}$$

При известном θ в качестве оценки снизу вероятности ошибки декодирования кодового слова можно использовать аппроксимацию [57] формулы Шеннона [102]:

$$\begin{aligned} P_{\text{sh}}(n, R, \frac{E_b}{N_0}) \;&\approx\; \frac{1}{\sqrt{n\pi}} \frac{1}{\sqrt{1+G^2}\sin\theta} \times \\ &\times\; \frac{\left(G\sin\theta \exp\left(-\frac{E_\text{b}}{2N_0} + \frac{1}{2}\sqrt{\frac{E_\text{b}}{N_0}} G\cos\theta\right)\right)^n}{\sqrt{\frac{E_\text{b}}{N_0}} G\sin^2\theta - \cos\theta}, \end{aligned} \tag{3.24}$$

где

$$G = \frac{1}{2}\left(\sqrt{\frac{E_b}{N_0}}\cos\theta + \sqrt{\frac{E_b}{N_0}\cos^2\theta + 4}\right)$$

и $E_\text{b}/N_0 = E_s/(N_0 R)$ — отношение сигнал/шум на бит.

Этими формулами можно смело пользоваться при длине кодов меньше 1000 и при вероятности ошибки меньше 0.1. Листинг 3.2 МАТЛАБ-программы поможет в этом.

Точные формулы границы Шеннона и обсуждение вычислительных аспектов их применения для произвольных длин кодов можно найти в работе [110].

Программа 3.2. Подсчет границы Шеннона вероятности ошибки декодирования по максимуму правдоподобия

```
function PE=boutros_Lbound(N,r,sigma)
% Аппроксимация Boutros-Patel нижней границы
% Шеннона на вероятность ошибки
% n - длина кода
% r - скорость кода
% sigma=sqrt(N0/2) - средне-квадратическое
% отклонение шума
%  PE - оценка вероятности ошибки
global n R;
n=N;
R=r;
%find interval where function has a root
interval= fzero(@sp_equation,1.0);
% refine the root
theta=fzero(@sp_equation,interval);

sint=sin(theta);
cost=cos(theta);
ct=cost/sigma;
st=sint/sigma;
G=0.5*(ct+sqrt(ct^2+4));
EL=-1/2/sigma^2+ct*G/2;
A=G*sint;
numer_q=(A^n)*exp(EL*n);
denom_q=sqrt(1+G^2)*sint*(G*st*sint-cost)*...
sqrt(pi*n);
PE=numer_q/denom_q;
end
%-------------------------------------%
function y=cone_angle(x)
global n R;
y=R-((1+log2(pi*n))/2+log2(cos(x))+ ...
(1-n)*log2(sin(x)))/n;
end
%-------------------------------------%
```

```
function y = sp_equation(x)
global n R;
B=log(2)+log(pi)*n/2-gammaln(n/2);
y=n*R*log(2)-B+logomega(x);
end
%-------------------------------------%
function x = logomega(theta)
global n;
x=log(2)+ (n-1)/2*log(pi)-gammaln((n-1)/2)+...
log(quad(@sinn,0,theta));
end
%-------------------------------------%
function y=sinn(x)
global n;
    y=sin(x).^(n-2);
end
```

Задачи

В задачах данного раздела понадобятся таблицы лучших известных кодов. Постоянно обновляемые таблицы таких кодов и границ на их параметры общедоступны на сайте [77]. Выдержка из этих таблиц приведена на рис. 3.2, 3.3. Лучшие нелинейные коды представлены на сайте [90].

1. Докажите, что коды Хэмминга и код Голея (23,12), $d = 7$, удовлетворяют границе Хэмминга с равенством.

2. Сопоставьте характеристики кодов, дуальных к кодам Хэмминга, с границами на минимальное расстояние.

3. Задайтесь параметрами n и d. Примените границы этой главы для того, чтобы определить диапазон возможных значений размерности кода k.

4. Задайтесь параметрами n и k. Примените границы этой главы для того, чтобы определить диапазон возможных значений минимального расстояния кода d.

n/k	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
7	4	3	2	2	1													
8	4	4	2	2	2	1												
9	4	4	3	2	2	2	1											
10	5	4	4	3	2	2	2	1										
11	6	5	4	4	3	2	2	2	1									
12	6	6	4	4	4	3	2	2	2	1								
13	7	6	5	4	4	4	3	2	2	2	1							
14	8	7	6	5	4	4	4	3	2	2	2	1						
15	8	8	7	6	5	4	4	4	3	2	2	2	1					
16	8	8	8	6	6	5	4	4	4	2	2	2	2	1				
17	9	8	8	7	6	6	5	4	4	3	2	2	2	2	1			
18	10	8	8	8	7	6	6	4	4	4	3	2	2	2	2	1		
19	10	9	8	8	8	7	6	5	4	4	4	3	2	2	2	2	1	
20	11	10	9	8	8	8	7	6	5	4	4	4	3	2	2	2	2	1
21	12	10	10	8	8	8	8	7	6	5	4	4	4	3	2	2	2	2
22	12	11	10	9	8	8	8	8	7	6	5	4	4	4	3	2	2	2
23	12	12	11	10	9	8	8	8	8	7	6	5	4	4	4	3	2	2
24	13	12	12	10	10	8	8	8	8	8	6	6	4	4	4	4	3	2
25	14	12	12	11	10	9	8	8	8	8	6	6	5	4	4	4	4	3
26	14	13	12	12	11	10	9	8	8	8	7	6	6	5	4	4	4	4
27	15	14	13	12	12	10	10	9	8	8	8	7	6	6	5	4	4	4
28	16	14	14	12	12	11	10	10	8	8	8	8	6	6	6	5	4	4
29	16	15	14	13	12	12	11	10	9	8	8	8	7	6	6	6	5	4
30	16	16	15	14	12	12	12	11	10	9	8	8	8	7	6	6	6	5
31	17	16	16	15	13	12	12	12	11	10	9	8	8	8	7	6	6	6
32	18	16	16	16	14	13	12	12	12	10	10	8-9	8	8	8	6-7	6	6
33	18	16	16	16	14	14	12	12	12	11	10	9-10	8-9	8	8	7-8	6-7	6
34	19	17	16	16	15	14	13	12	12	12	10	10	9-10	8-9	8	8	7-8	6-7
35	20	18	16	16	16	15	14	12-13	12	12	11	10	10	9-10	8	8	8	7-8
36	20	18	17	16	16	16	14	13-14	12-13	12	12	11	10	10	8-9	8	8	8
37	20	19	18	17	16	16	15	14	13-14	12-13	12	12	10-11	10	9-10	8-9	8	8
38	21	20	18	18	16	16	16	14	14	13-14	12	12	11-12	10-11	10	9-10	8-9	8
39	22	20	19	18	17	16	16	15	14	14	12-13	12	12	11-12	10-11	10	9-10	8-9
40	22	20	20	18	18	16	16	16	15	14	13-14	12-13	12	12	11-12	10-11	10	9-10
n/k	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**

Рис. 3.2. Минимальные расстояния лучших известных линейных кодов

n/k	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
21	1														
22	2	1													
23	2	2	1												
24	2	2	2	1											
25	2	2	2	2	1										
26	3	2	2	2	2	1									
27	4	3	2	2	2	2	1								
28	4	4	3	2	2	2	2	1							
29	4	4	4	3	2	2	2	2	1						
30	4	4	4	4	3	2	2	2	2	1					
31	5	4	4	4	4	3	2	2	2	2	1				
32	6	5	4	4	4	4	2	2	2	2	2	1			
33	6	6	5	4	4	4	3	2	2	2	2	2	1		
34	6	6	6	4	4	4	4	3	2	2	2	2	2	1	
35	6-7	6	6	5	4	4	4	4	3	2	2	2	2	2	1
36	7-8	6-7	6	6	5	4	4	4	4	3	2	2	2	2	2
37	8	7-8	6-7	6	6	5	4	4	4	4	3	2	2	2	2
38	8	8	6-8	6-7	6	6	5	4	4	4	4	3	2	2	2
39	8	8	7-8	6-8	6-7	6	6	5	4	4	4	4	3	2	2
40	8-9	8	8	7-8	6-8	6-7	6	6	5	4	4	4	4	3	2
n/k	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**	**32**	**33**	**34**	**35**

Рис. 3.3. Минимальные расстояния лучших известных линейных кодов (продолжение)

5. Для лучших коротких кодов со скоростью $R = 1/2$ сравните их характеристики с неасимптотическими формами границ Варшамова–Гилберта и Хэмминга.

6. Постройте бесконечные последовательности хороших кодов с числом информационных символов $k = 1, 2, 3, 4, \ldots$. Сравните результаты с границей Грайсмера.

7. Используя границу Грайсмера, постройте последовательность оценок длин оптимальных кодов с минимальным расстоянием 8. Сравните полученные параметры кодов с параметрами наилучших известных кодов.

8. Напишите программу построения линейных кодов, гарантированно достигающих границу Варшамова–Гилберта. Оцените сложность алгоритма. До каких длин кодов и расстояний ваша программа позволяет получать коды за разумное время (несколько минут на один код)?

9. Проанализируйте поведение чисел вдоль диагоналей таблиц на рис. 3.2, 3.3. Дайте формальное объяснение. Сколько оценок на параметры кодов и какие именно надо уточнить, чтобы в вопросе о минимальном расстоянии кодов длины до 40 не оставалось белых пятен?

Приложение. Тождество Мак-Вильямс

Приведем без доказательства важный и неожиданный результат, называемый тождествами Мак-Вильямс. Эти тождества связывают линейными уравнениями спектр произвольного линейного кода со спектром дуального к нему кода. Коэффициенты уравнений зависят только от длины кодов и не зависят от самих кодов.

Пусть A_w обозначает число слов веса w в линейном (n, k)-коде. Однородный многочлен двух формальных переменных x и y

$$W(x, y) = \sum_{w=0}^{n} A_w x^{n-w} y^w$$

называется *весовой функцией* кода. В теории кодирования и теории связи чаще используется функция одного аргумента

$$W(y) = W(x,y)\Big|_{x=1} = \sum_{w=0}^{n} A_w y^w,$$

называемая *порождающей функцией спектра кода.*

В дальнейшем знак $\perp$ в верхнем индексе переменной указывает на то, что переменная относится к дуальному коду. Например, $W^{\perp}(x,y)$ — весовая функция дуального кода.

Теорема 3.11. *Теорема Мак-Вильямс [10]. Весовые функции q-ичного (n,k)-кода и дуального к нему $(n,n-k)$-кода связаны соотношениями*

$$W^{\perp}(x,y) = q^{-k} W(x+y, x-y). \tag{3.25}$$

Заметим, что обе части (3.25) — линейные функции спектров, т.е. один спектр получается линейным преобразованием другого. Запишем (3.25) в форме

$$A_h^{\perp} = q^{-k} \sum_{w=0}^{n} A_w P_h(w;n). \tag{3.26}$$

Коэффициенты $P_h(w;n)$ уравнений (3.26) являются многочленами степени h переменной w с коэффициентами, зависящими от n. Эти многочлены называют *многочленами Кравчука.* Для кодов над алфавитом размера q они определяются соотношениями

$$P_h(x;n) = \sum_{i=0}^{h} (-1)^i (q-1)^{h-i} \binom{x}{i} \binom{n-x}{h-i}. \tag{3.27}$$

В матричной форме (3.26) имеет вид

$$\boldsymbol{A}^{\perp} = q^{-k} \boldsymbol{P} \boldsymbol{A}, \tag{3.28}$$

где $\boldsymbol{A} = \begin{pmatrix} A_0 & \dots & A_n \end{pmatrix}^{\mathrm{T}}$, $\boldsymbol{A}^{\perp} = \begin{pmatrix} A_0^{\perp} & \dots & A_n^{\perp} \end{pmatrix}^{\mathrm{T}}$ — векторы-столбцы коэффициентов спектра, $\boldsymbol{P} = \{p_{i,j}\}$, $i,j = 0,\dots,n$ — матрица из значений многочленов Кравчука, $p_{i,j} = P_i(j;n)$.

Для практических вычислений удобны рекуррентные формулы [38]

$$p_{i+1,j+1} = p_{i+1,j} - p_{i,j+1} - (q-1)p_{i,j} \tag{3.29}$$

с начальными значениями

$$p_{0,j} = 1, \quad j = 0,,1,...,n \tag{3.30}$$

$$p_{i,0} = \binom{n}{i}(q-1)^i, \quad i = 0,,1,...,n. \tag{3.31}$$

Пример 3.1. Для двоичных кодов длины $n = 7$ формулы (3.28), (3.29) дают следующую матрицу преобразования спектров

$$\boldsymbol{P} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 7 & 5 & 3 & 1 & -1 & -3 & -5 & -7 \\ 21 & 9 & 1 & -3 & -3 & 1 & 9 & 21 \\ 35 & 5 & -5 & -3 & 3 & 5 & -5 & -35 \\ 35 & -5 & -5 & 3 & 3 & -5 & -5 & 35 \\ 21 & -9 & 1 & 3 & -3 & -1 & 9 & 21 \\ 7 & -5 & 3 & -1 & -1 & 3 & -5 & 7 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \end{pmatrix}.$$

Например, дуальным по отношению к коду Хэмминга (7,4) со спектром $\boldsymbol{A} = \begin{pmatrix} 1 & 0 & 0 & 7 & 7 & 0 & 0 & 1 \end{pmatrix}^{\mathrm{T}}$, служит симплексный код (7,3). По формуле (3.28) находим его спектр

$$\boldsymbol{A}^{\perp} = \boldsymbol{P}\boldsymbol{A} = \begin{pmatrix} 1 & 0 & 0 & 0 & 15 & 0 & 0 & 0 \end{pmatrix}^{\mathrm{T}}$$

.

4. Декодирование коротких кодов по максимуму правдоподобия

В этой главе мы рассмотрим методы декодирования, которые могут быть применены к коротким кодам. Эти методы имеют практическую значимость, поскольку короткие коды применяются как сами по себе для передачи коротких сообщений, так и в качестве элементов каскадных конструкций.

Методы декодирования, обсуждаемые в данной главе, в последние годы обрели особую значимость в связи с развитием принципа «распространения доверия» («belief propagation») для декодирования любых кодов, построенных как комбинация других кодов. Основой такого декодирования служат декодеры, обрабатывающие мягкие решения на входе и формирующие мягкие решения на выходе. Метод Бала–Кука–Джелинека–Равива, применяемый к решетке кода, — одно из лучших решений этой задачи.

4.1. Декодирование по максимуму правдоподобия

Предполагается, что канал связи без памяти задан переходными вероятностями $\{p(y|x), x \in X, y \in Y\}$. Такая запись подразумевает,

что канал дискретный. Мы приняли эту модель только для простоты записи. Все рассматриваемые алгоритмы без изменений могут быть применены к наиболее важной с точки зрения практики ситуации, канала с мягкими решениями, когда вход канала дискретный (обычно двоичный), а выход — непрерывный.

Предположим, что для передачи сообщений по этому каналу используется код $C = \{\boldsymbol{c}_m, m = 1, \ldots, M\} \subseteq X^n$ длины n объема M. Напомним, что декодирование последовательности $\boldsymbol{y} = (y_1, \ldots, y_n)$ на выходе канала по максимуму правдоподобия (МП) состоит в выборе такого кодового слова $\boldsymbol{c}_m$, для которого максимальна функция правдоподобия $p(\boldsymbol{y}|\boldsymbol{c}_m)$.

В силу предположения об отсутствии памяти в канале для произвольной входной последовательности $\boldsymbol{x} = (x_1, \ldots, x_n)$ имеем

$$p(\boldsymbol{y}|\boldsymbol{x}) = \prod_{i=1}^{n} p(y_i|x_i).$$

В дальнейшем задача декодирования будет сведена к задаче поиска кратчайшего пути на некотором графе. По этой причине удобнее иметь не мультипликативный, а аддитивный критерий выбора кодового слова. С этой целью вместо функции правдоподобия рассматривают ее логарифм

$$\Lambda_m(\boldsymbol{y}) = \ln p(\boldsymbol{y}|\boldsymbol{c}_m) = \sum_{i=1}^{n} \ln p(y_i|c_{mi}) = \sum_{i=1}^{n} \lambda_m(y_i).$$

Оптимальное в смысле МП декодирование сводится к поиску индекса m, для которого эта сумма максимальна.

В случае двоичных кодов кодовые символы принимают одно из двух значений множества $\{0, 1\}$. Поэтому для каждого значения y на выходе канала можно вычислить так называемое *отношение правдоподобия*

$$L(y) = \ln \frac{p(y|1)}{p(y|0)}. \tag{4.1}$$

При передаче 1 ожидаемое значение отношения правдоподобия положительно, при передаче 0 — отрицательно. Таким образом, посимвольное принятие «жестких» решений по максимуму правдоподобия производится сравнением $L(y)$ с нулем. Чем сильнее отличается от нуля $L(y)$, т.е. чем больше величина $\alpha = |L(y)|$, тем большей

кажется надежность соответствующих жестких решений. Поэтому вектор, составленный из значений $\alpha_i = L(y_i)$, называют *вектором надежностей*.

Введем обозначения $L(\boldsymbol{y}) = (L(y_1), \ldots, L(y_n))$, $\boldsymbol{\xi}_m = (\xi_{m1}, \ldots, \xi_{mn})$, где $\xi_{mi} = 2c_{mi} - 1$, т.е. вектор $\boldsymbol{\xi}$ получается из вектора $\boldsymbol{c}$ заменой нулей и единиц вещественными числами -1 и $+1$ соответственно.

Упражнение 4.1. Докажите, что для канала с двоичным входом декодирование по МП сводится к поиску кодового слова, для которого максимально скалярное произведение

$$L_m(\boldsymbol{y}) = (\boldsymbol{\xi}_m, L(\boldsymbol{y}))\,. \tag{4.2}$$

Решение. Рассмотрим два кодовых слова $\boldsymbol{c}_m$ и $\boldsymbol{c}_{m'}$, $m' \neq m$, и убедимся в том, что из неравенства $\Lambda_m(\boldsymbol{y}) \geq \Lambda_{m'}(\boldsymbol{y})$ следует $L_m(y) \geq L_{m'}(y)$. Обозначим через I_{uv} множество номеров позиций, на которых в слове $\boldsymbol{c}_m$ записан двоичный символ u, а в $\boldsymbol{c}_{m'}$ — символ v. Имеем

$$\begin{aligned}
L_m(\boldsymbol{y}) - L_{m'}(\boldsymbol{y}) &= \sum_{j\in I_{10}} (+1-(-1))L(y_j) + \sum_{j\in I_{01}} (-1-(+1))L(y_j) = \\
&= 2\sum_{j\in I_{10}} L(y_j) - 2\sum_{j\in I_{01}} L(y_j) = \\
&= 2\sum_{j\in I_{10}} \ln\frac{p(y|c_{mj})}{p(y|c_{m'j})} - 2\sum_{j\in I_{01}} \ln\frac{p(y|c_{m'j})}{p(y|c_{mj})} = \\
&= 2\sum_{j=1}^{n} \ln\frac{p(y|c_{mj})}{p(y|c_{m'j})} = \Lambda_m(\boldsymbol{y}) - \Lambda_{m'}(\boldsymbol{y}).
\end{aligned}$$

□

Обратим внимание, что вектор $L(y)$ отношений правдоподобия вычисляется один раз. Затем декодирование по МП сводится к выполнению $n-1$ сложений для каждого из M кодовых слов кода и последующему выбору наибольшего значения. С увеличением числа кодовых слов сложность такого декодирования перебором по кодовым словам становится, конечно, неприемлемо высокой.

Ниже мы рассмотрим некоторые способы сокращения перебора при сохранении экспоненциального характера роста сложности с

увеличением длины кода. Более изощренные методы декодирования будут рассмотрены позже.

Приведем примеры метрик для декодирования по МП.

Пример 4.1. Декодирование по МП в ДСК. Рассмотрим ДСК с переходной вероятностью p_0. Из (4.1) получаем, что

$$L(y) = \begin{cases} \ln \frac{1-p_0}{p_0}, & y = 1; \\ \ln \frac{p_0}{1-p_0}, & y = 0. \end{cases} \tag{4.3}$$

Обозначим $\delta = \ln((1-p_0)/p_0)$. Тогда слагаемые в скалярном произведении (4.2) равны δ при $y_i = c_{mi}$ и $-\delta$ в противном случае. Следовательно,

$$L_m(\boldsymbol{y}) = \delta\left(n - 2d(\boldsymbol{y}, \boldsymbol{c}_m)\right),$$

где $d(\boldsymbol{y}, \boldsymbol{c}_m)$ — расстояние Хэмминга между $\boldsymbol{y}$ и $\boldsymbol{c}_m$. Как и следовало ожидать, декодирование по МП при $\delta > 0$ (т.е. при $p_0 < 1/2$) эквивалентно декодированию по минимуму расстояния Хэмминга.

Пример 4.2. Декодирование по МП в канале с аддитивным белым гауссовским шумом (АБГШ) Предположим, что канал описывается условными плотностями

$$f(y|0) = \frac{1}{\sqrt{\pi N_0}} e^{-\frac{\left(y+\sqrt{E}\right)^2}{N_0}}; \quad f(y|1) = \frac{1}{\sqrt{\pi N_0}} e^{-\frac{\left(y-\sqrt{E}\right)^2}{N_0}}. \tag{4.4}$$

Тогда (4.1) принимает вид

$$L(y) = y\frac{4\sqrt{E}}{N_0}.$$

Постоянный множитель при y не играет роли при принятии решений, поэтому, в соответствии с (4.2), декодирование по МП сводится к максимизации скалярного произведения $(\boldsymbol{\xi}_m, \boldsymbol{y})$.

4.2. Поиск кратчайшего пути в решетке. Алгоритм Витерби

Представление логарифма функции правдоподобия (либо логарифма отношения правдоподобия) кодового слова в виде суммы позволяет интерпретировать эти величины как аддитивную метрику

слова или как его «длину». Если мы найдем удобное представление кода в виде графа, то задача сведется к поиску кратчайшего пути на графе. Такой подход, первоначально предложенный Витерби для декодирования сверточных кодов, впоследствии оказался весьма продуктивным как для анализа, так и для декодирования блоковых кодов.

Пример графа, называемого в теории кодирования *решетчатой диаграммой* или просто *решеткой*, приведен на рис. 4.1. Мы видим, что решетка — это граф, обладающий следующими специальными свойствами.

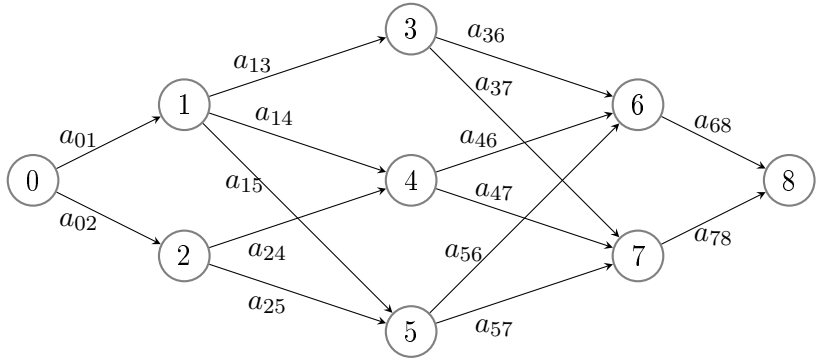


Рис. 4.1. Пример решетки

1. Вершины графа разбиты на непересекающиеся подмножества, называемые *уровнями* или *ярусами*. Вершины каждого уровня связаны только с вершинами соседних уровней.

2. Нулевой ярус и последний ярус содержат по одному узлу.

3. Граф направленный, возможно движение только от уровня с меньшим номером к уровню с большим номером.

4. Ребрам графа приписаны метрики, их называют также весами или длинами. Метрики могут быть как положительными, так и отрицательными. Длина (метрика) пути измеряется как сумма длин (метрик) ребер.

Задача состоит в отыскании кратчайшего пути от начального узла решетки к конечному узлу. (Хотя требование наличия только одного начального и одного конечного узла решетки при описании блоковых кодов решетками не является обязательным, в рамках данного раздела нас будут интересовать только такие решетки.)

Тривиальное решение задачи состоит в полном переборе по множеству всех путей. В решетке на рис. 4.1 таких путей 10 (проверьте правильность подсчета). Для вычисления метрики каждого пути нужно выполнить 3 сложения, т.е. всего 30 сложений. Чтобы принять решение о самом коротком пути, потребуется 9 операций сравнения и выбора.

Нетривиальное решение основано на принципе динамического программирования. Проще говоря, оно основано на том, что в тех узлах, где происходит слияние путей, можно без потери оптимальности принимать решения о лучшем пути и отбрасывать путь с худшей метрикой. В нашем примере такими являются узлы с номерами 4, 5, 6, 7. Окончательное решение будет принято в узле 8 при сравнении двух выживших кандидатов.

Например, в узле 5 мы сравниваем две суммы: $a_{01} + a_{15}$ и $a_{02} + a_{25}$. На последующих ярусах решетки к этим суммам будут прибавляться одни и те же слагаемые. Поэтому без потери оптимальности из этих двух сумм можно выбрать меньшую и закрепить ее за узлом. При использовании такого подхода для выбора наилучшего пути достаточно 13 сложений и 7 операций сравнения и выбора.

Сформулируем алгоритм поиска кратчайшего пути в решетке в общем виде. Алгоритм 4.1, называемый в теории кодирования алгоритмом Витерби, указывает оптимальное правило вычисления кратчайшего пути на графе.

Для того чтобы воспользоваться этим алгоритмом, мы построим описание линейных блоковых кодов с помощью решеток, с ребрами которых ассоциированы кодовые символы кода. После получения последовательности $\boldsymbol{y}$ на выходе канала декодер заменяет кодовые символы на логарифмы их функций правдоподобия или эквивалентные им величины (см. примеры 4.1, 4.2). Например, при декодировании по минимуму расстояния Хэмминга решетка размечается расстояниями Хэмминга между кодовым символом и соот-

Алгоритм 4.1. Алгоритм Витерби поиска кратчайшего пути на графе

Input: Решетка кода с заданными метриками ребер ;
Output: Кратчайший путь в решетке от яруса 0 до яруса на глубине N как последовательность номеров ребер, исходящих из узлов решетки ;
`Инициализация:` Метрика единственного узла нулевого яруса (начального узла) приравнивается нулю, за этим узлом закрепляется «пустой» путь.;
`Цикл по ярусам решетки`;
for *l=1* **to** *N* **do**
  V_l `-- множество узлов на ярусе` l;
  foreach $node \in V_l$ **do**
    1. Находим метрику каждого из путей, ведущих в данный узел, как сумму метрик предшествующих узлов и метрик ребер, связывающих узлы-предшественники с данным узлом.
    2. Среди кандидатов выбираем путь с минимальной метрикой, эту метрику приписываем данному узлу.
    3. Путь, ведущий в узел, вычисляется дописыванием к пути, ведущему в выбранный предшествующий узел, номера ребра, соответствующего переходу из узла-предшественника в данный узел.
  end
end
Вывод: Выбранный путь в конечный узел

ветствующим символом, полученным на выходе канала, а в гауссовском канале записываются просто значения на выходе канала со знаком, который определяется кодовым символом.

В случае ДСК алгоритм Витерби используется для отыскания пути с минимальной метрикой, а в случае гауссовского канала — для отыскания пути с максимальной накопленной вдоль пути ре-

шетки суммой, что соответствует максимуму скалярного произведения в (4.2) или максимуму функции правдоподобия.

4.3. Минимальная решетка кода

Рассмотрим множество кодовых слов произвольного линейного кода. Попробуем построить для него решетку с наименьшим возможным числом узлов. В решетке кода ребрам приписываются значения кодовых символов. Мы будем сначала рассматривать решетки, в которых каждому ребру приписывается ровно один кодовый символ, хотя в общем случае можно приписывать блоки кодовых символов, и тогда решетку называют *секционированной. Несекционированная (побитовая)* решетка кода длины n состоит из $n+1$ ярусов.

Начнем с простого примера. Код $(3, 2)$ с проверкой на четность, заданный порождающей матрицей

$$G = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}, \tag{4.5}$$

может быть описан решеткой, показанной на рис. 4.2. Мы не указываем на этом и следующих рисунках направления ребер, полагая их по умолчанию направленными слева направо. Ребрам решетки приписаны двоичные кодовые символы.

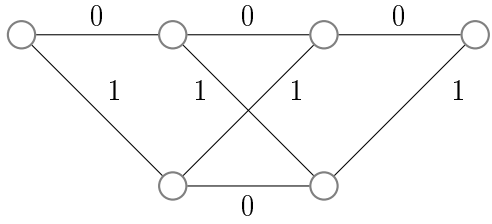


Рис. 4.2. Решетка кода $(3, 2)$

При известной последовательности на выходе канала $\boldsymbol{y} = (y_1, \ldots, y_n)$ кодовые символы $c_i = 0$ или $c_i = 1$ можно заменить функциями правдоподобия $\ln p(y_i|0)$ и $\ln p(y_i|1)$. Тем самым декодирование по максимуму правдоподобия может быть сведено к поиску кратчайшего пути в решетке по алгоритму Витерби.

Упражнение 4.2. При передаче по каналу с гауссовским шумом одного из кодовых слов кода, заданного матрицей (4.5), на выходе канала получена последовательность $\boldsymbol{y} = (-0.83, 0.68, -1.09)$. Выполните декодирование с помощью алгоритма Витерби.

Подсказка: решетка, размеченная с учетом правила, выведенного в примере 4.2, показана на рис. 4.3.

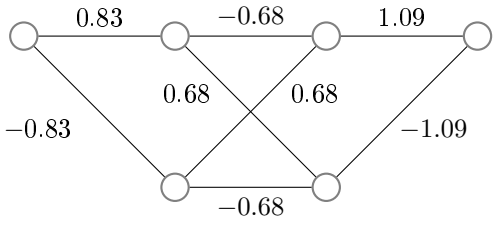


Рис. 4.3. Размеченная решетка кода $(3, 2)$ при получении на выходе канала последовательности $\boldsymbol{y} = (-0.83, 0.68, -1.09)$

Сложность декодирования, очевидно, определяется числом узлов решетки. Таким образом, задача построения эффективного декодера сводится к задаче построения решетки с наименьшим числом узлов.

В общем случае для одного и того же кода можно построить различные решетки, имеющие различную сложность. Последовательность, в которой ξ_i представляет собой число узлов на i-м ярусе решетки, называется *профилем сложности* решетчатого представления кода.

Решетка с профилем $\boldsymbol{\xi} = (\xi_0, \xi_1, ..., \xi_n)$ для данного кода называется *минимальной*, если профиль сложности любой другой решетки этого кода $\boldsymbol{\xi'} = (\xi_0', \xi_1', \ldots, \xi_n')$ удовлетворяет неравенствам $\xi_i' \geq \xi_i$ при всех $i = 0, 1, \ldots, n$.

Из этого определения следует, что мы требуем от минимальной решетки, чтобы она была проще любой другой решетки на каждом ярусе. Это требование кажется чрезмерно жестким, но, как мы увидим, минимальная решетка для каждого кода не только существует, но и может быть довольно легко построена.

Для произвольного (не обязательно линейного) кода $C = \{\boldsymbol{c}_m, m = 1, ..., M\}$ длины n выпишем список его кодовых

слов, $\boldsymbol{c}_m = (c_{m1}, \ldots, c_{mn})$, $m = 1, \ldots, M$. Выберем некоторый индекс $i \in \{0, \ldots, n\}$ и разобьем каждое слово на две части, длины i и $n - i$, $\boldsymbol{c}_m = (\boldsymbol{c}_m^p, \boldsymbol{c}_m^f)$. Верхние индексы «$p$» и «$f$» указывают на «прошлое» (past) и «будущее» (future). Заметим, что в любой решетке кода пути, приходящие в некоторый узел, имеют «общее будущее», и пути, исходящие из одного узла — «общее прошлое». В последнем предложении было бы точнее говорить о кодовых словах, соответствующих путям, но мы и дальше для краткости будем пользоваться этим сленгом, подразумевая существование взаимно-однозначного соответствия между множеством путей и множеством кодовых слов кода.

Рассмотрим множество:

$$F_i = \{\boldsymbol{c}^f : (\boldsymbol{c}^p, \boldsymbol{c}^f) \in C \text{ для некоторого } \boldsymbol{c}^p\}.$$

Заметим, что в F_i каждому $\boldsymbol{c}^p$ могут соответствовать одно или несколько продолжений $\boldsymbol{c}^f$. Следовательно, F_i можно разбить единственным образом на подмножества $F_i(\boldsymbol{c}^p) = \{\boldsymbol{c}^f : (\boldsymbol{c}^p, \boldsymbol{c}^f) \in C\}$. Сопоставим различным таким подмножествам узлы яруса i. Узел v яруса i свяжем ребром с узлом v' яруса $i + 1$, если для некоторого кодового слова прошлое, соответствующее v', является продолжением на один символ одной из последовательностей, ведущих в узел v. Этот кодовый символ приписывается ребру, соединяющему v с v'. Выполнив эту процедуру при $i = 0, ..., M$, получим некоторую решетку T.

Пример 4.3. Рассмотрим линейный код с порождающей матрицей

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

Код содержит 8 кодовых слов

$$\begin{array}{llll} \boldsymbol{c}_0 = (000000), & \boldsymbol{c}_1 = (110100), & \boldsymbol{c}_2 = (101010), & \boldsymbol{c}_3 = (011110) \\ \boldsymbol{c}_4 = (101101), & \boldsymbol{c}_5 = (011001), & \boldsymbol{c}_6 = (000111), & \boldsymbol{c}_7 = (110011) \end{array}$$

В табл. 4.1 приведены множества $F_i(\boldsymbol{c}^p)$ для всех $i = 0, \ldots, 6$. Каждому множеству сопоставлен узел решетки на ярусе i, узлы

пронумерованы буквами из множества $\{a, ..., d\}$. В таблице приведены предшествующие состояния и кодовые символы, соответствующие переходам. С помощью данных, представленных в табл. 4.1, легко построить решетку, показанную на рис. 4.4. Профиль сложности решетки определяется числом различных $F_i(\boldsymbol{c}^p)$ при $i = 0, \ldots, 6$. В данном случае профиль имеет вид (1,2,4,4,4,2,1).

Таблица 4.1. Построение минимальной решетки для кода примера 4.3

i	$\boldsymbol{c}^p$	$F_i(\boldsymbol{c}^p)$	v_i	$\{v_{i-1}\}$	$\{c_i\}$
0	$\emptyset$	$\{\boldsymbol{c}_m, m = 0, \ldots, 7\}$	a	–	–
1	0	00000,11110,11001,00111	a	a	0
	1	10100,01010,01101,10011	b	a	1
2	00	0000,0111	a	a	0
	01	1110,1001	b	a	1
	10	1010,1101	c	b	0
	11	0100,0011	d	b	1
3	000	000,111	a	a	0
	011	110,001	b	b	1
	101	010,101	c	c	1
	110	100,011	d	d	0
4	0000,1101	00	a	a, d	0,1
	0001,1100	11	b	a, d	1,0
	0111,1010	10	c	b, c	1,0
	0110,1011	01	d	b, c	0,1
5	00000,11010,10101,01111	0	a	a, c	0,1
	10110,01100,00011,11001	1	b	b, d	1,0
6	$\{\boldsymbol{c}_m, m = 0, \ldots, 7\}$	$\emptyset$	a	a, b	0,1

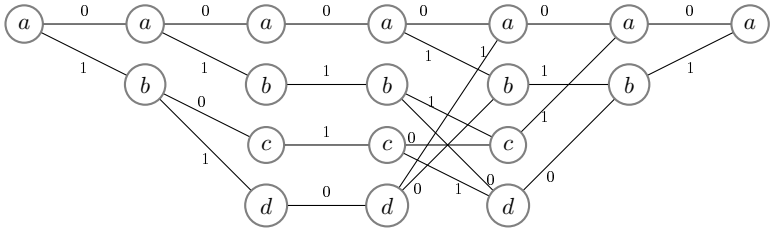


Рис. 4.4. Решетка, соответствующая табл. 4.1

Можно убедиться в том, что полученная решетка минимальна. Для этого рассмотрим произвольную решетку данного кода T'. Как и в любой другой решетке, в T' два слова $\boldsymbol{c}_1 = (\boldsymbol{c}_1^p, \boldsymbol{c}_1^f)$ и $\boldsymbol{c}_2 = (\boldsymbol{c}_2^p, \boldsymbol{c}_2^f)$ могут иметь общую вершину на ярусе i только в том случае, если $F_i(\boldsymbol{c}_1^p) = F_i(\boldsymbol{c}_2^p)$ (два слова имеют общее множество возможных продолжений). Поэтому два пути, проходящие через общий узел в T', также проходят через общий узел и в T, по построению множества F_i. Обратное, вообще говоря, неверно. Поэтому число узлов на ярусе i в T' не может быть меньше числа узлов на этом же ярусе в T при всех i; иными словами, решетка T минимальна.

Подведем итоги построения в виде теоремы.

Теорема 4.1. *Всякий код имеет минимальную решетку, все минимальные решетки совпадают с точностью до нумерации узлов каждого яруса.*

Более точное утверждение и детальное доказательство можно найти, в частности, в работе [112].

Заметим, что при получении минимальной решетки мы объединяли пути, имеющие общее прошлое. В равной степени можно было изменить направление построения и объединять пути, имеющие общее будущее. Подводя итоги, можем сформулировать следующее утверждение.

Признак минимальности решетки: если на каждом ярусе все пути, имеющие общее будущее (прошлое), проходят через один узел решетки, то такая решетка для данного кода минимальна.

4.4. Построение решетки кода по порождающей матрице

Принцип построения минимальных решеток, рассмотренный в предыдущем параграфе, требует анализа всего множества кодовых слов. Следовательно, сложность построения пропорциональна числу кодовых слов, т.е растет экспоненциально с длиной кода. Как и следует ожидать, для линейных кодов задача решается намного проще. Достаточно иметь дело с порождающей либо проверочной матрицей кода.

Опишем одну из возможных конструкций решетки линейного кода.

Для произвольной последовательности $\boldsymbol{x} = (x_1, \ldots, x_n)$ ее *началом* $b(\boldsymbol{x})$ назовем номер позиции первого ненулевого элемента, *концом* $e(\boldsymbol{x})$ — номер позиции последнего ненулевого элемента. Элементы на позициях с номерами $b(\boldsymbol{x}), \ldots, e(\boldsymbol{x}) - 1$ назовем *активными*, число активных элементов называется *спэном* последовательности.

Например, для последовательности $\boldsymbol{x} = (0, 0, 1, 0, 1, 1, 1, 0, 0, 0)$ получаем $b(\boldsymbol{x}) = 3$, $e(\boldsymbol{x}) = 7$, спэн равен 4, позиции с третьей по шестую активны.

Порождающая матрица называется приведенной к *минимальной спэновой форме*(МСФ), если все начала строк различны и все концы строк различны. Для определенности потребуем также, чтобы строки матрицы были упорядочены по возрастанию начал строк.

Не составляет труда привести произвольную порождающую матрицу к минимальной спэновой форме, заменяя ее строки их суммами с другими строками.

Узлы на ярусе i решетки нумеруются значениями информационных символов, соответствующих строкам, активным в i-й позиции.

Пример 4.4. Приведем к МСФ порождающую матрицу

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

Для этого прибавим первую строку ко второй и третьей и переставим вторую и третью строки. Получим

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

Строки упорядочены по началам. Теперь прибавим вторую строку к первой и последнюю строку ко второй. Получаем матрицу в МСФ

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

На нулевом ярусе имеем единственный начальный узел. На первом ярусе решетки номера узлов 0 и 1 определяются значениями первого информационного символа, а переходы из узлов в узлы следующего яруса — совместно номером узла и возможными значениями второго информационного символа и т.д. Решетка кода показана на рис. 4.5.

Рассмотрим, например, построение четвертого яруса решетки. Узлы предыдущего яруса соответствуют значениям информационных символов m_2, m_1, поскольку в столбце с номером 3 активны первая и вторая строки. Узлы следующего яруса соответствуют значениям пар m_3, m_2, поскольку активны вторая и третья строки. Если, например, $m_2 = 1, m_1 = 0$, т.е. предыдущее состояние (1,0), то при $m_3 = 0$ следующим состоянием станет (0,1), т.к. $m_3 = 0, m_2 = 1$. При $m_3 = 1$ произойдет переход в (1,1). На ребрах пишутся значения, соответствующие четвертой позиции кодового слова, т.е. четвертому столбцу порождающей матрицы. При переходе из (1,0) в (0,1) значения информационных символов $m_1 = 0, m_2 = 1, m_3 = 0$ после умножения на соответствующие элементы столбца $(1\ 1\ 1)^{\mathrm{T}}$ и суммирования порождают кодовый символ 1. Аналогично, при переходе в (1,1) информационные символы $m_1 = 0, m_2 = 1, m_3 = 1$ порождают $(0\ 1\ 1)(1\ 1\ 1)^{\mathrm{T}} = 0$.

Профиль сложности решетки (1,2,4,4,4,2,1), т.е. совпадает с оптимальным профилем, построенным в предыдущем параграфе. Немного более простую решетку можно получить, объединив второй и третий ярусы в одну секцию. Секционированная решетка показана на рис. 4.6. Ее профиль сложности (1,2,4,4,2,1).

Отметим, что в некоторых случаях секционирование позволяет не только улучшить профиль сложности, но и уменьшить (не больше чем в 2 раза) максимальное число узлов на ярусах решетки.

Этот пример легко обобщить на произвольную матрицу в МСФ. Номера узлов решетки на ярусе i определяются значениями информационных символов, соответствующих строкам, которые активны в позиции с номером i. Кодовые символы, соответствующие переходам из узла в узел, вычисляются как линейная комбинация активных элементов соответствующего столбца порождающей матрицы.

Рассмотрим вопрос о минимальности.

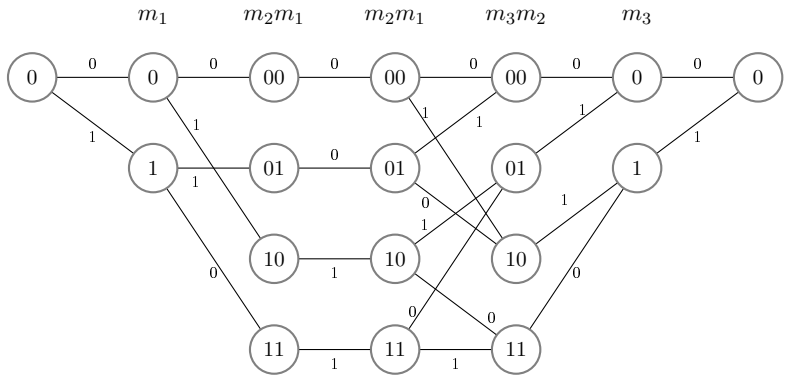


Рис. 4.5. Побитовая решетка кода примера 4.4

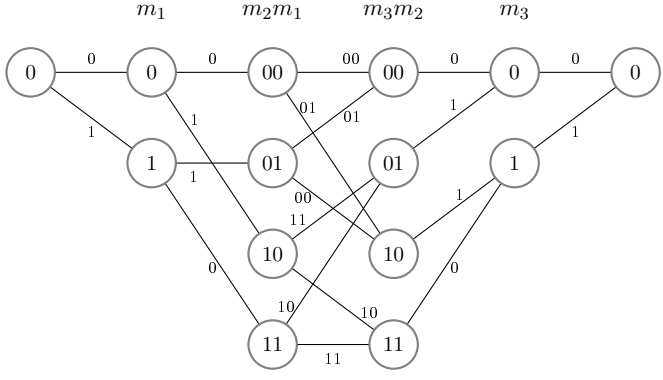


Рис. 4.6. Секционированная решетка кода примера 4.4

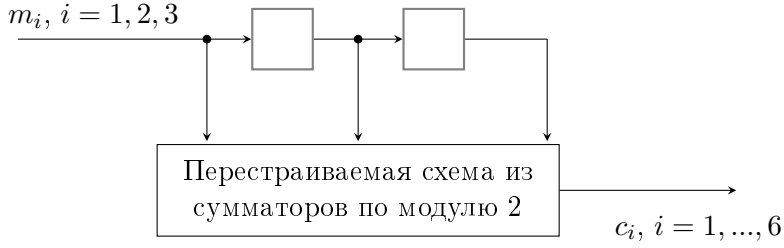


Рис. 4.7. Кодер на основе конечного автомата 4.4

Теорема 4.2. *Решетка, получаемая из порождающей матрицы в МСФ, минимальна.*

Доказательство. Достаточно показать, что пути, определяющие кодовые слова с одинаковыми последовательностями $\boldsymbol{c}^f$ некоторой длины $n-l$, не проходят через различные узлы на ярусе с номером l.

Заметим, что для данного слова узел, через который проходит путь на ярусе l, определяется значениями информационных символов, соответствующих активным на этом ярусе строкам. Эти строки линейно независимы и заканчиваются на ярусах с номерами больше l. Поэтому нетривиальные линейные комбинации этих строк отличны от нуля хотя бы на одной из позиций старше l.

Предположим теперь, что найдутся два слова с одинаковым будущим, проходящие через разные узлы на ярусе l. Их сумма образует слово, которое активно на ярусе l и равно нулю на позициях с номерами больше l. Поскольку таких слов быть не может, слова с одинаковым будущим проходят через одни и те же узлы, т.е. решетка минимальна. □

Решетки, показанные на рис. 4.5 и 4.6, допускают следующую важную интерпретацию. Узлам решетки могут быть сопоставлены состояния регистра сдвига, на который последовательно поступают информационные символы. Этот регистр может быть использован для формирования кодовых слов, поскольку кодовые символы зависят только от символов, находящихся на данном такте в регистре сдвига. Кодер, построенный по такому принципу, показан на рис. 4.7. До начала работы кодера в регистре записаны нули. Затем, такт за тактом, на вход регистра поступают информационные символы, а перестраиваемая схема из сумматоров по модулю 2 вычисляет соответствующие кодовые символы. Очевидно, связи в решетке кода в точности отображают переходы регистра из состояния в состояние, а разметка решетки полностью определяется связями ячеек регистра с сумматорами. Такая интерпретация важна для понимания связи между блоковыми и сверточными кодами и подсказывает возможность получения блоковых кодов с малой сложностью решетки из сверточных кодов.

4.5. Построение решетки кода по проверочной матрице

Логика предшествующего рассмотрения подсказывает, что для высокоскоростных кодов (со скоростью больше $1/2$) представление кодов проверочной матрицей компактнее, чем порождающей матрицей. Сложность декодирования должна определяться параметрами проверочной матрицы, и решетка, построенная по проверочной матрице, должна быть проще.

С другой стороны, теорема 4.1 утверждает, что минимальная решетка кода единственна с точностью до нумерации узлов ярусов решетки. Интересно, что теория решетчатого описания блоковых кодов началась именно с построения так называемых синдромных решеток (см. работы [43] и [116]) для кодов со скоростью больше $1/2$. И только через несколько лет пришло понимание того, что решетчатое представление может быть эффективным для декодирования любых линейных кодов, и что существует несколько одинаково эффективных способов построения минимальных решеток.

Хотя в нашем распоряжении уже есть хороший способ построения минимальных решеток, полезно рассмотреть еще один способ, исторически предшествовавший предыдущему. Начнем с примера.

Пример 4.5. Проверочная матрица кода из примера 4.4 может быть записана, в частности, в виде

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

Построим побитовую решетку кода, нумеруя узлы каждого яруса накопленными синдромами, т.е. произведениями начальной части кодового слова на соответствующую подматрицу проверочной матрицы. Начальному узлу решетки соответствует нулевой синдром. Синдром последовательностей длины 1 может быть равен одному из двух векторов множества $\{(0,0,0), (1,0,0)\}$. Синдромы последовательностей длины 2 принадлежат множеству $\{(0,0,0), (1,0,0), (1,0,1), (0,0,1)\}$ и т.д. На начальном этапе построения на каждом следующем ярусе различных синдромов становится вдвое

больше, чем на предыдущем ярусе. Поскольку синдромов не может быть больше чем $2^{n-k} = 2^r = 8$, понятно, что синдромы начнут совпадать и общее число узлов на каждом ярусе ограничено сверху величиной $2^r = 8$. Полученная *синдромная решетка* данного кода показана на рис. 4.8.

Построение, описанное в примере, тривиально обобщается на произвольный линейный код.

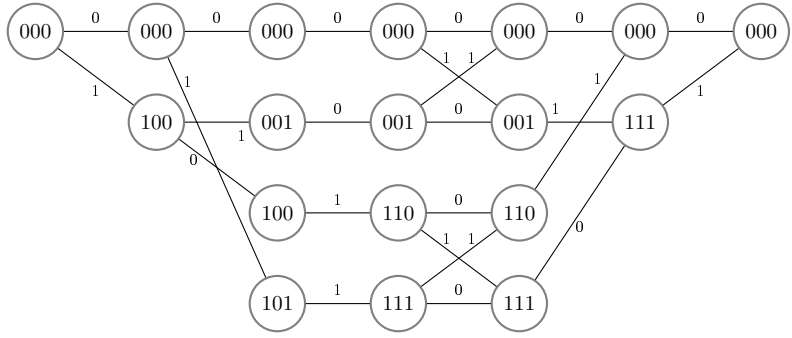


Рис. 4.8. Синдромная решетка кода примера 4.5

Теорема 4.3. *Синдромная решетка минимальна.*

Доказательство. Достаточно показать, что пути, определяющие кодовые слова с одинаковыми последовательностями $\boldsymbol{c}^f$, никогда не проходят через различные узлы. Рассмотрим кодовое слово $\boldsymbol{c} = (\boldsymbol{c}^p, \boldsymbol{c}^f)$. Частичные синдромы, вычисленные по $\boldsymbol{c}^p$ и $\boldsymbol{c}^f$, совпадают. Следовательно, все совпадающие $\boldsymbol{c}^f$ исходят из одного и того же узла, определяемого частичным синдромом, вычисленным по $\boldsymbol{c}^p$. □

Заканчивая обсуждение способов построения решеток, заметим, что минимальная решетка каждого кода однозначно определена с точностью до нумерации узлов уровней, но решетки эквивалентных кодов, отличающихся порядком следования символов, могут иметь различную и довольно сильно отличающуюся сложность. Тривиальной верхней границей на максимальную сложность (максимальное число узлов на ярусах решетки) для любого кода является граница

$$\xi_{\max} = \max_i \xi_i \leq \min\{2^k, 2^{n-k}\}.$$

Истинное значение сложности может быть заметно меньше. Поиск кода с минимальной сложностью среди эквивалентных кодов, отличающихся порядком следования символов, является экспоненциально сложной задачей. Еще сложнее задача построения хороших кодов, сочетающих оптимальную корректирующую способность и малую сложность декодирования по решетке.

4.6. Декодирование по максимуму апостериорной вероятности с мягкими решениями. Алгоритм БКДР

До сих пор, говоря о декодировании по максимуму апостериорной вероятности (МАВ), мы подразумевали декодирование в пользу сообщения или кодового слова, апостериорная вероятность которого при известной последовательности на выходе канала максимальна. При известных и неравных априорных вероятностях сообщений МАВ-декодирование, вообще говоря, эффективнее МП-декодирования. Можно рассматривать в качестве сообщений отдельные информационные или избыточные символы. Тогда декодирование по МАВ будет формировать побитово-оптимальное решение в отличие от МП-декодирования, при котором формируется решение обо всем кодовом слове. Может оказаться, что посимвольное МАВ-декодирование обеспечивает меньшую вероятность ошибки на бит передаваемой информации, чем декодирование по МП, даже при равных вероятностях входных символов.

Начиная с середины 90-х годов прошлого века, актуальность посимвольного МАВ-декодирования резко возросла, что привлекло значительное внимание к маленькой заметке Бала–Кука–Джелинека–Равива (БКДР), опубликованной в 1974 году. Причина в том, что появились на свет способы каскадирования кодов, для которых оказалось эффективным итеративное декодирование. Поясним идею, лежащую в его основе.

Одни и те же информационные символы кодируются двумя или несколькими кодерами. После передачи данных по каналу мягкие решения демодулятора поступают на вход одного из декодеров. Результаты декодирования поступают на другой декодер. Решения, принятые первым декодером, рассматриваются этим вторым декодером как выход некоторого канала, и решения второго декодера снова поступят на вход первого декодера в надежде на то, что часть ошибок уже исправлена, и во второй попытке первый декодер лучше сделает свою работу, чем в первой. Обычно выполняется несколько десятков итераций. Такого роды методы получили общее название алгоритмов на основе распространения доверия (belief propagation).

Мы уже знаем, что декодирование мягкого выхода канала много эффективнее, чем декодирование по жестким решениям. Следовательно, в такой итеративной схеме каждый декодер должен не только принимать решения, но и выдавать информацию об их надежности.

Итак, в схемах итеративного декодирования востребованы побитовые MAB-декодеры, обрабатывающие мягкие решения на входе и формирующие мягкие решения на выходе. На английском языке такие декодеры получили аббревиатуру SISO (soft input, soft output). Именно такую задачу решает декодер БКДР, рассматриваемый ниже в данном параграфе.

Рассмотрим двоичный блоковый код $C = \{\boldsymbol{c}_m, m = 1, \ldots, M\} \subseteq \subseteq \{0,1\}^n$, описываемый некоторой решеткой, и последовательность $\boldsymbol{y} = (y_1, \ldots, y_n) \in Y^n$ на выходе канала. Апостериорные вероятности символов $c \in \{0,1\}$ на позиции с номером t записываются как

$$p(c_t = c|\boldsymbol{y}) = \frac{p(c_t = c, \boldsymbol{y})}{p(\boldsymbol{y})}, \tag{4.6}$$

где

$$p(c_t = c, \boldsymbol{y}) = \sum_{\boldsymbol{c} \in C_t(c)} p(\boldsymbol{c}, \boldsymbol{y}), \tag{4.7}$$

и $C_t(c)$ обозначает ту часть кодовых слов, в которых символ на позиции с номером t равен c. Сложность вычислений по этим формулам имеет тот же порядок, что и число кодовых слов в коде.

Наша задача — упростить вычисления, используя решетчатое представление кода. Идея, как и в алгоритме Витерби, состоит в том, чтобы при вычислениях апостериорных вероятностей для яруса t максимально использовать результаты вычислений, уже выполненных для других ярусов.

Суммирование по кодовым словам в (4.7) мы заменим суммированием по множеству состояний (узлов решетки). Для этого введем обозначение s_t для состояния (точнее, номера состояния) на ярусе t. Мы начнем с вычисления условных вероятностей

$$\Pr\left(s_{t-1}=m', s_t=m|\boldsymbol{y}\right)=\frac{\Pr\left(s_{t-1}=m', s_t=m, \boldsymbol{y}\right)}{p\left(\boldsymbol{y}\right)}. \tag{4.8}$$

Числитель наиболее интересен, поскольку знаменатель не зависит от информационных и кодовых символов. Введем специальное обозначение для числителя:

$$\sigma_t(m', m)=\Pr(s_{t-1}=m', s_t=m, \boldsymbol{y}). \tag{4.9}$$

Как мы увидим позже, по этим вероятностям легко находятся искомые апостериорные вероятности (4.6). Далее будет использована сокращенная запись $\boldsymbol{y}_i^j=(y_i, y_{i+1}, \ldots, y_j)$ для подпоследовательностей последовательности $\boldsymbol{y}$. Из (4.9), разбивая событие $(s_{t-1}=m', s_t=m, y)$ на прошлое, настоящее и будущее, получаем

$$\sigma_t(m', m)=\Pr\left((s_{t-1}=m', \boldsymbol{y}_1^{t-1}), (s_t=m, y_t), (\boldsymbol{y}_{t+1}^n)\right). \tag{4.10}$$

Совместная вероятность трех событий раскладывается в произведение $\Pr(A_1A_2A_3)=\Pr(A_1)\Pr(A_2|A_1)\Pr(A_3|A_1A_2)$. В нашем случае мы должны принять во внимание, что символы последовательности $\boldsymbol{y}=\boldsymbol{y}_1^n=(\boldsymbol{y}_1^{t-1}, y_t, \boldsymbol{y}_{t+1}^n)$ зависимы, поскольку получены в результате передачи некоторого кодового слова. В то же время, по предположению об отсутствии памяти в канале, символы последовательности $\boldsymbol{y}_1^n$ условно независимы при известном переданном кодовом слове. Поэтому при известном кодовом символе c_t условное распределение вероятностей для y_t однозначно определено. В свою очередь, кодовый символ c_t при известном номере узла s_{t-1} уже не зависит от предыдущих кодовых символов, следовательно, $\boldsymbol{y}_{t+1}^n$

не зависит от $\boldsymbol{y}_1^t$ при известном s_t. Поэтому вероятности событий, входящих в (4.10), равны

$$\Pr(A_1) = \Pr(s_{t-1} = m', \boldsymbol{y}_1^{t-1});$$

$$\Pr(A_2|A_1) = \Pr(s_t = m, y_t|s_{t-1} = m', \boldsymbol{y}_1^{t-1}) = \Pr(s_t = m, y_t|s_{t-1} = m');$$

$$\Pr(A_3|A_1A_2) = \Pr(\boldsymbol{y}_{t+1}^n|s_{t-1} = m', s_t = m, \boldsymbol{y}_1^t) = \Pr(\boldsymbol{y}_{t+1}^n|s_t = m).$$

Введем вспомогательные обозначения

$$\alpha_t(m) = \Pr(A_1) = \Pr(s_t = m, \boldsymbol{y}_1^t); \tag{4.11}$$

$$\gamma_t(m', m) = \Pr(s_t = m, y_t|s_{t-1} = m'); \tag{4.12}$$

$$\beta_t(m) = \Pr(\boldsymbol{y}_{t+1}^n|s_t = m). \tag{4.13}$$

В этих обозначениях

$$\sigma_t(m', m) = \alpha_{t-1}(m')\gamma_t(m', m)\beta_t(m). \tag{4.14}$$

Следующий шаг — вывод рекуррентных соотношений для сомножителей α_t, γ_t и β_t. По формуле полной вероятности запишем

$$\alpha_t(m) = \sum_{m'} \Pr(s_{t-1} = m', \boldsymbol{y}_1^{t-1}) \Pr(s_t = m, y_t|s_{t-1} = m', \boldsymbol{y}_1^{t-1}).$$

Условие $\boldsymbol{y}_1^{t-1}$ при известном s_{t-1} можно опустить, поэтому получаем рекуррентное соотношение

$$\alpha_t(m) = \sum_{m'} \alpha_{t-1}(m')\gamma_t(m', m). \tag{4.15}$$

с начальными условиями

$$\alpha_0(m) = \begin{cases} 1, & m = 0; \\ 0, & m \neq 0. \end{cases} \tag{4.16}$$

Аналогично, но как бы двигаясь в обратном направлении, запишем

$$\begin{aligned}
\beta_t(m) &= \sum_{m'} \Pr(s_{t+1}=m', \boldsymbol{y}_{t+1}^n | s_t = m) = \\
&= \sum_{m'} \Pr(s_{t+1}=m', y_{t+1}, \boldsymbol{y}_{t+2}^n | s_t = m) = \\
&= \sum_{m'} \Pr(s_{t+1}=m', y_{t+1} | s_t = m, \boldsymbol{y}_{t+2}^n) \times \\
&\quad \times \Pr(\boldsymbol{y}_{t+2}^n | s_t = m, s_{t+1}=m', y_{t+1}) = \\
&= \sum_{m'} \Pr(s_{t+1}=m', y_{t+1} | s_t = m) \Pr(\boldsymbol{y}_{t+2}^n | s_{t+1}=m').
\end{aligned}$$

В результате приходим к рекурсии

$$\beta_t(m) = \sum_{m'} \beta_{t+1}(m') \gamma_{t+1}(m', m) \tag{4.17}$$

с граничными условиями

$$\beta_n(m) = \begin{cases} 1, & m = 0; \\ 0, & m \neq 0. \end{cases} \tag{4.18}$$

Далее,

$$\begin{aligned}
\gamma_t(m', m) &= \Pr(s_t = m, y_t | s_{t-1} = m') = \\
&= p(s_t = m | s_{t-1} = m') p(y_t | s_t = m, s_{t-1} = m') = \\
&= p(c_t) p(y_t | c_t) = \qquad (4.19) \\
&= p(u_t) p(y_t | c_t) \qquad (4.20)
\end{aligned}$$

для тех пар (m, m'), которые связаны ребрами решетки, и $\gamma_t(m', m) = 0$ для остальных пар. В формулах (4.19) и (4.20) мы приняли во внимание, что переходы из состояния в состояние определяются однозначно соответствующими кодовыми (информационными) символами. (Если в решетке есть параллельные ребра, формула изменится, появится суммирование по соответствующим кодовым или информационным символам.) Вероятность $p(y_t|c_t)$ определяется переходными вероятностями (или условными плотностями) канала, а вероятность $p(c_t)$ (или $p(u_t)$) представляет собой априорную вероятность кодового символа c_t (или информационного u_t).

Мы пользуемся (4.19) или (4.20) в зависимости от того, какие априорные вероятности известны при декодировании.

Пусть теперь $S_t(c)$ обозначает множество таких пар (m, m'), которым соответствует значение $c_t = c$. Тогда из (4.6—4.9) получаем

$$p(c_t = c|\boldsymbol{y}) = \frac{\sum\limits_{(m',m)\in S_t(c)} \sigma_t(m', m)}{p(\boldsymbol{y})}. \tag{4.21}$$

Если нас интересуют апостериорные вероятности не кодовых, а информационных символов $u_t \in \{0, 1\}$, управляющих переходами в решетке, то множество переходов на ярусе t разбивается на подмножества $U_t(u)$ и вместо (4.21) используют

$$p(u_t = u|\boldsymbol{y}) = \frac{\sum\limits_{(m',m)\in U_t(u)} \sigma_t(m', m)}{p(\boldsymbol{y})}. \tag{4.22}$$

В обоих случаях неизвестную вероятность $p(\boldsymbol{y})$ можно определить, например, из условия нормировки $p(c_t = 0|\boldsymbol{y})+p(c_t = 1|\boldsymbol{y}) = 1$. Кроме того,

$$p(\boldsymbol{y}) = \alpha_n(0) = \beta_0(0).$$

В знании $p(\boldsymbol{y})$ нет необходимости, если нас интересуют мягкие решения в форме логарифмов отношений апостериорных вероятностей

$$\lambda_t = \ln \frac{p(c_t = 1|\boldsymbol{y})}{p(c_t = 0|\boldsymbol{y})} = \ln \frac{\sum\limits_{(m',m)\in S_t(1)} \sigma_t(m', m)}{\sum\limits_{(m',m)\in S_t(0)} \sigma_t(m', m)}. \tag{4.23}$$

Жесткое решение по МАВ принимается по правилу

$$\hat{c}_t = \begin{cases} 1, & \lambda_t > 0; \\ 0, & \lambda_t \le 0, \end{cases} \tag{4.24}$$

а величина $|\lambda_t|$ характеризует надежность решения.

Подведем итоги в форме алгоритма 4.2 декодирования по МАВ.

Для наглядности мы представили алгоритм в виде пяти циклов, хотя при практической реализации достаточно двух проходов по решетке, прямого и обратного. Величины γ и α вычисляются при прямом проходе, а σ и λ — при обратном.

Алгоритм 4.2. Алгоритм БКДР

Input: Последовательность $\boldsymbol{y}$ на выходе канала, априорные вероятности информационных символов $u_1, \ldots, u_k$ либо кодовых символов $c_1, \ldots, c_n$;

Output: Последовательность логарифмов отношений правдоподобия $\lambda_1, \ldots, \lambda_n$;

`Инициализация`;

for t=0 **to** n **do**

для всех пар (m', m) вычисляются $\gamma_t(m', m)$ по формуле (4.19) или (4.20);

end

`Прямой проход`;

for t=0 **to** n **do**

для всех m вычисляются величины $\alpha_t(m)$;

end

`Обратный проход`;

for t=0 **to** n **do**

для всех m вычисляются величины $\beta_t(m)$;

end

`Подсчет` σ;

for t=0 **to** n **do**

для всех пар (m', m) вычисляются $\sigma_t(m', m)$ по формуле (4.14)

end

for t=0 **to** n **do**

по формуле (4.23) находим λ_t,

end

Вывод: Последовательность λ_t, $t = 1, 2, \ldots, n$

Пример 4.6. Рассмотрим код с порождающей матрицей

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

Предположим, что в результате передачи по ДСК с переходной вероятностью $p_0 = 0.1$ принята последовательность $\boldsymbol{y} = (0, 0, 1, 0, 1)$. Примем входные вероятности кодовых слов одина-

ковыми и подсчитаем апостериорные вероятности кодовых слов, информационных символов и отдельных кодовых символов. Секционированная решетка для этого кода показана на рис. 4.9,*а*. Над решеткой выписана полученная из канала последовательность.

Вычисления, выполненные для «прямого» решения задачи (без использования решетки), показаны в табл. 4.2. Эти вычисления будут служить ориентиром для проверки правильности вычислений по решетке.

Таблица 4.2. Прямое вычисление результатов декодирования для примера 4.6

Информационные символы u_1, u_2	Кодовое слово $\boldsymbol{c} = (c_1, .., c_5)$	Функция правдоподобия $p(\boldsymbol{y}\|\boldsymbol{c})$	Апостериорная вероятность $p(\boldsymbol{c}\|\boldsymbol{y}) = \frac{p(\boldsymbol{y}\|\boldsymbol{c})p(\boldsymbol{c})}{p(\boldsymbol{y})}$
$\boldsymbol{y} = (0,0,1,0,1)$			
00	00000	$p_0^2(1-p_0)^3 = 0.00729$	0.45
01	01011	$p_0^3(1-p_0)^2 = 0.00081$	0.05
10	11100	$p_0^3(1-p_0)^2 = 0.00081$	0.05
11	10111	$p_0^2(1-p_0)^3 = 0.00729$	0.45
$p(\boldsymbol{y}) = \sum_{m=0}^{3} p(\boldsymbol{y}\|\boldsymbol{c}_m)p(\boldsymbol{c}_m) = 0.00405$			
Символы		Логарифмы отношений апостериорных вероятностей $\ln \frac{p(1\|\boldsymbol{y})}{p(0\|\boldsymbol{y})}$	
u_1		0	
u_2		0	
c_1, c_3, c_4, c_5		0	
c_2		-2.1972	

На рис. 4.9 показаны все шаги декодирования по алгоритму БКДР. Для формирования отношений правдоподобия в соответствии с (4.23), нужно сначала просуммировать значения σ_t, соот-

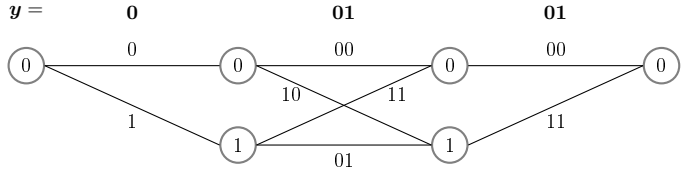
ветствующие значениям информационных символов 1. Затем эту сумму нужно поделить на сумму σ_t, соответствующих значениям 0. Для наглядности, переходы по 1 показаны жирными линиями. Выполняя подсчеты, убеждаемся в совпадении результатов декодирования по МАП с результатами, представленными в табл. 4.2. По данной решетке мы можем оценить логарифмы отношений апостериорных вероятностей для информационных символов. Чтобы получить оценки для кодовых символов, нужно суммировать σ_t с учетом значений соответствующих кодовых символов. Например, в данном примере значению $c_2 = 0$ соответствуют переходы $0 \to 0$ и $1 \to 1$. Другие два перехода соответствуют $c_2 = 1$.

Имеем

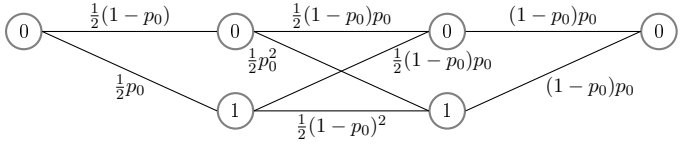
$$\ln \frac{0.0002025 + 0.0002025}{0.0018225 + 0.0018225} = -2.1972,$$

что в точности совпадает с величиной в табл. 4.2.

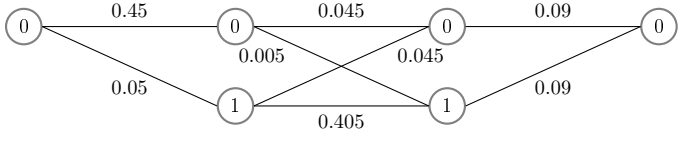
В заключение отметим важный аспект практического применения алгоритма БКДР — высокие требования к точности вычислений. В формулах фигурируют значения вероятностей последовательностей. При большой длине последовательностей эти вероятности становятся маленькими вещественными числами, и для их записи потребуется представление в формате плавающей запятой. При реализации декодирования в виде микросхем или с помощью специализированных вычислителей использование вычислений с плавающей запятой снижает быстродействие и в разы увеличивает сложность и, в конечном итоге, стоимость устройств. Поэтому на практике вместо формул (4.14)–(4.23) используются их аппроксимации, основанные на выполнении действий только над логарифмами вероятностей. Мы вернемся к этому вопросу позже при изучении итеративных алгоритмов декодирования (см. параграф 10.3.3).



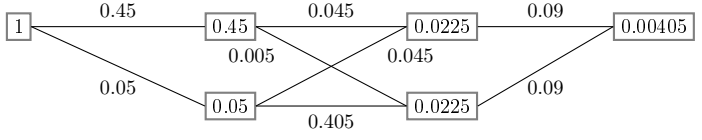
а) Решетка кода и принятая последовательность $\boldsymbol{y}$



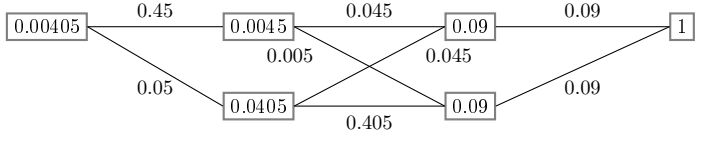
б) Формулы для подсчета γ



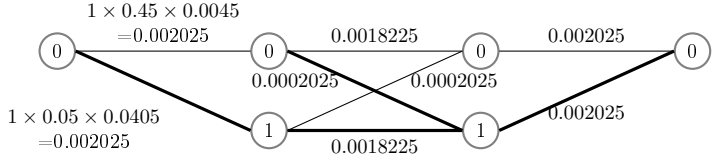
в) Подсчет γ



г) Подсчет α



д) Подсчет β



е) Подсчет σ

Рис. 4.9. МАП-декодирование по алгоритму БКДР для кода примера 4.6

4.7. Сложность решеток линейных кодов и сложность декодирования по максимуму правдоподобия

Каково место рассмотренных выше решетчатых представлений кодов в теории и практике кодирования? Представление кодов решетками, к сожалению, не позволяет уйти от экспоненциальной сложности декодирования. Несмотря на очень быстрый прогресс в области микроэлектроники, вычислительная и аппаратная сложность более 2^8–2^{12} арифметических операций или элементов памяти на один символ кодового слова остается предельно допустимой величиной. Исходя из этой очень грубой оценки, можно сразу сказать, что при скорости кодов около $k/n = 1/2$ коды длин порядка 50–100 могут быть декодированы с помощью решеток, только если для них существуют эффективные представления с малой сложностью. Приведенные в этом параграфе оценки сложности решеток оказываются довольно точными как в асимптотике, так и для кодов конечной длины.

4.7.1. Свойства минимальных решеток линейных кодов

Вернемся к обозначениям параграфа 4.3, в частности, для последовательности $\boldsymbol{a}$ длины i через $F_i(\boldsymbol{a})$ будем обозначать множество всех $\boldsymbol{c}^f$ таких, что $\boldsymbol{c} = (\boldsymbol{a}, \boldsymbol{c}^f)$ — кодовое слово кода C.

Свойство 4.1. *Для линейного кода множество $F_i(\mathbf{0})$ — линейный код длины $n - i$.*

Это свойство непосредственно вытекает из линейности исходного кода длины n.

Свойство 4.2. *Для линейного кода любое непустое множество $F_i(\boldsymbol{a})$ при $\boldsymbol{a} \neq \mathbf{0}$ либо совпадает с $F_i(\mathbf{0})$, либо не пересекается с ним.*

Доказательство Предположим сначала, что $(\boldsymbol{a}, \mathbf{0})$ принадлежит коду. Тогда из $\boldsymbol{b} \in F_i(\boldsymbol{a})$ следует $(\boldsymbol{a}, \boldsymbol{b}) \in C$ и, в силу линейности, $(\mathbf{0}, \boldsymbol{b}) \in C$, т.е. $\boldsymbol{b} \in F_i(\mathbf{0})$. Следовательно, $F_i(\boldsymbol{a}) = F_i(\mathbf{0})$.

В противном случае, когда $(\boldsymbol{a}, \mathbf{0})$ — не кодовое слово, а $(\boldsymbol{a}, \boldsymbol{b})$ — кодовое слово, их сумма $(\mathbf{0}, \boldsymbol{b})$ не может принадлежать коду, а значит, множества $F_i(\boldsymbol{a})$ и $F_i(\mathbf{0})$ не пересекаются. □

Из этого свойства следует еще одно свойство.

Свойство 4.3. *Для линейного кода, если множество $F_i(\boldsymbol{a})$ не пусто и не совпадает с $F_i(\mathbf{0})$, то оно может быть получено из $F_i(\mathbf{0})$ «сдвигом», т.е. прибавлением ко всем его элементам одного и того же вектора $\boldsymbol{b}$ такого, что $(\boldsymbol{a}, \boldsymbol{b})$ — кодовое слово.*

В терминах алгебры множеств $F_i(\mathbf{0})$ образует линейное подпространство линейного пространства всех таких последовательностей $\boldsymbol{b}$, что $(\boldsymbol{a}, \boldsymbol{b})$ — кодовое слово при некотором $\boldsymbol{a}$. При этом $F_i(\boldsymbol{a})$ либо совпадает с $F_i(\mathbf{0})$, либо является смежным классом в этом пространстве, рассматриваемом как группа относительно операции покомпонентного сложения векторов. Из свойств смежных классов заключаем:

Свойство 4.4. *Для линейного кода множества $F_i(\boldsymbol{a})$ при различных $\boldsymbol{a}$ либо совпадают, либо не пересекаются.*

Еще одно очевидное следствие из сформулированных выше свойств:

Свойство 4.5. *Для исходного линейного кода с минимальным расстоянием d множество $F_i(\mathbf{0})$ — линейный код длины $n - i$ с минимальным расстоянием не меньше d.*

Аналогично множеству продолжений кодовых слов $F_i(\boldsymbol{a})$ можно рассматривать множества начал кодовых слов, которые мы обозначаем как $P_i(\boldsymbol{a})$ («P» — аббревиатура от «past»). Все свойства множества продолжений справедливы для множества начал, в частности, $P_i(\mathbf{0})$ — линейный код длины i с минимальным расстоянием не меньше минимального расстояния d исходного кода.

Мы установили, что для любого линейного кода для любой фиксированной точки i можно вычислить размерность k_i^p подкода $P_i(\mathbf{0})$ пространства начал и подкода $F_i(\mathbf{0})$ пространства продолжений k_{n-i}^f . Индекс $n - i$ указывает длину последовательностей, образующих подкод $F_i(\mathbf{0})$.

Следующая теорема была практически одновременно установлена в работах Мудера [94] и Форни [74].

Теорема 4.4. *Для линейного (n,k)-кода C с профилем сложности его минимальной решетки $\boldsymbol{\xi} = (\xi_0, \xi_1, \ldots, \xi_n)$ при всех $i = 1, \ldots, n-1$ имеет место тождество*

$$k = k_i^p + \log_2 \xi_i + k_{n-i}^f. \tag{4.25}$$

Доказательство. Множество путей, приходящих в один узел яруса i решетки, соответствует одному из множеств $P_i(\boldsymbol{a})$. Множество путей, исходящих из этого узла, соответствует одному из множеств $F_i(\boldsymbol{b})$. Поскольку мощности этих множеств не зависят от конкретных $\boldsymbol{a}$ и $\boldsymbol{b}$, заключаем, что через каждый узел проходит одинаковое число путей $2^{k_i^p + k_{n-i}^f}$. Множества $P_i(\boldsymbol{a})$ и $F_i(\boldsymbol{b})$, соответствующие разным узлам решетки, не пересекаются, а общее число путей равно числу кодовых слов. Поэтому, умножив число путей через узел на число узлов, получаем

$$2^k = \xi \times 2^{k_i^p + k_{n-i}^f}.$$

После логарифмирования получим утверждение теоремы. □

4.7.2. Границы сложности решеток

Теорема 4.4 связывает параметры кода (длину и размерность) с профилем сложности решетки через параметры некоторых подкодов кода. Манипулируя границами на характеристики кодов, мы попытаемся установить границы на сложность решеток. Этот подход к анализу асимптотической сложности решеток был впервые использован Зябловым и Сидоренко в [37]. Более детальный анализ как асимптотики сложности, так границ сложности для кодов конечной длины, представлен в [86].

В случае линейного кода число узлов на каждом ярусе является степенью двойки. Нам будет удобно измерять сложность логарифмом числа узлов, т.е. мы используем логарифмический профиль сложности: $\zeta_i = \log_2 \xi_i$, $i = 0, \ldots, n$. *Логарифмической сложностью решетки* называется максимальная величина

$$\zeta = \max_i \zeta_i.$$

Начнем с кодов конечной длины. Обозначим через $K(n,d)$ максимальную возможную размерность кода длины n с минимальным расстоянием d. Непосредственно из (4.25) для любого (n,k)-кода получаем

$$\zeta \geq \max_i k - K(i,d) - K(n-i,d). \qquad (4.26)$$

Пример 4.7. Рассмотрим код Голея (23,12) с расстоянием $d = 7$. Параметр разбиения i можно выбрать из множества {7,...,12}. Например, при $i = 7...12$ с помощью таблицы лучших линейных кодов находим оценки

$$\begin{aligned}
\zeta &\geq 12 - K(7,7) - K(16,7) = 12 - 1 - 5 = 6 \\
\zeta &\geq 12 - K(8,7) - K(15,7) = 12 - 1 - 5 = 6 \\
\zeta &\geq 12 - K(9,7) - K(14,7) = 12 - 1 - 4 = 7 \\
\zeta &\geq 12 - K(10,7) - K(13,7) = 12 - 1 - 3 = 8 \\
\zeta &\geq 12 - K(11,7) - K(12,7) = 12 - 2 - 2 = 8
\end{aligned}$$

Отсюда заключаем, что для любой решетки этого кода $\zeta \geq 8$. В данном случае эта оценка сложности точна, решетка с такой сложностью для кода Голея существует (см. [52]).

Из (4.26) можно вывести асимптотическую версию нижней границы сложности. Скорость и относительное расстояние (n,k)-кода с минимальным расстоянием d обозначим через $R(\delta) = k/n$ и $\delta = d/n$. Относительная сложность решетки $\kappa = \zeta/n$ может быть интерпретирована как показатель экспоненты сложности решетки или, что то же самое, показатель экспоненты сложности декодирования по МП с мягкими решениями.

Поделив обе части (4.26) и выбрав длину кода n достаточно большой, получим

$$\kappa \geq \max_{\gamma \in [0,1/2]} R(\delta) - \gamma R_{\max}\left(\frac{\delta}{\gamma}\right) - (1-\gamma) R_{\max}\left(\frac{\delta}{1-\gamma}\right), \qquad (4.27)$$

где $R_{\max}(\delta)$ обозначает максимально достижимую скорость кода с заданным относительным расстоянием. В качестве $R(\delta)$ можно выбрать границу Варшамова–Гилберта, $R(\delta) = 1 - h(\delta)$. Подставив в неравенство (4.26) известные верхние границы на функцию $R_{\max}(\delta)$,

получим асимптотические нижние оценки показателя экспоненты сложности решетки для кодов, удовлетворяющих границе ВГ.

Если мы примем верной гипотезу о том, что граница ВГ асимптотически точна для двоичных кодов, то можно в качестве $R_{\max}(\delta)$ тоже использовать границу ВГ. Используя свойства выпуклых функций, или приравняв производную по γ нулю, нетрудно убедиться в том, что максимум по γ в правой части достигается при $\gamma = 1/2$. Приходим к оценке для экспоненты сложности

$$\kappa(R) \geq \begin{cases} h(2\delta) - h(\delta), & R > 1 - h(0.25), \\ 1 - h(\delta), & R \leq 1 - h(0.25) \end{cases} \tag{4.28}$$

при $R = 1 - h(\delta)$.

Для того чтобы утверждать, что эта граница достижима, нужно убедиться в существовании таких кодов, параметры которых достигают границу ВГ, и при этом выполняются дополнительные условия: для любого i два кода, один из которых задан проверочной матрицей, составленной из первых i столбцов исходной проверочной матрицы, а другой задан последними $n - i$ ее столбцами, достигали границу ВГ. Хотя параметры случайно выбранного кода удовлетворяют границе ВГ с вероятностью близкой к единице, далеко не очевидно, что существуют хорошие коды, для которых все разбиения на две части порождают два хороших кода. Тем не менее, коды, достигающие (4.28), существуют (это вытекает, в частности, из результатов работы [41]).

Тем самым (4.28) можно рассматривать как точный ответ на вопрос о сложности решеток линейных кодов в той степени, в которой точна граница ВГ как граница на минимальное расстояние двоичных линейных кодов.

Удивительным представляется тот факт, что граница сложности решеток в точности совпала с границей сложности декодирования по методу соседей нулевого слова (2.19). Это косвенно свидетельствует о том, что почти для каждого узла минимальной решетки найдется сосед нулевого слова, соответствующий некоторому пути в решетке, проходящему через этот узел. Если эта интерпретация верна, то декодирование по решетке можно рассматривать как эффективную реализацию декодирования по соседям нуля для канала с мягкими решениями.

Более точные формулировки и анализ сложности решеток линейных кодов можно найти в [86, 112]. Нижние границы сложности для коротких кодов приведены в [52].

4.8. Практические алгоритмы субоптимального декодирования

Все рассмотренные выше методы декодирования имеют экспоненциальную по длине кода сложность. Для декодирования с мягкими решениями мы предложили пока только декодирование по решетке кода. Дополнительную информацию об асимптотической сложности оптимального или близкого к оптимальному декодирования кодов, параметры которых удовлетворяют границе ВГ, можно найти в [69].

Общий вывод состоит в том, что показатель экспоненты сложности оптимального декодирования может быть существенно меньше, чем для полного перебора по множеству кодовых слов, но все же накладывает очень жесткие ограничения на параметры кодов, для которых такое декодирование может быть реализовано на практике. Задача состоит в том, чтобы существенно упростить декодирование за счет некоторого (предполагается, что незначительного) проигрыша по вероятности ошибки.

В таком виде задача кажется плохо сформулированной: неясно, какой именно проигрыш по вероятности ошибки считать допустимым, и какое упрощение считать значительным. Более того, критерии сложности реализации для конечных длин кодов формализовать довольно сложно.

Неточность формулировки компенсируется актуальностью проблемы. Если вновь предлагаемый метод проигрывает оптимальному декодированию порядка 0.1–0.2 дБ, результаты считаются приемлемыми. Что касается сложности, если с использованием нового метода удается оценить характеристики кодов длины 100–200 при вероятности ошибки на блок порядка 10^{-6}, то алгоритм рассматривается как перспективный для практических целей.

Важная особенность рассматриваемых в этом параграфе алгоритмов — случайное число операций, зависящее от конкретной реа-

лизации шума в канале связи. Характеристиками сложности являются как среднее, так и максимальное число операций на декодированный бит переданной информации.

4.8.1. BEAST

Название алгоритма BEAST является аббревиатурой полного англоязычного названия *B*idirectional *E*fficient *A*lgorithm for *S*earching *T*rees, т.е. двусторонний эффективный алгоритм для анализа деревьев. Идея, как видно из названия, состоит в замене поиска по решетке двусторонним встречным поиском по дереву, причем при поиске игнорируются возможные слияния путей-кандидатов. Первоначально алгоритм рассматривался как способ вычисления характеристик сверточных кодов [49] (см. главу 8). Сложность BEAST применительно к декодированию в канале с мягкими решениями обсуждалась в [53], асимптотическая сложность — в [50].

Рассмотрим линейный (n, k)-код, обозначим через $\boldsymbol{y} = (y_1, ..., y_n)$ последовательность длины n на выходе канала при передаче одного из кодовых слов. Требуется найти то из кодовых слов, которое минимизирует расстояние $\mu(\boldsymbol{v}, \boldsymbol{y})$ по всему множеству кодовых слов $\boldsymbol{v} = (v_1, ..., v_n)$. Для описания алгоритма конкретный вид декодирующей функции $\mu(\boldsymbol{v}, \boldsymbol{y})$ не имеет значения, для нас важны только следующие свойства:

- Аддитивность: $\mu(\boldsymbol{v}, \boldsymbol{y}) = \sum_{i=1}^{n} \mu(v_i, y_i)$.
- Неотрицательность: $\mu(v, y) \geq 0$ для любых пар (v, y).
- Ограниченность: $\mu(v, y) \leq \mu_0$ для любых пар (v, y), где неотрицательная константа μ_0 зависит от параметров канала и не зависит от кода.

Заметим, что в случае ДСК и канала с АБГШ декодирование по МП может быть сведено к декодированию по минимуму метрики μ, удовлетворяющей этим требованиям. (Мы по привычке называем μ метрикой, хотя формально μ не обязательно удовлетворяет аксиомам метрики.)

Представим рассматриваемый код с помощью решетки. Задача, как и прежде, состоит в том, чтобы найти кратчайший путь от на-

чального узла решетки к конечному узлу. Предположим, что $\hat{T}$ — метрика искомого оптимального пути $\hat{\boldsymbol{v}} = (\hat{v}_1, ..., \hat{v}_n)$. Тогда найдется индекс t, для которого выполняются два условия

$$\mu(\hat{\boldsymbol{v}}_1^t, \boldsymbol{y}_1^t) < \hat{T}/2; \tag{4.29}$$

$$\mu(\hat{\boldsymbol{v}}_{t+1}^n, \boldsymbol{y}_{t+1}^n) \geq \hat{T}/2, \tag{4.30}$$

где использовано обозначение $\boldsymbol{x}_i^j = (x_i, ..., x_j)$. Значение t, удовлетворяющее условиям (4.29) и 4.30), будет определено однозначно, если мы потребуем, чтобы это было максимальное значение среди всех таких t.

Факт существования такого симметричного разбиения на «прошлое» и «будущее» подсказывает простую идею, лежащую в основе алгоритма BEAST: чтобы найти все пути с метрикой не больше T, алгоритм строит множество путей $\mathcal{F}$ («forward»), начинающихся в начале решетки и имеющих вес меньше $T/2$, и второе множество путей $\mathcal{B}$ («backward»), растущих в противоположном направлении, от конечного узла к начальному, и таких, что их метрика превышает $T/2$, но укороченные слева на одно ребро пути имеют метрику меньше или равную $T/2$. Пути каждого подмножества сортируются по номеру узла решетки, в котором закончился путь. После этого находим согласованные пути-кандидаты, такие, что начальная часть из $\mathcal{F}$ и продолжение из $\mathcal{B}$ образует вместе кодовое слово. Понятно, что если путь с метрикой меньше или равной T существует, он обязательно будет в числе выбранных кандидатов. Если таких путей несколько, выбираем путь с лучшей метрикой, если же таких нет, то нужно увеличить порог T и повторить процедуру.

Формальное описание BEAST представлено в виде алгоритма 4.3.

В описании алгоритма присутствует параметр Δ, определяющий шаг приращения порога метрики в случае, если при предыдущем значении порога решение не было получено. Величина Δ влияет на производительность декодера: при слишком маленьком значении параметра возрастает число попыток декодирования, а при слишком большом — может оказаться высокой сложность той попытки, на которой будет найдено решение. Однако в любом случае число итераций ограничено сверху линейной функцией длины кода.

Алгоритм 4.3. Алгоритм BEAST

Input: Решетка кода с заданными метриками ребер, вычисленными по последовательности $\boldsymbol{y}$ на выходе канала.

Output: Кодовое слово, соответствующее кратчайшему пути в решетке от яруса 0 до конечного яруса на глубине N

ζ^{P} и ζ^{C} -- соответственно предок и потомок узла ζ,
$W_{\mathrm{F}}(\zeta)$ -- метрика пути из начального узла в ζ,
$W_{\mathrm{B}}(\zeta)$ -- метрика пути из ζ в конечный узел.
Инициализация:
Flag $= 0$; $T = 0$;
while Flag$==0$ **do**
$T \leftarrow T + \Delta$
1. Построение прямого дерева

$$\mathcal{F} = \{\zeta | W_{\mathrm{F}}(\zeta) \geq T/2,\, W_{\mathrm{F}}(\zeta^{\mathrm{P}}) < T/2\}$$

2. Построение обратного дерева

$$\mathcal{B} = \{\zeta | W_{\mathrm{B}}(\zeta) < T/2\}$$

3. Поиск совпадений

- Сортировка путей в $\mathcal{F}$ и $\mathcal{B}$ по номерам узлов ζ, в которых заканчиваются пути.
- Если некоторый узел ζ присутствует в обоих списках, проверяем, равна ли сумма длины пути из $\mathcal{F}$ и длины пути из $\mathcal{B}$ длине кодового слова n. В случае равенства включаем путь в список кандидатов.
- Если множество кандидатов не пусто, из числа кандидатов выбираем кандидата с наименьшей метрикой. Flag $= 1$

end

Алгоритм 4.3 в случае неудачной попытки начинает строить множества $\mathcal{F}$ и $\mathcal{B}$ заново, тем самым повторяя все вычисления, выполненные в процессе предыдущей попытки декодирования. Несложно поправить алгоритм и устранить эти избыточные вычисления, хотя на порядок сложности это не повлияет.

Эксперименты с алгоритмом показали, что для конкретных кодов более существенное улучшение быстродействия может быть получено за счет выбора неодинаковых порогов для $\mathcal{F}$ и $\mathcal{B}$ и перевычисления не обоих, а только одного из двух деревьев (меньшего по размеру) на очередной итерации.

Заметим, что не двусторонний характер поиска, а сортировка является ключевым моментом, обеспечивающим выигрыш по сложности. Постараемся объяснить это следующими очень грубыми эвристическими оценками сложности.

Можно считать число путей (не обязательно одинаковой длины), расположенных на расстоянии не более T вокруг принятой последовательности, экспоненциальной функцией числа T, скажем,

$$\kappa = e^{\alpha T},$$

где α — неотрицательная величина, и величину κ примем в качестве оценки сложности декодирования при одностороннем поиске по дереву. Размеры прямого и обратного деревьев можно принять одинаковыми и равными

$$\kappa_F = \kappa_B = e^{\alpha T/2}.$$

Это и есть грубая оценка сложности BEAST-декодирования без учета сложности процедуры поиска согласованных путей.

Если бы эта процедура выполнялась попарным сравнением путей, то сложность была бы пропорциональна

$$\kappa_F \times \kappa_B = \left(e^{\alpha T/2}\right)^2 = e^{\alpha T},$$

т.е. примерно равна сложности декодирования односторонним поиском по дереву! В то же время сложность быстрой сортировки массива из N элементов оценивается как $N \log N$. Поэтому полная сложность BEAST-декодирования имеет порядок

$$\kappa_{\text{BEAST}} = \alpha T e^{\alpha T/2},$$

т.е. показатель экспоненты сложности вдвое меньше, чем при одностороннем поиске по дереву.

Напрашивается контраргумент: сравнивать надо не с поиском по дереву, а с поиском по решетке, ведь экспонента сложности решетки может быть существенно ниже экспоненты сложности декодирования перебором по кодовым словам. Удивительно, но при выполнении определенных условий учет слияния путей в решетке не уменьшает сложности декодирования.

Теорема 4.5. *Пусть кодовым словам соответствуют сигналы, представляющие собой точки n-мерного метрического (например, евклидова) пространства и* $d_{\min}$ *— минимальное расстояние между точками, соответствующими кодовым словам. Тогда при* $T < d_{\min}$ *ни в множестве* $\mathcal{F}$*, ни в* $\mathcal{B}$ *нет путей, проходящих через общий узел решетки кода.*

Доказательство. В силу линейности кода сумма двух путей, приходящих в один узел, — кодовое слово. Поэтому расстояние между такими путями не меньше $d_{\min}$. В силу неравенства треугольника сумма расстояний от путей до принятой последовательности, т.е. сумма метрик путей тоже не меньше $d_{\min}$, а значит одна из метрик больше $d_{\min}/2 > T/2$, что противоречит правилу построения множеств $\mathcal{F}$ и $\mathcal{B}$. Стало быть, предположение о существовании путей, приходящих в один узел, приводит к противоречию. □

Заметим, что для доказательства теоремы мы вынуждены были предположить, что метрика декодирования, помимо сформулированных выше требований, удовлетворяет неравенству треугольника. Это ограничение может оказаться существенным. Например, в канале с АБГШ декодирование по МП может быть реализовано как декодирование по минимуму квадрата евклидова расстояния. Для квадрата метрики Евклида неравенство треугольника не выполняется.

Рассмотрим декодирование в канале с АБГШ подробнее. Представим последовательность на входе декодера линейного (n, k)-кода в виде $\boldsymbol{y} = (y_1, ..., y_n) = (\alpha_1\beta_1, ..., \alpha_n\beta_n)$, где α_i и β_i — соответственно знаки входных символов декодера и надежности жестких решений,

т.е.

$$\alpha_i = \begin{cases} -1, & y_i < 0 \\ 1, & y_i \geq 0 \end{cases}, \quad \beta_i = |y_i|, \quad i = 1, ..., n.$$

По знакам жесткие двоичные решения восстанавливаются по формуле

$$r_i = \frac{1}{2}(1 - \alpha_i), \quad i = 1, ..., n.$$

Чтобы ввести подходящую метрику декодирования, зафиксируем неотрицательный вектор надежностей $\boldsymbol{\beta}$. Для произвольной двоичной последовательности $\boldsymbol{c} = (c_1, ..., c_n)$ обозначим через I_0 и I_1 индексы расположений, соответственно, нулей и единиц в $\boldsymbol{c}$. Определим вес $\boldsymbol{c}$ как

$$\mu(\boldsymbol{c}) = \sum_{i \in I_1} \beta_i \, .$$

Расстояние между $\boldsymbol{c}$ и $\boldsymbol{c}'$ определяется как $\mu(\boldsymbol{c} + \boldsymbol{c}')$, т.е. как вес суммы векторов по модулю 2.

Нетрудно видеть, что для $\mu(\boldsymbol{c})$ выполняются все свойства метрики. Назовем эту метрику обобщенной метрикой Хэмминга. В [109] эта метрика названа эллипсоидной метрикой.

Теорема 4.6. *Пусть $\boldsymbol{y}$ — выходная последовательность канала с АБГШ, $\boldsymbol{x}$ — вычисленная по ней последовательность жестких решений. Для (n, k)-кода $C = \{\boldsymbol{c}_i\}$, $i = 0, ..., 2^k - 1$ декодирование по максимуму правдоподобия эквивалентно декодированию по минимуму обобщенной метрики Хэмминга $\mu(\boldsymbol{c}_i + \boldsymbol{x})$.*

Доказательство. Двоичный вектор $\boldsymbol{c}_i + \boldsymbol{x}$ содержит единицы на позициях, где жесткие решения отличаются от символов кодового слова. Для некоторого кодового слов $\boldsymbol{c} = (c_1, ..., c_n)$ обозначим через I_0 множество тех позиций, где жесткие решения совпадают с кодовыми символами, т.е. $x_i = c_i$, и через I_1 дополнение к I_0. В канале с АБГШ декодирование по МП сводится к декодированию по максимуму скалярного произведения (см. (4.2)) $(\boldsymbol{y}, 2\boldsymbol{c} - 1)$, в которое надежности символов на позициях из I_0 входят со знаком «+», символы из I_1 — со знаком «−». Таким образом, решение о $\boldsymbol{c}$ по МП должно максимизировать

$$\sum_{j \in I_0} \beta_j - \sum_{j \in I_1} \beta_j = \sum_{j=1}^{n} \beta_j - 2 \sum_{j \in I_1} \beta_j = \sum_{j=1}^{n} \beta_j - 2\mu(\boldsymbol{c} + \boldsymbol{x}),$$

и, следовательно, минимизировать $\mu(\boldsymbol{c}+\boldsymbol{x})$, что и требовалось доказать. □

Хорошей характеристикой практической сложности алгоритма служит длина кодов, для которых может быть выполнено моделирование декодирования в разумное время. Разумеется, эта длина зависит от используемых вычислительных средств, качества программирования и т.п. Однако эта зависимость не так сильна, как может показаться. Поскольку сложность растет экспоненциально с длиной кода, ускорение программы или производительности компьютера в 2 раза позволит увеличить длину кодов всего на единицу...

Сложность алгоритма BEAST позволяет декодировать коды со скоростью $R = 1/2$ длины порядка $n = 100$.

4.8.2. Метод порядковых статистик

Применительно к декодированию в ДСК в главе 2 мы рассматривали методы декодирования, не связанные с решетками. Наиболее оптимистичными оказались оценки сложности для методов, основанных на переборе по информационным совокупностям кода в надежде найти информационную совокупность, не содержащую ошибок. Кажется естественным попытаться воспользоваться этим подходом для декодирования с мягкими решениями. Более того, можно ожидать дополнительного упрощения, поскольку надежности символов подсказывают, какие информационные совокупности наиболее надежны, и потому с большей вероятностью приведут к успешному декодированию.

Метод порядковых статистик [75] использует всего одну, самую надежную, информационную совокупность и для поиска решения выполняет перебор по возможным комбинациям ошибок на этой совокупности. Отметим, что введенное авторами название метода не отражает его сути, оно связано, скорее с методами анализа его эффективности.

В алгоритме 4.4 метрикой декодирования μ может служить, например, обобщенное расстояние Хэмминга, введенное в предыдущем параграфе, посвященном алгоритму BEAST.

Алгоритм 4.4. Алгоритм на основе порядковых статистик

Input: Знаки и надежности жестких решений, $\boldsymbol{\alpha}, \boldsymbol{\beta}$.
Output: Решение о кодовом слове $\hat{\boldsymbol{c}}$

`Инициализация:`
$\mu_0 \leftarrow \infty$, $\hat{\boldsymbol{c}} = \mathbf{0}$

1. Найти перестановку π такую, что $\pi(\boldsymbol{\beta})$ — неубывающая последовательность.

2. Применить перестановку π к столбцам порождающей матрицы кода и найти наименьшее число $h \geq k$ такое, что первые h столбцов переставленной порождающей матрицы образуют матрицу полного ранга k.

3. Среди h столбцов переставленной порождающей матрицы найти набор из k линейно-независимых столбцов и привести переставленную матрицу к диагональной форме на соответствующих k позициях. Обозначим новую матрицу $\tilde{G}$

4. **for** $w = 0$ **to** $w_{\max}$ **do**

   Для каждого из $\binom{k}{w}$ возможных расположений w позиций $i_1, ..., i_w$ на k позициях информационной совокупности

   - инвертировать w жестких решений α_j на позициях $j \in \{i_1, ..., i_w\}$
   - восстановить кодовое слово $\boldsymbol{c}$ умножением соответствующих информационных символов на $\tilde{G}$
   - найти метрику $\mu(\boldsymbol{c})$ кодового слова $\boldsymbol{c}$.
   - **if** $\mu(\boldsymbol{c}) < \mu_0$ **then** $\hat{\boldsymbol{c}} \leftarrow \boldsymbol{c}, \quad \mu_0 \leftarrow \mu(\boldsymbol{c})$

end

5. Восстановить порядок следования символов в кодовом слове

$$\hat{\boldsymbol{c}} \leftarrow \pi^{-1}(\hat{\boldsymbol{c}})$$

Параметр $w_{\max}$, ограничивающий максимальный вес корректируемых комбинаций ошибок на позициях информационной совокупности, определяет компромисс между вычислительной сложностью декодирования и эффективностью. В работе [75] на примерах показано, что для кода с минимальным расстоянием d при $w_{\max} = d/4$ достигается вероятность ошибки, близкая к вероятности ошибки декодирования по МП.

Дальнейшее развитие эта техника декодирования получила в работе [109]. Алгоритм, названный «Box and Match», использует не k, а некоторое большее количество $s > k$ самых надежных позиций, включающих информационную совокупность. Избыточные $s-k$ позиций применяются для сокращения перебора по информационным позициям. Общий эффект состоит в некотором снижении количества вычислений за счет использования дополнительной памяти, объем которой растет экспоненциально с длиной кода.

Вычислительная сложность алгоритма Box and Match позволила его авторам выполнить моделирование декодирования (почти по максимуму правдоподобия) кода (192,96).

Выводы

В этой главе мы изучали оптимальное или близкое к оптимальному декодирование линейных кодов. Кажется вполне естественным построить хороший код и затем применить к нему хороший метод декодирования. Однако на этом пути мы не можем построить реализуемых на практике методов кодирования и декодирования, близких по своим характеристикам к теоретическим пределам. Основное ограничение — сложность. Все рассмотренные нами методы декодирования имеют экспоненциальную по длине кода сложность декодирования. В следующих главах мы будем изучать специальные классы кодов, для которых существуют полиномиально-сложные декодеры и даже декодеры с линейной сложностью.

Сформулируем основные результаты данной главы.

1. Задача декодирования по максимуму правдоподобия сводится к задаче поиска кратчайшего пути на графе. Правило вычисления длин путей определяется моделью канала.

2. Любой код можно представить решетчатой диаграммой. Поиск кратчайшего пути может быть выполнен с помощью алгоритма Витерби.

3. Для каждого кода однозначно (с точностью до изоморфизма) определена минимальная решетка — решетка с наименьшим возможным числом узлов на каждом ярусе.

4. Минимальная решетка получается приведением порождающей матрицы кода к минимальной спэновой форме, либо строится как синдромная решетка по проверочной матрице кода.

5. Декодирование по максимуму апостериорной вероятности с мягкими решениями на выходе (SISO-декодирование) выполняется по решетке кода с помощью алгоритма БКДР.

6. Асимптотическая сложность декодирования по решетке, характеризуемая показателем экспоненты сложности, существенно меньше сложности декодирования перебором по кодовым словам.

7. Примеры альтернативных алгоритмов декодирования — алгоритмы BEAST, декодирование по порядковым статистикам, Box and Match. Эти алгоритмы позволяют декодировать коды длин порядка 100–200 с вероятностью ошибки, близкой к вероятности ошибки декодирования по максимуму правдоподобия.

Задачи

1. Как алгоритмы декодирования применить к задаче поиска минимального расстояния линейных кодов?

2. Напишите программу переборного декодирования по МП и МАВ для кодов, задаваемых матрицами из задачи 3 главы 2. С помощью моделирования постройте зависимость вероятности ошибки на бит от отношения сигнал/шум для ДСК и канала с АБГШ для двух методов декодирования, а также энерге-

тический выигрыш кодирования при вероятности ошибки на бит 10^{-5}.

3. Пусть

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

и при передаче по ДСК с переходной вероятностью $p_0 = 0.1$ принята последовательность (0, 1, 1, 1, 1). Вычислите апостериорные вероятности кодовых слов, кодовых символов и информационных символов. Определите решение МП-декодера и МАВ-декодера.

4. Для примеров матриц из задачи 3, интерпретируемых как порождающие или проверочные матрицы, постройте минимальные решетки приведением к МСФ и синдромные решетки.

5. Постройте решетку для кода из задачи 3 и решите задачу 3 с применением алгоритма БКДР.

6. Продумайте схему переборного декодирования, декодеров Витерби и БКДР. Оцените объем памяти, необходимой для хранения промежуточных данных, как функцию параметров кода и сложности решетчатого описания. Сопоставьте сложность декодирования перебором по кодовым словам, по алгоритму Витерби и по алгоритму БКДР.

7. Для одного из кодов длины n =20–32 определите его спектр. Воспользуйтесь программой моделирования (задача 2) и границами Полтырева и Шеннона (глава 3) для оценивания вероятности ошибки как функции отношения сигнал/шум в ДСК и в канале с АБГШ. Сопоставьте результаты.

.

5. Циклические коды

Циклические коды — подкласс линейных кодов. Эти коды обладают тем замечательным свойством, что все циклические сдвиги любого кодового слова принадлежат коду. В частности, порождающая матрица может быть составлена из сдвигов одного слова. Таким образом, для хранения полной информации о коде длины n достаточно n бит. Трудно ожидать, что коды такого узкого подкласса будут также хороши, как и линейные коды общего вида. Действительно, асимптотически эти коды плохи, но на удивление большая часть известных хороших кодов конечных длин — циклические коды. Более того, циклическими являются и коды Рида–Соломона, которые сегодня являются безусловными рекордсменами по числу реализованных в реальных устройствах кодеров и декодеров, поскольку они применяются во всех накопителях на жестких дисках, в устройствах записи информации на CD, DVD, флеш-памяти и т.п. Среди преимуществ циклических кодов, помимо простоты их описания, — возможность построения простых схем кодирования и декодирования. Об этом пойдет речь позже, после изучения специальных классов циклических кодов — БЧХ-кодов и кодов Рида–Соломона.

5.1. Порождающий и проверочный полиномы циклического кода

Начнем прямо с определения.

Определение 5.1. *Линейный (n, k)-код над полем F_q называется циклическим, если циклический сдвиг любого слова также является кодовым словом.*

Применительно к циклическим кодам удобно использовать представление двоичных последовательностей длины n в виде полиномов степени не выше $n-1$.

Пример 5.1. Последовательностям 10110, 10001, 00100 соответствуют полиномы $1+x^2+x^3$, $1+x^4$, x^2.

В этом примере рассмотрены полиномы с коэффициентами из поля F_2. Точно так же записываются полиномы с коэффициентами из произвольного конечного поля F_q. До определенного момента нас будут интересовать в основном двоичные коды.

По мере изложения мы все больше будем опираться на свойства групп, колец и полей. Самые необходимые определения даны в приложении к данной главе.

Пусть имеется полином

$$f(x) = f_0 + f_1 x + ... + f_{n-1} x^{n-1}$$

с коэффициентами из поля F_q, причем значения коэффициентов могут быть, в том числе, и нулевыми. Умножим этот полином на x. Получим новый полином

$$xf(x) = 0 + f_0 x + ... + f_{n-2} x^{n-1} + f_{n-1} x^n.$$

Видно, что последовательность коэффициентов полинома $xf(x)$ получается сдвигом вправо последовательности коэффициентов полинома $f(x)$. Если мы теперь приведем правую часть последнего соотношения по модулю $x^n - 1$, т.е. примем, что $x^n = 1$, то получим

$$xf(x) = f_{n-1} + f_0 x + ... + f_{n-2} x^{n-1} \mod (x^n - 1).$$

Таким образом, умножение $f(x)$ на x по модулю $x^n - 1$ дает циклический сдвиг $f(x)$.

Упражнение 5.1. Докажите, что множество вычетов всех полиномов по модулю $x^n - 1$ образует кольцо.

Мы называем полином *приведенным*, если коэффициент при его старшей степени равен единице.

С этого момента мы не различаем последовательностей и полиномов. Для нас теперь (n, k)-код $C = \{c(x)\}$ — это линейное подпространство размерности k в пространстве полиномов $c(x)$ степени не выше $n - 1$. Мы уже убедились в том, что представление кодовых слов в виде полиномов удобно хотя бы потому, что циклические сдвиги описываются умножением на мономы.

Теорема 5.1. *Пусть имеется циклический (n, k)-код. Если $g(x)$ – кодовое слово, то и $m(x)g(x) \mod x^n - 1$ тоже будет кодовым словом для любого $m(x)$.*

Упражнение 5.2. Докажите теорему 5.1.

Теорема 5.2. *Пусть имеется циклический (n, k)-код. Среди слов кода существует только один приведенный полином $g(x)$ наименьшей степени h.*

Доказательство. Если бы нашлось два различных полинома степени h, то их сумма (разность в недвоичном случае) оказалась бы полиномом степени меньше h, что невозможно по условию теоремы. □

Теорема 5.3. *Пусть имеется циклический (n, k)-код и $g(x)$ — кодовое слово наименьшей степени h. Тогда любое слово кода делится на $g(x)$.*

Доказательство. Поскольку степень любого слова не меньше степени $g(x)$, запишем произвольное слово $c(x)$ в виде

$$c(x) = g(x)q(x) + r(x), \tag{5.1}$$

где $q(x)$ и $r(x)$ — частное и остаток от деления $c(x)$ на $g(x)$. Из этого представления по теореме 5.1 следует, что $r(x)$ — кодовое слово. Но

степень $r(x)$ строго меньше степени $g(x)$, что невозможно по теореме 5.2. Следовательно, $r(x)$ должен быть равен нулю, а кодовое слово $c(x)$, согласно (5.1), кратно $g(x)$. □

Следствие 1. *Пусть имеется циклический (n,k)-код и $g(x)$ — кодовое слово наименьшей степени h. Тогда $g(x)$ является делителем x^n-1.*

Упражнение 5.3. Докажите следствие 1 из теоремы 5.3. Подсказка: запишите x^n-1 через частное и остаток от деления на $g(x)$ аналогично (5.1). Поскольку все выражение равно нулю по модулю x^n-1, остаток должен делиться на $g(x)$, либо он тоже должен быть равен нулю.

Теорема 5.4. *Пусть имеется циклический (n,k)-код и $g(x)$ — кодовое слово наименьшей степени h. Тогда $h=n-k$.*

Доказательство. Из теоремы 5.3 вытекает, что все слова кода кратны $g(x)$. Для доказательства надо подсчитать размерность множества полиномов, кратных $g(x)$. Полиномы $g(x)$, $xg(x)$,..., $x^{n-h-1}g(x)$ линейно независимы и являются базисом линейного пространства размерности $n-h$. В виде их линейной комбинации можно получить любое слово вида $m(x)g(x) \bmod x^n-1$, т.е. это множество линейных комбинаций порождает весь код, и его размерность равна размерности кода, т.е. $k=n-h$, что и требовалось доказать. □

Определение 5.2. *Полином $g(x)$, фигурирующий в теоремах 5.1—5.4, называется порождающим полиномом циклического кода.*

Пример 5.2. Пусть $n=7$ и $g(x)=1+x^2+x^3$. Этот порождающий полином, согласно теореме 5.4, порождает циклический код размерности 4. Выписывая базис $g(x)$, $xg(x)$,..., $x^{n-h-1}g(x)$ в виде двоичных последовательностей, получаем порождающую матрицу

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

Нетрудно подсчитать, что минимальное расстояние кода равно 3, т.е. мы получили циклическое представление кода Хэмминга (7,4).

Предыдущий пример подсказывает, что в общем случае порождающая матрица легко выписывается через коэффициенты порождающего полинома:

$$G = \begin{pmatrix} g_0 & g_1 & g_2 & \cdots & g_r & & & & \\ & g_0 & g_1 & \cdots & g_{r-1} & g_r & & & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & & \\ & & & & & g_0 & \cdots & g_{r-1} & g_r \end{pmatrix}, \tag{5.2}$$

где пустые позиции заполняются нулями.

Следуя логике рассмотрения линейных кодов, нам предстоит найти проверочную матрицу, точнее, ее представление с помощью полинома или полиномов.

Рассмотрим полином

$$h(x) = \frac{x^n - 1}{g(x)}. \tag{5.3}$$

Согласно Следствию 1, деление в правой части выполняется без остатка. Из этого равенства имеем

$$h(x)g(x) = 0 \mod x^n - 1. \tag{5.4}$$

Более того, для любого кодового слова $c(x) = m(x)g(x)$ имеем

$$c(x)h(x) = 0 \bmod x^n - 1. \tag{5.5}$$

Определение 5.3. *Полином $h(x)$, определенный соотношением (5.3), называется* проверочным полиномом *циклического кода длины n, заданного порождающим полиномом $g(x)$.*

Степень проверочного полинома равна размерности кода $k = n - r$.

Чтобы выписать проверочную матрицу кода, перепишем (5.4) через коэффициенты полиномов:

$$\begin{array}{llll} g_0h_0 & & & = 0 \\ g_0h_1+g_1h_0 & & & = 0 \\ g_0h_2+g_1h_1+g_2h_0 & & & = 0 \\ \dots & \dots & \dots & \dots \\ & & g_rh_k & = 0 \end{array}$$

Дополнительные уравнения можно получить из тождеств

$$h(x)(x^j g(x)) = 0 \mod x^n - 1, \quad j = 1, ..., k-1.$$

Нетрудно видеть, что всем уравнениям удовлетворяет проверочная матрица

$$H = \begin{pmatrix} h_k & h_{k-1} & h_{k-2} & \cdots & h_0 & & & & \\ & h_k & h_{k-1} & \cdots & h_1 & h_0 & & & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & & \\ & & & & & h_k & \cdots & h_1 & h_0 \end{pmatrix}. \qquad (5.6)$$

Заметим, что в строках матрицы H коэффициенты проверочного полинома следуют в инверсном порядке. Полином, инверсный полиному $h(x)$, можно записать в форме $x^k h(x^{-1})$. Итак, доказано следующее утверждение.

Теорема 5.5. *Дуальный код циклического (n,k)-кода с порождающим многочленом $g(x)$ и проверочным многочленом $h(x)$ является циклическим $(n, n-k)$ кодом с порождающим многочленом*

$$g^{\perp}(x) = x^k h(x^{-1}).$$

Из теоремы следует, в частности, что проверочный полином может рассматриваться как порождающий многочлен кода, эквивалентного дуальному коду. Его кодовые слова могут быть получены инверсией (переписыванием в обратном порядке) слов дуального кода.

Упражнение 5.4. Найдите порождающий многочлен кода дуального коду Хэмминга (7,4) (см. пример 5.2).

Упражнение 5.5. Что можно сказать о сложности декодирования циклического кода с помощью решетки? (См. задачу 8 в конце главы.)

5.2. Примеры циклических кодов

Построить циклический код, как мы знаем из предыдущего параграфа, несложно. Для этого нужно разложить на множители многочлен $x^n - 1$ и выбрать любой (не обязательно простой) делитель $x^n - 1$ в качестве порождающего многочлена циклического кода длины n. Чем больше степень делителя, тем меньше скорость, и, возможно, больше минимальное расстояние кода.

Пример 5.3. Положим $n = 7$. Имеем

$$x^7 + 1 = (1 + x)(1 + x + x^3)(1 + x^2 + x^3).$$

Отсюда следует, что множество циклических кодов длины 7 включает в себя

А. (7,6)-код с $g(x) = 1 + x$,

Б. (7,4)-коды с $g(x) = 1 + x + x^3$ и $g(x) = 1 + x^2 + x^3$,

В. (7,1)-код с $g(x) = (1 + x + x^3)(1 + x^2 + x^3) = 1 + x + x^2 + x^3 + x^4 + x^5 + x^6$.

Г. (7,3)-коды с $g(x) = (1 + x)(1 + x^2 + x^3) = 1 + x + x^2 + x^4$ и $g(x) = (1 + x)(1 + x + x^3) = 1 + x^2 + x^3 + x^4$.

Коды А и В — пара дуальных кодов: код с проверкой на четность и код с повторениями. Два кода Б — коды Хэмминга, отличающиеся порядком следования символов: один из них получается переписыванием кодовых символов другого кода в обратном порядке. Коды Г — два кода Хэмминга с дополнительной проверкой на четность, т.е. симплексные коды, дуальные по отношению к кодам Хэмминга.

В этом примере рассматриваются коды над полем характеристики 2, поэтому знаки «+» и «−» не различаются.

Поскольку $x^n - 1$ имеет корнем 1, то многочлен $x + 1$ всегда его делит. Поэтому при любом n можно выбрать $g(x) = x + 1$ в качестве порождающего полинома. При любом $m(x)$ произведение $c(x) = m(x)(x + 1)$ имеет четное число слагаемых, т.е. получаемый код — это $(n, n - 1)$-код с проверкой на четность.

Точно так же двойственный к нему код — код с повторениями — тоже циклический код при всех n.

Пример 5.4. (Двоичные коды Хэмминга.) Пусть $n = 2^m - 1$. В качестве порождающего многочлена $g(x)$ выберем любой примитивный многочлен $p(x)$ поля $GF(2^m)$ (см. приложение). Тогда число избыточных символов будет равно m, размерность кода $k = 2^m - m - 1$. Осталось убедиться в том, что минимальное расстояние кода равно 3. Предположим, что среди кодовых слов нашлось слово веса 2, кратное $g(x) = p(x)$. Слово веса 2 имеет вид $c(x) = x^i + x^j = x^i(x^{j-i} + 1)$, $j > i$. В этом случае $p(x)$ оказывается делителем многочлена вида $x^h - 1$ при $h < m$, т.е. порядок корней многочлена меньше m, что невозможно в силу предположения о примитивности $p(x)$. Следовательно, в коде нет слов веса меньше 3, и мы имеем $(n = 2^m - 1, k = 2^m - m - 1)$-код Хэмминга.

Проверочная матрица кода может быть выбрана в виде

$$H = \begin{pmatrix} 1 & \alpha & \ldots & \alpha^{2^m-3} & \alpha^{2^m-2} \end{pmatrix},$$

где α — примитивный элемент поля. Подразумевается, что вместо элементов поля выписываются в виде столбцов их двоичные представления. Двоичная последовательность при умножении на H даст ноль только в том случае, если α является корнем соответствующего многочлена, т.е. только если этот многочлен делится на примитивный многочлен.

Пример 5.5. (Двоичные коды максимальной длины.) Более точно назвать эти коды кодами из последовательностей максимальной длины. Их применение много шире рамок теории кодирования. Они используются для установления синхронизации, в криптографии, для генерации псевдослучайных последовательностей и т.д.

Циклический код длины $n = 2^m - 1$, проверочным многочленом $h(x)$ которого является примитивный многочлен $p(x)$ поля $GF(2^m)$, называется *кодом максимальной длины*.

По определению, код максимальной длины дуален коду Хэмминга. Порождающий многочлен кода равен $g(x) = (x^{2^m-1} - 1)/h(x)$. Кодирование, как и для любого циклического кода, можно выполнить умножением информационного полинома $m(x)$ на $g(x)$.

На рис. 5.1 показана одна из наиболее распространенных блок-схем генератора последовательности максимальной длины. Эта же схема может служить кодером для кода максимальной длины. В момент начала работы кодера в ячейки его регистра сдвига записывается начальная последовательность (информационные символы). На каждом из n тактов работы схемы на выход поступает некоторый кодовый символ, содержимое регистра сдвигается вправо (умножается на x). Как только на выходе появляется ненулевое значение, оно умножается на $p(x)$ и вычитается из содержимого регистра, тем самым производится приведение по модулю $p(x)$. Выходом схемы является последовательность максимальной длины. Поясним работу схемы примером.

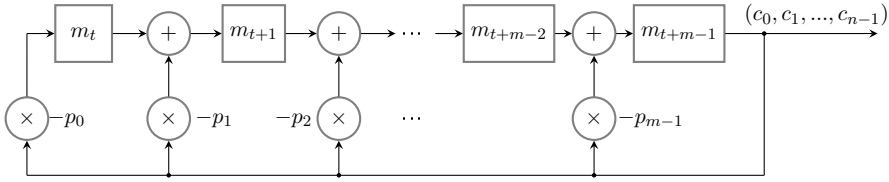


Рис. 5.1. Генератор последовательностей максимальной длины или кодер кода максимальной длины

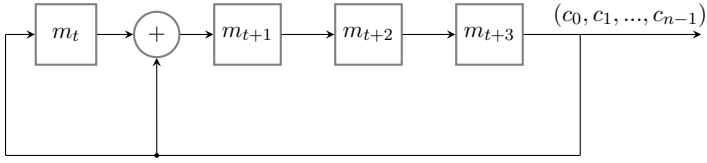


Рис. 5.2. Генератор последовательностей максимальной длины при $p(x) = 1 + x + x^4$ или кодер кода (15,4)

На рис. 5.2 показан пример кодера для случая, когда $h(x) = p(x) = 1 + x + x^4$ — примитивный многочлен поля $GF(2^4)$. Поло-

жим состояние регистра равным $m(x) = m_0 + m_1 x + m_2 x^2 + m_3 x^3 = x^2 + x^3$, что соответствует кодированию информационной последовательности $(0, 0, 1, 1)$. В табл. 5.1 показаны вычисления, выполняемые кодером, такт за тактом.

Таблица 5.1. Работа генератора последовательности максимальной длины

Такт	Состояние регистра	Выходной символ
0	0011	1
1	1101	1
2	1010	0
3	0101	1
4	1110	0
5	0111	1
6	1111	1
7	1011	1
8	1001	1
9	1000	0
10	0100	0
11	0010	0
12	0001	1
13	1100	0
14	0110	0

Нетрудно видеть, что, если мы обозначим через $m_t(x)$ полином, описывающий состояние регистра на такте t, то последовательность состояний окажется циклическим сдвигом последовательности полиномов 1, x, $x^2, \ldots, x^{2^m-2}$ по модулю $p(x)$. В табл. 5.1 этой последовательности соответствуют строки с номерами 9, 10, ..., 8. Иными словами, множество состояний регистра пробегает все возможные ненулевые состояния.

Для произвольного постоянного во времени конечного автомата с памятью m бит выход автомата определяется детерминированной функцией содержимого его ячеек памяти. Максимальное число различных состояний равно 2^m, и эта же величина определяет мак-

симальную длину псевдослучайной последовательности, генерируемой автоматом. В случае линейного автомата нулевому начальному состоянию соответствует нулевая выходная последовательность, поэтому максимальная длина оказывается равной $2^m - 1$, отсюда и название последовательности — последовательность максимальной длины.

Поскольку кодовые слова отличаются только начальными состояниями регистра, они являются сдвигами друг друга. Нулевому начальному состоянию соответствует нулевое кодовое слово. Минимальное расстояние (минимальный вес кодового слова) равно числу единиц в последовательности максимальной длины. На рис. 5.1 и 5.2 видно, что единиц в кодовом слове будет столько, сколько ненулевых элементов поля заканчивается на 1, т.е. минимальное расстояние равно 2^{m-1}.

Докажем, что все сдвиги порождающего многочлена $g(x) = (x^{2^m} - 1)/h(x)$ различны. Предположим, что два сдвига совпадают, т.е. для некоторых i и j имеет место равенство

$$x^i g(x) = x^j g(x) \mod x^n - 1.$$

Это означает, что для некоторого $r(x)$

$$x^i g(x) = x^j g(x) + r(x)(x^n - 1).$$

Поделив обе части на $g(x)$, получаем

$$x^i - x^j = r(x)p(x),$$

откуда следует, что $p(x)$ является делителем двучлена меньшей степени, чем $n = 2^m - 1$, что невозможно, поскольку $p(x)$ — примитивный многочлен. Тем самым мы убедились в том, что все $2^m - 1$ сдвигов последовательностей максимальной длины различны, и они вместе с нулевым словом образуют $(2^m - 1, m)$-код с расстоянием 2^{m-1}.

Пример 5.6. (Код Голея.) Положим $n = 23$. Можно, перемножив многочлены, проверить, что

$$x^{23} - 1 = (x - 1)g(x)\tilde{g}(x),$$

где

$$g(x) = x^{11} + x^{10} + x^6 + x^5 + x^4 + x^2 + 1,$$

$$\tilde{g}(x) = x^{11} + x^9 + x^7 + x^6 + x^5 + x + 1.$$

При этом многочлены $g(x)$ и $\tilde{g}(x)$ — взаимные друг к другу, т.е. коэффициенты одного многочлена получаются переписыванием в обратном порядке коэффициентов другого многочлена, $\tilde{g}(x) = x^{11}g(x^{-1})$. Получается, что существует два нетривиальных циклических кода длины 23: код (23,12), порождаемый $g(x)$, и дуальный к нему код (23,11), порождаемый многочленом $(x-1)\tilde{g}(x)$.

Докажем, что расстояния кода Голея и дуального к нему кода равны соответственно 7 и 8.

Код, порождаемый $g(x)$, — частный случай БЧХ-кодов, которые будут рассматриваться в следующем параграфе. Из теоремы БЧХ мы получим оценку $d \geq 5$. Покажем, что в коде Голея нет слов веса 5 и 6. Для этого сначала рассмотрим так называемый расширенный код Голея (24,12), получаемый дописыванием проверки на четность к словам кода (23,12). При этом слова четного веса, конечно, не изменяются. Порождающая матрица расширенного кода Голея состоит из 12 сдвигов $g(x)$ (порождающая матрица исходного кода) и столбца из всех единиц. Все строки в результате имеют вес 8.

Дальнейшие рассуждения базируются на трех леммах.

Лемма 5.6. *Расширенный код Голея самодуален.*

Доказательство. Этот факт можно проверить непосредственно, помножив порождающую матрицу на транспонированную порождающую матрицу. Другое доказательство основано на том, что проверочный многочлен кода Голея равен $h(x) = (x+1)\tilde{g}(x)$. Следовательно, порождающий многочлен дуального (23,11)-кода равен $g^{\perp}(x) = x^{12}h(x^{-1}) = (1+x)g(x)$. Отсюда следует, что код (23,11) — подкод (23,12), составленный из слов четного веса. Если дописать к порождающей матрице кода (23,11) строку из всех единиц, то получим альтернативную порождающую матрицу для (23,12), которая ортогональна себе, если не принимать во внимание последнюю строку. После дописывания столбца общей проверки на четность

(столбец из всех нулей и одной единицы на нижней позиции) все строки, включая последнюю, станут ортогональными коду. □

Лемма 5.7. *В расширенном коде Голея вес кодовых слов кратен 4.*

Доказательство. Пусть $\boldsymbol{a}$ и $\boldsymbol{b}$ — ненулевые слова. Тогда скалярное произведение $(\boldsymbol{a}, \boldsymbol{b})$ равно нулю, если эти слова имеют четное число совпадающих позиций. Отсюда следует, что $w(\boldsymbol{a}+\boldsymbol{b}) = = w(\boldsymbol{a}) + w(\boldsymbol{b}) - 4s$, где s — целое. Поскольку в порождающей матрице расширенного кода Голея все строки имеют вес 8, и код самодуален, вес всех линейных комбинаций кратен 4. □

Лемма 5.8. *В коде Голея нет слов веса 5 и 6.*

Доказательство. Наличие таких слов в коде Голея привело бы к существованию слов веса 6 в расширенном коде Голея, что противоречит лемме 5.7. □

Из доказанных утверждений следует, что расширенный код Голея (24,12) имеет расстояние 8, откуда вытекает, что код Голея (23,12) имеет расстояние 7.

Упражнение 5.6. Убедитесь, что код Голея (23,12) — *совершенный код*, т.е. удовлетворяет границе Хэмминга.

5.3. Кодирование и вычисление синдрома

Казалось бы, при современном уровне техники нет смысла останавливаться на деталях реализации устройств — подобные задачи сводятся к программированию в той или иной среде. Однако возможность чрезвычайно эффективной реализации элементов кодеров и декодеров на регистрах сдвига — примечательная особенность циклических кодов, немало способствовавшая их широкому распространению.

Начнем с задачи кодирования. Формула, связывающая кодовое слово $c(x)$ с соответствующим информационным многочленом $m(x)$ и порождающим многочленом $g(x)$, имеет вид

$$c(x) = m(x)g(x). \tag{5.7}$$

Степени при x можно рассматривать как индексы времени, умножение на x соответствует сдвигу на один такт. Поэтому операция (5.7) аналогична операции свертки двух последовательностей или операции фильтрации входной последовательности $m(x)$ фильтром с конечным откликом (или с конечной импульсной характеристикой (КИХ-фильтром)) $g(x)$.

Пример 5.7. Рассмотрим код Хэмминга (15,11). Выберем в качестве порождающего многочлена $g(x) = 1 + x + x^4$. На рис. 5.3 представлен фильтр-умножитель, который можно использовать в качестве кодера.

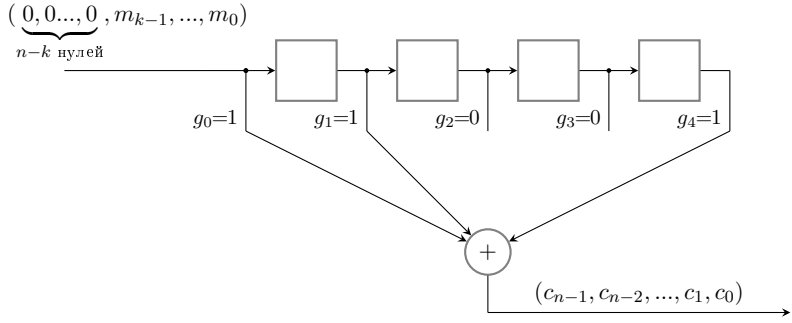


Рис. 5.3. Несистематический кодер циклического кода

До начала работы кодера в регистр записаны нули. Затем один за другим поступают информационные символы и складываются, умножаясь на порождающий многочлен. Выходами кодера будут последовательно $c_0 = m_0 g_0$, $c_1 = m_0 g_1 + m_1 g_0$, и т.д., что в точности соответствует (5.7). После подачи k информационных символов поступают один за другим $n-k$ нулей, и последним кодовым символом будет, как и должно быть, $c_{n-1} = m_k g_k$.

Аналогично, вычисление синдрома можно выполнить в виде умножения на проверочный полином кода $h(x)$ принятой последовательности $b(x) = c(x) + e(x)$, где $e(x)$ — полином, соответствующий вектору ошибок.

Итак, на базе КИХ-фильтров, похожих на фильтр, приведенный на рис. 5.3, можно строить как кодеры, так и вычислители синдромов. Однако это решение не самое удачное. Во-первых, желательно использовать систематические коды, т.к. при этом уменьшается сложность восстановления информации в декодере. Во-вторых, по крайней мере, если $k > n-k$, желательно уменьшить сложность вычисления синдрома, сделать ее пропорциональной $n-k$, а не k. Обе задачи, как мы увидим, успешно решаются применением фильтров с обратной связью, или, в терминах цифровой обработки сигналов, БИХ-фильтров (фильтров с бесконечной импульсной характеристикой).

Посмотрим, как можно сделать кодирование систематическим. Для этого запишем кодовое слово в виде

$$c(x) = x^{n-k}m(x) + t(x). \tag{5.8}$$

Эта запись означает, что в старших разрядах слова записана информационная последовательность, а полином $t(x)$ должен иметь степень не выше $n-k-1$ и быть таким, чтобы $c(x)$ делился на порождающий многочлен $g(x)$. Этого можно добиться, выбрав в качестве $t(x)$ остаток от деления $-x^{n-k}m(x)$ на $g(x)$, т.е. найти такие $t(x)$ и $q(x)$, что

$$-x^{n-k}m(x) = q(x)g(x) + t(x).$$

В этом случае (см. (5.8)) $c(x) = -q(x)g(x)$, т.е. является кодовым словом. Для построения кодового слова само частное $q(x)$ не требуется, достаточно к информационной последовательности дописать остаток $t(x)$ от деления $-x^{n-k}m(x)$ на $g(x)$.

Вычисление остатка от деления с помощью регистра очень похоже на школьное деление чисел «столбиком». Начиная со старших разрядов, мы вычитаем делитель, умноженный на такой элемент поля, который обеспечивает равенство нулю старшего разряда делимого.

Пример 5.8. Схема кодера для кода Хэмминга (15,11) с порождающим многочленом $g(x) = 1 + x + x^4$ показана на рис. 5.4. Кроме сумматоров и элементов задержки схема содержит два переключателя, на положениях которых надписаны номера соответствующих

тактов. Начальное положение регистра — нулевое. На первых одиннадцати тактах информационные символы поступают одновременно в обратную связь регистра и на выход схемы. За 11 шагов в регистре формируется остаток от деления информационной последовательности на порождающий многочлен. Этот остаток на последних 4 тактах подается на выход как проверочные символы к данным информационным символам. Кодирование информационной последовательности $m = (m_0, \ldots, m_{10}) = (0,1,0,0,0,0,1,1,0,1,1)$ систематическим кодером, показанным на рис. 5.4, шаг за шагом поясняется в табл. 5.2. Обратите внимание, что сначала поступают старшие разряды информационной последовательности. Казалось бы, отсутствует умножение на $x^{n-k} = x^4$. Оно присутствует неявно: за информационными символами как бы следуют нули, и деление осуществляется, как и должно быть, в течение 11 тактов.

Результат вычислений в точности совпадает с результатом деления в столбик (единицы соответствуют коэффициентам при соответствующих степенях x):

$$
\begin{array}{l|l}
\mathbf{11011000010}0000 & \mathbf{10011} \\
\underline{10011} & \underline{1100\ldots} \\
10000 & \\
\underline{10011} & \\
11000 & \\
\underline{10011} & \\
1011 & \\
\underline{10011} & \\
10000 & \\
\underline{10011} & \\
11000 & \\
\underline{10011} & \\
\mathbf{1011} &
\end{array}
$$

Рассмотрим теперь задачу вычисления синдрома. Если на выходе канала наблюдается последовательность $b(x) = c(x) + e(x)$, то синдромный полином $s(x)$ вычисляется либо умножением на $h(x)$ (по аналогии с несистематическим кодированием, см. рис. 5.3), либо делением на порождающий многочлен $g(x)$:

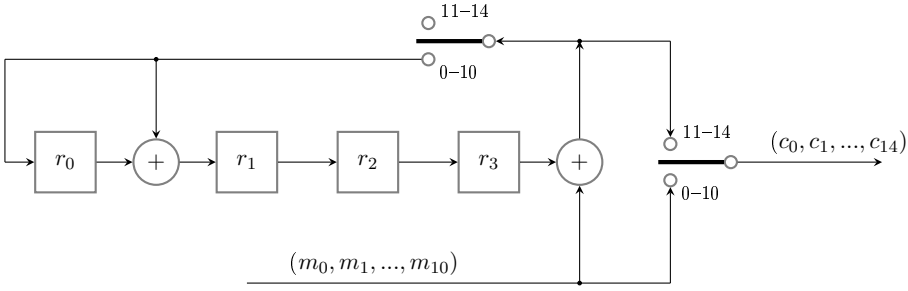


Рис. 5.4. Систематический кодер кода $(15, 11)$

Таблица 5.2. Кодирование систематическим кодером кода Хэмминга (15,11)

Такт	Вход	Обратная связь	Состояние регистра	Выходной символ
—	—	—	0000	—
0	1	1	1100	1
1	1	1	1010	1
2	0	0	0101	0
3	1	0	0010	1
4	1	1	1101	1
5	0	1	1010	0
6	0	0	0101	0
7	0	1	1110	0
8	0	0	0111	0
9	1	0	0011	1
10	0	1	1101	0
11	—	—	0110	1
12	—	—	0011	0
13	—	—	0001	1
14	—	—	0000	1

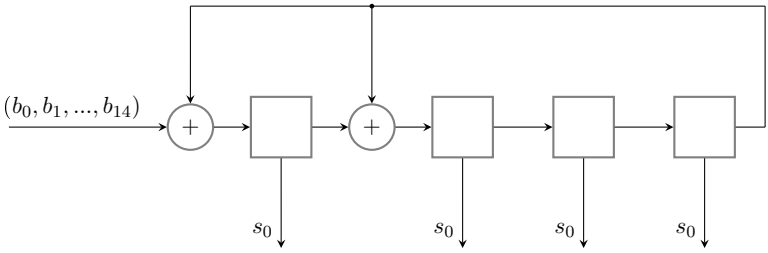


Рис. 5.5. Вычислитель синдрома для кода (15,11)

$$s(x) = b(x) \bmod g(x).$$

Пример 5.9. Схема вычисления синдрома для кода (15,11) показана на рис. 5.5. Предположим, что последовательность на выходе канала $(b_0, b_1, \ldots, b_{14}) = (010101101101100)$, что соответствует полиному $b(x) = x^{13}+x^{11}+x^9+x^8+x^6+x^5+x^3+x^2$. Предполагается, что начальное состояние регистра — нулевое, и последовательность поступает на вход, начиная со старших коэффициентов. Вычисления, выполняемые схемой на каждом такте работы, показаны в табл. 5.3. Честное деление в столбик показывает, что синдром вычислен правильно.

Для исправления ошибок нужно не только подсчитать синдром, но и найти соответствующий этому синдрому вектор ошибок. Для этого можно использовать запоминающее устройство, в котором для каждого из $2^{n-k} - 1$ ненулевых синдромов хранится соответствующий ему вектор ошибок длины n. Поскольку код циклический, можно немного сократить объем необходимой памяти.

Теорема 5.9. *Если*

$$s(x) = b(x) \mod g(x),$$

то

$$(xb(x)(\mathrm{mod}(x^n - 1)))\,(\mathrm{mod} g(x)) = xs(x)(\mathrm{mod} g(x)).$$

Доказательство. Обозначим через $q_b(x)$ частное от деления $b(x)$ на $g(x)$. Тогда

$$b(x) = g(x)q_b(x) + s(x)$$

Таблица 5.3. Вычисление синдрома кода Хэмминга (15,11)

Такт	Вход	Обратная связь	Состояние регистра
—	—	—	0000
0	0	0	0000
1	0	0	0000
2	1	0	1000
3	1	0	1100
4	0	0	0110
5	1	0	1011
6	1	1	0001
7	0	1	1100
8	1	0	1110
9	1	0	1111
10	0	1	1011
11	1	1	0001
12	0	1	1100
13	1	0	1110
14	0	0	0111

и

$$xb(x) = xg(x)q_b(x) + xs(x).$$

Поскольку $xb(x)(\bmod (x^n-1)) = xb(x)-b_{n-1}(x^n-1)$ и $g(x)$ делит x^n-1, из последнего тождества после приведения по модулю $g(x)$ получаем утверждение теоремы. □

Иными словами, теорема утверждает, что циклическому сдвигу последовательности длины n соответствует циклический сдвиг синдрома, выполняемый с приведением по модулю порождающего многочлена.

При декодировании циклического кода принятую последовательность можно записать в циклический регистр сдвига из n ячеек. Для каждого циклического сдвига заново вычисляется синдром и проверяется наличие вычисленного вектора в памяти декодера. Если он там есть, соответствующий вектор ошибок прибавляется к

принятому слову, и на этом декодирование заканчивается. Из Теоремы 5.9 следует, что достаточно хранить только такие синдромы, которые не являются циклическими сдвигами других синдромов, уже помещенных в память декодера. Кажется, что таким способом можно сократить объем памяти примерно в $r = n - k$ раз, но на самом деле многие последовательности длины r имеют много меньше, чем r различных сдвигов (например, последовательность из всех единиц). По этой причине память уменьшается не в r раз, а примерно в 4 раза. Декодер, использующий эту возможность, называют *декодером Меггитта*. Платой за экономию памяти является значительное увеличение времени работы декодера.

Задачи

1. Постройте циклические коды длин $n = 3, ..., 9$. Определите скорость и минимальное расстояние кодов и дуальных к ним кодов.

2. Постройте все поля характеристики 2 из 8 элементов. Проверьте, что они изоморфны друг другу.

3. Определите порядки элементов мультипликативной группы поля $GF(q)$ при $q = 2, 3, 4, 5, 7, 8, 9$.

4. Постройте поле $GF(2^4)$ как кольцо вычетов по модулю $p(x) = = 1 + x + x^4$. Найдите обратные элементы к элементам поля.

5. Постройте поле $GF(2^4)$ как кольцо вычетов по модулю $p(x) = = 1 + x + x^2 + x^3 + x^4$. Найдите обратные элементы к элементам поля.

6. Представьте порождающую матрицу циклического кода Хэмминга (7,4) в систематической форме и выпишите соответствующую проверочную матрицу. Как будет выглядеть декодер Меггитта для этого кода?

7. Предположим, что кодом Хэмминга (15,11) передается сообщение (10111101001) и в канале ошибка произошла в позиции с номером 7. Повторите вычисления, выполняемые кодером

при систематическом кодировании и декодером при вычислении синдрома. Используя результаты выполнения задачи 4, покажите, как произойдет исправление этой ошибки декодером Меггитта.

8. Докажите, что минимальная решетка циклического кода имеет максимальную по всем уровням сложность $\min\{2^k, 2^{n-k}\}$. Эта величина совпадает с максимальной возможной сложностью минимальной решетки линейного кода. Означает ли это, что сложность решетки кода Голея не может быть меньше 2^{12}?

Приложение. Конечные поля

Кольцо вычетов

Определение 5.4. *Кольцом называется множество R, на котором определены две операции, называемые сложением и умножением, и для которого справедливы аксиомы:*

1. *R образует абелеву группу по сложению.*
2. *R замкнуто относительно операции умножения.*
3. *Ассоциативность: $a(bc) = (ab)c$.*
4. *Дистрибутивность: $a(b + c) = ab + ac$.*

Обратим внимание на то, что в кольце может не быть единичного элемента по умножению. Если же он есть, кольцо называется *кольцом с единицей*.

В общем случае не требуется, чтобы операция умножения была коммутативной. Если же это условие выполняется, то имеем *коммутативное кольцо*.

Отметим также, что в кольце с единицей не обязательно определены обратные элементы для каждого элемента кольца. Если бы они были, то можно было бы говорить о том, что ненулевые элементы образуют группу по умножению, и тогда это уже было бы поле.

Определение 5.5. *Полем называется множество F, на котором определены две операции, называемые сложением и умножением, и для которого справедливы аксиомы:*

1. *F образует абелеву группу по сложению.*
2. *Множество ненулевых элементов F образует абелеву группу по умножению.*

Пример 5.10. При любом q множество целых чисел по модулю числа q является кольцом, при условии, что операции сложения и умножения выполняются с последующим приведением по модулю q. Это кольцо является коммутативным кольцом с единицей. Оно является полем, если число q — простое.

Упражнение 5.7. Пусть R_q обозначает кольцо вычетов целых чисел по модулю q. Для $q = 2, \ldots, 7$ определите, какие из элементов полей и колец имеют обратные элементы. Сформулируйте общее утверждение о существовании обратных элементов в таких кольцах.

Поле из q элементов в дальнейшем будем обозначать $GF(q)$. GF — аббревиатура словосочетания Galois Field (поле Галуа), связанного с именем французского ученого с очень необычной биографией. Рекомендуем почитать о нем хотя бы в Интернете.

Кольцо многочленов

Рассмотренные примеры подсказывают, что при простых q существуют поля с q элементами — это поля целых чисел по модулю q. Можно построить поля мощности q при составном q. Чтобы построить такие поля, рассмотрим специальное кольцо — кольцо многочленов.

Элементами кольца являются многочлены вида $a(x) = a_0 + a_1 x + \ldots + a_m x^m$, где коэффициенты выбираются из некоторого поля $GF(q)$. Операции сложения и умножения многочленов выполняются обычным образом. Такое кольцо обозначим $F[x]$.

Напомним, что *приведенным* называется многочлен, у которого коэффициент при старшей степени аргумента равен 1.

Определение 5.6. *Для приведенного многочлена $p(x)$ степени больше единицы кольцом многочленов по модулю $p(x)$ называется множество всех многочленов степени меньше степени $p(x)$ с операциями умножения и сложения, определенными как сложение и умножение по модулю $p(x)$. Это кольцо обозначим $F_{p(x)}[x]$.*

Теорема 5.10. *Кольцо $F_{p(x)}[x]$ является полем тогда и только тогда, когда $p(x)$ — простой (т.е. приведенный и неразложимый) многочлен.*

Доказательство. Начнем с доказательства необходимости. Покажем, что при простом $p(x)$ каждый элемент кольца $F_{p(x)}[x]$ имеет обратный элемент по умножению. Для этого выберем произвольный элемент $s(x)$ и покажем, что все произведения вида $s(x)a(x) \bmod p(x)$ различны. Это будет означать, что одно из произведений равно 1. Соответствующий элемент $a(x)$ будет обратным к $s(x)$. Предположим противное, т.е. что для некоторых $a(x)$ и $b(x)$ имеет место $s(x)a(x) = s(x)b(x) \bmod p(x)$, или, что то же самое, $s(x)d(x) = 0 \bmod p(x)$ имеет место для некоторого $d(x)$. Это означает, что $s(x)d(x) = t(x)p(x)$ при некотором $t(x)$. Один из многочленов в левой части обязан делиться на $p(x)$, поскольку $p(x)$ — простой. Это невозможно, поскольку степень каждого из этих двух многочленов меньше степени $p(x)$. Тем самым доказано существование обратного элемента, и данное кольцо — поле.

Чтобы доказать необходимость, заметим, что при непростом $p(x) = u(x)v(x)$ делители $p(x)$ не имеют обратных, поскольку иначе можно было бы записать

$$u(x) = [u(x)v(x)]\, v^{-1}(x) = p(x)v^{-1}(x) = 0 \mod p(x).$$

Отсюда следует, что $F_{p(x)}[x]$ не удовлетворяет аксиомам поля при непростом $p(x)$. □

Поле, построенное как кольцо многочленов по модулю неприводимого $p(x)$ степени m с коэффициентами из $GF(q)$, содержит q^m элементов. Оно называется *расширением* $GF(q)$ и обозначается как $GF(q^m)$.

Итак, любой простой многочлен может быть использован для построения поля Галуа $GF(q^m)$.

Пример 5.11. Два примера колец многочленов по модулю многочленов второй степени приведены в табл. 5.4. Кольцо многочленов по модулю $1 + x + x^2$ является полем.

Таблица 5.4. Два кольца многочленов по модулю полинома второй степени

$p(x) = 1 + x^2$		$p(x) = 1 + x + x^2$	
Элемент	Обратный элемент	Элемент	Обратный элемент
0	—	0	—
1	1	1	1
x	x	x	$1 + x$
$1 + x$	не определен	$1 + x$	x

Хотя поле Галуа, как мы видели, может быть задано любым простым многочленом, оно единственно с точностью до изоморфизма. Для построения кодов и вычислений в поле Галуа оказывается удобным описание мультипликативной группы поля в виде набора степеней одного из элементов поля.

Остановимся подробнее на свойствах мультипликативной группы поля.

Мультипликативная группа поля Галуа

Пусть имеется произвольная группа по умножению, состоящая из $q - 1$ элементов. Выберем некоторый элемент группы a. Очевидно, вместе с этим элементом группе принадлежат все элементы вида a^i, $i = 1, 2, \ldots$ Поскольку в группе конечное число элементов, для некоторого числа s получим $a^s = 1$.

Определение 5.7. *Наименьшее s такое, что $a^s = 1$, называется порядком элемента a.*

Согласно теореме 2.9, порядок подгруппы является делителем порядка группы, поэтому $q - 1$ делится на s и, в частности, может совпадать с s. Таким образом, каждый ненулевой элемент поля удовлетворяет уравнению

$$x^{q-1} - 1 = 0 \tag{5.9}$$

и любой элемент поля — уравнению

$$x^q - x = 0. \tag{5.10}$$

Это утверждение называется *малой теоремой Ферма*. Оно приводит к важному свойству, формулируемому ниже в виде теоремы 5.13, доказательство которой станет простым, благодаря двум элементарным леммам.

Лемма 5.11. *Если порядки s_a и s_b элементов a и b взаимно просты, то порядок их произведения ab равен произведению их порядков $s_{ab} = s_a s_b$.*

Доказательство. Поскольку $(ab)^{s_a s_b} = 1$, достаточно доказать, что порядок ab не может быть меньше, чем $s_a s_b$. Если для некоторого l имеет место $(ab)^l = 1$, то $(ab)^{ls_a} = b^{ls_a} = 1$. Это возможно, только если показатель степени ls_a кратен порядку s_b элемента b. Поскольку s_a и s_b взаимно просты, делаем вывод, что l кратно s_b. Аналогично можно убедиться, что l кратно s_a. В итоге заключаем, что $l = s_a s_b$. □

Лемма 5.12. *Если число s^t является делителем $q-1$, то в поле Галуа найдется элемент порядка s^t.*

Доказательство. Положим $q - 1 = s^t u$. Уравнению $x^u = 1$ удовлетворяет $u - 1 < q - 1$ элементов, т.е. не все элементы поля. Пусть a не удовлетворяет этому уравнению, т.е. $a^u \neq 1$. При этом для элемента $b = a^u$ имеет место $b^{s^t} = a^{q-1} = 1$, т.е. его порядок либо равен s^t, либо делит его, т.е. равен s^j, $j < t$. Поскольку множество всех элементов поля, которые удовлетворяют уравнению $x^{s^t} = 1$, содержит все элементы, удовлетворяющие $x^{s^j} = 1$, и при этом число решений первого уравнения больше, чем второго, непременно найдется элемент, удовлетворяющий только первому из этих двух уравнений, т.е. элемент порядка s^t (см. пример 5.12 ниже). □

Теорема 5.13. *В поле из q элементов найдется элемент порядка $q-1$.*

Доказательство. При простом $q-1$ все нетривиальные элементы имеют такой порядок, т.к. $q-1$ — единственный делитель порядка группы. Пусть число $q-1$ составное, его разложение на простые множители запишем в виде $q-1=\prod_{i=1}^{s} p_i^{\nu_i}=\prod_{i=1}^{s}\theta_i$.

Из лемм 5.11 и 5.12 теперь следует, что найдется элемент порядка $q-1$. □

Теорема 5.13, по сути, утверждает, что все ненулевые элементы поля Галуа можно получить возведением в степень одного элемента поля. Таким образом, в терминологии теории групп мультипликативная группа поля Галуа является *циклической группой*. Элемент, из которого последовательным применением групповой операции можно получить всю группу, называется *образующим элементом циклической группы*. Этот элемент играет настолько большую роль в теории групп, что у него есть еще одно название.

Определение 5.8. *В поле $GF(p^m)$ элемент порядка p^m-1 называется примитивным.*

Пример 5.12. Мультипликативная группа поля $GF(5^2)$ содержит 24 элемента, т.е. порядки элементов являются числами из множества $\{1,2,3,4,6,8,12,24\}$. Если x — примитивный элемент, то x^s имеет порядок 2 при $s=12$, порядок 3 при $s=8,16$, порядок 4 при $s=6,18$, порядок 6 при $s=4,20$, порядок 8 при $s=3,9,15,21$, порядок 12 при $s=2,10,14,22$. Элементы степеней 5,7,11,13,17,19,23 (взаимно простые с 24) имеют порядок 24, т.е. являются примитивными.

Мы показали выше, что любой неприводимый многочлен можно использовать для построения поля Галуа. Теперь стало ясно, что поле может быть описано заданием одного примитивного элемента. Наша следующая цель — связать эти два представления, полиномиальное и степенное.

Пример 5.13. Рассмотрим кольцо многочленов по модулю $p(x)=1+x+x^3$. Поскольку многочлен неразложим над $GF(2)$, это кольцо является полем $GF(2^3)$. В нем должен существовать элемент порядка 7. Возможно, элемент x является таковым. Чтобы

проверить этот факт, нужно подсчитать его степени по модулю $p(x)$. Вычисления сведены в табл. 5.5. Каждый новый элемент полиномиального представления получен умножением предыдущего на x по модулю $p(x)$.

Таблица 5.5. Мультипликативная группа поля $GF(2^3)$

Степень примитивного элемента	Полиномиальное представление	Двоичная запись полинома
x	x	010
x^2	x^2	100
x^3	$x^3 = 1 + x$	011
x^4	$x + x^2$	110
x^5	$x^2 + x^3 = 1 + x + x^2$	111
x^6	$x + x^2 + x^3 = 1 + x^2$	101
$x^7 = x^0$	$x + x^3 = 1$	001

Вычисления подтвердили, что x — примитивный элемент. Как мы увидим позже, совсем не любой неразложимый многочлен примитивен и может быть использован для описания ненулевых элементов в виде циклической группы.

Упражнение 5.8. Постройте поле $GF(3^2)$.

Определение 5.9. *Простой многочлен $p(x)$ степени m с коэффициентами из $GF(p)$ называется примитивным многочленом, если в поле полиномов по модулю $p(x)$ элемент x является примитивным.*

Рассмотрим произвольное поле $GF(q)$ и число 1 в этом поле. Суммируя различное число единиц, будем получать различные элементы поля, и, в силу конечности поля, для некоторого p выполняется равенство

$$\underbrace{1 + 1 + \ldots + 1}_{p \text{ слагаемых}} = 0.$$

Минимальное число p, для которого выполняется это равенство, называется *характеристикой поля*. Число p — простое, т.к. иначе

мы получили бы равенство $p = st = 0$, из которого бы следовало, что либо r, либо s равны нулю.

Само поле называют *полем характеристики* p. В таком поле имеют место тождества

$$px = 0; \tag{5.11}$$

$$(x+y)^p = x^p + y^p. \tag{5.12}$$

Первое следует из определения, а для доказательства второго нужно воспользоваться формулой бинома Ньютона. При этом окажется, что все члены разложения, кроме x^p и y^p, содержат в качестве сомножителей биномиальные коэффициенты $\binom{p}{i}$, кратные p, т.к. p — простое, и равные нулю в силу (5.11).

Минимальные многочлены

Корнем многочлена $p(x)$ с коэффициентами из поля $GF(p)$ называется элемент x, для которого имеет место равенство $p(x) = 0$. Если a является корнем $p(x)$, то $p(x)$ делится на двучлен $(x - a)$. Таким образом, многочлен степени m может иметь не больше m корней (не обязательно принадлежащих тому же полю, что и коэффициенты многочлена $p(x)$).

Поскольку все элементы поля $GF(q)$ удовлетворяют (5.10), имеет место разложение

$$x^q - x = \prod_{\alpha \in GF(q)} (x - \alpha). \tag{5.13}$$

Отсюда следует, что в поле $GF(p^m)$ всякий элемент является корнем некоторого многочлена степени p^m или меньше.

Определение 5.10. *Минимальным многочленом $M(x)$ над полем $GF(p)$ элемента α в поле $GF(p^m)$ называется нормированный многочлен наименьшей степени, корнем которого является α.*

Свойства минимальных многочленов:

Свойство 5.1. *Минимальный многочлен элемента α неприводим.*

Доказательство. В противном случае один из сомножителей имел бы корнем α, и его степень была бы меньше степени $M(x)$. □

Свойство 5.2. *Любой многочлен, имеющий корень α, делится на $M(x)$.*

Доказательство. Пусть для некоторого $f(x)$ имеет место $f(\alpha) = 0$. Тогда остаток $f(x)$ от деления $f(x)$ на $M(x)$ тоже должен иметь корень α. Но степень остатка меньше степени $M(x)$, что противоречит предположению о минимальности $M(x)$. □

Свойство 5.3. *$x^{p^m} - x$ делится на $M(x)$.*

Доказательство. Это свойство непосредственно следует из свойства 2. □

Свойство 5.4. *Степень $M(x)$ не превышает m.*

Доказательство. Поле $GF(p^m)$ представляет собой m-мерное линейное пространство полиномов степени не более $m-1$. Элементы $1, \alpha, \alpha^2, ...\alpha^m$ линейно зависимы и, следовательно, для некоторого набора коэффициентов $f_0, f_1, ..., f_m$ имеет место равенство $f_0 + f_1\alpha + ... + f_m\alpha^m = f(\alpha) = 0$. □

Свойство 5.5. *Степень минимального многочлена $M(x)$ примитивного элемента равна m.*

Доказательство. Если бы степень была меньше, например d, то степени корней $M(x)$, приведенные по модулю $M(x)$, образовали бы поле из $p^d < p^m$ элементов, и порядок примитивного элемента был бы меньше числа элементов поля, что противоречит определению примитивного элемента. □

6. БЧХ-коды и РС-коды

Не будет преувеличением сказать, что БЧХ и РС-коды — рекордсмены по практическому использованию в разнообразных системах и устройствах хранения и передачи данных. Кроме того, эти коды важны с теоретической точки зрения, поскольку, по сути, все остальные кодовые конструкции алгебраических блоковых кодов являются в той или иной степени их обобщением и развитием. Большая часть лучших известных кодов получается модификацией кодов этого класса.

Коды БЧХ получили свое название от имен первооткрывателей Боуза, Рой-Чоудхури ([56], 1960 г.) и Хоквингема ([79], 1959 г.), название РС-кодов связано с их авторами Ридом и Соломоном ([98], 1960 г.).

Важно отметить, что коды БЧХ и РС мы рассматриваем как «длинные» коды с «простым» декодированием. Эти слова надо понимать следующим образом. Коды БЧХ образуют бесконечную последовательность кодов, для которых известен способ построения и способ их декодирования с полиномиальной сложностью в канале с жесткими решениями с исправлением ошибок кратности до половины «конструктивного расстояния». Конструктивное расстояние БЧХ-кодов при фиксированном размере алфавита асимптотически плохое, доля исправляемых ошибок убывает с ростом длины кодов, тем не менее, при конечных длинах эти коды достаточно хороши. Коды Рида–Соломона имеют наибольшее возможное расстояние, удовлетворяющее границе Синглтона, но алфавит кода растет с его длиной.

Итак, достоинство рассматриваемых в данном разделе кодов — простота конструкции и алгоритма декодирования в канале с жест-

кими решениями. Недостаток — отсутствие разумного по сложности алгоритма декодирования в канале с мягкими решениями. Этот недостаток частично преодолевается использованием предложенного Форни декодирования по МОР (минимуму обобщенного расстояния), которое сводится к многократному декодированию в жестком канале. Этот метод будет рассмотрен позже в параграфе 7.5.

6.1. Определение БЧХ-кода

Прежде чем будут даны формальные определения, попробуем самостоятельно построить код с минимальным расстоянием 5, т.е. код, исправляющий две ошибки. В примерах на построение линейных кодов мы легко научились строить коды с расстоянием $1, 2, 3, 4$. Не может быть, чтобы следующий шаг оказался для нас непосильным.

Запишем проверочную матрицу кода Хэмминга (15,11) в виде

$$H = \begin{pmatrix} 1 & \alpha^1 & \alpha^2 & \cdots & \alpha^{14} \end{pmatrix},$$

где α обозначает примитивный элемент поля $GF(2^4)$, построенного, например, с помощью примитивного многочлена $p(x) = 1 + x + x^4$. Чтобы получить двоичную форму матрицы (она нам сейчас не нужна), достаточно записать степени примитивного элемента в виде двоичных коэффициентов их разложения по степеням α.

Упражнение 6.1. Постройте поле $GF(2^4)$.

Результат выполнения этого упражнения приведен в табл. 6.1.

Одну ошибку, как и должно быть, код исправляет. Пусть $b(x) = c(x) + e(x)$ — полином, соответствующий принятой из канала последовательности при передаче кодового слова $c(x)$, полином $e(x)$ соответствует вектору ошибок. Тогда синдром мы запишем как

$$s = \sum_{i=0}^{n-1} b_i \alpha^i = \sum_{i=0}^{n-1} e_i \alpha^i = b(\alpha) = e(\alpha).$$

Если, скажем, ошибка произошла в позиции с номером i, то $b(x) = x^i$ и $s = \alpha^i$, и по s мы однозначно определим номер ошибочной позиции i. Посмотрим, что произойдет в случае двух ошибок.

Пусть $e(x) = x^i + x^j$, $i \neq j$. Получим

$$s = \alpha^i + \alpha^j. \tag{6.1}$$

В правой части получили некоторый элемент поля, который ошибочно указывает нам на некоторую позицию, где ошибки нет. Очевидно, имея только одну строку в матрице H, мы не сможем исправить две ошибки. Уравнение (6.1) — одно уравнение с двумя неизвестными. Исправить положение могло бы второе уравнение. Таким образом, нужна дополнительная избыточность в виде еще одной строки, и тогда у нас была бы система уравнений вида

$$\begin{cases} s_1 = \alpha^i + \alpha^j; \\ s_2 = f(\alpha^i) + f(\alpha^j). \end{cases} \tag{6.2}$$

Если удачно выбрать функцию f, то из этих двух уравнений с двумя неизвестными мы смогли бы находить единственную неизвестную пару (i, j) и тем самым гарантировать исправление двух ошибок, т.е. гарантировать минимальное расстояние 5. Заметим, что линейные функции не годятся, поскольку уравнения окажутся линейно зависимыми. Из нелинейных функций первое, что приходит на ум — возведение в квадрат.

Это правильное решение, но не для любого поля. Для полиномов с коэффициентами из поля $GF(2)$ имеет место тождество $b(x^2) = (b(x))^2$, из которого следует $s_2 = (s_1)^2$. Это означает, что любое решение первого уравнения одновременно является решением второго.

Выберем функцию $f(x) = x^3$. Тогда новая проверочная матрица примет вид

$$H = \begin{pmatrix} 1 & \alpha^1 & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 & \alpha^7 & \alpha^8 & \alpha^9 & \alpha^{10} & \alpha^{11} & \alpha^{12} & \alpha^{13} & \alpha^{14} \\ 1 & \alpha^3 & \alpha^6 & \alpha^9 & \alpha^{12} & 1 & \alpha^3 & \alpha^6 & \alpha^9 & \alpha^{12} & 1 & \alpha^3 & \alpha^6 & \alpha^9 & \alpha^{12} \end{pmatrix}. \tag{6.3}$$

Во второй строке записаны элементы первой строки, возведенные в третью степень.

Пусть, как и прежде, i и j — номера ошибочных позиций. Введем обозначения $x = \alpha^i$, $y = \alpha^j$. Тогда, обозначив скалярное произведение первой строки с принятой последовательностью через s_1,

Таблица 6.1. Поле $GF(2^4)$, порожденное многочленом $p(x) = 1 + x + x^4$

Степень	Многочлен	Последовательность
$-\infty$	0	0000
0	1	0001
1	x	0010
2	x^2	0100
3	x^3	1000
4	$1 + x$	0011
5	$x + x^2$	0110
6	$x^2 + x^3$	1100
7	$1 + x + x^3$	1011
8	$1 + x^2$	0101
9	$x + x^3$	1010
10	$1 + x + x^2$	0111
11	$x + x^2 + x^3$	1110
12	$1 + x + x^2 + x^3$	1111
13	$1 + x^2 + x^3$	1101
14	$1 + x^3$	1001

второй — через s_3, получаем уравнения

$$\begin{cases} x + y = s_1; \\ x^3 + y^3 = s_3. \end{cases} \tag{6.4}$$

При одной или двух ошибках $s_1 \neq 0$, поэтому, деля второе уравнение на первое, находим

$$x^2 + y^2 + xy = s_3/s_1.$$

Здесь мы воспользовались тем, что в поле характеристики 2 вычитание и сложение эквивалентны. Поскольку в этом поле $(a + b)^2 = a^2 + b^2$, имеем

$$xy = s_3/s_1 - s_1^2.$$

По теореме Виета, произведение и сумму двух чисел можно интерпретировать как коэффициенты квадратного уравнения с кор-

нями x и y:

$$z^2 + \sigma_1 z + \sigma_2 = 0, \tag{6.5}$$

где

$$\sigma_1 = s_1, \quad \sigma_2 = s_3/s_1 - s_1^2. \tag{6.6}$$

Итак, задачу декодирования можно считать решенной при условии, что мы умеем решать квадратные уравнения над конечными полями. Хотя решения уравнений небольшой степени (до 4-й) можно искать неперебо́рными методами, на практике чаще всего их решают просто подстановкой в уравнение вида (6.5) всех ненулевых элементов поля. Порядок сложности этой процедуры примерно такой же, как порядок сложности вычисления синдрома. Поэтому логическая простота такого решения делает его предпочтительным. Подытожим процедуру декодирования данного кода в виде алгоритма 6.1.

Пример 6.1. Рассмотрим код с проверочной матрицей (6.3). Предположим, что на выходе канала наблюдается последовательность (1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1). Вычисляем синдром: $s_1 = \alpha^{11}$ и $s_3 = \alpha^4$, находим коэффициенты уравнения, $\sigma_1 = \alpha^{11}$, $\sigma_2 = \alpha^{11}$. Поочередной подстановкой элементов поля убеждаемся в том, что корнями являются α^2 и α^9. Это означает, что ошибочными являются позиции с номерами 3 и 10. Кодовое слово с исправленными ошибками имеет вид (1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1). Информационные символы (1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1) выдаются получателю (в предположении, что использовано систематическое кодирование).

Мы убедились в том, что код, заданный проверочной матрицей вида (6.3), исправляет любые 2-кратные ошибки. Можно утверждать, что его минимальное расстояние не меньше 5, а число проверочных символов не больше 8 (оно равно 8, если строки H линейно независимы, что имеет место в данном случае). Построенный код — БЧХ-код (15,7) с минимальным расстоянием 5.

Название кода — аббревиатура от имен авторов конструкции Боуза и Чоудхури [56] (1960) и Хоквингема [79] (1959).

Алгоритм 6.1. Исправление двух ошибок

Input: Двоичная последовательность $\boldsymbol{b}$, полученная из канала ;

Output: Кодовое слово $\boldsymbol{c}$;

`Инициализация:` $\boldsymbol{c} \leftarrow \boldsymbol{b}$, флаг успешного декодирования $F = 1$;

Вычисляем компоненты синдрома s_1 и s_3;

if $s_1 = s_3 = 0$ **then**
  `Имеем кодовое слово. Декодирование закончено`
else
  if $s_1 = 0$, $s_3 \neq 0$ **then**
    `Ошибок больше 2. Декодирование невозможно` ;
    $F \leftarrow 0$;
  else
    if $s_1 \neq 0$, $s_3 = s_1^3$ **then**
      `Одиночная ошибка` ;
      Находим позицию ошибки по s_1 и исправляем, инвертируя соответствующую позицию в $\boldsymbol{c}$.
    else
      Находим коэффициенты уравнения (6.5) по формулам (6.6);
      Поочередной подстановкой элементов 1, α^1, ..., α^{n-1} в уравнение (6.5) находим корни уравнения;
      if *корни найдены* **then**
        исправляем ошибки в $\boldsymbol{c}$
      else
        `корни не найдены`;
        число ошибок не меньше 3; $F \leftarrow 0$;
      end
    end
  end
end

Вывод: Флаг F, последовательность $\boldsymbol{c}$

Успешный опыт построения кода с расстоянием 5 подсказывает прямой путь к построению кодов с еще большими расстояниями. Прежде, чем мы продвинемся дальше по этому пути, имеет смысл рассмотреть подробнее свойства построенного кода.

Первая строка в (6.3) показывает, что $c(x)$ — кодовое слово, если $c(\alpha) = 0$, откуда немедленно следует, что $\alpha, \alpha^2, \alpha^4, \ldots$ являются корнями $c(x)$, т.е. все корни минимального многочлена являются корнями $c(x)$. Это означает, что $c(x)$ делится на минимальный многочлен $M_1(x) = p(x)$ примитивного элемента α. Точно так же мы устанавливаем, что $c(x)$ делится на минимальный многочлен $M_3(x)$ элемента α^3. Отсюда следует, что построенный код — циклический с порождающим многочленом

$$g(x) = M_1(x)M_3(x).$$

Теорема 6.1. *(Граница БЧХ.) Пусть $g(x)$ — порождающий многочлен циклического кода, и α — примитивный элемент конечного поля. Если для некоторых чисел $b \geq 0$ и $d \geq 2$ имеют место равенства*

$$g(\alpha^i) = 0, \quad i = b, \ldots, b + d - 2, \tag{6.7}$$

то минимальное расстояние кода не меньше d.

Иными словами, если $d-1$ последовательных степеней примитивного элемента поля обращают в ноль порождающий многочлен кода, то минимальное расстояние не меньше d. В рассмотренном выше примере кода, исправляющего 2 ошибки, такими степенями были 1, 2, 3, 4. При этом, как мы убедились, расстояние кода было не меньше 5.

Доказательство. Из условия теоремы следует, что каждое кодовое слово $c(x)$ обращается в ноль при подстановке вместо x любого из $d-1$ элементов вида α^i, $i = b, \ldots, b+d-2$. Отсюда следует, что проверочная матрица кода может быть записана в виде

$$H = \begin{pmatrix} 1 & \alpha^b & \alpha^{2b} & \cdots & \alpha^{(n-1)b} \\ 1 & \alpha^{b+1} & \alpha^{2(b+1)} & \cdots & \alpha^{(n-1)(b+1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{b+d-2} & \alpha^{2(b+d-2)} & \cdots & \alpha^{(n-1)(b+d-2)} \end{pmatrix}. \tag{6.8}$$

Докажем, что любые $d-1$ столбцов линейно независимы. Для этого рассмотрим произвольный набор индексов $i_1, \ldots, i_{d-1}$ и соот-

ветствующую подматрицу

$$\begin{pmatrix} \alpha^{i_1 b} & \alpha^{i_2 b} & \dots & \alpha^{i_{d-1} b} \\ \alpha^{i_1 (b+1)} & \alpha^{i_2 (b+1)} & \dots & \alpha^{i_{d-1} (b+1)} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha^{i_1 (b+d-2)} & \alpha^{i_2 (b+d-2)} & \dots & \alpha^{i_{d-1} (b+d-2)} \end{pmatrix}.$$

Ее определитель равен

$$\alpha^{(i_1+\ldots+i_{d-1})b} \det \begin{pmatrix} 1 & 1 & \dots & 1 \\ \alpha^{i_1} & \alpha^{i_2} & \dots & \alpha^{i_{d-1}} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha^{i_1 (d-2)} & \alpha^{i_2 (d-2)} & \dots & \alpha^{i_{d-1} (d-2)} \end{pmatrix}. \qquad (6.9)$$

Известно тождество

$$\det \begin{pmatrix} 1 & 1 & \dots & 1 \\ a_1 & a_2 & \dots & a_m \\ \vdots & \vdots & \ddots & \vdots \\ a_1^{m-1} & a_2^{m-1} & \dots & a_m^{m-1} \end{pmatrix} = \prod_{j=1}^{m-1} \prod_{i=j+1}^{m} (a_i - a_j), \qquad (6.10)$$

в котором матрица в левой части называется *матрицей Вандермонда*. Ее определитель называют *определителем Вандермонда*. Он отличен от нуля, если элементы $a_1, \ldots, a_m$ различны.

Нетрудно видеть, что определитель в (6.9) является частным случаем определителя Вандермонда. Поскольку α — примитивный элемент, его различные степени различны, определитель не равен нулю, ранг матрицы H не меньше $d-1$, и, следовательно, минимальное расстояние кода не меньше d. □

Определение 6.1. *Циклический код длины n над $GF(q)$ называется кодом БЧХ с конструктивным расстоянием d, если для некоторого $b \geq 0$ порождающий многочлен кода равен*

$$g(x) = НОК\{M_i(x), i = b, b+1, \ldots, b+d-2\}. \qquad (6.11)$$

Заметим, что минимальные многочлены различных элементов поля либо взаимно просты либо совпадают. В определение мы пишем наименьшее общее кратное, а не произведение минимальных

многочленов, поскольку минимальные многочлены часто совпадают.

Например, рассмотренный выше код с расстоянием 5 имел корнями примитивный элемент в степени с первой по четвертую. Для степеней 1, 2 и 4 минимальным многочленом служил общий примитивный многочлен, по которому было построено само поле. Поэтому порождающий многочлен — произведение не четырех, а только двух минимальных многочленов, соответствующих первой и третьей степеням примитивного элемента.

Теорема 6.2. *Пусть код БЧХ длины n с конструктивным расстоянием d над полем $GF(p)$ задан порождающим многочленом, корнями которого являются $d-1$ последовательных степеней элемента α порядка n в некотором расширении $GF(p^m)$ поля $GF(p)$. Тогда минимальное расстояние кода не меньше d, а размерность кода не меньше $n - m(d-1)$.*

Доказательство. Обоснования требует только оценка размерности кода. Поскольку степень каждого минимального многочлена не превышает m, а число сомножителей, образующих порождающий многочлен, не превышает $d-1$, степень произведения не больше $m(d-1)$. Эта величина является верхней оценкой избыточности кода. □

Коды БЧХ длины $n = q^m - 1$ называют *примитивными*. Длины непримитивных БЧХ-кодов являются делителями числа $q^m - 1$.

Для двоичных БЧХ-кодов оценка теоремы 6.2 может быть существенно улучшена. Поскольку минимальный многочлен $M_{2i}(x)$ совпадает с $M_i(x)$, выбрав в качестве порождающего многочлена

$$g(x) = \text{НОК}\left\{M_1(x), M_3(x), \ldots, M_{2t-1}(x)\right\}, \tag{6.12}$$

можно гарантировать конструктивное расстояние $d = 2t+1$. Отсюда получаем оценку размерности двоичного БЧХ-кода с минимальным расстоянием $d = 2t + 1$:

$$k \geq n - mt. \tag{6.13}$$

6.2. Построение БЧХ-кодов. Примеры

Из предыдущего параграфа, в принципе, понятно, как можно построить БЧХ-код с заданным числом исправляемых ошибок. Для этого нужно подобрать последовательность корней порождающего многочлена в соответствии с определением 6.1 и построить порождающий многочлен в соответствии с (6.11) или (6.12).

Заметим, что истинная избыточность кода зависит от того, какими конкретно окажутся наборы примитивных многочленов.

Те элементы поля, которые имеют одинаковые минимальные многочлены, называют *сопряженными элементами.* Поскольку в поле характеристики p значения любого полинома в точках β и β^p одинаковы, то подмножества сопряженных элементов можно получать возведением элементов в степень p. Пусть для некоторого s имеет место равенство $\beta = \alpha^s$, где α — примитивный элемент поля, и пусть m_s — наименьшее число, такое, что $p^{m_s} s = s \pmod{p^m - 1}$. Наборы показателей степеней

$$\{s, ps, p^2 s, \ldots, p^{m_s} s\}$$

при различных s образуют непересекающиеся множества, называемые *циклотомическими классами.* В каждом классе наименьшее число s называется *представителем класса.*

Циклотомические классы не пересекаются и покрывают все множество показателей степеней мультипликативной группы поля. Оценки параметров БЧХ-кодов определяются структурой циклотомических классов и выбранным расширением конечного поля.

Пример 6.2. Циклотомическими классами по модулю 15 являются множества

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

Исходя из структуры циклотомических классов, легко построить таблицу БЧХ-кодов длины 15. Их параметры приведены в табл. 6.2.

Таблица 6.2. Примеры БЧХ-кодов длины 15

Показатели степеней корней	Порождающий многочлен	Размерность $n - \deg g(x)$	Расстояние
0	$M_0(x) = x + 1$	14	2
1,2,4,8	$M_1(x)$	11	3
0,1,2,4,8	$M_0(x)M_1(x)$	10	4
1,2,3,4,6,9,12	$M_1(x)M_3(x)$	7	5
1,...,5,6,8,9, 10,12	$M_1(x)M_3(x)\times$ $\times M_5(x)$	5	7
1,2,...,14	$M_1(x)M_3(x)\times$ $\times M_5(x)M_7(x)$	1	15

Упражнение 6.2. Постройте циклотомические классы и таблицу кодов длины 31.

Пример 6.3. Приведем пример непримитивного БЧХ-кода. Положим $n = 23$. Циклотомические классы имеют вид:

$$C_0 = \{0\};$$
$$C_1 = \{1, 2, 4, 8, 16, 9, 18, 13, 3, 6, 12\};$$
$$C_5 = \{5, 10, 20, 17, 11, 22, 21, 19, 15, 7, 14\}.$$

Примеры кодов приведены в табл. 6.3. Код во второй строке — код Голея. На этом примере мы видим, что оценка БЧХ на минимальное расстояние бывает неточной.

Итак, для получения параметров БЧХ-кода достаточно знания структуры циклотомических классов поля, из которого выбираются корни порождающего многочлена.

Для нахождения порождающего многочлена нужно знать минимальные многочлены корней. Один из способов их нахождения минимальных многочленов:

$$M_i(x) = \prod_{j \in C_i} (x - \alpha^j),$$

где произведение вычисляется по всем элементам циклотомического класса.

Таблица 6.3. Примеры БЧХ-кодов длины 23

Показатели степеней корней	Порождающий многочлен	Размерность $n - \deg g(x)$	Расстояние	
			конструктивное	истинное
0	$M_0(x) = x + 1$	22	2	2
1,2,3,4,6,8,...	$M_1(x)$	12	5	7
...**19,20,21,22,**...	$M_5(x)$	12	5	7
0,1,2,3,4,6,...	$M_0(x)M_1(x)$	11	6	8
1,...,23	$M_1(x)M_5(x)$	1	23	23

Таблица 6.4. Двоичные примитивные многочлены

Степень	Полином	Степень	Полином
2	2,1,0	16	16,12,3,1,0
3	3,1,0	17	17,3,0
4	4,1,0	18	18,7,0
5	5,2,0	19	19,5,2,1,0
6	6,1, 0	20	20,3,0
7	7,3,0	21	21,2,0
8	8,4, 3, 2,0	22	22,1,0
9	9,4,0	23	23,5,0
10	10,3,0	24	24,7,2,1,0
11	11,2,0	25	25,3,0
12	12,6,4,1,0	26	26,6,2,1,0
13	13,4,3,1,0	27	27,5,2,1,0
14	14,10,6,1,0	28	28,3,0
15	15,1,0	29	29,2,0

Пример 6.4. Минимальный многочлен элемента α^5 в поле $GF(2^4)$, заданном примитивным полиномом $p(x) = 1 + x + x^4$, равен

$$M_5(x) = (x + \alpha^5)(x + \alpha^{10}) =$$
$$= x^2 + (\alpha^5 + \alpha^{10})x + \alpha^0 =$$
$$= x^2 + x + 1$$

Таблица 6.5. Разложения чисел $2^m - 1$ на простые сомножители

m	Разложение	m	Разложение
3	7	14	3×43×127
4	3×5	15	7×31×151
5	31	16	3×5×17×257
6	3×3×7	17	131071
7	127	18	$3^3 \times 7 \times 19 \times 73$
8	3×5×17	19	524287
9	7×73	20	3×5×5×11×31×41
10	3×11×31	21	7×7×127×337
11	23×89	22	3×23×89×683
12	3×3×5×7×13	23	47×178841
13	8191	24	3×3×5×7×13×17×241

При выполнении этих вычислений использована табл. 6.1. Согласно этой таблице двоичными представлениями α^5 и α^{10} служат (0110) и (0111). Сумма этих элементов равна единице.

Для построения двоичных БЧХ-кодов нужны примитивные многочлены, задающие поля $GF(2^m)$. Примеры таких многочленов представлены в табл. 6.4. В таблице указаны степени, входящие в многочлен с ненулевыми коэффициентами.

Для построения непримитивных кодов БЧХ полезны разложения чисел вида $2^m - 1$ на простые сомножители. Разложения для относительно малых значений m приведены в табл. 6.5.

6.3. Коды Рида–Соломона

Хотя это не подчеркивалось особо, при изучении кодов БЧХ мы, в основном, интересовались двоичными кодами. По крайней мере, все коды в примерах были двоичными. Вся теория, на которой базируются БЧХ-коды, в равной степени верна и для недвоичных кодов. Недвоичные коды могут быть полезны в системах связи с недвоичной модуляцией. В этом смысле среди недвоичных кодов особенно интересны коды, алфавиты которых — расширения двоичного поля.

В определенном смысле корректирующая способность недвоичных кодов при одинаковом числе гарантированно исправляемых ошибок несколько лучше, чем двоичных кодов: если несколько двоичных ошибок приходятся на один недвоичный символ недвоичного кода, они рассматриваются как одна ошибка. Это означает, что код над полем $GF(2^m)$, исправляющий t ошибок, в действительности может исправить t «синхронизированных» пакетов ошибок длины m каждый, т.е. вплоть до tm ошибок. Во многих системах передачи и хранения информации ошибки группируются, и поэтому коды над большими алфавитами могут оказаться весьма практичными.

По сути, коды Рида–Соломона [98] (РС-коды, 1960 г.) — частный случай БЧХ-кодов. В соответствии с (6.11), избыточность БЧХ-кода тем меньше, чем меньше степень минимальных многочленов, образующих порождающий многочлен циклического кода. Степень не может быть меньше 1, и она равна единице, если формальная переменная x принимает значения в том же поле, что и поле $GF(2^m)$, использованное для построения проверочной матрицы кода. Действительно, в таком поле минимальным многочленом элемента β служит $x-\beta$. Поскольку степеням переменной x соответствуют позиции кодового слова, длина кода должна быть не больше числа элементов мультипликативной группы поля.

Определение 6.2. *Кодом Рида–Соломона над $GF(q)$ называется код БЧХ длины $n = q - 1$.*

Из определения следует, что порождающий многочлен кода Рида–Соломона с конструктивным расстоянием d имеет вид

$$g(x) = (x - \alpha^b)(x - \alpha^{b+1}) \times \cdots \times (x - \alpha^{b+d-2}). \tag{6.14}$$

Чаще всего полагают $b = 1$, но другие значения b также используются и, более того, иногда приводят к более простым схемам кодирования.

Из (6.14) и теорем 6.1 и 6.2 следует, что параметры РС-кода связаны соотношением

$$d - 1 = n - k, \tag{6.15}$$

т.е. удовлетворяют границе Синглтона. Поскольку граница Синглтона одновременно является верхней границей на минимальное

расстояние кодов, РС-коды являются оптимальными в смысле минимального расстояния, их еще называют кодами с максимально достижимым расстоянием (МДР-кодами).

Пример 6.5. Рассмотрим код длины $n = 15$ над полем $GF(2^4)$. Для построения кода с минимальным расстоянием 5 положим $b = 1$, и тогда

$$g(x) = (x-\alpha)(x-\alpha^2)(x-\alpha^3)(x-\alpha^4) = x^4+\alpha^{13}x^3+\alpha^6x^2+\alpha^3x+\alpha^{10}.$$

Имеем (15,11)-код с $d = 5$. Напомним, что двоичный код с таким же расстоянием имел параметры (15,7). Если записывать информационные и кодовые символы $GF(2^4)$ в двоичном виде, то получим линейный код (60, 44) с расстоянием 5. Как двоичный код этот код неоптимален, т.к. для кода с такой длиной и размерностью достижимое расстояние точно неизвестно, оно лежит в интервале от 6 до 7.

Одно из основных применений РС-кодов — использование в качестве компонентных кодов в каскадных конструкциях, которые будут рассматриваться позже (см. задачу 7 в конце главы).

При заданном размере алфавита желательно иметь код как можно большей размерности. В этом смысле полезны расширенные коды Рида–Соломона длины $n = q + 1$, размерности k с расстоянием $d = n - k + 1$. Проверочная матрица такого кода может быть представлена, в частности, в виде

$$H = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 & 0 \\ 0 & \alpha & \alpha^2 & \cdots & \alpha^{(q-1)} & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \alpha^{q-k} & \alpha^{2(q-k)} & \cdots & \alpha^{(q-k)(q-1)} & 1 \end{pmatrix}.$$

Легко убедиться в том, что любые $n-k$ столбцов этой матрицы линейно независимы.

Задачи

1. Постройте матрицу вида (6.3), но с возведением элементов первой строки не в третью, а во вторую степень. Определите минимальное расстояние и размерность кода.

2. Напишите программы для построения поля, нахождения минимальных многочленов, нахождения корней уравнений перебором по элементам поля. Напишите программу удобного МАТЛАБ-калькулятора для выполнения арифметических операций в конечном поле. Этот калькулятор пригодится при решении задач на декодирование БЧХ и РС-кодов.

3. Укажите параметры БЧХ-кодов длины 31 и 63. Сравните параметры с параметрами лучших известных линейных кодов.

4. Укажите параметры непримитивных БЧХ-кодов длины 17 и 21. Сравните их с параметрами лучших известных линейных кодов.

5. Построить порождающие многочлены двоичных БЧХ-кодов и РС-кодов длины 31, исправляющих 2-кратные ошибки.

6. Код БЧХ длины 31, исправляющий 2-кратные ошибки, использован для передачи сообщений. Примитивный многочлен, использованный для построения кода $p(x) = 1+x^2+x^5$. На выходе двоичного канала связи наблюдается последовательность y =403415447 (это двоичная последовательность, записанная в восьмеричной форме). Определите переданное сообщение.

7. *Каскадные коды [18].* Выберем два кода:

   - (N, K)-код Рида–Соломона с минимальным расстоянием D над полем $GF(2^k)$ (внешний код);
   - двоичный (n, k)-код с минимальным расстоянием d (внутренний код).

   Кодирование для каскадного кода выполняется в два этапа. Последовательность из kK двоичных информационных символов разбивается в блоки по k бит, каждый блок интерпретируется как элемент поля $GF(2^k)$ и последовательность из K расширенных информационных символов кодируется внешним кодом. Затем N кодовых символов этого кода интерпретируются как двоичные последовательности длины k и кодируются внутренним кодом. Полученная на выходе кодера после-

довательность представляет собой кодовое слово каскадного кода.

- Найдите параметры (длину, скорость и расстояние) каскадного кода.
- Является ли код линейным?
- По аналогии с декодированием итеративных кодов (см. задачу 4 главы 2) предложите процедуры декодирования каскадного кода.
- Сравните каскадные и итеративные коды по скорости, расстоянию, сложности построения, сложности декодирования.

7. Декодирование БЧХ- и РС-кодов

В предыдущей главе на примере кодов, исправляющих двукратные ошибки, мы рассмотрели одну из возможных процедур декодирования. Непосредственно из структуры проверочной матрицы видно, что задача декодирования сводится к решению системы нелинейных уравнений относительно позиций ошибок (точнее, элементов поля, соответствующих позициям ошибок). Огромное практическое значение кодов БЧХ и РС послужило стимулом к поиску более эффективных алгоритмов. Одно из первых решений задачи — алгоритм Питерсона (1960) и Горенстейна–Цирлера (1961), который сводит задачу исправления t ошибок к решению системы из t линейных уравнений. Сложность такого декодирования пропорциональна t^3.

Берлекэмп (1968), используя особенности матрицы коэффициентов системы уравнений, уменьшил сложность решения до величины порядка t^2. Месси (1969) сумел интерпретировать алгоритм Берлекэмпа как алгоритм построения рекуррентного фильтра, тем самым упростив и понимание алгоритма, и его реализацию. В окончательном виде этот алгоритм получил название алгоритма Берлекэмпа–Месси. Дальнейшее упрощение декодирования связано с применением быстрых преобразований Фурье над конечными полями. Основные исследования в этой области были выполнены Блэйхутом (1981). В результате достигнута асимптотическая сложность декодирования кода длины n, пропорциональная $n \log n$. Тем не менее, на практике при реализации декодирования на БИС чаще всего применяется обычная версия алгоритм Берлекэмпа–

Месси. В данной главе рассматриваются алгоритм Питерсона–Горенстейна–Цирлера (ПГЦ) и алгоритм Берлекэмпа–Месси (БМ).

7.1. Алгоритм Питерсона–Горенстейна–Цирлера

При описании декодирования кодов БЧХ и РС нет необходимости различать эти два класса кодов. Достаточно рассматривать БЧХ-коды над произвольным полем $GF(q)$. Объяснения будут немного проще, если мы предположим, что последовательными корнями порождающего многочлена являются $\alpha, \alpha^2, ..., \alpha^{d-1}$, где α — примитивный элемент поля. Через d мы обозначили конструктивное расстояние кода, ему соответствует кратность исправляемых ошибок $t = \lfloor (d-1)/2 \rfloor$. Напомним, что $\lfloor a \rfloor$ обозначает округление a вниз до ближайшего целого.

Переданное кодовое слово $c(x)$, вектор ошибок $e(x)$ и принятую из канала последовательность $v(x)$ запишем в виде

$$\begin{aligned} c(x) &= c_0 + c_1 x + ... + c_{n-1} x^{n-1}; \\ e(x) &= e_0 + e_1 x + ... + e_{n-1} x^{n-1}; \\ v(x) &= c(x) + e(x) = v_0 + v_1 x + ... + v_{n-1} x^{n-1}. \end{aligned}$$

Поскольку $c(\alpha^j) = 0$ при $j = 1, ..., d-1$, для соответствующих компонент синдрома имеем

$$S_j = v(\alpha^j) = e(\alpha^j).$$

Предположим, что число ошибок (вес вектора ошибок) $\nu < t$ и что ошибки произошли на позициях с номерами $i_1, i_2, ..., i_\nu$. Тогда

$$S_j = e_{i_1} \alpha^{i_1 j} + e_{i_2} \alpha^{i_2 j} + \cdots + e_{i_\nu} \alpha^{i_\nu j},$$

где e_{i_h} обозначает значение ошибки на позиции h. Запись упростится, если использовать обозначения $Y_h = e_{i_h}$, $X_h = \alpha^{i_h}$. Получаем систему уравнений

$$S_j = Y_1 X_1^j + Y_2 X_2^j + \cdots + Y_\nu X_\nu^j, \quad j = 1, ..., d-1. \qquad (7.1)$$

Величины Y_h и X_h называют соответственно *значениями* и *локаторами* ошибок. При $\nu \le t$ комбинация ошибок может быть исправлена, система имеет единственное решение, но найти его непросто, поскольку система нелинейна и число неизвестных велико. Ключ к решению задачи — в замене переменных: вместо X_h мы будем искать коэффициенты полинома, корнями которого служат X_h^{-1} . Этот полином называется *полиномом локаторов ошибок*.

Итак, введем полином

$$\Lambda(x) = \prod_{j=1}^{\nu}(1 - xX_j) = 1 + \Lambda_1 x + ... + \Lambda_\nu x^\nu. \tag{7.2}$$

Поясним дальнейшие выкладки на примере случая $\nu = 3$. Из (7.1) имеем

$$\begin{array}{l} Y_1X_1 + Y_2X_2 + Y_3X_3 = S_1; \\ Y_1X_1^2 + Y_2X_2^2 + Y_3X_3^2 = S_2; \\ Y_1X_1^3 + Y_2X_2^3 + Y_3X_3^3 = S_3, \\ \ldots \end{array} \tag{7.3}$$

а из (7.2) последовательной подстановкой $x = X_1^{-1}$, $x = X_2^{-1}$ и $x = X_3^{-1}$ получаем

$$\begin{cases} X_1^3 + \Lambda_1 X_1^2 + \Lambda_2 X_1 + \Lambda_3 = 0; \\ X_2^3 + \Lambda_1 X_2^2 + \Lambda_2 X_2 + \Lambda_3 = 0; \\ X_3^3 + \Lambda_1 X_3^2 + \Lambda_2 X_3 + \Lambda_3 = 0. \end{cases} \tag{7.4}$$

Неизвестными для нас служат коэффициенты многочлена локаторов ошибок Λ_i. В последней системе уравнений в столбцах локаторы стоят в одинаковых степенях. Это наводит на мысль, что линейные комбинации столбцов могут порождать новые уравнения, в которых коэффициентами при Λ_i будут степенные суммы из (7.3), т.е. синдромы. Действительно, умножая уравнения (7.4), соответственно, на X_1Y_1, X_2Y_2 и X_3Y_3 и суммируя, с учетом (7.3), получаем уравнение

$$S_4 + \Lambda_1 S_3 + \Lambda_2 S_2 + \Lambda_3 S_1 = 0.$$

Еще два уравнения получим, умножая (7.4) на $X_i^2Y_1$, $i = 1, 2, 3$, а потом на $X_i^3Y_1$, $i = 1, 2, 3$. Получим систему линейных уравнений, которая в матричной форме может быть записана как

$$\begin{pmatrix} S_1 & S_2 & S_3 \\ S_2 & S_3 & S_4 \\ S_3 & S_4 & S_5 \end{pmatrix} \begin{pmatrix} \Lambda_3 \\ \Lambda_2 \\ \Lambda_1 \end{pmatrix} = \begin{pmatrix} -S_4 \\ -S_5 \\ -S_6 \end{pmatrix}. \tag{7.5}$$

Контуры будущего алгоритма становятся понятными: из (7.5) находим коэффициенты полинома локаторов ошибок, потом из (7.2) — его корни, т.е. локаторы ошибок. После этого из (7.3) найдем значения ошибок. Чтобы окончательно сформулировать алгоритм, введем обозначения

$$\mathbf{M}_\mu = \begin{pmatrix} S_1 & S_2 & \cdots & S_\mu \\ S_2 & S_3 & \cdots & S_{\mu+1} \\ \vdots & \vdots & \ddots & \vdots \\ S_\mu & S_{\mu+1} & \vdots & S_{2\mu-1} \end{pmatrix}, \tag{7.6}$$

$$\mathbf{L}_\mu = \begin{pmatrix} X_1 & X_2 & \cdots & X_\mu \\ X_1^2 & X_2^2 & \cdots & X_\mu^2 \\ \vdots & \vdots & \ddots & \vdots \\ X_1^\mu & X_2^\mu & \vdots & X_\mu^\mu \end{pmatrix}. \tag{7.7}$$

Формулы (7.3) и (7.5) в общем случае имеют вид

$$\mathbf{L}_\mu \begin{pmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_\mu \end{pmatrix} = \begin{pmatrix} S_1 \\ S_2 \\ \vdots \\ S_\mu \end{pmatrix}; \tag{7.8}$$

$$\mathbf{M}_\mu \begin{pmatrix} \Lambda_\mu \\ \Lambda_{\mu-1} \\ \vdots \\ \Lambda_1 \end{pmatrix} = \begin{pmatrix} -S_{\mu+1} \\ -S_{\mu+2} \\ \vdots \\ -S_{2\mu} \end{pmatrix}. \tag{7.9}$$

Теорема 7.1. *Матрицы $\mathbf{M}_\mu$, $\mathbf{L}_\mu$ невырождены, если μ равно числу ошибок ν, и матрица $\mathbf{M}_\mu$ вырождена, если $\nu < \mu$.*

Доказательство. Докажем теорему на примере случая $\mu = 3$. Положим сначала $\nu = \mu = 3$ и запишем $\mathbf{M}_3$ в виде

$$\mathbf{M}_3 = \begin{pmatrix} 1 & 1 & 1 \\ X_1 & X_2 & X_3 \\ X_1^2 & X_2^2 & X_3^2 \end{pmatrix} \begin{pmatrix} Y_1X_1 & 0 & 0 \\ 0 & Y_2X_2 & 0 \\ 0 & 0 & Y_3X_3 \end{pmatrix} \begin{pmatrix} 1 & X_1 & X_1^2 \\ 1 & X_2 & X_2^2 \\ 1 & X_3 & X_3^2 \end{pmatrix}. \tag{7.10}$$

Матрица $\mathbf{M}_3$ невырождена, поскольку является произведением трех невырожденных матриц (две из них — матрицы Вандермонда). Определитель матрицы $\mathbf{L}_3$ можно также выразить через определитель Вандермонда, и эта матрица также невырождена.

Допустим теперь, что $\mu = 3$, но $\nu = 2$. При этом разложение (7.10) остается в силе, если положить $Y_3 = 0$. Очевидно, определитель матрицы $\mathbf{M}_3$ станет равным нулю, т.к. центральная матрица в (7.10) окажется вырожденной. □

Формально декодирование ПГЦ описывается алгоритмом 7.1.

Пример 7.1. Рассмотрим РС-код (15,9) над полем $GF(2^4)$. Поле задано примитивным многочленом $p(x) = 1 + x + x^4$, список элементов поля приведен в табл. 6.1. Пример декодирования со всеми промежуточными вычислениями представлен в табл. 7.1.

7.2. Алгоритм Берлекэмпа–Месси

Алгоритм ПГЦ сводит задачу нахождения позиций и значений ν ошибок к решению двух систем линейных уравнений порядка ν. Для решения можно воспользоваться, например, методом Гаусса, и тогда сложность вычислений будет иметь порядок ν^3. Можно заметить, что в (7.5) и (7.9) мы имеем дело с весьма специфической матрицей коэффициентов: строки являются сдвигами предыдущих строк. Такие матрицы (матрицы с постоянными диагоналями) называются теплицевыми. Для них существуют более простые способы обращения, чем для матриц общего вида. Рассматриваемый ниже алгоритм БМ использует сходство структуры матрицы коэффициентов со структурой, которая часто встречается во многих приложениях, в частности, в задачах цифровой обработки сигналов.

Алгоритм 7.1. Алгоритм ПГЦ. Исправление до $\nu \le t = \lfloor (d-1)/2 \rfloor$ ошибок

Input: Выход канала $v(x)$;
Output: Кодовое слово $c(x)$;
Вычислить компоненты синдрома
for $j = 1$ **to** $d-1$ **do** $S_j = v(\alpha^j)$;

Положить $\nu = t+1$, $D = 0$;
while $D == 0$ **do**
| $\nu \leftarrow \nu - 1$; $D = \det(\mathbf{M}_\nu)$.
end

Вычислить коэффициенты полинома $\Lambda(x)$ локаторов ошибок как решение системы (7.9).
Найти локаторы ошибок как величины, обратные корням $\Lambda(x)$.
Найти значения ошибок как решение системы (7.8).
Исправить ошибки: $c(x) = v(x) - e(x)$.
Выход: Последовательность $c(x)$.

Вернемся к системе уравнений (7.5). Каждое из уравнений этой системы получается увеличением на единицу индексов при коэффициентах предыдущего уравнения. Уравнение с номером r имеет вид:

$$S_r = -\sum_{j=1}^{\nu} \Lambda_j S_{r-j} = -\Lambda_1 S_{r-1} - \Lambda_2 S_{r-2} - ... - \Lambda_\nu S_{r-\nu}. \qquad (7.11)$$

Можно интерпретировать это соотношение как предсказание величины следующей компоненты синдрома по предыдущим. Если коэффициенты предсказания $\Lambda_1, \Lambda_2, ..., \Lambda_\nu$ известны, то по ним последовательным применением рекуррентного соотношения (7.11) из компонент синдрома $S_1, ..., S_\nu$ получаются компоненты $S_{\nu+1}, ..., S_{2\nu}$. Устройство, реализующее (7.11), в цифровой обработке сигналов называют рекурсивным фильтром. Таким образом, задача вычисления коэффициентов многочлена локаторов ошибок сводится к зада-

Таблица 7.1. Декодирование РС-кода с исправлением трех ошибок с помощью алгоритма Питерсона–Горенстейна–Цирлера

Порождающий многочлен $g(x) = \alpha^6 + \alpha^9 x + \alpha^6 x^2 + \alpha^4 x^3 + \alpha^{14} x^4 + \alpha^{10} x^5 + x^6$	
Кодовое слово $c(x) = \alpha + \alpha^3 x + \alpha^{10} x^2 + \alpha^{12} x^3 + \alpha^{13} x^4 + \alpha^3 x^5 + \alpha^9 x^6 + \alpha^5 x^7 + +x^8 + \alpha^{10} x^9 + \alpha^8 x^{10} + \alpha x^{11} + \alpha x^{12} + \alpha^{13} x^{13} + \alpha^2 x^{14}$	
Выход канала $v(x) = \alpha + \alpha^3 x + \alpha^7 x^2 + \alpha^{12} x^3 + \alpha^{13} x^4 + \alpha^{14} x^5 + \alpha^9 x^6 + \alpha^5 x^7 + +x^8 + \alpha^{10} x^9 + \alpha^8 x^{10} + \alpha x^{11} + \alpha^{10} x^{12} + \alpha^{13} x^{13} + \alpha^2 x^{14}$	
Синдромный многочлен	$S(x) = \alpha^8 + \alpha^2 x + \alpha^{10} x^2 + x^3 + \alpha^{12} x^5$
Система уравнений для коэффициентов многочлена локаторов ошибок	$\begin{pmatrix} \alpha^8 & \alpha^2 & \alpha^{10} \\ \alpha^2 & \alpha^{10} & 1 \\ \alpha^{10} & 1 & 0 \end{pmatrix} \begin{pmatrix} \Lambda_3 \\ \Lambda_2 \\ \Lambda_1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ \alpha^{12} \end{pmatrix}$
Многочлен локаторов ошибок	$\Lambda(x) = 1 + \alpha^{13} x + \alpha^5 x^2 + \alpha^4 x^3$
Локаторы ошибок	$\alpha^2, \alpha^5, \alpha^{12}$
Система уравнений для значений ошибок	$\begin{pmatrix} \alpha^2 & \alpha^5 & \alpha^{12} \\ \alpha^4 & \alpha^{10} & \alpha^9 \\ \alpha^6 & 1 & \alpha^6 \end{pmatrix} \begin{pmatrix} Y_1 \\ Y_2 \\ Y_3 \end{pmatrix} = \begin{pmatrix} \alpha^8 \\ \alpha^2 \\ \alpha^{10} \end{pmatrix}$
Значения ошибок	$\alpha^6, 1, \alpha^8$
Вектор ошибок	$e(x) = \alpha^6 x^2 + x^5 + \alpha^8 x^{12}$

че построения кратчайшего рекурсивного фильтра по его выходной последовательности. Фильтр, соответствующий уравнению (7.11), показан на рис. 7.1. Отличие рассматриваемой задачи от обычной задачи построения рекурсивного фильтра состоит в том, что коэффициенты фильтра и наблюдаемая последовательность принимают значения в конечном поле.

Для построения фильтра надо найти его длину L и коэффициенты полинома $\Lambda(x)$. Фильтр строится шаг за шагом. Начиная с $r = 1$, строим фильтр, который порождает последовательность $S_1, ..., S_\nu$. На r-й итерации, имея уже построенный на предыдущей

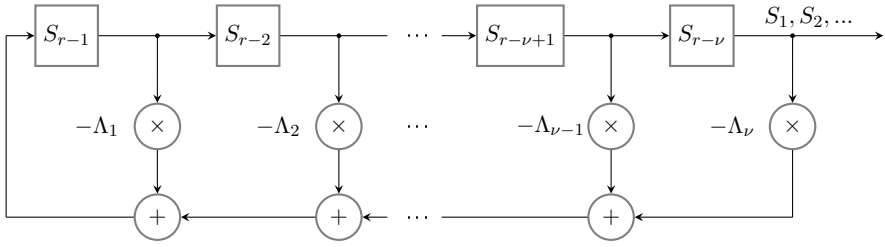


Рис. 7.1. Интерпретация вычисления многочлена локаторов ошибок как построения БИХ-фильтра

итерации фильтр $\Lambda^{(r-1)}(x)$, вычисляем очередное значение

$$\hat{S}_r = -\sum_{j=1}^{r-1} \Lambda_j^{(r-1)} S_{r-j}$$

(для упрощения записи мы игнорируем тот факт, что порядок фильтра, порождающего последовательность длины r, может быть меньше r). Это предсказанное значение $\hat{S}_r$ может не совпадать с требуемым значением S_r. В этом случае вычисляется так называемая *невязка* (ошибка предсказания)

$$\Delta_r = S_r - \hat{S}_r = S_r + \sum_{j=1}^{r-1} \Lambda_j^{(r-1)} S_{r-j} = \sum_{j=0}^{r-1} \Lambda_j^{(r-1)} S_{r-j}.$$

Если невязка равна нулю, итерация выполнена успешно, $\Lambda^{(r)}(x) = \Lambda^{(r-1)}(x)$. Если же невязка не равна нулю, то ранее вычисленные коэффициенты должны быть подправлены, чтобы сделать ее нулевой, сохранив нулевыми все предыдущие невязки. Положим

$$\Lambda^{(r)}(x) = \Lambda^{(r-1)}(x) + Ax^{r-m}\Lambda^{(m-1)}(x), \tag{7.12}$$

где A — элемент поля, а $\Lambda^{(m-1)}(x)$ — один из многочленов, полученных на предыдущих итерациях. Параметры A и m нужно подобрать так, чтобы невязка стала равной нулю.

Выберем $m < r$ таким, что $\Delta_m = \sum_{j=0}^{r-1} \Lambda_j^{(m-1)} S_{r-j-l} \neq 0$.

Заметим, что умножение на x^{r-m} в (7.12) соответствует сдвигу содержимого регистра сдвига «вперед» на $r-m$ тактов. Поэтому фильтр $x^{r-m}\Lambda^{(m-1)}(x)$, соответствующий второму слагаемому в

(7.12), до такта с номером $r-1$ получает на вход те же значения, что поступали на фильтр $\Lambda^{(m-1)}(x)$ до такта $m-1$. Невязка на этих шагах была нулевой, следовательно, вклад второго слагаемого до такта с номером $r-1$ равен нулю, а на такте r вклад равен $A\Delta_m$. Положим $l = r - m$, $A = -\Delta_r/\Delta_m$.

Тогда новая невязка равна

$$\Delta'_r = \Delta_r - \frac{\Delta_r}{\Delta_m}\Delta_m = 0.$$

Тем самым мы убедились в том, что модифицированный в соответствии с (7.12) фильтр порождает правильную последовательность синдромов.

Подсчитаем длину L_r фильтра Λ_r, получаемого на шаге r. Эта длина, конечно, зависит от выбора параметра m. Для построения фильтра, порождающего нужную последовательность, можно выбрать любое m, при котором $\Delta_m \neq 0$, но длина фильтра зависит от m, а решаемая задача — построение фильтра минимально возможной длины (длина фильтра равна весу предполагаемой комбинации ошибок).

Порядок L_r фильтра Λ_r равен степени многочлена в правой части (7.12), т.е.

$$L_r = \max\{L_{r-1}, r - m + L_{m-1}\}. \tag{7.13}$$

Отсюда видим, что порядок фильтра либо не меняется, либо растет от шага к шагу. Выберем m равным номеру последнего шага, на котором увеличился порядок фильтра. Следующая лемма показывает, насколько может увеличиться порядок фильтра за одну итерацию.

Лемма 7.2. *Пусть на шаге r невязка $\Delta_r \neq 0$, и предыдущее увеличение порядка фильтра произошло на шаге m при невязке $\Delta_m \neq 0$. Тогда фильтр, порождающий $S_1, S_2, ..., S_r$, задается полиномом*

$$\Lambda^{(r)}(x) = \Lambda^{(r-1)}(x) - \frac{\Delta_r}{\Delta_m}x^{r-m}\Lambda^{(m-1)}(x) \tag{7.14}$$

и имеет порядок

$$L_r = \max\{L_{r-1}, r - L_{r-1}\}. \tag{7.15}$$

Доказательство. Тождество (7.14) уже доказано, осталось доказать (7.15). Для этого воспользуемся индукцией по r, причем интересны только те шаги, на которых увеличивается порядок фильтра.

Положив в качестве начального значения $L_0 = 0$, при появлении первого ненулевого элемента S_j в последовательности $S_1, S_2, ..., S_j, ...$ порядок фильтра, порождающего $0, 0, ..., S_j$, равен j (см. лемму 7.11 в приложении), т.е. удовлетворяет (7.15). Тем самым, мы проверили справедливость леммы для начального шага индукции.

Предположим теперь, что на всяком шаге j, когда увеличивалась длина фильтра, имело место равенство $L_j = j - L_{j-1}$. В частности, на шаге с номером $j = m$ длина фильтра увеличилась и, по предположению леммы, стала равной $L_m = L_{r-1}$, следовательно,

$$L_m = m - L_{m-1} = L_{r-1}.$$

Подстановка этого соотношения в (7.13) приводит к (7.15). □

Формальное описание алгоритма, который вытекает из доказанных утверждений, сформулируем в виде теоремы.

Теорема 7.3. Алгоритм Берлекэмпа–Месси. *Пусть заданы* $S_1, S_2, ..., S_{2t}$ *и начальные условия* $\Lambda^{(0)}(x) = 1$, $B^{(0)}(x) = 1$, $L_0 = 0$. *При* $r = 1, 2, ..., 2t$ *рекуррентно вычислим*

$$\Delta_r = \sum_{j=0}^{r-1} \Lambda_j^{(r-1)} S_{r-j}; \tag{7.16}$$

$$L_r = \delta_r(r - L_{r-1}) + (1 - \delta_r)L_{r-1}; \tag{7.17}$$

$$\begin{pmatrix} \Lambda^{(r)}(x) \\ B^{(r)}(x) \end{pmatrix} = \begin{pmatrix} 1 & -\Delta_r x \\ \Delta_r^{-1}\delta_r & (1-\delta_r)x \end{pmatrix} \begin{pmatrix} \Lambda^{(r-1)}(x) \\ B^{(r-1)}(x) \end{pmatrix}, \tag{7.18}$$

где $\delta_r = 1$, *если одновременно* $\Delta_r \neq 0$ *и* $2L_{r-1} \leq r - 1$, *и* $\delta_r = 0$ *в противном случае. Тогда* $\Delta^{(2t)}(x)$ *является многочленом наименьшей степени, коэффициенты которого удовлетворяют равенствам* $\Lambda_0^{(2t)} = 1$ *и*

$$S_r + \sum_{j=1}^{r-1} \Lambda_j^{(2t)} S_{r-j} = 0, \quad r = L_{2t} + 1, ..., 2t.$$

Алгоритм 7.2. Алгоритм Берлекэмпа–Месси

```
Вычисление коэффициентов многочлена локаторов ошибок
по алгоритму Берлекэмпа-Месси
Input: Синдромный многочлен S(x)
Output: Многочлен локаторов Λ(x)
1. Инициализация
L = 0; % Текущая длина регистра
Λ(x) = 1; % Многочлен локаторов
B(x) = 1; % Многочлен компенсации невязки
2. Основной цикл
for r = 1 to d − 1 do
    Δ = Σ_{j=0}^{L} Λ_j S_{r−j}; % Невязка
    B(x) = xB(x); % Сдвиг
    if Δ ≠ 0 then
        % ЛРОС модифицируется
        T(x) = Λ(x) − ΔB(x); % Вспомогательный многочлен
        if 2L ≤ r − 1 then
            % Длина увеличивается
            B(x) = Δ^{−1}Λ(x);
            L = r − L;
        end
        Λ(x) = T(x);
    end
end
3. Формирование результата
if deg Λ(x) == L then
    Многочлен локаторов = Λ(x);
end
else
    Число ошибок больше t;
end
```

Обсуждение. В вычислениях по формуле (7.18) присутствует обратный элемент к Δ_r, который не определен при $\Delta_r = 0$. Однако этот обратный элемент востребован только при $\delta_r = 0$. В этом

случае принимаем $\Delta_r^{-1}\delta_r = 0$. Заметим также, что переменная δ_r играет роль переключателя: в зависимости от ее значения изменяется способ вычисления «модифицирующего многочлена» $B^{(r)}(x)$, который, по сути, представляет собой второе слагаемое в (7.14). Заметим, что выражение $\Delta_r^{-1}\Lambda^{(r-1)}(x)$ достаточно пересчитывать только при изменении длины регистра, а сам регистр модифицируется всякий раз, когда невязка отлична от нуля.

Алгоритм 7.2 представляет собой псевдокод программы, реализующей формулы теоремы 7.3. Длина регистра L неявно участвует во всех операциях с полиномами, т.к. определяет число нетривиальных элементов в соответствующих массивах.

Доказательство. Все соотношения в формулировке теоремы уже доказаны, за исключением того, что построенный ЛРОС является кратчайшим среди всех ЛРОС, порождающих заданную последовательность синдромных компонент. В приложении к главе мы обсудим подробнее задачу построения регистра, порождающего заданную последовательность, и докажем оптимальность алгоритма БМ. □

7.3. Алгоритм Форни

В предыдущем параграфе мы рассмотрели быстрый алгоритм нахождения локаторов ошибок. Быстрым он является в том смысле, что решает задачу нахождения решения системы из t линейных уравнений за число операций порядка t^2, вместо числа t^3, которое потребовалось бы для системы уравнений общего вида. Чтобы декодирование в целом имело сложность порядка t^2, нужно упростить решение системы уравнений (7.8) для нахождения значений ошибок по известным локаторам. Эта задача была решена Форни [70] (1965). Для описания алгоритма введем многочлен значений ошибок

$$\Omega(x) = S(x)\Lambda(x) \qquad \bmod x^{2t}. \tag{7.19}$$

Теорема 7.4. *Имеет место соотношение*

$$\Omega(x) = \sum_{i=1}^{\nu} Y_i X_i \prod_{j\neq i}(1 - X_j x). \tag{7.20}$$

Доказательство. Напомним обозначения:

$$\begin{aligned} S(x) &= \sum_{j=1}^{2t} S_j x^{j-1} = \sum_{j=1}^{2t} \sum_{i=1}^{\nu} Y_i X_i^j x^{j-1}; \\ \Lambda(x) &= \prod_{j=1}^{\nu} (1 - X_j x). \end{aligned}$$

Подставим эти выражения в (7.19). Получим

$$\Omega(x) = \left[\sum_{j=1}^{2t} \sum_{i=1}^{\nu} Y_i X_i^j x^{j-1} \right] \left[\prod_{j=1}^{\nu} (1 - X_j x) \right]. \qquad (7.21)$$

Следующий шаг использует тождество

$$\frac{1 - a^n}{1 - a} = 1 + a + ... + a^{n-1}.$$

При $a = X_i x$ и $n = 2t$

$$\sum_{j=1}^{2t} X_i^{j-1} x^{j-1} = \frac{1 - X_i^{2t} x^{2t}}{1 - X_i x}.$$

Подставив в (7.21), получим

$$\Omega(x) = \sum_{i=1}^{\nu} Y_i X_i \left(1 - X_i^{2t} x^{2t}\right) \prod_{j \neq i} (1 - X_j x).$$

После приведения по модулю x^{2t} получаем требуемый результат. □

Теорема 7.5. *Алгоритм Форни.*

$$Y_i = \frac{X_i^{-1} \Omega(X_i^{-1})}{\prod_{j \neq i} \left(1 - X_j X_i^{-1}\right)}. \qquad (7.22)$$

Доказательство. В сумме в правой части (7.20) при подстановке $x = X_i^{-1}$ остается только одно ненулевое слагаемое. Поэтому из Теоремы 7.4 имеем

$$\Omega(X_i^{-1}) = Y_i X_i \prod_{j \neq i} \left(1 - X_j X_i^{-1}\right).$$

Отсюда следует (7.22). □

Таким образом, теорема 7.5 дает нам формулу для прямого вычисления значений ошибок. Сложность непосредственного вычисления каждого значения пропорциональна степени многочлена значений, следовательно, общая сложность алгоритма имеет порядок t^2.

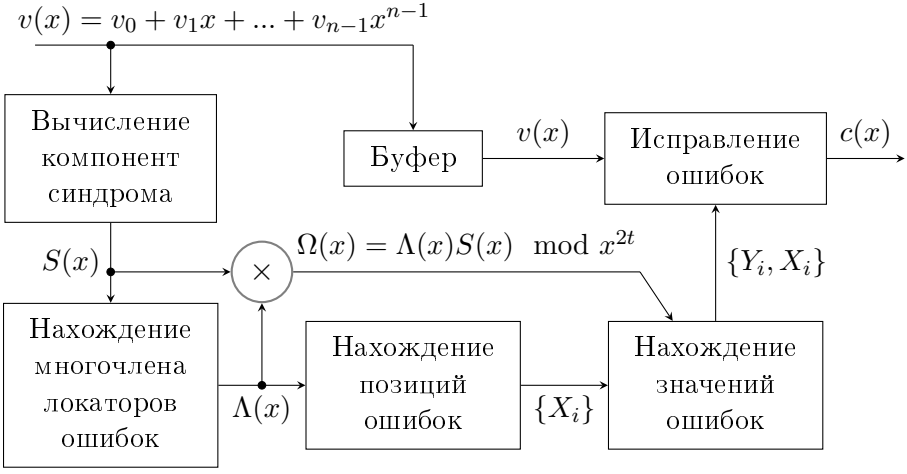


Рис. 7.2. Блок-схема декодера БЧХ-кода

Полная схема декодера БЧХ-кода приведена на рис. 7.2. Ее входом служит последовательность $v(x) = v_0 + v_1 x + ... + v_{n-1} x^{n-1}$ на выходе канала. Она хранится в буфере до тех пор, пока не будут выполнены основные этапы декодирования: вычисление компонент синдрома, многочлена локаторов ошибок, значений ошибок и локаторов ошибок. В случае двоичных кодов блок нахождения значений ошибок не нужен. Выходом блока исправления ошибок служат информационные символы исправленного кодового слова либо сигнал о неисправимой ошибке.

Блок-схема алгоритма работы модуля нахождения многочлена локаторов ошибок показана на рис. 7.2. Алгоритм в точности следует утверждению теоремы 7.3. Участвующий в теореме и алгорит-

ме полином коррекции невязки $B(x)$ хранит последнюю ненулевую невязку, на каждом шаге «сдвигая» ее умножением на x. После того как снова появляется ненулевая невязка, содержимое $B(x)$ обновляется. Выходом алгоритма является либо вычисленный многочлен локаторов, либо сообщение о том, что число ошибок превышает корректирующую способность кода.

Упражнение 7.1. Тот же код и тот же вектор ошибок, что и в примере 7.1, декодируем по алгоритму Берлекэмпа–Месси. Все промежуточные вычисления представлены в табл. 7.2.

7.4. Исправление ошибок и стираний

Как было объяснено в главе 1, эффективность кодирования намного повышается при использовании в декодере информации о надежности символов. Простейшей формой использования такой информации является стирание ненадежных символов. Манипулируя порогом стирания и многократно декодируя при различных значениях порога, можно добиться эффективности декодирования, близкой к декодированию по максимуму правдоподобия. Такое декодирование называют декодированием по минимуму обобщенного расстояния [71, 18] (Форни, 1966). Точная формулировка алгоритма будет приведена в параграфе 7.5.

Упражнение 7.2. Докажите, что код с минимальным расстоянием d гарантированно исправляет комбинации из ν ошибок и f стираний при условии

$$2\nu + f + 1 \leq d. \tag{7.23}$$

Подсказка: нужно убедиться в том, что расстояние Хэмминга, подсчитанное по нестертым позициям, является метрикой, а затем воспользоваться неравенством треугольника.

В случае двоичных БЧХ-кодов такая корректирующая способность достигается двукратным декодированием с исправлением только ошибок. Для этого при первом декодировании нужно положить все стертые позиции равными нулю, а при втором — единице. В одной из двух попыток декодирования на f стертых позициях будет не больше $f/2$ ошибок. Условием правильного декодирования будет $2(f/2 + v) + 1 \leq d$, что эквивалентно (7.23).

Таблица 7.2. Декодирование РС-кода с исправлением 3 ошибок с помощью алгоритма Берлекэмпа–Месси

<table>
<tr><td colspan="5">Порождающий многочлен
$g(x) = \alpha^6 + \alpha^9 x + \alpha^6 x^2 + \alpha^4 x^3 + \alpha^{14} x^4 + \alpha^{10} x^5 + x^6$</td></tr>
<tr><td colspan="5">Кодовое слово
$c(x) = \alpha + \alpha^3 x + \alpha^{10} x^2 + \alpha^{12} x^3 + \alpha^{13} x^4 + \alpha^3 x^5 + \alpha^9 x^6 + \alpha^5 x^7 +$
$+ x^8 + \alpha^{10} x^9 + \alpha^8 x^{10} + \alpha x^{11} + \alpha x^{12} + \alpha^{13} x^{13} + \alpha^2 x^{14}$</td></tr>
<tr><td colspan="5">Выход канала
$v(x) = \alpha + \alpha^3 x + \alpha^7 x^2 + \alpha^{12} x^3 + \alpha^{13} x^4 + \alpha^{14} x^5 + \alpha^9 x^6 + \alpha^5 x^7 +$
$+ x^8 + \alpha^{10} x^9 + \alpha^8 x^{10} + \alpha x^{11} + \alpha^{10} x^{12} + \alpha^{13} x^{13} + \alpha^2 x^{14}$</td></tr>
<tr><td colspan="5">Синдромный многочлен $S(x) = \alpha^8 + \alpha^2 x + \alpha^{10} x^2 + x^3 + \alpha^{12} x^5$</td></tr>
<tr><td>r</td><td>Δ</td><td>$B(x)$</td><td>$\Lambda(x)$</td><td>L</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>α^8</td><td>α^7</td><td>$1 + \alpha^8 x$</td><td>1</td></tr>
<tr><td>2</td><td>α^5</td><td>$\alpha^7 x$</td><td>$1 + \alpha^9 x$</td><td>1</td></tr>
<tr><td>3</td><td>α^{14}</td><td>$\alpha + \alpha^{10} x$</td><td>$1 + \alpha^9 x + \alpha^6 x^2$</td><td>2</td></tr>
<tr><td>4</td><td>α^{10}</td><td>$\alpha x + \alpha^{10} x^2$</td><td>$1 + \alpha^2 x + \alpha^9 x^2$</td><td>2</td></tr>
<tr><td>5</td><td>α^{10}</td><td>$\alpha^5 + \alpha^7 x + \alpha^{14} x^2$</td><td>$1 + \alpha^2 x + \alpha^2 x^2 + \alpha^5 x^3$</td><td>3</td></tr>
<tr><td>6</td><td>α^9</td><td>$\alpha^5 x + \alpha^7 x^2 + \alpha^{14} x^3$</td><td>$1 + \alpha^{13} x + \alpha^5 x^2 + \alpha^4 x^3$</td><td>3</td></tr>
<tr><td colspan="5">Многочлен локаторов ошибок $\Lambda(x) = 1 + \alpha^{13} x + \alpha^5 x^2 + \alpha^4 x^3$</td></tr>
<tr><td colspan="5">Локаторы ошибок $\alpha^2, \alpha^5, \alpha^{12}$</td></tr>
<tr><td colspan="5">Многочлен значений ошибок
$S(x)\Lambda(x) = \alpha^8 + \alpha^3 x + \alpha^7 x^2 + \alpha^2 x^6 + \alpha^2 x^7 + \alpha x^8$
$\Omega(x) = S(x)\Lambda(x)(\mod x^6) = \alpha^8 + \alpha^3 x + \alpha^7 x^2$</td></tr>
<tr><td colspan="5">Значения ошибок $\alpha^6, 1, \alpha^8$</td></tr>
<tr><td colspan="5">Вектор ошибок $e(x) = \alpha^6 x^2 + x^5 + \alpha^8 x^{12}$</td></tr>
</table>

В случае недвоичных кодов БЧХ или кодов РС такой метод не годится, его непосредственное применение потребовало бы перебора по множеству значений ошибок на стертых позициях. К счастью, как мы увидим ниже, небольшая модификация описанных выше декодеров ПГЦ и БМ позволяет декодировать ошибки и стирания с вычислительной сложностью примерно того же порядка, что и сложность декодирования с исправлением только ошибок.

Рассмотрим БЧХ-код, заданный определением 6.1, при $b = 1$. Запишем кодовое слово, принятую последовательность и вектор ошибок в виде полиномов

$$\begin{aligned} c(x) &= c_0 + c_1 x + ... + c_{n-1} x^{n-1}; \\ v(x) &= v_0 + v_1 x + ... + v_{n-1} x^{n-1}; \\ e(x) &= e_0 + e_1 x + ... + e_{n-1} x^{n-1}. \end{aligned}$$

Поскольку позиции стираний известны, при вычислении позиций ошибок стертые позиции можно игнорировать (положить равными нулю). Соответствующие последовательности обозначаем как $c'(x)$, $v'(x)$ и $e'(x)$, а соответствующие компоненты (модифицированного) синдрома — как S'_i, $i = 1, 2, ..., d-1$. Модифицированным синдромом мы назвали синдром, вычисленный по нестертым позициям принятой последовательности.

Пусть $i_1, i_2, ..., i_f$ — номера позиций стираний, и $U_j = \alpha^{i_j}$, $j = 1, 2, ..., f$. По аналогии с полиномом локаторов ошибок $\Lambda(x)$ вводим в рассмотрение полином локаторов стираний

$$\Psi(x) = \prod_{j=1}^{f} (1 - xU_j). \tag{7.24}$$

Нам потребуется также полином локаторов ошибок и стираний, который вычисляется как

$$\tilde{\Lambda}(x) = \Psi(x)\Lambda(x). \tag{7.25}$$

Очевидно, зная $\tilde{\Lambda}(x)$, можно построить многочлен значений ошибок по формуле (7.19) и затем по алгоритму Форни найти значения ошибок. Многочлен локаторов стираний известен, следовательно, достаточно найти второй сомножитель в (7.25). Можно поступить иначе, предположив, что ошибок было $\nu + f$, но для f из них локаторы уже найдены. Тогда для нахождения оставшихся локаторов можно воспользоваться теоремой БМ, заменив начальные условия $\Lambda^{(0)}(x) = B^{(0)}(x) = 1$ на $\Lambda^{(0)}(x) = B^{(0)}(x) = \Psi(x)$. Заметим, что вычисление полинома локаторов стираний по формуле (7.24) также можно организовать в рекуррентной форме. Полностью процедура исправления ошибок и стираний представлена в виде алгоритма. 7.3.

Алгоритм 7.3. Алгоритм Берлекэмпа–Месси для исправления ошибок и стираний

Вычисление многочлена локаторов ошибок и стираний по алгоритму Берлекэмпа-Месси
Input: Синдромный многочлен $S(x)$;
Число стираний f;
Локаторы стираний $U_1, ..., U_f$;
Output: Многочлен локаторов $\Lambda(x)$
1. Инициализация
$L = f$; % Текущая длина регистра
$\Lambda(x) = 1$; % Многочлен локаторов
$B(x) = 1$; % Многочлен компенсации невязки
2. Построение полинома локаторов стираний $\Psi(x)$
for $r = 1$ **to** f **do**
  $\Lambda(x) = \Lambda(x) - U_r x \Lambda(x)$
end
$B(x) = \Lambda(x)$;
for $r = f + 1$ **to** $d - 1$ **do**
  $\Delta = \sum_{j=0}^{L} \Lambda_j S_{r-j}$; % Невязка
  $B(x) = xB(x)$; % Сдвиг
  if $\Delta \neq 0$ **then**
    % ЛРОС модифицируется
    $T(x) = \Lambda(x) - \Delta B(x)$; % Вспомогательный многочлен
    if $2L \leq r + f - 1$ **then**
      % Длина увеличивается
      $B(x) = \Delta^{-1}\Lambda(x)$;
      $L = r - L + f$;
    end
    $\Lambda(x) = T(x)$;
  end
end
4. Формирование результата
if $\deg \Lambda(x) == L$ **then**
  Многочлен локаторов ошибок и стираний $= \Lambda(x)$.
end
else
  Отказ от декодирования.
end

Упражнение 7.3. Снова рассмотрим РС-код (15,9) над полем $GF(2^4)$. Пример декодирования комбинации ошибок кратности 2 при наличии двух стираний подробно разобран в табл. 7.3.

Таблица 7.3. Декодирование РС-кода с исправлением 2 ошибок и 2 стираний с помощью алгоритма Берлекэмпа–Месси

<table>
<tr><td colspan="5">Порождающий многочлен
$g(x) = \alpha^6 + \alpha^9 x + \alpha^6 x^2 + \alpha^4 x^3 + \alpha^{14} x^4 + \alpha^{10} x^5 + x^6$</td></tr>
<tr><td colspan="5">Кодовое слово
$c(x) = \alpha + \alpha^3 x + \alpha^{10} x^2 + \alpha^{12} x^3 + \alpha^{13} x^4 + \alpha^3 x^5 + \alpha^9 x^6 + \alpha^5 x^7 +$
$+ x^8 + \alpha^{10} x^9 + \alpha^8 x^{10} + \alpha x^{11} + \alpha x^{12} + \alpha^{13} x^{13} + \alpha^2 x^{14}$</td></tr>
<tr><td colspan="5">Выход канала
$v(x) = \alpha + \alpha^3 x + \alpha^7 x^2 + \alpha^{12} x^3 + \emptyset x^4 + \alpha^{14} x^5 + \alpha^9 x^6 + \alpha^4 x^7 +$
$+ x^8 + \alpha^{10} x^9 + \emptyset x^{10} + \alpha x^{11} + \alpha x^{12} + \alpha^{13} x^{13} + \alpha^2 x^{14}$
Символ $\emptyset$ обозначает стирание</td></tr>
<tr><td colspan="5">Локаторы стираний α^4, α^{10}</td></tr>
<tr><td colspan="5">Синдромный многочлен $S(x) = \alpha^{12} x + x^2 + \alpha^6 x^3 + \alpha^{11} x^4 + \alpha^{10} x^6$</td></tr>
<tr><td colspan="5">Полином локаторов стираний
$\Psi(x) = (1 + \alpha^4 x)(1 + \alpha^{10} x) = 1 + \alpha^2 x + \alpha^{14} x^2$</td></tr>
<tr><td>r</td><td>Δ</td><td>$B(x)$</td><td>$\Lambda(x)$</td><td>L</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>$1 + \alpha^4 x$</td><td>$1 + \alpha^4 x$</td><td>1</td></tr>
<tr><td>2</td><td>0</td><td>$1 + \alpha^2 x + \alpha^{14} x^2$</td><td>$\Psi(x) = 1 + \alpha^2 x + \alpha^{14} x^2$</td><td>2</td></tr>
<tr><td>3</td><td>α^5</td><td>$\alpha^{10} + \alpha^{12} x + \alpha^9 x^2$</td><td>$1 + \alpha x + \alpha x^2 + \alpha^4 x^3$</td><td>3</td></tr>
<tr><td>4</td><td>α^8</td><td>$\alpha^{10} x + \alpha^{12} x^2 + \alpha^9 x^3$</td><td>$1 + \alpha^9 x + \alpha^2 x^2 + \alpha^{10} x^3$</td><td>3</td></tr>
<tr><td>5</td><td>α^2</td><td>$\alpha^{13} + \alpha^7 x + x^2 + \alpha^8 x^3$</td><td>$1 + \alpha^9 x + \alpha^7 x^2 + \alpha^{11} x^3 + \alpha^{11} x^4$</td><td>4</td></tr>
<tr><td>6</td><td>α^{10}</td><td>$\alpha^{13} x + \alpha^7 x^2 + x^3 + \alpha^8 x^4$</td><td>$1 + \alpha^5 x + \alpha^9 x^2 + \alpha^7 x^3 + \alpha^6 x^4$</td><td>4</td></tr>
<tr><td colspan="5">Многочлен локаторов ошибок и стираний
$\Lambda(x) = 1 + \alpha^5 x + \alpha^9 x^2 + \alpha^7 x^3 + \alpha^6 x^4$</td></tr>
<tr><td colspan="5">Локаторы ошибок и стираний $\alpha^2, \alpha^4, \alpha^5, \alpha^{10}$</td></tr>
<tr><td colspan="5">Многочлен значений ошибок
$S(x)\Lambda(x) = \alpha^{12} + \alpha^8 x + \alpha^5 x^2 + \alpha^{14} x^3 + \alpha^5 x^6 + \alpha^{10} x^7 + \alpha^2 x^8 + \alpha x^9$
$\Omega(x) = S(x)\Lambda(x) (\mod x^6) = \alpha^{12} x + \alpha^8 x^2 + \alpha^5 x^3 + \alpha^{14} x^4$</td></tr>
<tr><td colspan="5">Значения ошибок $\alpha^6, 1$</td></tr>
<tr><td colspan="5">Значения стираний α^{13}, α^8</td></tr>
<tr><td colspan="5">Вектор ошибок $e(x) = \alpha^6 x^2 + x^5$</td></tr>
<tr><td colspan="5">Вектор стираний $e(x) = \alpha^{13} x^4 + \alpha^8 x^{10}$</td></tr>
</table>

7.5. Декодирование по минимуму обобщенного расстояния

В предыдущих главах мы не раз подчеркивали, что декодирование с мягкими решениями, т.е. с учетом оценок надежности принимаемых символов, намного эффективнее декодирования с жесткими решениями. Очевидный недостаток рассмотренных выше алгоритмов декодирования БЧХ- и РС-кодов состоит в том, что они не могут быть непосредственно применены для декодирования с мягкими решениями. Наиболее практичным среди известных методов учета надежности входных символов при сохранении преимуществ алгебраического декодирования кодов является декодирование по критерию минимума обобщенного расстояния [71, 18].

Цель метода — сведение декодирования в канале с мягкими решениями к многократным попыткам алгебраического декодирования с исправлением ошибок и стираний. В двух словах суть алгоритма в том, что сначала выполняется обычное декодирование без стираний, с исправлением максимально возможного числа ошибок, затем декодирование с двумя стертыми наименее надежными символами, с четырьмя, и т.д. вплоть до $d-1$ стертых символов. Из всех результатов декодирования выбирается наиболее близкий к полученной из канала последовательности.

Напомним некоторые обозначения, использованные в главе 4.

Предположим, что символы двоичного (n,k)-кода $C=\{\boldsymbol{c}_m, m=1,...,2^k\}$ передаются по каналу с гауссовским шумом. Двоичные кодовые слова $\boldsymbol{c}_m$ преобразуются в кодовые слова $\boldsymbol{\xi}_m$ из евклидова пространства заменой нулей и единиц на вещественные числа -1 и $+1$ соответственно.

Предположим теперь, что значения последовательности $\boldsymbol{\alpha}=(\alpha_1,\alpha_2,...,\alpha_n)$ на выходе демодулятора (на входе декодера) нормированы и ограничены так, что $|\alpha_i|\le 1$, $i=1,2,...,n$.

Знаки компонент $\boldsymbol{\alpha}$ представляют собой жесткие решения, а их модули — надежности принятых символов. Если α_i вычислены как логарифмы отношения правдоподобия, то, как было показано в параграфе 4.1, в отсутствие ограничения на значения компонент декодирование по максимуму скалярного произведения $(\boldsymbol{\alpha},\boldsymbol{\xi}_m)$ рав-

носительно декодированию по максимуму правдоподобия. Введение нормировки не влияет на результат декодирования, зато дает возможность доказать следующую теорему.

Теорема 7.6. *В коде с длиной n и минимальным расстоянием d найдется не больше одного слова $\boldsymbol{\xi}_m$ такого, что*

$$(\boldsymbol{\alpha}, \boldsymbol{\xi}_m) > n - d, \tag{7.26}$$

если $|\alpha_i| \leq 1$, $i = 1, 2, ..., n$.

Доказательство. Найдем слово $\boldsymbol{\xi}_{m'}$, отличающееся от $\boldsymbol{\xi}_m$ ровно в d позициях, и обозначим через S множество позиций, в которых знаки $\boldsymbol{\xi}_m$ и $\boldsymbol{\xi}_{m'}$ не совпадают, $|S| = d$. Положим

$$\begin{aligned} A &= \sum_{i \notin S} \alpha_i \xi_{mi}; \\ B &= \sum_{i \in S} \alpha_i \xi_{mi}. \end{aligned}$$

Заметим, что $A \leq n - d$, поскольку $|S^{\mathrm{c}}| = n - d$ и по условию теоремы каждое слагаемое меньше единицы. Для двух скалярных произведений имеем

$$\begin{aligned} (\boldsymbol{\alpha}, \boldsymbol{\xi}_m) &= A + B; \\ (\boldsymbol{\alpha}, \boldsymbol{\xi}_{m'}) &= A - B. \end{aligned}$$

Их сумма

$$(\boldsymbol{\alpha}, \boldsymbol{\xi}_m) + (\boldsymbol{\alpha}, \boldsymbol{\xi}_{m'}) = 2A \leq 2(n - d).$$

Отсюда следует, что только одно из двух слагаемых может быть строго больше $n - d$. □

Определение 7.1. *Декодирование по минимуму евклидова расстояния между вектором $\boldsymbol{\alpha}$ и кодовыми словами $\boldsymbol{\xi}_m$ называется декодированием по минимуму обобщенного расстояния (МОР).*

Упражнение 7.4. Докажите, что декодирование по МОР эквивалентно декодированию по максимуму скалярного произведения $(\boldsymbol{\alpha}, \boldsymbol{\xi}_m)$.

Подсказка: рассмотрите квадрат нормы $\|\boldsymbol{\alpha} - \boldsymbol{\xi}_m\|^2$.

Упражнение 7.5. Докажите, что декодирование в ДСК по минимуму расстояния Хэмминга и декодирование в ДСтК по минимуму расстояния, вычисленного по нестертым позициям, являются частными случаями декодирования по МОР. Запишите аналоги неравенства (7.26) для этих частных случаев.

Следующая теорема подсказывает способ нахождения единственного решения $\boldsymbol{\xi}_m$ неравенства (7.26), если такое решение существует.

Теорема 7.7. *Если для некоторого m имеет место неравенство $(\boldsymbol{\alpha}, \boldsymbol{\xi}_m) > n-d$, то при некотором $f \in \{0, 1, ..., d-1\}$ при стирании f наименее надежных позиций декодер, исправляющий f стираний и комбинации ошибок кратности до $t = \lfloor (d-f-1)/2 \rfloor$, вынесет решение в пользу кодового слова $\boldsymbol{\xi}_m$.*

Доказательство. Для того чтобы упростить запись, будем считать, что перед декодированием символы упорядочены по возрастанию надежности, т.е. $|\alpha_1| \le |\alpha_2| \le ... \le |\alpha_n|$. Символы кодовых слов, конечно, должны быть соответствующим образом перенумерованы.

Предположим сначала, что f принимает любые значения из множества $\{0, 1, ..., n\}$. Обозначим

$$\begin{aligned} p_0 &= |\alpha_1|; \\ p_1 &= |\alpha_2| - |\alpha_1|; \\ \dots &\quad \dots \quad \dots; \\ p_{n-1} &= |\alpha_n| - |\alpha_{n-1}|; \\ p_n &= 1 - |\alpha_n|. \end{aligned}$$

Числа $\{p_i\}$ неотрицательны, и их сумма равна единице, что дает возможность рассматривать их как распределение вероятностей.

Введем вспомогательные векторы $\boldsymbol{\beta}^{(f)} = (\beta_1^{(f)}, \beta_2^{(f)}, ..., \beta_n^{(f)})$, где

$$\begin{aligned} \beta_i^{(f)} &= \begin{cases} 0, & i \le f; \\ +1, & i > f, \alpha_i \ge 0; \\ -1, & i > f, \alpha_i < 0, \end{cases} \\ f &= 0, 1, ..., n. \end{aligned}$$

Нетрудно подсчитать, что

$$\boldsymbol{\alpha} = \sum_{f=0}^{n} p_f \boldsymbol{\beta}^{(f)}. \tag{7.27}$$

В то же время, скалярные произведения $(\boldsymbol{\beta}^{(f)}, \boldsymbol{\xi}_m)$ представляют собой метрику декодирования с жесткими решениями при наличии f стираний, и неравенство $(\boldsymbol{\beta}^{(f)}, \boldsymbol{\xi}_m) > n - d$ может выполняться только для одного слова, которое и является решением при данном f. Из (7.27) имеем

$$(\boldsymbol{\alpha}, \boldsymbol{\xi}) = \sum_{f=0}^{n} p_f (\boldsymbol{\beta}^{(f)}, \boldsymbol{\xi}). \tag{7.28}$$

Итак, $(\boldsymbol{\alpha}, \boldsymbol{\xi})$ равно среднему по всем f значению $(\boldsymbol{\beta}^{(f)}, \boldsymbol{\xi})$ при распределении вероятностей $\{p_f\}$ на f. Среднее значение не больше максимального. По условию теоремы среднее значение больше $n-d$, а значит и одно из скалярных произведений в правой части (7.28) больше $n-d$. Следовательно, искомое решение будет вынесено при одном из $f \in \{0, 1, ..., n\}$.

С другой стороны, декодер, исправляющий ошибки и стирания, может выносить решения только при числе стираний, меньшем d, — отсюда следует ограничение на f в формулировке теоремы. □

Отметим, что при четной разности $d-f$ решение декодера будет таким же, как и при на единицу большем числе стираний. Поэтому максимальное число попыток декодирования примерно равно $d/2$.

Кроме того, как только выполнено неравенство $(\boldsymbol{\alpha}, \boldsymbol{\xi}_m) > n - d$, декодирование может быть остановлено. Поэтому среднее число попыток будет заметно меньше числа $d/2$.

Задачи

1. Дана последовательность (0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1) над полем $GF(2)$. Постройте минимальный ЛРОС, генерирующий эту последовательность. Нарисуйте схему генератора. Продлите последовательность еще на 10 двоичных символов.

2. Какие РС-коды можно построить над полями $GF(5)$, $GF(7)$? Приведите пример кода, проверьте на нем работу алгоритмов ПГЦ, БМ.

3. Постройте код Рида–Соломона длины 7, исправляющий двойные ошибки. Приведите пример вектора ошибок веса 2 и выполните декодирование по алгоритмам ПГЦ и БМ.

4. Для кода из задачи 3 приведите пример вектора ошибок веса 1 и вектора стираний веса 2. Примените алгоритм БМ для нахождения кодового слова.

5. Оцените сложность в числе операций в расширении поля для каждого из этапов декодирования при декодировании РС-кодов по алгоритму ПГЦ и БМ.

6. Постройте двоичный БЧХ-код длины 31, исправляющий 3 ошибки. Приведите пример вектора, содержащего 3 ошибки. Выполните декодирование, используя алгоритмы ПГЦ и БМ. Убедитесь в том, что в результате декодирования действительно получено кодовое слово.

7. Используя опыт решения предыдущей задачи, предложите упрощения, возможные при использовании описанных в разделе алгоритмов для декодирования двоичных БЧХ-кодов. Изменится ли асимптотическое поведение сложности декодирования как функции длины кодов и числа исправляемых ошибок для двоичных кодов по сравнению с q-ичными?

8. БЧХ-коды и коды РС могут быть укорочены удалением информационных символов. При этом расстояние кода не уменьшается, но код теряет свойство цикличности. Как воспользо-

ваться описанными в данной главе алгоритмами для декодирования укороченных циклических кодов?

Приложение. Линейная сложность последовательностей

Пусть имеется последовательность $\boldsymbol{s} = (s_1, s_2, ...)$ конечной или бесконечной длины с элементами из конечного поля $GF(q)$, и нужно построить линейную схему, которая ее порождает. Решением задачи может служить линейный регистр с обратной связью (ЛРОС). Его можно описать разностным уравнением, выражающим очередное значение последовательности в виде линейной комбинации предыдущих значений

$$s_{n+1} + c_1 s_n + ... + c_L s_{n-L+1} = 0, \tag{7.29}$$

где L — длина регистра, а вектор $\boldsymbol{c} = (1, c_1, ..., c_L)$ задает весовые коэффициенты умножителей в цепи обратной связи (по аналогии с рис. 7.1, с соответствующей корректировкой обозначений). Альтернативной формой описания регистра служит полином $c(x) = 1 + c_1 x + ... + c_L x^L$.

Если вся последовательность может быть порождена регистром длины L и не может быть порождена более коротким регистром (разностным уравнением более низкого порядка), то L называется *линейной сложностью последовательности.*

Пример 7.2. Простейшее разностное уравнение

$$s_n - c_1 s_{n-1} = 0$$

нетривиально при $c_1 \neq 0$. При начальном состоянии s_1 оно порождает последовательность $(c_1 s_1, c_1^2 s_1, ..., c_1^n s_1, ...)$. Сложность последовательности равна 1, ей соответствует полином $c(x) = 1 - c_1 x$.

Пример 7.3. Разностное уравнение

$$s_n - s_{n-L} = 0$$

(в полиномиальной записи $c(x) = 1 - x^L$) при начальном состоянии

$$(\underbrace{0, 0, ..., 0}_{L-1 \text{ нулей}}\ a)$$

периодически повторяет эту последовательность. Сложность этой бесконечной последовательности равна L и совпадает с ее периодом.

Пример 7.4. Нетрудно проверить, что в поле $GF(2)$ при любом ненулевом начальном состоянии уравнение

$$s_n = s_{n-1} + s_{n-3}$$

(в полиномиальной записи $c(x) = 1 + x + x^3$) порождает последовательность с периодом 7. Это максимально возможный период для последовательностей, задаваемых полиномом третьей степени, поскольку общее число ненулевых состояний регистра равно 7.

В общем случае, если в качестве $c(x)$ выбрать примитивный многочлен степени L, получим на выходе регистра последовательность длины $2^L - 1$, которая представляет собой кодовое слово кода максимальной длины, двойственного коду Хэмминга (см. параграф 5.2). Действительно, выписав соответствующее рекуррентное уравнение, можно заметить, что сдвиги примитивного многочлена $c(x)$ являются проверочными соотношениями для любой последовательности, порождаемой данным регистром. Следовательно, проверочная матрица линейного пространства таких последовательностей совпадает с порождающей матрицей кода Хэмминга.

В этих примерах бесконечная (периодическая) последовательность задавалась полиномом конечной степени. Нетрудно догадаться, что в общем случае при длине регистра L период последовательности не может быть больше, чем $q^L - 1$ (число различных ненулевых состояний регистра).

С другой стороны, любая последовательность длины (периода) n может быть порождена тривиальным РЛОС: достаточно записать последовательность в регистр и соединить выход регистра с его входом.

Таким образом, сложность L последовательности длины (периода) n может находиться в широких пределах от $L = \log_q(n+1)$ до $L = n$.

Рассмотрим задачу нахождения минимального регистра, порождающего заданную последовательность.

Предположим сначала, что L известно и нужно только найти коэффициенты обратной связи (коэффициенты уравнения (7.29)). Задача решается просто. Выпишем (7.29) при $n = 1, 2, ..., 2L - 1$. Запишем полученную систему уравнений в матричном виде

$$\begin{pmatrix} s_1 & s_2 & \cdots & s_L \\ s_2 & s_3 & \cdots & s_{L+1} \\ \vdots & \vdots & \ddots & \vdots \\ s_L & s_{L+1} & \cdots & s_{2L-1} \end{pmatrix} \begin{pmatrix} c_L \\ c_{L-1} \\ \vdots \\ c_1 \end{pmatrix} = \begin{pmatrix} -s_{L+1} \\ -s_{L+2} \\ \vdots \\ -s_{2L} \end{pmatrix}. \qquad (7.30)$$

Если гипотеза о линейной сложности L верна, то существует единственное решение этой системы уравнений, оно и является решением этой задачи. Приходим к следующему утверждению.

Теорема 7.8. *Для вычисления коэффициентов ЛРОС, описывающего последовательность сложности L, достаточно иметь любые $2L$ последовательных значений на выходе фильтра.*

Возвращаясь к примеру 7.4, заметим, что последовательность, порожденная регистром, заданным примитивным многочленом степени L, может быть восстановлена полностью по любому отрезку длины $2L$. Этот факт препятствует применению таких ЛРОС в криптографии.

То, что сложность L неизвестна заранее, не намного усложняет дело, поскольку можно по очереди проверять гипотезы $L = 1, 2, ...$ Решение системы из L уравнений имеет сложность не более L^3, поэтому общая сложность нахождения регистра не превышает L^4, т.е. остается в любом случае полиномиальной.

Алгоритм Берлекэмпа–Месси решает совместно обе задачи — определение линейной сложности и нахождение коэффициентов полинома ЛРОС со сложностью порядка L^2. Это упрощение обусловлено тем, что матрица коэффициентов имеет специфический вид: строки коэффициентов являются сдвигами предыдущих строк.

Решение задачи мы уже описали в параграфе 7.2. Остаток параграфа посвящен доказательству его оптимальности, т.е. минимальности длины регистра, получаемого по алгоритму БМ.

Лемма 7.9. *Пусть* $\boldsymbol{c}^{(r-1)}$ *длины* L_{r-1} *— регистр минимальной длины, генерирующий* $\boldsymbol{s}_{r-1} = (s_1, s_2, ..., s_{r-1})$, *а* $\boldsymbol{c}^{(r)}$ *длины* L_r *— регистр минимальной длины, генерирующий* $\boldsymbol{s}_r = (s_1, s_2, ..., s_r)$ *и пусть* $\boldsymbol{c}^{(r-1)} \neq \boldsymbol{c}^{(r)}$. *Тогда*

$$L_r \geq \max\{L_{r-1}, r - L_{r-1}\}.$$

Доказательство. Неравенство $L_r \geq L_{r-1}$ очевидно, поскольку регистр, генерирующий некоторую последовательность, генерирует также любую последовательность, продолжением которой эта последовательность является.

Докажем неравенство $L_r \geq r - L_{r-1}$. Для сокращения записи обозначим $\boldsymbol{a} = \boldsymbol{c}^{(r-1)}$, $\boldsymbol{b} = \boldsymbol{c}^{(r)}$, $L_a = L_{r-1}$, $L_b = L_r$.

Доказательство от противного. Предположим, что $r > L_a + L_b$. Из условий леммы имеем

$$s_j = -\sum_{i=1}^{L_a} a_i s_{j-i}, \quad j = L_{a+1}, ..., r-1; \tag{7.31}$$

$$s_r \neq -\sum_{i=1}^{L_a} a_i s_{r-i}; \tag{7.32}$$

$$s_j = -\sum_{k=1}^{L_b} b_k s_{j-k}, \quad j = L_{b+1}, ..., r, \tag{7.33}$$

поскольку $\boldsymbol{a}$ порождает последовательность длины $r-1$, но не порождает последовательность длины r, а $\boldsymbol{b}$ порождает последовательность длины r.

Применяя (7.33) при $j = r$, получаем после подстановки (7.31)

$$s_r = -\sum_{k=1}^{L_b} b_k s_{r-k} = -\sum_{k=1}^{L_b} b_k \sum_{i=1}^{L_a} a_i s_{r-k-i}. \tag{7.34}$$

Здесь существенно то, что индексы при s остаются положительными в силу предположения $r > L_a + L_b$. С другой стороны, подстановка (7.33) в (7.32) дает

$$s_r \neq -\sum_{i=1}^{L_a} a_i s_{r-i} = -\sum_{i=1}^{L_a} a_i \sum_{k=1}^{L_b} b_k s_{r-i-k}. \tag{7.35}$$

Очевидно, (7.34) противоречит (7.35). □

Хотя доказательство несложное, приведем еще одно, опирающееся больше на интуицию, чем на формальные вычисления. Для этого сформулируем вспомогательные утверждения.

Лемма 7.10. *Линейная сложность суммы последовательностей не превышает суммы их линейных сложностей.*

Доказательство. Устройство, порождающее отдельные слагаемые и суммирующее их — неоптимальная (возможно, не самая короткая) версия минимального ЛРОС. □

Лемма 7.11. *Линейная сложность последовательности* $\boldsymbol{s}$ *длины* r *вида* $\boldsymbol{s} = (\underbrace{0, 0, ..., 0}_{r-1 \text{ нулей}}\ a)$ *равна* r.

Доказательство. Один (не единственный возможный) полином обратной связи, генерирующий $\boldsymbol{s}$, равен $a - x^r$ (см. пример 7.3). Следовательно, линейная сложность не больше r. С другой стороны, регистр длины меньше r не может породить серию из $r - 1$ нулей. Значит, сложность равна r. □

Доказательство леммы 7.9. Пусть $\boldsymbol{s}_r$ — последовательность длины r и $\hat{\boldsymbol{s}}_r$ — последовательность длины r, порождаемая фильтром длины L_{r-1}. По условию леммы 7.9 $\boldsymbol{s}_r - \hat{\boldsymbol{s}}_r = (0, 0, ..., 0, \Delta)$, где $\Delta \neq 0$ — невязка. Сложность суммы, в соответствии с леммой 7.11, оказалась равной r. Из леммы 7.10 имеем $L_r + L_{r-1} \geq r$, что и требовалось доказать. □

Из леммы 7.9 и теоремы 7.3, следует, что алгоритм Берлекэмпа–Месси строит регистр минимально возможной длины, т.е. для заданной последовательности синдромов он отыскивает вектор ошибок минимального веса.

Пример 7.5. Предположим, что на выходе троичной линейной схемы наблюдается последовательность $\boldsymbol{x} = (2\,1\,0\,1\,2\,1\,0\,1\,2\,1\,0\,1\,...)$. Построим минимальный ЛРОС, порождающий эту последовательность. Результаты вычислений, выполняемых по алгоритму БМ шаг за шагом, приведены в табл. 7.4, а соответствующие схемы фильтров показаны на рис. 7.3.

Поскольку в алгоритме БМ (см. параграф 7.2) в качестве фильтра нулевой длины выбран фильтр $\Lambda = 1$, невязка на первом шаге подсчитывается относительно единицы. Это разумное соглашение, поскольку любое поле содержит единицу. Фильтр первого порядка после первых двух итераций успешно сформировал первые два символа последовательности, но третий (он выделен жирным шрифтом на рис. 7.3, *а*) неверен. На следующих итерациях в качестве исходного состояния фильтра принимаем уже аппроксимированные элементы последовательности. На пятой итерации приходится увеличивать порядок фильтра до трех. В результате оказалось, что заданная последовательность длины 12 описывается фильтром третьего порядка.

Таблица 7.4. Построение ЛРОС по последовательности на его выходе

r	s	Δ	$B(x)$	$\Lambda(x)$	L
0	—	0	1	1	0
1	2	2	2	$1+x$	1
2	1	0	$2x$	$1+x$	1
3	0	1	$1+x$	$1+x+x^2$	2
4	1	2	$x+x^2$	$1+2x+2x^2$	2
5	2	1	$1+2x+2x^2$	$1+2x+x^2+2x^3$	3
6	1	0	$x+2x^2+2x^3$	$1+2x+x^2+2x^3$	3
7	0	0	$x^2+2x^3+2x^4$	$1+2x+x^2+2x^3$	3

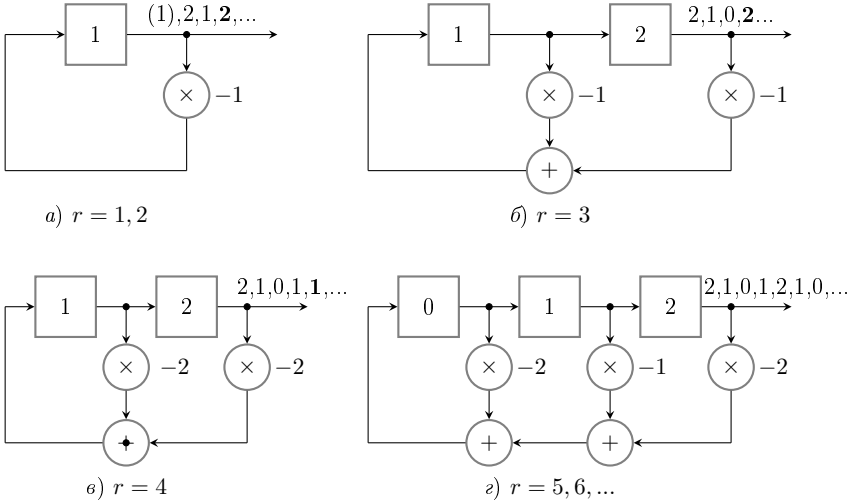


Рис. 7.3. Построение фильтра по заданной последовательности

8. Сверточные коды

Сверточные коды широко применяются в самых различных областях техники передачи и хранения информации. Примерами их эффективного применения являются системы космической связи, системы мобильной связи, модемы для телефонных каналов. В частности, протоколы V.32, V.34, ADSL, HDSL используют для защиты от ошибок сверточные коды в сочетании с декодированием по максимуму правдоподобия по алгоритму Витерби. Часто помимо сверточного кода передаваемые данные защищаются дополнительно с помощью циклической проверки на четность, либо применением кода Рида–Соломона. Тем самым получается код-произведение либо каскадный код, эти коды мы кратко рассмотрим в главе 10.

Среди специалистов по теории кодирования бытует шутка: блоковые коды хороши для статей и диссертаций, а сверточные — для надежной передачи информации. В этой шутке есть немного правды. Сверточные коды эффективны, но теория сверточных кодов на сегодняшний день не так глубока и изящна, как теория блоковых кодов, все практически значимые коды найдены с помощью компьютерного поиска, а декодирование имеет экспоненциальную сложность и реализуется методами, по сути, заимствованными из общей теории дискретного программирования. Другая возможная интерпретация сложившейся в этой области знаний ситуации состоит в том, что теория сверточных кодов сложнее, чем блоковых, и основные научные и практические результаты еще ждут своих первооткрывателей.

В связи с этим изучение сверточных кодов разбито на два этапа. Сначала мы рассмотрим очень простой подход, который вполне достаточен для понимания того, каким образом эти коды применя-

ются в технике связи. В следующей главе мы кратко рассмотрим основы алгебраической теории сверточных кодов.

8.1. Представление сверточного кода

При блоковом кодировании поступающие от источника информации данные разбиваются на блоки, и каждому блоку сопоставляется отдельное кодовое слово, зависящее только от данных, содержащихся в данном блоке. Альтернативой такому кодированию является сверточное кодирование, при котором блоки данных, по которым вычисляются в каждый момент времени кодовые символы, перекрываются с предыдущими.

Слово «сверточные» в названии кодов подразумевает наличие вычислителя свертки входных данных с некоторыми фиксированными последовательностями, определяющими код. Как мы увидим, под сверточными кодами часто понимается более широкий класс кодов, среди которых могут быть и нелинейные. Использование термина «сверточные коды» объясняется традицией.

Проще всего пояснить понятие сверточного кода примером. Одна из возможных схем кодера двоичного сверточного кода представлена на рис. 8.1. Кодер содержит k двоичных регистров сдвига длин ν_1, ν_2, ..., ν_k. Входами регистров сдвига являются информационные символы. Выходы ячеек регистров соединены с многовходовыми сумматорами по модулю 2, таких сумматоров n, по числу выходных символов, формируемых кодером на каждом такте работы.

На каждом такте на вход кодера поступает блок из k информационных символов. Эти символы и символы, хранящиеся в данный момент в регистрах кодера, поступают на входы тех сумматоров, которые подключены к соответствующим ячейкам. Результаты сложения по модулю 2 поступают на выход схемы. После этого в каждом из регистров происходит сдвиг, новые информационные символы записываются в первые ячейки, а содержимое остальных ячеек сдвигается на один разряд.

Содержимое $\nu = \nu_1 + \nu_2 + ... + \nu_k$ ячеек регистров сдвига в каждый конкретный момент времени называется текущим *состоянием*

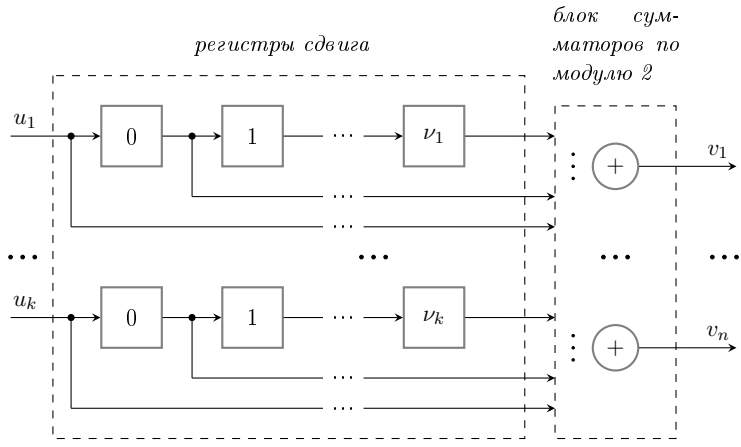


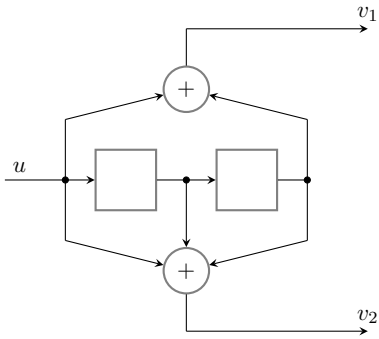
Рис. 8.1. Сверточный кодер со скоростью $R = k/n$

кодера. Предположим, что в начальный момент времени кодер находится в некотором заранее известном декодеру состоянии. Примем для определенности это начальное состояние нулевым. Рассмотрим процесс кодирования полубесконечной информационной последовательности. Значения на выходах сумматоров на каждом такте работы называются *кодовыми символами* сверточного кода. Полубесконечная последовательность кодовых символов называется *кодовым словом* сверточного кода. Множество всевозможных кодовых слов образует *сверточный код*.

Отношение $R = k/n$ называется *скоростью сверточного кода*. Примеры кодеров сверточных кодов приведены на рис. 8.2.

Суммарная длина регистров сверточного кодера ν называется *длиной кодового ограничения кода*, а максимальная длина регистров $m = \max_i\{m_i\}$ называется *памятью* или *задержкой* кодера. Для кодов со скоростью $R = 1/n$, т.е. при $k = 1$, память и кодовое ограничение совпадают.

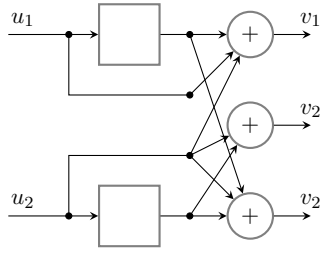
Относительно блоковых кодов мы знаем, что для одного и того же кода можно построить много порождающих матриц, описывающих код, или, другими словами, кодирование может быть выполнено разными кодерами. От выбора кодера зависит отображение информационных последовательностей на конкретные кодовые



а) $R = 1/2$, $m = \nu = 2$

$$G(D) = \begin{pmatrix} 1 + D^2 & 1 + D + D^2 \end{pmatrix}$$

$$G = \begin{pmatrix} 11 & 01 & 11 & 00 & \cdots & \cdots \\ 00 & 11 & 01 & 11 & 00 & \cdots \\ \cdots & 00 & 11 & 01 & 11 & \cdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots \end{pmatrix}$$



б) $R = 2/3$, $m = 1$, $\nu = 2$

$$G(D) = \begin{pmatrix} 1 + D & 0 & D \\ 1 & 1 + D & 1 + D \end{pmatrix}$$

$$G = \begin{pmatrix} 100 & 101 & 000 & \cdots & \cdots \\ 111 & 011 & 000 & \cdots & \cdots \\ \cdots & 100 & 101 & 000 & \cdots \\ \cdots & 111 & 011 & 000 & \cdots \\ \cdots & \ddots & \ddots & \ddots & \ddots \end{pmatrix}$$

Рис. 8.2. Примеры сверточных кодеров

слова, а это означает, что даже при одинаковых решениях, выносимых декодером максимального правдоподобия, средняя вероятность ошибки одного бита при использовании разных кодеров может быть разной. По отношению к сверточным кодам выбор кодера, реализующего данный код, имеет еще бóльшее значение.

Кодеры, которые мы рассматривали до сих пор, можно рассматривать как схемы, построенные на основе КИХ-фильтров (фильтров с конечным откликом), или фильтров без обратной связи. Альтернативные схемы могут быть построены на основе БИХ-фильтров, т.е. фильтров с обратной связью. В следующей главе мы покажем, что для всякого кода, порождаемого кодером с обратной связью, может быть найден кодер без обратной связи. Поэтому многие вопросы, связанные со сверточными кодами, достаточно изучать на примере кодеров без обратной связи.

В связи с появлением в 1990-х годах турбо-кодов кодеры с обратной связью стали привлекать к себе особенное внимание. Отчасти потому, что при скорости кода, близкой к пропускной способности канала, различие между характеристиками различных кодеров одного и того же кода становятся значимыми. Мы обсудим эти вопросы более подробно при изучении итеративного декодирования.

Кодеры сверточных кодов на основе регистров без обратной связи полностью описываются связями ячеек регистров сдвига с выходными сумматорами. Существует несколько общепринятых форм представления этих связей. Начнем с двоичных кодов со скоростью $R = 1/n$. Связи каждого из n сумматоров с ячейками одного регистра длины ν записываются в виде двоичного вектора $\boldsymbol{g}_i = (g_{i0}, g_{i1}, ..., g_{i\nu})$, $i = 1, 2, ..., n$. Значению $g_{ij} = 0$ соответствуют отсутствие связи j-й ячейки памяти регистра сдвига регистра с i-м сумматором, $g_{ij} = 1$ — наличие связи.

Векторы $\boldsymbol{g}_i$ называют порождающими векторами или генераторами кода. В таблицах кодов порождающие векторы приводят в восьмеричной форме. Например, генератор $\boldsymbol{g} = (1010111)$ будет записан как (127). Генераторы кодера, приведенного на рис. 8.2, *а*, в восьмеричной форме записываются как 5 и 7, поэтому мы будем называть его кодом (5,7).

Порождающие векторы записывают также в виде полиномов формальной переменной D. Например, порождающие полиномы кодера (5,7), показанного на рис. 8.2, *а*, имеют вид

$$\begin{aligned} g_1(D) &= 1 + D^2; \\ g_2(D) &= 1 + D + D^2. \end{aligned}$$

Эту совокупность полиномов можно также записать в виде полиномиальной порождающей матрицы

$$G(D) = \begin{pmatrix} 1 + D^2 & 1 + D + D^2 \end{pmatrix}.$$

В общем случае кодер со скоростью $R = k/n$ задается матрицей размерности $k \times n$. Элементами матрицы будут полиномы, если кодер построен на основе регистров без обратной связи. Линейный сверточный кодер общего вида может быть описан матрицей, элементы которой — дробно-рациональные функции формальной переменной D.

Пример сверточного кода со скоростью $2/3$ приведен на рис. 8.2, *б*.

Матричное представление кодера позволяет записать процесс кодирования для кода со скоростью $R = 1/n$ в виде умножения информационного полинома $u(D) = u_0 + u_1 D + ...$ на матрицу $G(D)$,

$$\boldsymbol{v}(D) = u(D)G(D).$$

При скорости $R = k/n$ входом кодера служит вектор

$$\begin{aligned} \boldsymbol{u}(D) &= \begin{pmatrix} u_1(D) & u_2(D) & ... & u_k(D) \end{pmatrix} = \\ &= \boldsymbol{u}_0 + \boldsymbol{u}_1 D + ..., \quad \boldsymbol{u}_i \in \{0,1\}^k \end{aligned}$$

из k информационных полиномов $u_1(D), ..., u_k(D)$. Через $\boldsymbol{u}_i$ обозначен вектор из k информационных бит, поступивших на вход кодера в момент времени i. Выход кодера — кодовый вектор из n полиномов

$$\begin{aligned} \boldsymbol{v}(D) &= \begin{pmatrix} v_1(D) & v_2(D) & ... & v_n(D) \end{pmatrix} = \\ &= \boldsymbol{v}_0 + \boldsymbol{v}_1 D + ..., \quad \boldsymbol{v}_i \in \{0,1\}^n, \end{aligned}$$

вычисляемый как

$$\boldsymbol{v}(D) = \boldsymbol{u}(D)G(D). \tag{8.1}$$

Порождающую матрицу $G(D)$ кодера с памятью m можно записать также в виде разложения по степеням переменной D:

$$G(D) = G_0 + G_1 D + ... + G_m D^m. \tag{8.2}$$

Множество кодовых слов образует линейное пространство полубесконечных двоичных последовательностей.

Мы знаем, что удобной формой представления блоковых линейных кодов является описание с помощью порождающей матрицы кода, строками которой, как известно, служат базисные векторы кода. Хотя размерность пространства кодовых слов сверточного кода бесконечна, его регулярная структура позволяет в явном виде указать базис пространства и выписать порождающую матрицу сверточного кода.

Рассмотрим сначала коды со скоростью $R = 1/n$. Заметим, что информационные последовательности вида 1, D, D^2,... образуют в совокупности (бесконечный) базис линейного пространства входных последовательностей кодера, поскольку любая информационная последовательность может быть единственным образом представлена в виде линейной комбинации этих мономов. Умножая эти последовательности на $G(D)$, получим базис пространства кодовых слов. Эти базисные векторы кода являются сдвигами одной и той же строки. Обобщая эти рассуждения на коды с произвольной скоростью, можем записать

$$G = \begin{pmatrix} G_0 & G_1 & \cdots & G_m & 0_{k,n} & \cdots & & \\ 0_{k,n} & G_0 & G_1 & \cdots & G_m & 0_{k,n} & \cdots & \\ & 0_{k,n} & G_0 & G_1 & \cdots & G_m & 0_{k,n} & \cdots \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \end{pmatrix}, \qquad (8.3)$$

где через $0_{k,n}$ обозначена нулевая матрица размерности $k \times n$, а G_m — матричные коэффициенты разложения $G(D)$ по степеням D в соответствии с (8.2).

При обсуждении вопросов декодирования часто используют представление сверточных кодов с помощью графов.

Вернемся к коду (5,7), кодер которого показан на рис. 8.2, *а*. Считаем, что начальное состояние кодера было нулевым. Это состояние и все следующие за ним мы будем отображать в виде узлов дерева. Возможные переходы из состояния в состояние соответствуют ребрам дерева. Каждому такому ребру соответствует некоторый блок из $n = 2$ кодовых символов. Несколько начальных ярусов дерева, соответствующего коду (5,7), показано на рис. 8.3. Это дерево называют *кодовым деревом*. В связи с такой графической интерпретацией сверточного кода параметр n называют *длиной ребра*.

По кодовому дереву легко определить кодовое слово, соответствующее заданной информационной последовательности. На рис. 8.3 движение вверх соответствует информационному символу 0, движение вниз — символу 1. Например, информационной последовательности 010... соответствует кодовое слово 00 11 01.... Соответствующие переходы показаны на дереве жирными линиями.

Обращает на себя внимание то, что число узлов дерева быстро растет от яруса к ярусу. В то же время, каждому узлу соответству-

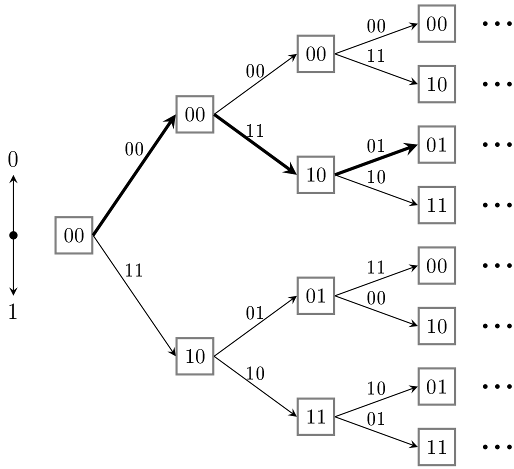


Рис. 8.3. Представление сверточного кода в виде дерева

ет некоторое состояние кодера. Число различных состояний кодера кода со скоростью $R = 1/2$ и с длиной кодового ограничения $\nu = 2$ равно $2^\nu = 4$. Следовательно, начиная с яруса с номером 3, найдутся узлы, соответствующие одинаковым состояниям. Поддеревья, начинающиеся в этих узлах, полностью идентичны. Чтобы сделать графическое представление кода более компактным, имеет смысл объединить узлы, соответствующие одинаковым состояниям, и отображать их в виде одного узла.

Полученный таким образом граф называют *решетчатой диаграммой* сверточного кода. Напомним, что у нас уже есть опыт построения решетчатых диаграмм для блоковых кодов (см. главу 4). Решетчатая диаграмм кода (5,7) приведена на рис. 8.4. В общем случае кода со скоростью $1/n$ и кодовым ограничением ν на ярусах с номерами $t =$1, 2, $\ldots$, $\nu - 1$ располагается 2^t узлов, и из каждого узла исходит $q = 2^k = 2$ ребра. На ярусах с номерами $t \geqslant \nu$ число узлов равно 2^ν, в каждый узел входит 2 ребра и из каждого узла исходит 2 ребра. В случае кода со скоростью k/n и кодовым ограничением ν максимальное число состояний равно 2^ν, число ребер, исходящих из каждого узла, равно 2^k.

По заданной информационной последовательности легко проследить кодовое слово. Поступление на вход кодера информацион-

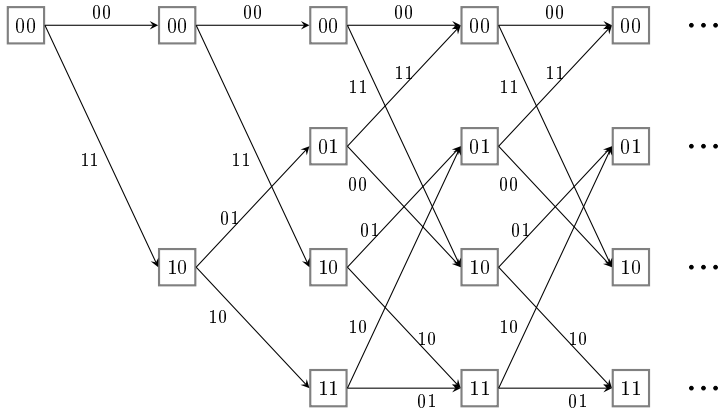


Рис. 8.4. Решетчатая диаграмма сверточного кода

ного символа 0 соответствует переходу в то из двух доступных из данного узла состояний, номер которого меньше. Поступление единицы приводит к переходу в состояние с бóльшим номером. На рис. 8.4 видно, что в первом случае мы двигаемся по решетке «вверх», во втором — «вниз».

Представление кода решетчатой диаграммой чрезвычайно удобно для анализа декодирования сверточных кодов. Для изучения характеристик сверточных кодов еще удобнее представление кодера в виде конечного автомата. Для кода (5,7) конечный автомат показан на рис. 8.5. Конечный автомат описывается графом, узлы которого соответствуют состояниям кодера, а ребра — переходам из состояние в состояние. Видно, что из каждого состояния выходят ровно два ребра, соответствующие двум возможным значениям входных символов. Каждому ребру приписаны кодовые символы, порождаемые кодером при данном переходе. В скобках указан информационный символ, соответствующий переходу.

Рассмотрим кодер и соответствующий конечный автомат, показанные на рис. 8.6.

Воспользовавшись схемой кодера либо диаграммой его состояний, легко убедиться, что информационной последовательности бесконечного веса 11...1...соответствует кодовая последовательность 11 01 00 00 ...00 ...веса 3. При передаче длинной после-

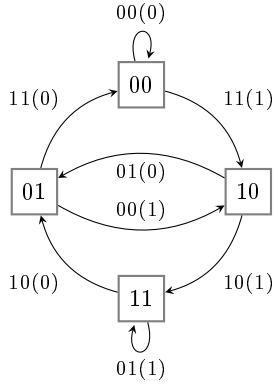


Рис. 8.5. Диаграмма состояний кодера сверточного кода

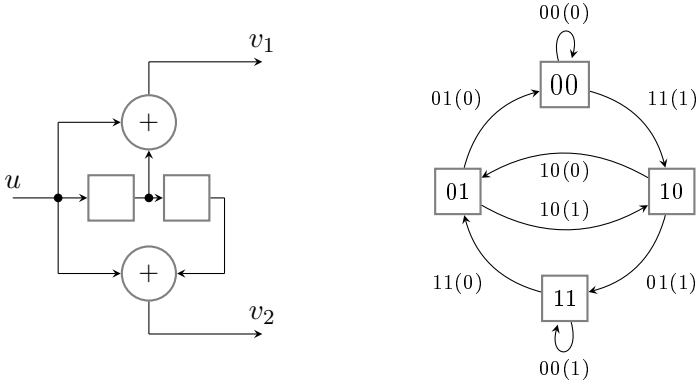


Рис. 8.6. Катастрофический кодер и его диаграмма состояний

довательности нулевых информационных символов всего 2 ошибки в канале связи достаточно для того, чтобы вместо нулевого кодового слова было принято записанное выше кодовое слово, соответствующее информационной последовательности из большого числа единиц. Этот пример показывает, что для некоторых сверточных кодеров небольшое число ошибок в канале связи может стать причиной большого числа ошибок декодирования.

Сверточные кодеры, содержащие кодовые слова конечного веса, соответствующие информационным последовательностям бесконечного веса, называются *катастрофическими*.

Сверточный кодер со скоростью $R = 1/n$ является катастрофическим, если его порождающие полиномы имеют общий делитель, не равный 1 или некоторому мономy D^i. Убедимся в этом.

Заметим, что интерес представляют только кодеры с порождающими матрицами $G(D)$ такими, что матрица G_0 в разложении (8.2) матрицы $G(D)$ отлична от нуля. В противном случае в регистрах кодера начальные ячейки невостребованы и могут быть удалены без изменения характеристик кода.

Пусть $a(D) = \text{НОД}(g_1(D), g_2(D), ..., g_n(D))$. Поскольку по меньшей мере один из генераторов не делится на D, общий делитель имеет вид $a(D) = 1 + Db(D)$, где $b(D) \neq 0$. Тогда при кодировании информационной последовательности бесконечного веса

$$u(D) = 1/a(D) = 1 + Db(D) + D^2b^2(D) + ...$$

будет получена последовательность кодовых блоков, полиномиальное представление которой имеет вид $v(d) = (g_1(D)/a(D), g_2(D)/a(D), ..., g_n(D)/a(D))$. Очевидно, эта кодовая последовательность имеет конечный вес.

Кодер, представленный на рис. 8.6, задается полиномиальной матрицей $G(D) = \begin{pmatrix} 1+D & 1+D^2 \end{pmatrix}$. Оба генератора имеют делителем полином $1 + D$, следовательно, кодер — катастрофический, в чем мы уже убедились, найдя слово конечного веса для информационной последовательности бесконечного веса. Найденная последовательность имеет вид (11 01 00, ...). Порождающая матрица может быть построена из этой последовательности и множества ее сдвигов на 2, 4 позиции и т.д. Сравнивая с (8.3), приходим к выводу, что тот же самый код описывается порождающей матрицей $G(D) = \begin{pmatrix} 1 & 1+D \end{pmatrix}$, задающей некатастрофический кодер данного кода.

Определить, является ли катастрофическим кодер кода со скоростью $R = k/n$, задаваемый произвольной матрицей $G(D)$ размера $k \times n$, не так просто. К этой задаче мы вернемся при обсуждении алгебраических свойств сверточных кодов.

8.2. Свободное расстояние и спектр сверточного кода

Как и в случае блоковых кодов, способность сверточного кода исправлять ошибки в канале связи определяется тем, насколько «далеко» друг от друга располагаются его кодовые слова. Для многих практических задач хорошей мерой близости служит расстояние Хэмминга между кодовыми словами.

Минимальным расстоянием блокового кода называют минимальное из попарных расстояний между кодовыми словами. Минимальное расстояние линейного кода может быть вычислено как минимальный вес его ненулевых кодовых слов. Особенность сверточного кода заключается в том, что его слова имеют бесконечную длину. Тем не менее, минимальное расстояние между парами кодовых слов конечно. В этом можно убедиться, например, вычислив вес кодового слова, порожденного информационной последовательностью веса 1.

Минимальное расстояние сверточного кода называют его *свободным расстоянием*. Обозначим его как d_f.

При поиске пути минимального веса по решетке сверточного кода достаточно рассматривать только пути, ответвляющиеся от нулевого на начальном ярусе и затем сливающиеся с нулевым путем. Множество таких путей называют *первым неправильным поддеревом*. Легко установить с помощью рис. 8.4, что свободное расстояние кода (5,7) равно $d_f = 5$. Слово минимального веса соответствует пути, который выходит из узла 00, затем проходит через узлы 10, 01, 00, 00, ... Этот же путь легко проследить и по диаграмме конечного автомата, представленной на рис. 8.5.

При анализе помехоустойчивости сверточных кодов выясняется, что на величину вероятности ошибки декодирования влияет не только свободное расстояние кода, но и то, как много слов малого веса содержит код. Обозначим через N_d число слов веса d, соответствующих путям, ответвляющимся от нулевого пути на нулевом ярусе (путям первого неправильного поддерева). Набор чисел $\{N_d\}$, $d = d_f, d_f + 1, ...$ называют *спектром сверточного кода*. Для вычисления спектров применяют метод производящих функций.

Рассмотрим дискретное множество $X = \{x_1, ..., x_N\}$, содержащее N элементов. Пусть на этом множестве определена некоторая функция $f(x)$, принимающая значения из дискретного множества $Z = \{z\}$, представляющего собой подмножество множества целых чисел $\mathbb{Z}$. Обозначим через N_z количество элементов x множества X, для которых $f(x) = z$. Полином вида

$$T(D) = \sum_{z \in Z} N_z D^z \tag{8.4}$$

называется производящей функцией для $f(x)$.

Пример 8.1. Пусть $Z = \{1, ..., 6\}$ — множество исходов бросания игральной кости. Производящая функция числа очков при одном бросании

$$T_1(D) = D + D^2 + ... + D^6 = \frac{D^7 - D}{D - 1}.$$

Пример 8.2. Рассмотрим игру в орлянку, при которой выпадение орла уменьшает капитал одного из игроков на единицу, выпадение решетки — увеличивает капитал на единицу. В этом случае $Z = \{-1, 1\}$ и производящая функция исходов одного бросания равна

$$T_2(D) = D^{-1} + D.$$

Главное свойство производящих функций, благодаря которому их используют для анализа сумм функций, состоит в том, что производящая функция суммы двух функций равна произведению производящих функций слагаемых. Действительно, если $T_U(D) = \sum_{u \in U} N_u D^u$ и $T_V(D) = \sum_{v \cup V} N_v D^v$ — две производящие функции, то их произведением будет

$$\begin{aligned} T_U(D) T_V(D) &= \sum_{u \in U} \sum_{v \in V} N_u N_v D^{u+v} = \\ &= \sum_z \left(\sum_{(u,v): u+v=z} N_u N_v \right) D^z = T_Z(D), \end{aligned}$$

где в правой части мы получили производящую функцию суммы $z = u + v$.

Пример 8.3. Монету бросают 10 раз. Каково число комбинаций исходов, при котором выигрыш игрока составит 6 единиц? Для ответа на этот вопрос запишем производящую функцию суммы исходов 10 опытов в виде

$$T_3(D) = T_2(D)^{10} = D^{-10}\left(1 + C_{10}^1 D^2 + C_{10}^2 D^4 + ... + C_{10}^9 D^{18} + D^{20}\right).$$

Поскольку коэффициент при D^6 равен C_{10}^8=120, именно в таком числе исходов из общего числа 1024 всевозможных исходов игрок получит доход в 6 единиц.

На рис. 8.7 приведен простой пример направленного графа. Числа, сопоставленные ребрам графа, интерпретируются как длины ребер. Задача анализа графа состоит в подсчете числа путей длины 0, 1, 2, ... из узла a в узел d. Обозначим через N_i число путей суммарной длины i. Непосредственный перебор по множеству путей показывает, что $N_0 = N_1 = N_2 = N_3 = 0$, $N_4 = 1$,.... Для этого простого примера нетрудно вывести общую формулу для N_i, но понятно, что для более сложного графа задача окажется трудной. Мы используем этот простой пример для иллюстрации метода производящих функций.

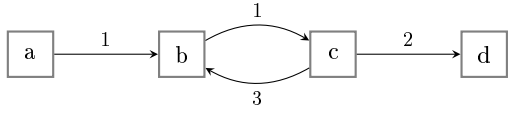


Рис. 8.7. Пример направленного графа

Обозначим через $g_s(D)$ производящую функцию длины путей из узла a в узел s. Задача состоит в нахождении $g_d(D)$.

Общие правила подсчета производящих функций весов путей графа таковы:

- Слиянию путей соответствует объединение двух множеств значений, поэтому при слиянии путей производящие функции суммируются.

- Последовательному соединению путей соответствует суммирование длин путей, поэтому соответствующие производящие функции перемножаются.

Рассмотрим узел b. В этот узел ведет один путь длины 1 из узла a и множество путей, проходящих по ребру длины 3 из узла c. Поскольку суммированию длин соответствует перемножение производящих функций, производящая функция для путей из a равна D, а для путей из c она равна $D^3 g_{\mathrm{c}}(D)$. В итоге имеем

$$g_{\mathrm{b}}(D) = D + D^3 g_{\mathrm{c}}(D).$$

Аналогично для узлов c и d имеем

$$g_{\mathrm{c}}(D) = D g_{\mathrm{b}}(D);$$
$$g_{\mathrm{d}}(D) = D^2 g_{\mathrm{c}}(D).$$

Решая систему из трех уравнений, находим

$$g_{\mathrm{d}}(D) = \frac{D^4}{1 - D^4}.$$

Чтобы найти коэффициенты производящей функции в явном виде, воспользуемся формулой

$$\frac{1}{1-x} = 1 + x + x^2 + ..., |x| < 1.$$

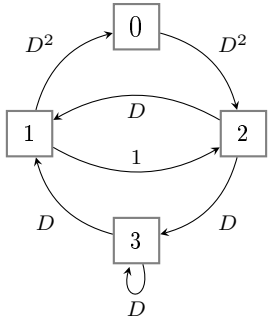
Окончательный результат имеет вид

$$g_{\mathrm{d}}(D) = D^4 + D^8 + D^{12} + ... \quad .$$

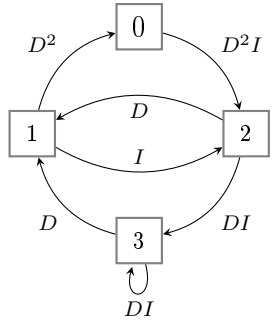
Отсюда следует, что, как можно было предвидеть, в нашем графе есть один путь из a в d длины 4, один путь длины 8 и т.д.

Рассмотрим теперь задачу вычисления *производящей функции спектра кода* на примере кода (5,7) со скоростью 1/2. На рис. 8.2, *а*, показана схема кодера и на рис. 8.5 — соответствующий кодеру конечный автомат. При подсчете спектра расстояний сверточного кода для нас не важна конкретная последовательность, сопоставленная конкретному ребру графа, важен только ее вес. Поэтому граф автомата приведен к виду, показанному на рис. 8.8, *а*, где ребрам приписаны производящие функции весов соответствующих последовательностей. Помимо этого, исключен переход из состояния 0 в состояние 0, поскольку нашей целью является анализ множества кодовых последовательностей первого неправильного поддерева, т.е.

последовательностей, отличающихся от нулевой последовательности в первом подблоке. Как и прежде, обозначим через $g_s(D)$ производящую функцию веса путей из узла 0 в узел с номером s. Задача теперь состоит в нахождении $g_0(D)$.



а) Разметка, учитывающая веса ребер



б) Разметка, учитывающая веса ребер и информационных последовательньостей

Рис. 8.8. Размеченные диаграммы состояний кода (5,7)

Аналогично рассмотренному выше примеру получаем систему уравнений

$$\begin{cases} g_0(D) &= D^2 g_1(D); \\ g_1(D) &= Dg_2(D) + Dg_3(D); \\ g_2(D) &= D^2 + g_1(D); \\ g_3(D) &= Dg_2(D) + Dg_3(D). \end{cases}$$

Решением системы является производящая функция

$$g_0(D) = \frac{D^5}{1-2D} = D^5 + 2D^6 + 4D^7 + \ldots \quad .$$

Итак, наши вычисления показывают, что среди кодовых слов первого неправильного поддерева имеется одно слово веса 5, два слова веса 6, четыре слова веса 7 и т.д.

Как уже говорилось, при расчете помехоустойчивости сверточных кодов важно не только расстояние между кодовыми словами, но и количество искаженных информационных символов, обусловленных принятием ошибочного решения в пользу того или иного

кодового слова. Таким образом, каждый путь в графе мы можем характеризовать двумя параметрами: вес Хэмминга соответствующего кодового слова и вес Хэмминга соответствующей информационной последовательности.

Метод производящих функций, который был выше описан для случая одного дискретного множества чисел, легко обобщается для совместного анализа двух и большего числа множеств. В частности, если обозначить через $t(d,i)$ число кодовых слов, порождаемых информационными последовательностями веса i и имеющих вес Хэмминга, равный d, то производящая функция числа таких путей определяется как функция двух формальных переменных D и I

$$T(D,I) = \sum_{d=1}^{\infty}\sum_{i=1}^{\infty} t(d,i) D^d I^i. \qquad (8.5)$$

Эту производящую функцию называют *расширенной производящей функцией сверточного кода*. Найдем $T(D,I)$ для кода (7,5). Для этого каждому ребру конечного автомата, описывающего работу кодера, сопоставим вычисленную для этого ребра совместную производящую функцию веса кодового блока и веса соответствующей информационной последовательности. Размеченная таким образом диаграмма состояний приведена на рис. 8.8, *б*.

Пусть $g_s(D,I)$ обозначает расширенную производящую функцию для путей, ведущих из состояния 0 в состояние s. Искомая производящая функция равна $T(D,I) = g_0(D,I)$. По диаграмме состояний кодера составляем систему уравнений для нахождения $g_0(D,I)$

$$\begin{cases} g_0(D,I) &= D^2 g_1(D,I); \\ g_1(D,I) &= D g_2(D,I) + D g_3(D,I); \\ g_2(D,I) &= D^2 I + I g_1(D,I); \\ g_3(D,I) &= DI g_2(D) + DI g_3(D,I). \end{cases} \qquad (8.6)$$

Решив систему, получим

$$\begin{aligned} T(D,I) &= g_0(D,I) = \\ &= \frac{D^5 I}{1-2DI} = D^5 I + 2D^6 I^2 + 4D^7 I^3 + 8D^8 I^4 \ldots \end{aligned} \qquad (8.7)$$

Этот результат означает, что код содержит одно слово веса 5, порождаемое информационной последовательностью веса 1. Каждое

из двух слов веса 6 порождается информационной последовательностью веса 2 и т.д.

Нам пригодится в дальнейшем производящая функция весов путей, не учитывающая весов соответствующих информационных последовательностей

$$T(D) = T(D, I)\,|_{I=1}\ . \tag{8.8}$$

Для практических расчетов вероятности ошибок часто используется производящая функция

$$F(D) = \sum_{d=1}^{\infty} f_d D^d, \tag{8.9}$$

в которой f_d обозначает суммарное количество единиц в информационных последовательностях, порождающих кодовые слова веса d. Понятно, что

$$f_d = \sum_{i=1}^{\infty} it(d, i). \tag{8.10}$$

Функцию $F(D)$ называют *производящей функцией весов для подсчета вероятности ошибки на бит* или, проще, *производящей функцией спектра ошибки на бит.*

Из (8.5) и (8.10) следует формула, связывающая функции $T(D, I)$ и $F(D)$:

$$F(D) = \sum_{d=1}^{\infty}\sum_{i=1}^{\infty} it(d, i)D^d = \tag{8.11}$$

$$= \frac{\partial}{\partial I} T(D, I)\,|_{I=1}\ . \tag{8.12}$$

Из (8.7) и (8.11) для кода с генераторами (5,7) получаем

$$F(D) = D^5 + 4D^6 + 12D^7 + 32D^8 + ...\quad .$$

Из этого примера видно, что даже для кода с кодовым ограничением 2 применение метода производящих функций позволило существенно упростить задачу вычисления спектральных характеристик кода по сравнению с перебором по множеству кодовых слов кода.

Заметим, что для получения функций $F(D)$ и $T(D)$ мы решили в символьном виде систему из 4 линейных уравнений. Для кода с кодовым ограничением ν число уравнений равно 2^ν, где ν — кодовое ограничение кода. Для практически интересных длин кодового ограничения задача относительно просто решается с помощью компьютера, но нужно иметь в виду, что метод Гаусса решения систем линейных уравнений в данном применении будет иметь сложность порядка $2^{3\nu}$. Понятно, что даже для небольших длин кодовых ограничений данный подход неэффективен.

Нетрудно заметить, однако, что матрица коэффициентов уравнений в каждой строке содержит всего 2 значащих элемента (2^k ненулевых элемента в случае кода со скоростью $R = k/n$). Следовательно, матрица является разреженной, что позволяет уменьшить сложность решения системы уравнений до величины порядка 2^ν. На практике применяются численные методы, позволяющие быстро найти первые коэффициенты разложения производящих функций по степеням D без нахождения точного аналитического выражения для самих функций, т.е. без решения системы уравнений. Сложность этих методов много ниже чем 2^ν, что позволяет найти свободные расстояния и спектры некоторых кодов с кодовым ограничением свыше 70. Примерами методов быстрого анализа сверточных кодов являются алгоритмы FAST [61] и BEAST [49].

8.3. Оценки вероятности ошибки

Рассмотрим сначала произвольный линейный код длины n с минимальным расстоянием d и спектром $N_d, N_{d+1}, ..., N_n$, где N_w — число слов веса w.

Оценивая вероятность ошибки при передаче одного из слов линейного кода (например, нулевого) сверху суммой вероятностей ошибок в пользу слов веса w при $w = d, d+1, ..., n$, приходим к оценке

$$P_e = \sum_{w=d}^{n} P_e(w) \le \sum_{w=d}^{n} N_w P(w), \tag{8.13}$$

где $P_e(w)$ — вероятность ошибки в пользу конкретного слова веса w,

а $P(w)$ — вероятность ошибки для кода из двух слов, отличающихся в w позициях.

Интуитивно понятно, что $P(w)$ в случае канала без памяти экспоненциально убывает с увеличением w. Примем как гипотезу, что для $P(w)$ имеет место оценка вида

$$P(w) \leq B(w) D_0^w, \tag{8.14}$$

где D_0 — константа, зависящая от характеристик канала, а $B(w)$ — невозрастающая по w функция, также зависящая от канала. Вычисления, подтверждающие эту гипотезу, для случаев ДСК и канала с АБГШ приведены ниже. После подстановки (8.14) в (8.13) имеем

$$P_e \leq \sum_{w=d}^{n} N_w B(w) D_0^w \leq B(d) \sum_{w=d}^{n} N_w D_0^w. \tag{8.15}$$

Введя обозначение

$$N(D) = \sum_{w=d}^{n} N_w D^w$$

для производящей функции спектра кода, приходим к формуле

$$P_e \leq B(d) N(D)|_{D=D_0}. \tag{8.16}$$

Точно так же для сверточного кода вероятность «*ошибочного события*», т.е. того, что будет принято решение в пользу одного из слов, отличающихся от правильного слова на первом ярусе решетки, удовлетворяет неравенству

$$P_e \leq B(d_f) T(D)|_{D=D_0}. \tag{8.17}$$

Аналогично для сверточного кода со скоростью $R = k/n$ вероятность ошибки на бит P_{eb}, т.е. доля ошибок в декодированной информационной последовательности, удовлетворяет неравенству

$$P_{eb} \le \frac{1}{k} B(d_f) F(D)|_{D=D_0}, \tag{8.18}$$

где $F(D)$ вычисляется по $T(D, I)$ по формуле (8.11).

Осталось указать формулы для D_0 и $B(d)$.

Рассмотрим сначала двоичный симметричный канал (ДСК).

Воспользовавшись формулой биномиального распределения, мы легко найдем вероятность половины или более половины ошибок на w позициях. После этого нужно будет воспользоваться формулой Стирлинга.

Упражнение 8.1. Докажите, что в случае ДСК с переходной вероятностью p_0

$$B(w) = \frac{1-p}{1-2p}\sqrt{\frac{2}{\pi w}}; \tag{8.19}$$

$$D_0 = 2\sqrt{p_0(1-p_0)}. \tag{8.20}$$

Подсказка: воспользуйтесь формулами из приложения к главе 1.

Чуть более точная формула, чем (8.19), приведена в [100]. Еще одно уточнение, основанное на том, что при четном w число ошибок, равное $w/2$, только в половине случаев приводит к ошибке декодирования, предложено Ван де Мибергом [111]. Окончательные формулы для ДСК имеют вид:

$$P_e < \frac{B(d_f)}{2} \left[(1+D)T(D) + (1-D)T(-D)\right]|_{D=2\sqrt{p(1-p)}}; \tag{8.21}$$

$$P_b < \frac{B(d_f)}{2k} \left[(1+D)F(D) + (1-D)F(-D)\right]|_{D=2\sqrt{p(1-p)}}, \tag{8.22}$$

где

$$B(w) = \begin{cases} \frac{2\sqrt{p(1-p)}}{\sqrt{2\pi(w+1)}(1-2p)} & \text{для нечетных } w; \\ \frac{1}{\sqrt{2\pi w}(1-2p)} & \text{для четных } w. \end{cases}$$

Обратимся к случаю канала с аддитивным гауссовским шумом.

Как и в главе 1, считаем, что на выходе канала при передаче нуля и единицы наблюдаются гауссовские случайные величины с дисперсией $N_0/2$ и математическими ожиданиями $-\sqrt{E}$ и $\sqrt{E}$ соответственно.

Упражнение 8.2. Докажите, что вероятность ошибки для кода из двух слов, отличающихся в w позициях, равна

$$P(w) = Q\left(\sqrt{\frac{2wE}{N_0}}\right), \tag{8.23}$$

где использовано обозначение

$$Q(x) = 1 - \frac{1}{\sqrt{2\pi}} \int\limits_{-\infty}^{x} e^{-\frac{t^2}{2}}\, dt.$$

Следующий шаг состоит в использовании неравенства

$$Q\left(\sqrt{x+y}\right) \leq \; Q\left(\sqrt{x}\right) e^{-y/2}, \; x, y \geqslant 0,$$

доказательство которого можно найти, например, в [25] (лемма 4.8). Получаем формулы

$$D_0 = e^{-\frac{E}{N_0}}; \tag{8.24}$$

$$B(w) = Q\left(\sqrt{\frac{2wE}{N_0}}\right) e^{wE/N_0}. \tag{8.25}$$

Эти значения используются совместно с (8.17) и (8.18) для вычисления, соответственно, вероятности ошибочного события и вероятности ошибки на бит.

Аддитивные границы вероятности ошибки грубы и почти бесполезны при малых отношениях сигнал/шум. При сильном шуме единственным способом анализа кодов остается компьютерное моделирование. В то же время, при высоких отношениях сигнал/шум и, соответственно, малой вероятности ошибки (ниже 10^{-6}) трудоемкость моделирования становится очень высокой, при этом аддитивные оценки, вычисленные по первым $5 \ldots 10$ коэффициентам производящей функции, дают вполне приемлемые по точности результаты.

Приведенные вычисления дают возможность оценить энергетический выигрыш кодирования при больших значениях отношения сигнал/шум на бит передаваемой информации.

Для случая ДСК переходная вероятность p_0 связана с отношением сигнал/шум формулой

$$p_0 = Q\left(\sqrt{\frac{2E}{N_0}}\right) \cong e^{-E/N_0} \quad ,$$

где знак $\cong$ мы используем для обозначения асимптотического равенства при $E/N_0 \to \infty$. В частности, в отсутствие кодирования вероятность ошибки на бит $P_{eb} = p_0$ и отношение сигнал/шум на бит $E_b/N_0 = E/N_0$, поэтому

$$P_{eb}(\text{Без кодирования}) \cong e^{-E_b/N_0}.$$

При использовании блоковых либо сверточных кодов в аддитивной границе вероятности ошибки (8.15)—(8.16) при $E/N_0 \to \infty$ имеет значение только первое слагаемое суммы, причем коэффициент при экспоненциальном члене вида D_0^d также не играет роли. Имеем

$$P_{eb} \cong D_0^d = \left(2\sqrt{p(1-p)}\right)^d \cong p^{d/2} \cong e^{-d\frac{E}{2N_0}}.$$

Напомним, что при скорости кода R имеет место равенство $E = RE_b$. Сопоставляя соотношения для вероятности ошибки при наличии и отсутствии кодирования, приходим к выводу, что при использовании кодов со скоростью R и минимальным расстоянием d асимптотический выигрыш кодирования при больших отношениях сигнал/шум равен

$$\gamma_{\text{ДСК}} \cong 10\lg(dR/2). \tag{8.26}$$

Для сверточного кода вместо d нужно подставить свободное расстояние d_f. Для гауссовского канала из (8.16) и (8.24) сразу получаем

$$P_{eb} \cong D_0^d = e^{-d\frac{E}{N_0}},$$

откуда следует, что энергетический выигрыш равен

$$\gamma_{\text{ГК}} \cong 10\lg(dR), \tag{8.27}$$

и в этой формуле вместо d подставляют d_f, если код сверточный.

Результаты моделирования декодирования по максимуму правдоподобия в канале с мягкими решениями для нескольких блоковых и сверточных кодов показаны на рис. 8.9. Заметим, что сложность декодирования расширенного кода Голея (24,12) соизмерима со сложностью декодирования сверточного кода с ограничением 8, а кода (48,24) — со сложностью декодирования сверточного кода с ограничением 12. Из графиков видно, что при сопоставимой сложности декодирования сверточные коды выигрывают почти 2 дБ по требуемому отношению сигнал/шум на бит по сравнению с блоковыми кодами.

Упражнение 8.3. Сравните результаты моделирования для сверточных кодов с аддитивной оценкой вероятности ошибки.

Упражнение 8.4. Сравните реальный энергетический выигрыш при вероятности ошибки на бит 10^{-5} с асимптотическим энергетическим выигрышем и с пределом Шеннона (см. главу 1).

8.4. Декодирование по максимуму правдоподобия

Для декодирования сверточных кодов по максимуму правдоподобия применяется алгоритм Витерби, который применительно к блоковым кодам уже был рассмотрен в главе 4. Чтобы уяснить работу декодера, сначала мы рассмотрим блоковый код, получаемый усечением сверточного, а уже потом обсудим декодирование бесконечной последовательности символов на выходе канала. Усечение сверточного кода с кодовым ограничением ν на длине L осуществляется подачей на вход кодера после информационной последовательности длины $L-\nu$ серии из ν нулей. Если скорость исходного кода была равна $R=k/n$, то скорость усеченного кода будет равна $R'=R\frac{L-\nu}{L}$ и будет мало отличаться от скорости сверточного кода при большой длине L.

Будем рассматривать декодирование сверточного кода на примере кода (7,5) с кодовым ограничением $\nu=2$. Фрагмент решетчатой диаграммы усеченного кода при $L=5$ показан на рис. 8.10.

Эта решетчатая диаграмма описывает множество кодовых слов *усеченного сверточного кода,* получаемого при подаче на вход ко-

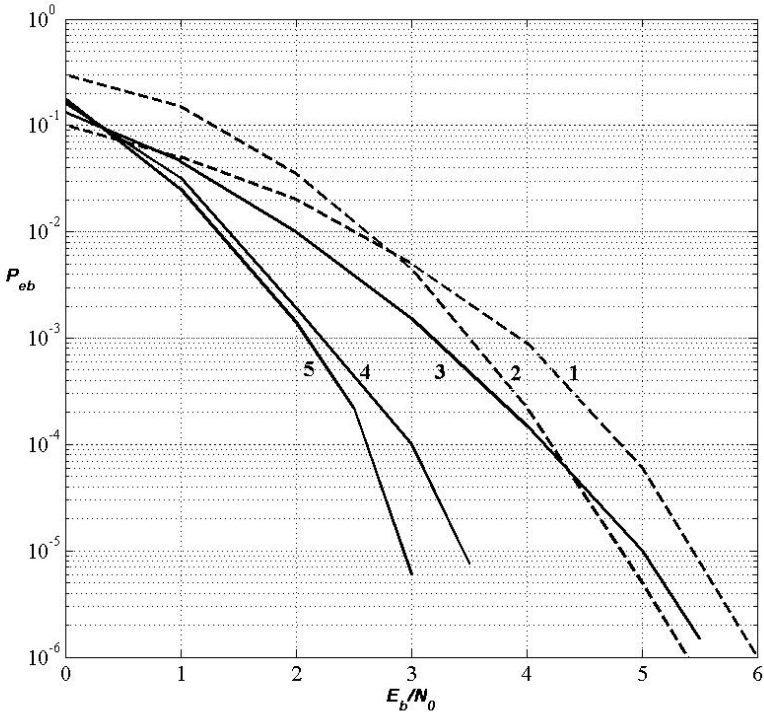


Рис. 8.9. Вероятность ошибки на бит в гауссовском канале для блоковых и сверточных кодов:
1: Код (24,12), d=8;
2: Код (48,24), d=12;
3: Сверточный код, $\nu = 4$, генераторы (35,23), d_f=7;
4: Сверточный код, $\nu = 8$, генераторы (657,435), d_f=12;
5: Сверточный код, $\nu = 12$, генераторы (15521,11677), d_f=16

дера некоторой последовательности информационных символов и вслед за ней — последовательности из ν нулей, где ν — длина кодового ограничения кода. В данном примере вслед за тремя информационными символами подается два нуля, в результате получается, по сути, линейный блоковый код длины 12 с четырьмя информационными символами. Рассмотрение усеченных сверточных кодов

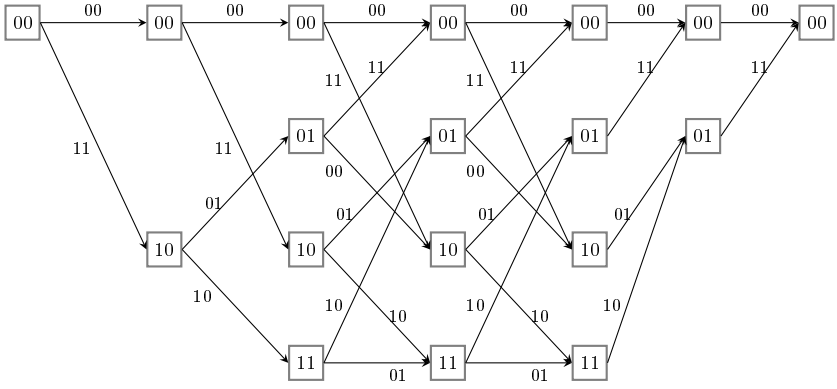


Рис. 8.10. Решетка усеченного сверточного кода

удобно с точки зрения описания алгоритма. Ниже будет объяснено, как алгоритм Витерби может быть использован для декодирования бесконечно длинных последовательностей.

Для каждого узла решетки мы называем предшествующими узлами те узлы предыдущего яруса, которые связаны ребрами с данным узлом. Путь, ведущий в данный узел на ярусе с номером t, задается последовательностью из t информационных символов, при подаче которой на вход кодера содержимое регистра кодера совпадет с номером узла.

При описании алгоритма мы предполагаем без потери общности, что задачей декодера является отыскание пути с **минимальной** метрикой. Метрикой узла мы называем метрику оптимального пути, ведущего в узел, метрикой ребра — сумму метрик соответствующих кодовых символов. Через L обозначим число ярусов решетчатой диаграммы, описывающей усеченный сверточный код.

Алгоритм 8.1. Алгоритм Витерби для сверточных кодов

Input: Выход канала $\boldsymbol{v}$
Output: Информационная последовательность $\boldsymbol{u}$
`1.Инициализация;`
t=0; `% Номер яруса`
Метрики всех узлов =0;
За каждым узлом закрепляется «пустой» путь нулевой длины;
`2. Основной цикл по ярусам решетки`
for $t = 1$ **to** L **do**
  `2. Цикл по узлам яруса (состояниям кодера)`
  for $s = 0$ **to** $2^{\nu} - 1$ **do**
    - Находим метрику каждого из путей, ведущих в узел, как сумму метрик предшествующих узлов и ребер, связывающ узлы-предшественники с данным узлом.
    - Находим путь с минимальной метрикой и эту метрику приписываем данному узлу.
    - Путь, ведущий в узел, вычисляется дописыванием к пути ведущему в выбранный предшествующий узел, информационного символа, соответствующего переходу из узла-предшественника в данный узел.

  end
end
`3. Формирование результата`
Путь, соответствующий единственному узлу на ярусе L, выдается получателю как результат декодирования.

Алгоритм Витерби 8.1 представляет собой частный случай более общего алгоритма 4.1, использованного для декодирования блоковых кодов в главе 4.

Пути, сходящиеся в одном узле решетки, называют конкурирующими, а выбранный из их числа наилучший путь называют выжившим. Коротко говоря, алгоритм Витерби на каждом ярусе решетки из каждого узла решетки «смотрит назад» на узлы предыдущего яруса, находит метрики конкурирующих путей и выбирает путь с

наименьшей метрикой. В результате к последнему ярусу выживает единственный путь, обладающий минимальной метрикой из всех возможных путей в решетке.

8.4.1. Реализация алгоритма Витерби

Описание, приведенное в предыдущем разделе, иллюстрирует основную идею алгоритма Витерби. При практической реализации возникает ряд проблем, самые важные проблемы мы обсудим в данном разделе. К их числу мы относим

- начало работы декодера;
- декодирование бесконечных последовательностей;
- организация памяти для хранения метрик путей;
- организация памяти для хранения выживших путей.

Начало работы декодера

Из приведенной на рис. 8.10 решетчатой диаграммы видно, что, пока номер яруса не превысил кодового ограничения кода ν, структура решетки отличается от структуры решетки на остальных ярусах. Эта нерегулярность усложняет как программную, так и аппаратную реализацию декодирования.

Эта проблема имеет очень простое решение. Предположим, что декодирование производится по критерию минимальной метрики. В начале работы декодера всем состояниям, кроме нулевого, приписываются одинаковые и достаточно большие значения метрик. Нулевому узлу приписывается нулевая метрика. Далее выполняются точно такие же действия, как при декодировании на ярусах с номерами, большими ν. Нетрудно убедиться, что при таком назначении начальных метрик на ярусе с номером ν декодер получит правильные значения метрик всех узлов. «Несуществующие» пути решетки будут отвергнуты декодером, поскольку их метрики будут больше метрик путей, исходящих из нулевого узла.

Применение алгоритма Витерби для декодирования бесконечных последовательностей

Из описания алгоритма следует, что окончательное решение выдается декодеру только после обработки всей принятой выходной последовательности канала. Понятно, что, если длина усечения кода велика или если используется неусеченный сверточный код, алгоритм не может быть использован из-за ограничений на задержку декодирования и сложность реализации.

Введем параметр T, представляющий собой максимальную задержку декодирования в числе информационных символов. Величина T существенно превышает кодовое ограничение кода ν. На ярусе с номером T память путей декодера будет заполнена информационными последовательностями, соответствующими 2^ν узлам решетки. Эти последовательности различны, но их число существенно меньше числа 2^T различных последовательностей длины T. На предыдущих шагах декодирования производилось сравнение метрик конкурирующих путей и отбрасывались пути с худшими значениями метрик. Если величина T достаточно велика, то среди путей, начавшихся T ярусов назад, сохранилась лишь небольшая доля путей, и все они имеют достаточно хорошие значения метрик. Наиболее вероятна ситуация, когда на ярусе с номером t, $t \geqslant T$, все пути имеют одинаковый бит, соответствующий ярусу с номером $t-T$. Этот бит можно выдать получателю, стереть из памяти путей, произвести сдвиг всех путей в регистрах памяти на одну ячейку и тем самым освободить место для очередного информационного символа.

В принципе, при сильном шуме в канале, возможно, что старшие биты в ячейках памяти путей различны. На этот случай нужно разработать стратегию формирования решений об информационных символах. На практике применяют одну из следующих стратегий:

- выбрать путь с наилучшей метрикой и выдать первый символ этого пути;
- подсчитать количество путей с различными значениями первого символа и принять решение по большинству;
- выдавать получателю первый символ одного и того же пути.

Нетрудно упорядочить эти стратегии по надежности и по сложности реализации, но можно заранее сказать, что при больших T разница между ними невелика. В практических схемах считается, что достаточно выбрать величину T приблизительно в 5—6 раз больше длины кодового ограничения кода.

Вычисление и хранение метрик путей

Основная область применения декодера Витерби — каналы с мягкими решениями, т.е. каналы, в которых значения на входе кодера — вещественные числа. Для дальнейшей обработки они должны быть представлены в цифровой форме. От точности представления зависит сложность аналого-цифрового преобразования, сложность устройств, выполняющих арифметические операции над метриками ребер и путей, объем памяти для хранения метрик.

В 70-е годы, когда разрабатывались первые реализации декодера Витерби, было принято считать, что для представления метрик символов достаточно 3 бит. По мере развития микроэлектроники ограничения на сложность арифметических устройств становились менее жесткими. Тем не менее, увеличение точности свыше 8 бит (для каналов с двоичным входным алфавитом) действительно не имеет смысла.

Итак, будем считать, что метрика одного символа записана в виде 1 байта. Сколько бит нужно для представления метрик путей?

Если длина пути составляет N символов канала, то максимальное значение метрики путей может достигнуть величины $2^8 N$ и для хранения такой метрики нужно отвести $(8 + \log_2 N)$ двоичных ячеек памяти. При $N \to \infty$ память любого объема неизбежно переполнится.

В решении этой проблемы определяющую роль играет следующее наблюдение. Пусть ν — кодовое ограничение кода, n — число кодовых символов, соответствующих ребру, μ_0 — максимальное значение метрики символа. Тогда разница между максимальной и минимальной метрикой путей, выживших на данном ярусе, не превышает величины $\nu n \mu_0$. Этот факт следует из того, что для любого узла решетки существует путь, ведущий в этот узел и отличающийся от пути с наименьшей метрикой не больше чем в νn символах.

Если на некотором ярусе из всех значений метрик вычесть минимальное, это не повлияет на выбор путей в будущем, и, значит, вероятность ошибки декодирования не изменится. Эта операция (ее называют *нормализацией* метрик) может выполняться на каждом ярусе или один раз после обработки нескольких ярусов. В результате число бит для хранения метрик каждого пути приблизительно равно $(8 + \log_2(\nu n))$ и при типичных значения параметров кодера не превышает 16 бит.

Организация памяти для хранения путей

Каждый из выживших путей записывается в виде двоичной последовательности длины T. Заметим, что декодер Витерби, обрабатывая очередной узел решетчатой диаграммы, вычисляет новый путь, ведущий в данный узел, но не может занести его в ту же ячейку памяти, в которой хранился предыдущий путь. Дело в том, что предыдущий путь может понадобиться при обработке других узлов текущего яруса. Значит, нужно хранить два массива путей: «старые» и «новые». После обработки очередного яруса массивы меняются названиями. Таким образом, объем памяти для хранения путей составляет $2T2^{\nu} \approx 10\nu 2^{\nu}$. Именно эта память на практике определяет основные затраты на реализацию декодера.

Существует альтернативный вариант реализации декодера, проигрывающий по скорости работы, но требующий вдвое меньше памяти для хранения путей. Этот вариант называют *декодированием с возвращениями* (trace-back декодер). Декодер с возвращениями на каждом ярусе работы после обработки узла заносит в память один бит, указывающий, какой из двух конкурирующих путей выбран.

Для выработки решения и выдачи получателю декодер, используя «обратный кодер», из текущего узла двигается назад по решетке и вычисляет путь в данный узел. Первый бит пути (k бит при скорости $R = k/n$) выдается получателю.

8.5. Высокоскоростные и переменные сверточные коды

В начале главы мы рассматривали двоичный сверточный кодер общего вида (см. рис. 8.1), получающий на свой вход k информационных символов на каждом такте работы и формирующий n двоичных кодовых символов. Скорость кода равна $R = k/n$. Прямолинейный подход к анализу и декодированию таких кодов основан на построении решетчатой диаграммы, из каждого узла которой выходит 2^k ребер, соответствующих всевозможным комбинациям входных символов. Такое же число ребер сходятся в каждом узле решетки, начиная с яруса, номер которого превышает длину кодового ограничения.

Посмотрим на такую решетку с точки зрения сложности декодирования по алгоритму Витерби. При длине кодового ограничения ν и числе регистров k сложность обработки одного яруса будет пропорциональна $2^k \times 2^\nu$, а сложность декодирования в расчете на бит передаваемой информации — $2^{\nu+k}/k$. Множитель $2^k/k$ — существенное увеличение сложности по сравнению с декодированием обычных (низкоскоростных) сверточных кодов.

Опираясь на теорию минимальных решеток (см. главу 4), можно указать простое решение проблемы — привести порождающую матрицу к минимальной спэновой форме и по ней построить побитовую или секционированную минимальную решетку кода. То, что код имеет бесконечную длину, не является препятствием, как мы увидим на следующем простом примере.

Пример 8.4. Рассмотрим сверточный код со скоростью $R = 2/3$ с полиномиальной порождающей матрицей вида

$$G(D) = \begin{pmatrix} 1+D & 1 & 0 \\ 1 & D & 1+D \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} + D \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \end{pmatrix}.$$

Ей соответствует двоичная порождающая матрица

$$G = \left(\begin{array}{ccc:ccc:ccc:c} 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & \cdots \\ \hdashline 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & \cdots \\ \hdashline 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & \cdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & \cdots \\ \hdashline \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{array}\right).$$

Чтобы привести эту матрицу к МСФ, достаточно прибавить первую строку ко второй, третью к четвертой и т.д. Этот же результат можно получить, исходя из полиномиальной формы матрицы, заменив вторую строку суммой двух строк:

$$G(D) = \begin{pmatrix} 1+D & 1 & 0 \\ D & 1+D & 1+D \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} + D \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}.$$

В итоге получим

$$G = \left(\begin{array}{ccc:ccc:ccc:c} 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & \cdots \\ \hdashline 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & \cdots \\ \hdashline 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & \cdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & \cdots \\ \hdashline \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{array}\right).$$

Эта матрица удовлетворяет условиям МСФ. Побитовая решетка, начиная с яруса 4, имеет логарифмическую сложность 2 (число состояний равно 4). Дополнительное упрощение можно получить, выполнив секционирование:

$$G = \left(\begin{array}{c:cc:cc:cc:cc:c} 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & \cdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & \cdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{array}\right).$$

Для секций с номерами 1, 3, 5, выходным символом является сумма одного информационного символа и двух символов, хранящихся в памяти. Это эквивалентно кодированию сверточным кодом

со скоростью 1, с генераторным полиномом $g(D) = 1 + D + D^2$. Для секций с номерами 2, 4, 6,... кодирование выполняется другим сверточным кодом, его скорость равна $1/2$, а генераторы — $g_1(D) = 1 + D + D^2$, $g_2(D) = 1 + D + D^2$ (уже хорошо знакомый нам код (7,5)).

Сверточный кодер, связи регистров которого с сумматорами переключаются от такта к такту, называют *переменным сверочным кодером*, соответствующий код — *переменным* («time-varying»).

Упражнение 8.5. Нарисуйте схему сверточного кодера и решетчатую диаграмму для кода из примера 8.4.

Итак, задача построения простого декодера для кода с произвольной рациональной скоростью $R = k/n$ имеет простое решение. Код нужно представить как k последовательно переключаемых сверточных кодов со скоростями $1/c_i$, такими, что $\sum_{i=1}^{k} c_i = n$. Декодирование по сложности будет мало отличаться от сложности декодирования двоичного кода с такой же длиной кодового ограничения.

Заметим, что числа k и n не обязательно должны быть взаимно простыми. Дело в том, что, например, множество кодов со скоростью $R = 2/4$ значительно шире множества кодов со скоростью $R = 1/2$. Среди переменных кодов (с кратными k и n) могут найтись коды с бóльшим свободным расстоянием, чем у постоянных кодов с такими же скоростью и кодовым ограничением. Известно несколько таких примеров, их можно найти в [48].

Исторически задача построения и декодирования высокоскоростных сверточных кодов была востребована раньше, чем была разработана теория минимальных решеток. Практичное решение было найдено еще в 1979 году в работах Дж. Кейна с соавт. (см. [7]), оно основано на «выкалывании» некоторых символов низкоскоростного кода для получения высокоскоростного кода.

Вернемся к примеру 8.4. Исходный код со скоростью $R = 2/3$ мы представили переменным кодом. Соответствующий переменный кодер на каждом такте переключается между постоянными кодерами с генераторами 7 и (7,5) (в восьмеричной записи). Этот код можно было получить из кода (7,5) следующим образом. Кодовые символы

этого кода получены применением генераторов (7, 5), (7, 5), (7, 5),... Вычеркнув из выходной последовательности кодера символы с номерами 2, 6, 10, ..., т.е. каждый четвертый, начиная со второго, мы получим код со скоростью $R = 2/3$ (на два информационных символа теперь тратится не 4, а 3 кодовых символа). Последовательность генераторов теперь (7), (7,5), (7), (7,5), ... , как и в примере 8.4. Этим способом мы можем получить код с любой скоростью из заданного кода со скоростью 1/2. Получаемые коды называются *выколотыми* сверточными кодами.

Для описания правила выкалывания вводится так называемая *матрица выкалывания*. Если исходный код имеет скорость $1/n_0$, то матрица выкалывания P для получения кода со скоростью $R = k/n$ имеет размер $n_0 \times k$. Для каждого из k информационных символов одного подблока столбцы матрицы выкалывания P указывают, какие именно из n_0 генераторов использованы при его кодировании.

Для кода примера 8.4 матрица выкалывания

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

Код со скоростью $R = 4/5$ с кодовым ограничением 2 из таблицы 8.6 задается матрицей

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

Понятно, что выколотые коды являются подмножеством кодов общего вида, которые можно рассматривать как выколотые коды с разрешенным числом генераторов равным n. Ограничение на число генераторов может быть полезным в некоторых задачах. Одно из преимуществ выколотых кодов — некоторое (очень небольшое) уменьшение сложности кодера и декодера. Еще одно проявляется в системах, где, в зависимости от текущих характеристик канала, меняется скорость кода. При этом ограничение на множество генераторов заметно упрощает логику работы кодера и декодера. Характеристики наборов таких «согласованных по скорости» кодов [78] не намного хуже характеристик кодов общего вида.

8.6. Построение блоковых кодов усечением сверточных кодов

8.6.1. Усеченные сверточные коды

Сверточные коды мы рассматриваем как линейные коды бесконечной длины. На практике, конечно, любые сеансы связи конечны. Для передачи блоков данных конечной длины используются усеченные сверточные коды (см. параграф 8.4 и рис. 8.10).

Кодовое слово усеченного сверточного кода с кодовым ограничением ν получается подачей на вход кодера последовательности информационных символов с дописанными к ней ν нулями. Дописывание «нулевого хвоста» приводит к некоторому снижению скорости, которое можно считать несущественным при большой длине блока.

Заметим, что с ростом длины блока при фиксированной длине кодового ограничения относительное расстояние соответствующего блокового кода стремится к нулю. Это говорит о том, что на этом пути из сверточных кодов мы получаем плохие в смысле минимального расстояния блоковые коды.

Можно ли построить хорошие блоковые коды из сверточных кодов? Ответ на этот вопрос является целью данного параграфа.

Теоретический (псимптотический) ответ на этот вопрос получен методом случайного кодирования в [73]. При заданных параметрах случайного переменного сверточного кода при правильной длине усечения почти все получаемые блоковые коды удовлетворяют границе Варшамова–Гилберта.

В свете теории минимальных решеток такой результат не выглядит неожиданным, т.к. любой линейный код можно описать его минимальной решеткой, а эту решетку можно интерпретировать как решетку переменного сверточного кода.

К сожалению, получить описание лучших известных линейных кодов в виде усеченных постоянных сверточных кодов не удается. Более продуктивным в этом смысле оказался класс циклически усеченных кодов, которые рассматриваются в следующем параграфе.

8.6.2. Циклически усеченные сверточные коды

В оригинальной работе Вулфа и Ма [91] (1986) эти коды были названы кодами с откушенными хвостами (tail-biting codes). В русскоязычной литературе за этими кодами укрепилось название циклически усеченных (ЦУ) кодов. Первооткрывателями этого класса кодов следует считать Соломона и Ван-Тилборга [105] (1979).

Идею циклического усечения иллюстрирует рис. 8.11 на примере сверточного кода со скоростью $R = 1/n_0$. Подлежащие кодированию k информационных символов записываются в циклический регистр сдвига, последние ν ячеек которого являются регистром сдвига сверточного кода. На каждом такте кодирования с выхода сверточного кода считываются очередные n_0 кодовых символов кодового слова. Таким образом, в результате k тактов работы кодера мы получим $n = kn_0$ кодовых символов.

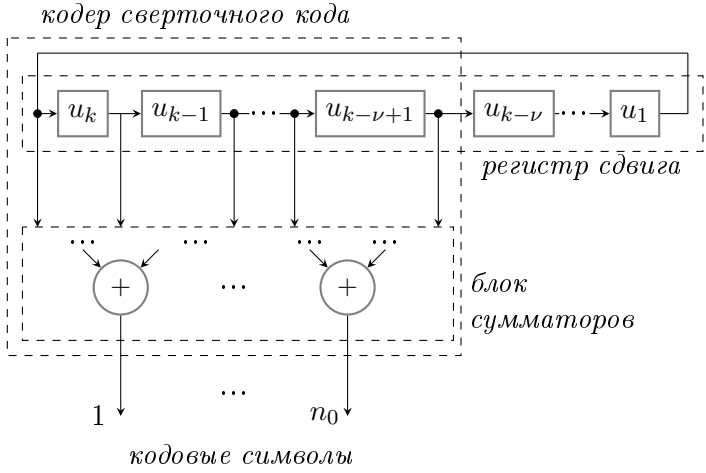


Рис. 8.11. Кодер циклически усеченного кода со скоростью $R = 1/n_0$

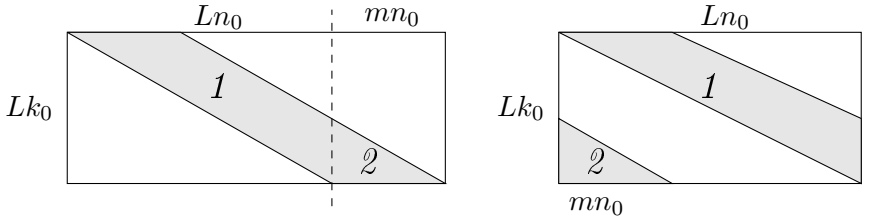
Аналогичная схема на основе сверточного кода со скоростью $R = k_0/n_0$ после $L = k/k_0$ тактов работы порождает кодовое слово линейного $(n = Lk_0, k = Ln_0)$-кода, который мы называем *циклически усеченным (ЦУ) сверточным кодом.*

Отметим очевидную разницу между усеченным и ЦУ кодом. Начальное и конечное состояние кодера усеченного кода — нулевое. Чтобы привести кодер в нулевое состояние, требуются лишние ν тактов работы кодера и, соответственно, уменьшение скорости кода по сравнению со скоростью исходного сверточного кода. Начальное состояние ЦУ кода совпадает с конечным, но оно может быть любым, оно определяется значениями ν последних информационных символов. При этом скорость циклически усеченного кода равна скорости исходного сверточного кода.

То, что начальное состояние неизвестно заранее, как мы увидим ниже, сказывается на сложности декодирования.

Из схемы кодера видно, что циклическому сдвигу информационной последовательности соответствует циклический сдвиг кодового слова на n_0 позиций. То же явление имеет место для кода с произвольной скоростью. Отсюда заключаем, что для ЦУ сверточного кода сдвиг циклический любого слова на n_0 позиций является кодовым словом. Коды, обладающие таким свойством, называются *квазициклическими*.

Квазициклическим кодам было посвящено много теоретических исследований. В частности, известно, что среди них есть коды с минимальным расстоянием, достигающим границы Варшамова–Гилберта [84]. Среди лучших известных линейных кодов [77] много квазициклических кодов. Как мы теперь видим, циклически усеченные сверточные коды являются одной из форм описания квазициклических кодов.



а) Усеченный код *б)* Циклически усеченный код

Рис. 8.12. Структура порождающих матриц усеченных и циклически усеченных кодов

На рис. 8.12 схематично показана структура порождающей матрицы усеченных и ЦУ кодов. В затененной части матрицы могут быть ненулевые и нулевые элементы, на остальных позициях — только нули. Через m обозначена память сверточного кода, которая для кодов со скоростью $R = 1/n_0$ совпадает с кодовым ограничением. При циклическом усечении «откусывается хвост», соответствующий нулевым информационным символам. Этот хвост копируется в начало порождающей матрицы. Как видно из рисунка, в отличие от обычного усечения, при циклическом усечении все выходные символы зависят от примерно одинакового числа информационных символов.

Пример 8.5. Рассмотрим сверточный код со скоростью $R = 1/2$ с генераторами (1,7) или $g(D) = (1, 1 + D + D^2)$. ЦУ (10, 5)-код задается порождающей матрицей

$$G = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 1 & & & & \\ & & 1 & 1 & 0 & 1 & 0 & 1 & & \\ & & & & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & & & & & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & & & & & 1 & 1 \end{pmatrix} \tag{8.28}$$

(пустые позиции подразумевают нули, как и на рис. 8.12). Решетка, описывающая код, приведена на рис. 8.13. Минимальное расстояние кода равно 4.

Напомним, что начальное состояние кодера ЦУ кода совпадает с конечным. Следовательно, кодовым словам кода соответствуют

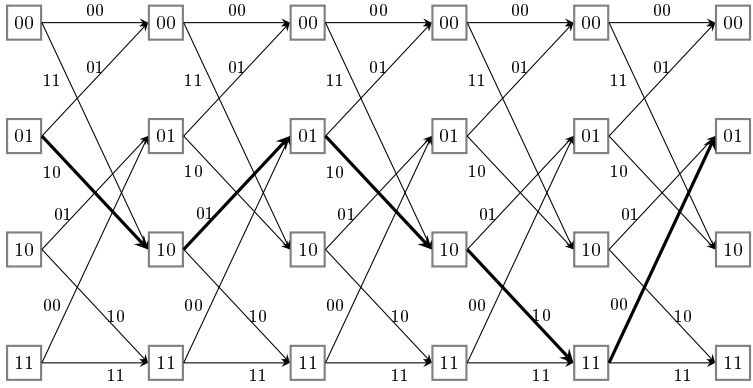


Рис. 8.13. Решетка циклически усеченного сверточного кода

пути, начинающиеся и заканчивающиеся в узлах с одинаковыми номерами.

На рис. 8.13 путь в решетке, соответствующий информационной последовательности (10110), показан жирными стрелками, ему соответствует кодовое слово (1 0 0 1 1 0 1 1 0 0).

Упражнение 8.6. Укажите схему кодера, порождающую матрицу и решетку для представления кода Хэмминга (8.4) в виде ЦУ сверточного кода.

Для решеток типа решеток ЦУ кодов можно развить теорию аналогичную теории минимальных решеток блоковых кодов, переопределив соответствующим образом множество активных позиций строки, спэн, минимальную спэновую форму. Этим вопросам, а также компьютерному поиску хороших блоковых (по сути, квазициклических) кодов с минимальной сложностью ЦУ решеток посвящена работа [54]. Оказалось, что для большинства лучших известных решеток можно найти довольно простые решетки и, более того, на этом пути были построены несколько кодов с бóльшим минимальным расстоянием, чем известные на тот момент лучшие коды.

Сложность ЦУ решетки можно оценить без построения самой решетки прямо по порождающей матрице кода, если она приведена к форме аналогичной (8.28), в которой можно выделить активные части строк и все позиции начала активных частей различны, так

же как и позиции их концов. По такой форме (*минимальной ЦУ спэновой форме*) не составляет труда построить ЦУ решетку.

Пример 8.6. ЦУ решетка кода Голея (24,12)

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

построена в [105, 59] из переменного сверточного кода со скоростью $R = 4/8$ с кодовым ограничением 4.

Рассмотрим кратко способы декодирования ЦУ кодов. Поскольку код описан решеткой, мы могли бы воспользоваться алгоритмом Витерби для поиска пути (слова) с максимальной функцией правдоподобия. Задачу декодирования усложняет то, что в отличие от обычного представления кода решеткой, неизвестны начальный и конечный узел решетки. Это препятствие преодолевается прямым перебором по множеству начальных узлов.

Для декодирования по максимуму правдоподобия ЦУ кода с кодовым ограничением ν нужно выполнить 2^ν попыток декодирования. Каждая попытка состоит в поиске наилучшего пути из некоторого начального узла в узел с тем же номером на последнем ярусе решетки. Сложность каждой попытки пропорциональна числу узлов 2^ν. Полная сложность декодирования оказалась равной $2^{2\nu}$, что намного больше сложности 2^ν декодирования обычного усеченного сверточного кода с той же длиной кодового ограничения.

Известно несколько упрощений декодирования ЦУ кодов ([104, 40]). Если приписать всем узлам начального яруса одинаковые (нулевые) метрики, то после прохода по решетке алгоритмом Витерби можно ожидать, что «правильный» начальный (он же конечный)

узел будет иметь бóльшую метрику, чем большинство неправильных узлов. Можно повторить попытку, приписав конечные метрики начальным узлам и т.д. В результате нескольких итераций можно значительно уменьшить число проверяемых кандидатов по сравнению с полным перебором.

В связи с этим интересно ответить на вопрос, помогает ли представление линейных кодов ЦУ решетками упростить задачу декодирования лучших известных линейных кодов. Ответ на этот вопрос — отрицательный. Декодирование по обычным минимальным решеткам, описанным в главе 4, и другие описанные в главе 4 методы более эффективны, чем декодирование по ЦУ решеткам. Прийти к такому заключению помогает следующее утверждение, называемое *границей квадратного корня*.

Теорема 8.1. *Если для данного линейного кода логарифмическая сложность минимальной решетки равна m, то логарифмическая сложность представления кода ЦУ решеткой не может быть меньше $m/2$.*

Упражнение 8.7. Докажите теорему 8.1.

Подсказка: покажите, что при приведении порождающей матрицы представленной в минимальной ЦУ спэновой форме к обычной минимальной спэновой форме сложность увеличится по крайней мере вдвое.

Теорема 8.1 утверждает, что сложность ЦУ решетки в числе состояний не меньше квадратного корня из числа состояний обычной решетки. При этом сложность декодирования равна квадрату числа состояний. Поэтому выигрыш по сложности МП-декодирования от ЦУ представления кода не достигается. В то же время, как уже было сказано, сам способ циклического усечения сверточных кодов — эффективный инструмент поиска и описания хороших линейных кодов в форме квазициклических кодов.

Задачи

1. Могут ли два кода с одинаковыми значениями свободного расстояния и одинаковой скоростью обладать различной помехоустойчивостью?

2. Могут ли два кода с одинаковой скоростью и одинаковыми спектрами весов обладать различной помехоустойчивостью?

3. Пусть $g_1(D), ..., g_c(D)$ — порождающие многочлены сверточного кода со скоростью $R = 1/c$ и все эти генераторы имеют четный вес. Целесообразно ли использовать такой код для передачи данных?

4. Пусть $g_1(D), ..., g_c(D)$ — порождающие многочлены сверточного кода со скоростью $R = 1/c$ и сумма весов этих генераторов четна. Что можно сказать о спектре весов кодовых слов кода?

5. Пусть $g_1(D), g_2(D)$ — порождающие многочлены сверточного кода со скоростью $R = 1/2$. Постройте дуальный код.

6. Для кода с порождающими многочленами (13,17) (в двоично-восьмеричной форме) найдите аналитическое выражение для производящей функции его весового спектра и постройте графики зависимости аддитивной оценки вероятности ошибки на бит от отношения сигнал/шум для ДСК и канала с АБГШ.

7. Воспользовавшись решением предыдущей задачи, найдите первые 10 коэффициентов весового спектра. Постройте графики зависимости аддитивной оценки вероятности ошибки на бит от отношения сигнал/шум, основываясь на усеченном до 5 и до 10 членов спектре весов. Сравните полученные результаты с решением предыдущей задачи.

8. Напишите программу моделирования декодера Витерби сверточного кода. Сопоставьте результаты решения двух предыдущих задач с оценками, полученными моделированием.

9. Постройте последовательности ЦУ кодов из лучших коротких кодов, приведенных в приложении к главе 8. Убедитесь на примере кодов со скоростью $R = 1/2$, что лучшие сверточные коды — далеко не лучшие кандидаты для построения линейных кодов с большим минимальным расстоянием.

10. Докажите, что минимальное расстояние ЦУ кода не превышает свободного расстояния исходного сверточного кода.

11. Предложите алгоритм нахождения минимального расстояния ЦУ кода. Оцените его сложность.

Приложение. Таблицы сверточных кодов

Как справочный материал мы приводим короткие таблицы «хороших» сверточных кодов со скоростями вида $R = 1/n$ и $R = (n-1)/n$. Критерием предпочтения служит, в первую очередь, свободное расстояние. При одинаковом свободном расстоянии выбирался код с лучшим спектром вероятности ошибки на бит. Такой выбор не гарантирует, что приведенные в таблице коды оптимальны по вероятности ошибки на бит, более того, при различных значениях отношения сигнал/шум лучшими будут разные коды. Тем не менее, при решении практических задач выбор кодов по расстоянию и спектрам оказывается вполне оправданным.

Все коды найдены компьютерным перебором по множеству кодов. Использовалось несколько эффективных тестов для быстрого исключения неперспективных кандидатов, для оставшихся кодов выполнялся поиск расстояния и спектра. Для коротких кодов (кодов с кодовым ограничением $\nu \leq 8$) младшие коэффициенты производящих функций спектров кодов найдены с помощью итеративного численного решения систем уравнений аналогичных (8.6) (см.[48]). Для анализа длинных кодов использовался алгоритм BEAST [49].

Генераторы кодов приведены в двоично-восьмеричной форме. Для высокоскоростных кодов приведены два представления порождающих матриц: полиномиальное и в форме переменного кода, как в примере 8.4. Помимо этого, дано описание кода в форме проверочной матрицы (очень простое, поскольку дуальным к коду со скоростью $R = (n-1)/n$ служит код со скоростью $R = 1/n$).

Более подробные таблицы кодов приведены в [48], [25].

.

Таблица 8.1. Сверточные коды со скоростью $R = 1/2$

ν	(g_1, g_2)	d_{free}	Весовой спектр кода Спектр ошибки на бит
2	(7, 5)	5	1,2,4,8,16,32,64,128, 256,512,... 1,4,12,32,80,192,448,1024,2304,5120,...
3	(17,13)	6	1,3,5,11,25,55,121,267,589,1299,... 2,7,18,49,130,333,836,2069,5060,12255,...
4	(35,23)	7	2,3,4,16,37,68,176,432,925,2156,... 4,12,20,72,225,500,1324,3680,8967,22270,...
5	(65,57)	8	1,8,7,12,48,95,281,605,1272,3334,... 2,36,32,62,332,701,2342,5503,12506,36234,...
6	(155,117)	10	11,0,38,0,193,0,1331,0,7275,0,... 36,0,211,0,1404,0,11633,0,77433,0,...
7	(345,237)	10	1,6,12,26,52,132,317,730,1823,4446,... 2,22,60,148,340,1008,2642,6748,18312,48478,...
8	(657,435)	12	11 0,50,0,286,0,1630,0,9639,0,... 33,0,281,0,2179,0,15035,0,105166,0,...
9	(1537,1131)	12	1,7,19,28,69,185,411,1010,2492,5963,... 2,21,100,186,474,1419,3542,9774,25950,66831,...
10	(3217,2473)	14	14,0,92,0,426,0,2595,0,15221,0,... 56,0,656,0,3708,0,27503,0,185997,0,...
11	(6747,4325)	15	14,21,34,101,249,597,1373,3317,8014,19559,... 66, 98,220,788,2083,5424,13771,35966,93970, 246720,...
12	(15521,11677)	16	17,39,41,143,381,894,1981,4835,11934,28474,... 76,221,378,1305,3340,8902,21688,57355,151288, 384644,...
13	(31737, 22535)	16	1,17,38,69,158,414,944,2210,5482,13372,... 2,99,234,513,1316,3890,9642,24478,65474,169616,...
14	(63057,44735)	18	26,0,165,0,845,0,4844,0,28513,0,... 133,0,1321,0,7901,0,54864,0,370057,0,...
15	(152711,126723)	19	30,67,54,167,632,1402,2812,7041,18178,43631,... 174,420,534,1712,5838,14210,32898,87786, 237228,609868,...

.

Таблица 8.2. Сверточные коды со скоростью $R = 1/3$

ν	(g_1, g_2, g_3)	d_{free}	Весовой спектр кода Спектр ошибки на бит
2	(7,7,5)	8	2,0,5,0,13,0,34,0,89,0,... 3,0,15,0,58,0,201,0,655,0,...
3	(17,15,13)	10	3,0,2,0,15,0,24,0,87,0,... 6,0,6,0,58,0,118,0,507,0,...
4	(37, 33 25)	12	5,0,3 ,0,13,0,62,0,108,0,... 12,0,12,0,56,0,320,0,693,0,...
5	(75,53,47)	13	1,3,6,4,5,12,14,33,66,106,... 1,8,26, 20,19,62,86,204,420,710,...
6	(155,127,117)	15	3,3,6,9,4,18,35,45,77,153,... 7,8,22,44,22,94,219,282,531,1104,...
7	(367,331,225)	18	1,0,8,0,24,0,51,0,133,0,... 1,0,24,0,113,0,287,0,898,0,...
8	(727,623,575)	18	1,4,10,9,17,14,39,56,85,207,... 2,10,50,37,92,92,274,402,600,1579,...
9	(1545,1375,1167)	20	3,6,14,14,27,24,51,91,178,270,... 6,16,72,68,170,162,340,629,1410,2090,...
10	(3477,2531,2335)	22	7,0,27,0,55,0,161,0,413 0,... 17,0,122,0,345,0,1102,0,3214 0,...
11	(6557,5175,4713)	24	13,0,32,0,78,0,202,0,614,0,... 43,0,162,0,507,0,1420,0,4857,0,...
12	(16713,11237,10655)	24	1,0,21,0,48,0,113,0,280,0,... 1,0,85,0,257,0,737,0,2106,0,...
13	(35363,33161,24575)	26	1,8,13,24,30,43,77,138,234,372,... 2,32,58,120,172,287,546,1054,1852,3160,...
14	(61675,53237,45571)	28	6,18,12,20,42,74,109,173,318,556,... 18,76,66,112,246,526,782,1275,2550,4656,...
15	(173263,156245,104737)	30	17,0,58,0,127,0,293,0,1002,0,... 73,0,373,0,922,0,2351,0,9135,0,...

.

Таблица 8.3. Сверточные коды со скоростью $R = 1/4$

ν	(g_1, g_2, g_3, g_4)	d_{free}	Весовой спектр кода Спектр ошибки на бит
2	(7,7,5,5)	10	1,0,2,0,4,0,8,0,16,0,... 1,0,4,0,12,0,32,0,80,0,...
3	(17,15,13,13)	13	2,1,0,3,1,4,8,4,15,16,... 4,2,0,10,3,16,34,18,77,84,...
4	(37,33,27,25)	16	4,0,2,0,4,0,15,0,30,0,... 8,0,7,0,17,0,60,0,140,0,...
5	(77,67,55,51)	18	3,0,3,0,9,0,13,0,26,0,... 5,0,9,0,34,0,59,0,142,0,...
6	(171,165,133,117)	20	2,0,6,0,7,0,15,0,25,0,... 3,0,17,0,32,0,66,0,130,0,...
7	(375,327,273,231)	22	1,2,2,2,5,8,10,14,14,18,... 2,4,4,6,18,32,50,78,82,92,...
8	(671,645,537,473)	24	1,0,6,0,8,0,23,0,37,0,... 1,0,15,0,33,0,111,0,210,0,...
9	(1455,1373,1247,1157)	27	3,6,5,4,5,10,17,26,25,35,... 7,20,19,20,21,56,101,150,155,212,...
10	(3475,3137,2565,2233)	29	3,3,1,4,13,17,9,15,42,51,... 7,8,3,16,61,84,49,88,242,312,...
11	(6713,6575,5237,4731)	32	8,0,8,0,28,0,53,0,104,0,... 20,0,34,0,131,0,312,0,670,0,...
12	(15523,13171,11375,10727)	33	1,3,5,11,11,9,28,29,38,55,... 1,6,15,48,55,40,156,172,210,342,...
13	(35311,32257,27735,23433)	36	7,0,15,0,33,0,50,0,87,0,... 16,0,68,0,177,0,286,0,573,0,...
14	(63057,46665,45167,43765)	37	1,7,14,9,10,21,23,38,53,66,... 1,20,62,42,48,110,129,226,377,460,...
15	(152711,126775,121547,117363)	40	5,10,12,8,13,15,30,59,61,97,... 12,40,58,42,66,79,192,393,410,693,...

.

Таблица 8.4. Сверточные коды со скоростью $R = 2/3$

ν	$\frac{(g_1)(g_2, g_3)}{(h_1, h_2, h_3)}$	G	d_{free}	Весовой спектр кода Спектр ошибки на бит
2	(7)(7,5) (3,5,7)	$\begin{pmatrix} 3\,1\,0 \\ 2\,3\,3 \end{pmatrix}$	3	$1, 4, 14, 40, 115, \ldots$ $1, 10, 54, 226, 853, \ldots$
3	(13)(17,11) (5,13,11)	$\begin{pmatrix} 3\,3\,2 \\ 4\,3\,1 \end{pmatrix}$	4	$2, 11, 34, 109, 366, \ldots$ $5, 41, 193, 808, 3299, \ldots$
4	(25)(35,23) (13,33,25)	$\begin{pmatrix} 7\,1\,2 \\ 0\,7\,5 \end{pmatrix}$	5	$5, 18, 54, 193, 714, \ldots$ $15, 88, 370, 1640, 7116, \ldots$
5	(75)(65,43) (37,67,51)	$\begin{pmatrix} 3\ \ 7\,4 \\ 16\,1\,5 \end{pmatrix}$	6	$15, 0, 188, 0, 2664, \ldots$ $56, 0, 1351, 0, 27315, \ldots$
6	(147)(147,135) (51,137,163)	$\begin{pmatrix} 15\ \ 5\ \ 2 \\ 12\,15\,17 \end{pmatrix}$	6	$1, 17, 59, 175, 668, \ldots$ $1, 81, 402, 1487, 6793, \ldots$
7	(373)(373,225) (242,313,337)	$\begin{pmatrix} 13\,17\ \ 1 \\ 17\,26\,34 \end{pmatrix}$	8	$66, 0, 706, 0, 10727, \ldots$ $395, 0, 6695, 0, 135288, \ldots$
8	(523)(775,447) (211,577,735)	$\begin{pmatrix} 27\ \ 7\ \ 12 \\ 20\,37\,31 \end{pmatrix}$	8	$9, 58, 161, 566, 2251, \ldots$ $38, 416, 1404, 5994, 27194, \ldots$
9	(1067)(1517,1051) (477,1551,1745)	$\begin{pmatrix} 25\,33\,20 \\ 70\,31\,15 \end{pmatrix}$	8	$1, 27, 82, 302, 1201, \ldots$ $1, 447, 602, 2901, 13543, \ldots$
10	(2347)(3367,2515) (1035,2761,3117)	$\begin{pmatrix} 73\,27\,10 \\ 60\,75\,67 \end{pmatrix}$	10	$94, 0, 1132, 0, 16955, \ldots$ $671, 0, 12125, 0, 235652, \ldots$
11	(7713)(7255,4023) (2411,6547,7443)	$\begin{pmatrix} 67\ \ 61\,50 \\ 116\,37\,41 \end{pmatrix}$	10	$16, 105, 320, 1169, 4433, \ldots$ $98, 911, 3248, 14257, 61066, \ldots$
12	(10515)(17703,10675) (4137,11151,16713)	$\begin{pmatrix} 155\ \ 47\ \ 34 \\ 40\ \ 117\,165 \end{pmatrix}$	11	$53, 189, 539, 2158, 8581, \ldots$ $399, 1702, 6482, 29122, \ldots$
13	(21015)(35731,21573) (10355,33001,36263)	$\begin{pmatrix} 45\ \ 135\,134 \\ 300\ \ 57\ \ 165 \end{pmatrix}$	12	$159, 0, 2223, 0, 32034, \ldots$ $1310, 0, 26556, 0, 488998, \ldots$
14	(45317)(65213,56341) (36547,66477,40045)	$\begin{pmatrix} 321\,157\ \ 32 \\ 334\,201\,227 \end{pmatrix}$	12	$45, 164, 557, 2221, 8298, \ldots$ $308, 1747, 6673, 30248,$ $127131, \ldots$

Таблица 8.5. Сверточные коды со скоростью $R = 3/4$

.

ν	$\frac{(g_1)(g_2)(g_3,g_4)}{(h_1,h_2,h_3,h_4)}$	G	d_{free}	Весовой спектр кода Спектр ошибки на бит
2	(5)(7)(5,7) (3,1,5,7)	$\begin{pmatrix} 1&1&1&1 \\ 2&1&0&1 \\ 0&2&1&1 \end{pmatrix}$	3	$6, 23, 80, 280, 904, \ldots$ $15, 104, 540, 2520,$ $9750, \ldots$
3	(17)(13)(17,13) (2,7,15,13)	$\begin{pmatrix} 3&0&1&1 \\ 2&3&1&0 \\ 2&2&3&3 \end{pmatrix}$	4	$29, 0, 532, 0, 9853, \ldots$ $124, 0, 4504, 0,$ $124337, \ldots$
4	(31)(31)(23,35) (12,13,31,27)	$\begin{pmatrix} 1&3&0&1 \\ 0&1&2&3 \\ 6&0&3&1 \end{pmatrix}$	4	$5, 42, 134, 662, 3643, \ldots$ $10, 290, 1188, 7174,$ $48976, \ldots$
5	(53)(53)(71,43) (37,25,61,63)	$\begin{pmatrix} 1&2&3&2 \\ 6&1&1&2 \\ 4&6&1&1 \end{pmatrix}$	5	$13, 71, 326, 1626, 8320, \ldots$ $51, 474, 2978, 18918,$ $116366, \ldots$
6	(157)(171)(165,127) (41,57,153,165)	$\begin{pmatrix} 7&1&1&3 \\ 4&7&3&2 \\ 6&2&5&5 \end{pmatrix}$	6	$45, 109, 844, 3444, 20880, \ldots$ $276, 843, 9588, 44046,$ $326876, \ldots$
7	(323)(241)(357,235) (133,115,311,367)	$\begin{pmatrix} 7&4&3&2 \\ 0&1&7&6 \\ 12&2&5&3 \end{pmatrix}$	6	$10, 86, 385, 1896, 10015, \ldots$ $35, 649, 3929, 23650,$ $147068, \ldots$
8	(521)(747)(631,417) (333,263,431,727)	$\begin{pmatrix} 1&5&6&6 \\ 12&7&3&4 \\ 4&12&1&5 \end{pmatrix}$	6	$1, 44, 216, 985, 5254, \ldots$ $1, 312, 2086, 12040,$ $76631, \ldots$
9	(1073)(1341)(1455,1227) (501,677,1633,1275)	$\begin{pmatrix} 15&2&0&7 \\ 14&13&7&4 \\ 4&2&15&11 \end{pmatrix}$	7	$11, 150, 460, 2952, 13944, \ldots$ $51, 1552, 5428, 43458,$ $231088, \ldots$
10	(2361)(3403)(3351,2267) (1365,1133,3301,3643)	$\begin{pmatrix} 7&11&2&6 \\ 4&11&17&10 \\ 24&2&3&17 \end{pmatrix}$	8	$53, 265, 1275, 6694, 34923, \ldots$ $429, 2700, 16850, 103690,$ $626769, \ldots$

Таблица 8.6. Сверточные коды со скоростью $R = 4/5$

.

ν	$\frac{(g1)(g2)(g3)(g4,g5)}{(h1,h2,h3,h4,h5)}$	G	d_{free}	Весовой спектр кода Спектр ошибки на бит
2	(7)(5)(5)(7,5) (3,2,3,5,7)	$\begin{pmatrix} 1\,0\,1\,0\,0 \\ 0\,1\,0\,1\,1 \\ 2\,0\,1\,1\,0 \\ 2\,2\,0\,1\,1 \end{pmatrix}$	2	$1, 12, 53, 228, 1007, \ldots$ $1, 36, 309, 1920,$ $11404, \ldots$
3	(15)(11)(11)(17,11) (6,7,5,13,15)	$\begin{pmatrix} 1\,0\,0\,1\,1 \\ 2\,1\,0\,1\,0 \\ 0\,2\,1\,1\,0 \\ 2\,0\,2\,1\,1 \end{pmatrix}$	3	$6, 32, 185, 1019, 5440, \ldots$ $11, 184, 1627, 12063,$ $80054, \ldots$
4	(27)(33)(27)(37,5) (13,15,16,36,37)	$\begin{pmatrix} 1\,0\,0\,1\,1 \\ 2\,1\,0\,1\,0 \\ 0\,2\,1\,1\,0 \\ 2\,0\,2\,1\,1 \end{pmatrix}$	4	$30, 126, 815, 4822,$ $28632, \ldots$ $159, 990, 9076,$ $66149, 478236, \ldots$
5	(75)(75)(71)(67,41) (37,25,31,71,43)	$\begin{pmatrix} 1\,3\,1\,1\,0 \\ 2\,1\,3\,0\,0 \\ 2\,2\,1\,3\,2 \\ 6\,2\,0\,3\,1 \end{pmatrix}$	4	$4, 46, 295, 1830,$ $11887, \ldots$ $11, 297, 2876,$ $23700, 191757, \ldots$
6	(153)(151)(123)(151,123) (71,57,65,111,143)	$\begin{pmatrix} 1\,1\,3\,1\,0 \\ 2\,1\,2\,2\,3 \\ 4\,2\,1\,1\,2 \\ 6\,4\,0\,1\,1 \end{pmatrix}$	5	$22, 146, 920, 5983,$ $39259, \ldots$ $99, 1184, 10987,$ $89453, 702809, \ldots$
7	(247)(217)(257)(361,233) (117,147,135,331,373)	$\begin{pmatrix} 4\,3\,2\,1\,2 \\ 6\,4\,3\,1\,0 \\ 4\,4\,4\,1\,3 \\ 2\,4\,6\,6\,6 \end{pmatrix}$	6	$134, 0, 6010, 0,$ $262004, \ldots$ $1015, 0, 86430, 0,$ $5363399, \ldots$
8	(637)(457)(767)(655,447) (313,207,337,741,525)	$\begin{pmatrix} 7\,2\,3\,1\,3 \\ 4\,5\,1\,2\,2 \\ 4\,6\,7\,3\,0 \\ 6\,4\,6\,5\,5 \end{pmatrix}$	6	$29, 257, 1519, 9854,$ $66094, \ldots$ $181, 2531, 20278,$ $161321, 1275983, \ldots$
9	(1777)(1251)(1605)(1515,1117) (425,507,643,1355,1641)	$\begin{pmatrix} 7\;\;4\,1\,3\,3 \\ 6\;\;3\,5\,2\,2 \\ 6\;\;0\,1\,5\,4 \\ 16\,6\,4\,1\,5 \end{pmatrix}$	6	$5, 130, 835, 5072,$ $33847, \ldots$ $18, 1361, 11127,$ $84038, 666455, \ldots$

9. Алгебраический подход к сверточным кодам

В предыдущей главе мы рассмотрели многие аспекты практического применения сверточных кодов: способы их описания, основные характеристики, методы получения оценок вероятности ошибки, алгоритмы декодирования. В этой главе мы предпримем попытку построения абстрактной алгебраической модели сверточного кодера. Структурный анализ алгебраической модели сверточного кодера позволит, в частности, ответить на вопросы о минимальности кодера, катастрофичности, построить дуальное пространство и проверочную матрицу.

9.1. Кодер сверточного кода общего вида

Сверточный кодер общего вида показан на рис. 9.1.

Схема начинает работу в момент времени $i = 0$. Входные и выходные символы мы считаем элементами некоторого конечного поля $GF(q)$. На каждом такте работы кодера на его вход поступает блок из k входных информационных символов $\boldsymbol{u}_i = (u_{i,1}, ..., u_{i,k})$, $i = 0, 1, ...$ Кодер представляет собой линейную схему с памятью. На каждом такте с выхода схемы считывается очередной кодовый блок из n кодовых символов $\boldsymbol{v}_i = (v_{i,1}, ..., v_{i,n})$. Таким образом, любой полубесконечной информационной последовательности $\boldsymbol{u} = (\boldsymbol{u}_0, \boldsymbol{u}_1, ...)$ сопоставляется полубесконечное кодовое слово $\boldsymbol{v} = (\boldsymbol{v}_0, \boldsymbol{v}_1, ...)$.

Определение 9.1. *Множество всевозможных последовательностей на выходе схемы, приведенной на рис. 9.1, назовем сверточным кодом со скоростью* $R = k/n$.

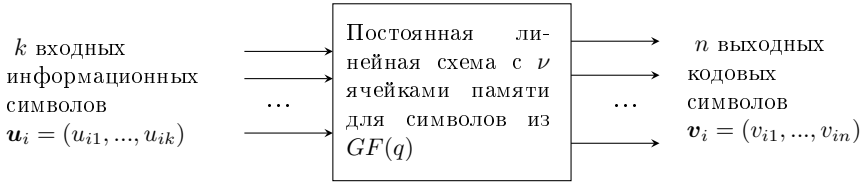


Рис. 9.1. Сверточный кодер со скоростью $R = k/n$

При изучении линейных блоковых кодов мы убедились в том, что произвольное линейное отображение множества последовательностей длины k на множество последовательностей длины n, $n \geq k$ может быть задано с помощью матрицы размера $k \times n$. Чтобы отображение было обратимым, матрица должна иметь полный ранг (ее строки должны быть линейно независимыми). Попробуем построить матричное описание кодера, показанного на рис. 9.1.

Для этого обозначим через $\boldsymbol{s}_i = (s_{i1}, ..., s_{i\nu})$ содержимое памяти кодера на такте i. Этот вектор будем называть *состоянием* кодера. Поскольку выход кодера и следующее состояние являются линейными функциями текущего состояния и входа, то для некоторых матриц A, B, E и F над полем $GF(q)$ размерности $\nu \times n$, $k \times n$, $\nu \times \nu$ и $k \times \nu$ соответственно, имеют место соотношения

$$\boldsymbol{v}_i = \boldsymbol{s}_i A + \boldsymbol{u}_i B; \tag{9.1}$$

$$\boldsymbol{s}_{i+1} = \boldsymbol{s}_i E + \boldsymbol{u}_i F. \tag{9.2}$$

Чтобы эти формулы имели смысл, нужно задать начальное состояние кодера. Примем $\boldsymbol{s}_0 = \boldsymbol{0}$. Для каждой последовательности векторов в этих формулах можно получить полиномиальную форму записи, используя формальную переменную D, имеющую смысл задержки на один такт. Домножив рекуррентные соотношения (9.1) и (9.2) на D^i, после суммирования по i приходим к соотношениям

$$\boldsymbol{v}(D) = \boldsymbol{s}(D)A + \boldsymbol{u}(D)B; \tag{9.3}$$

$$D^{-1}\boldsymbol{s}(D) = \boldsymbol{s}(D)E + \boldsymbol{u}(D)F. \tag{9.4}$$

Исключая из этих формул переменную $\boldsymbol{s}(D)$, получаем

$$\boldsymbol{v}(D) = \boldsymbol{u}(D)G(D), \tag{9.5}$$

где

$$G(D) = F(D^{-1}I_n - E)^{-1}A + B \tag{9.6}$$

и I_n обозначает единичную матрицу порядка n. В (9.6) мы предположили существование обратной матрицы в первом слагаемом суммы. Это предположение равносильно предположению о том, что последовательность состояний схемы однозначно определяется последовательностью поступающих на вход схемы блоков. Заметим, что это предположение для некоторых схем может не выполняться, но такие линейные схемы не представляют интереса в качестве схем сверточных кодеров.

Из формул (9.5) и (9.6) следует, что код общего вида может быть задан матрицей $G(D)$ размера $k \times n$, элементами которой являются дробно-рациональные функции формальной переменной D. Поэтому дальнейшее изучение теории сверточных кодов ограничивается кодами, задаваемыми матрицами $G(D)$ различного вида.

Матрицу $G(D)$, описывающую линейную схему, показанную на рис. 9.1, назовем *матричной передаточной функцией*. В дальнейшем мы сузим класс изучаемых передаточных функций, исключив те, которые заведомо не могут быть использованы для помехоустойчивого кодирования. По отношению к практически значимым матрицам мы будем применять термин «порождающая матрица сверточного кода».

Множество слов одного и того же сверточного кода может быть порождено различными кодерами или различными матрицами $G(D)$, причем некоторые матрицы могут быть предпочтительнее других по каким-то критериям. Например, при одном и том же наборе кодовых слов вероятность ошибки на бит передаваемой последовательности может различаться для различных реализаций кодера. Кроме того, сложность реализации кодирования и декодирования также зависит от способа описания кода. Рассмотрим подробнее свойства матриц $G(D)$ и их связь со свойствами соответствующих кодов.

Пусть $g(D)$ — один из элементов матрицы $G(D) = \{g_{ij}(D), i = 1, ..., k, j = 1, ..., n\}$. В общем случае он является отношением двух полиномов

$$g(D) = \frac{f(D)}{q(D)}. \tag{9.7}$$

При кодировании в соответствии с (9.5) для некоторых полиномов $y(D)$ и $x(D)$ придется вычислять произведение

$$y(D) = g(D)x(D).$$

Эту операцию можно интерпретировать как операцию фильтрации над конечным полем. Уместно назвать $g(D)$ передаточной функцией фильтра, а полиномы $x(D)$ и $y(D)$ — входом и выходом фильтра.

Определение 9.2. *Фильтр с передаточной функцией* $g(D) = f(D)/q(D)$ *называется реализуемым, если полином* $q(D) = q_0 + q_1 D + ... + q_m D^m$ *является полиномом «без задержки», т.е.* $q_0 \neq 0$.

Смысл этого определения состоит в том, что при $q_0 \neq 0$ выход фильтра в каждый момент времени зависит от предыдущих значений входной последовательности и не зависит от «будущих» значений.

Без потери общности будем считать степени числителя $f(D) = f_0 + f_1 D + ... f_m D^m$ и знаменателя $q(D)$ одинаковыми. Две канонические реализации фильтра с передаточной функцией $g(D)$ показаны на рис. 9.2 и 9.3. В каждом конкретном случае может быть отдано предпочтение одной из этих двух форм. Для нас существенно, что в любом случае количество ячеек памяти равно максимальной степени числителя и знаменателя передаточной функции фильтра.

Матрица $G(D)$ *реализуема*, если все ее элементы реализуемы. Отметим, что матрица, фигурирующая в (9.6), всегда реализуема.

Заметим, что некоторые линейные преобразования над строками матрицы изменяют матрицу $G(D)$, но сохраняют неизменным задаваемый ею код. Такие линейные преобразования над строками передаточной матрицы можно интерпретировать как изменения информационной последовательности, подаваемой на вход кодера.

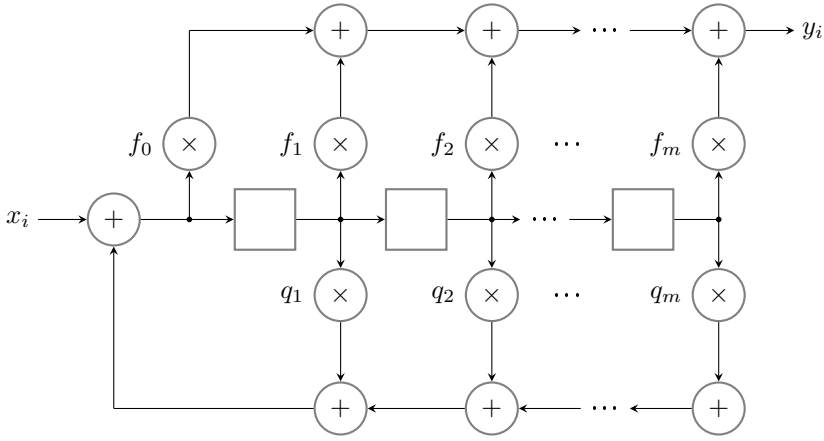


Рис. 9.2. Каноническая форма цифрового фильтра с вынесенными сумматорами (controller canonical form)

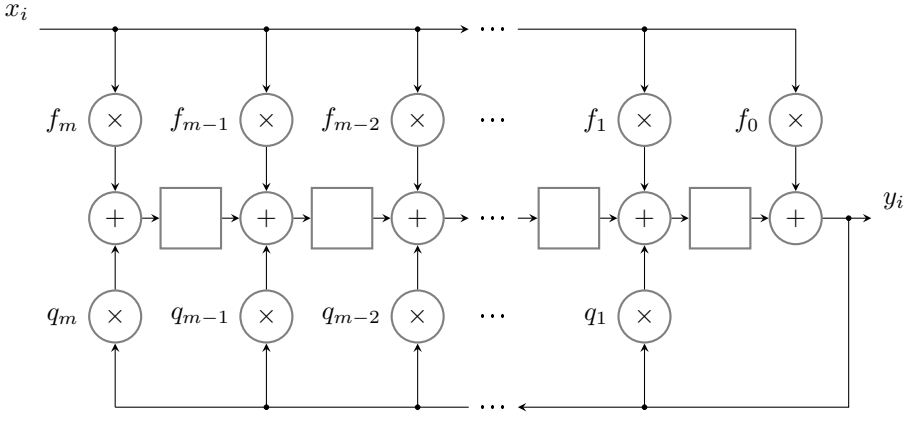


Рис. 9.3. Каноническая форма цифрового фильтра со встроенными сумматорами (observer canonical form)

Определение 9.3. *Матрицы $G_1(D)$ и $G_2(D)$ называют эквивалентными, если они порождают один и тот же код.*

Теорема 9.1. *Матрицы $G_1(D)$ и $G_2(D)$ ранга k эквивалентны, если и только если существует невырожденная матрица $T(D)$ такая, что*

$$G_1(D) = T(D)G_2(D). \tag{9.8}$$

Доказательство Пусть $\boldsymbol{u}(D)G_1(D) = \boldsymbol{v}(D)$. Кодовым словом для $\boldsymbol{u}'(D) = \boldsymbol{u}(D)T(D)$ в коде, порожденном $G_2(D)$, будет $\boldsymbol{v}(D)$. Это означает, что код, порождаемый $G_2(D)$, содержит код, порождаемый $G_1(D)$. В силу невырожденности $T(D)$ по аналогии доказывается обратное включение, т.е. коды совпадают и достаточность невырожденности имеет место.

Докажем необходимость этого условия. Пусть коды эквивалентны. Найдем соответствующую преобразующую матрицу $T(D)$ и докажем, что она невырождена. Каждая строка $\boldsymbol{g}_i(D)$ матрицы $G_1(D)$ может быть получена как некоторая линейная комбинация строк $G_2(D)$, т.е. $\boldsymbol{g}_i(D) = \boldsymbol{u}_i(D)G_2(D)$, где $\boldsymbol{u}_i(D)$ — некоторый вектор длины k, $i = 1, 2, ..., k$. Получим $G_1(D) = U(D)G_2(D)$, где строками $U(D)$ служат $\boldsymbol{u}_i(D)$, $i = 1, 2, ..., k$.. Поскольку ранг произведения матриц равен меньшему из рангов сомножителей, ранг $U(D)$ не может быть меньше ранга $G_1(D)$. Следовательно, ранг $U(D)$ равен k, т.е. $U(D)$ невырождена, что и требовалось доказать. □

Напомним, что в определении эквивалентных блоковых кодов допускалась перестановка кодовых символов. Сверточные коды имеют бесконечную длину, что создает трудности с введением аналогичного понятия эквивалентности. Тем не менее, понятно, что перестановка столбцов матрицы $G(D)$ сохраняет характеристики кода (такая перестановка эквивалентна перенумерации выходов сверточного кодера). Можно еще немного расширить понятие эквивалентности следующим образом.

Определение 9.4. *Матрицы $G_1(D)$ и $G_2(D)$ слабо эквивалентны, если для некоторой невырожденной рациональной матрицы $T(D)$ и полиномиальной перестановочной матрицы $\Pi(D)$ имеет место*

$$G_1(D) = T(D)G_2(D)\Pi(D).$$

Напомним, что перестановочной является матрица, содержащая ровно одну единицу в каждом столбце и каждой строке. Полиномиальной перестановочной матрицей мы называем перестановочную

матрицу, в которой единицы заменены мономами вида D^L конечной степени L.

Подытожим список преобразований матрицы, сохраняющих порождаемый ею код.

- Строки матрицы можно переставлять.
- Строки матрицы можно заменять их линейными комбинациями с другими строками.
- Если элементы строки матрицы имеют общим делителем моном aD^i, $i > 0$, $a \in GF(q)$, $a \neq 0$, то элементы строки можно поделить на этот моном.
- Если элементы столбца матрицы имеют общим делителем моном D^i, $i > 0$, то элементы столбца можно поделить на этот моном.

Рассматривая матрицу $G(D)$ как кодирующую матрицу, можно сформулировать некоторые естественные требования к ней.

Начнем с того, что матрица $G(D)$ должна быть реализуемой. Она должна быть *матрицей без задержки*, т.е. в разложении $G(D) = G_0 + G_1 D + ...$ матрица $G_0 = G(0)$ должна быть ненулевой (поскольку матрица и ее сдвиг $D^s G(D)$, $s > 0$, порождают один и тот же код).

Помимо этого, матрица $G(D)$ должна иметь полный ранг, равный k (в противном случае информационная последовательность не может быть однозначно восстановлена по последовательности кодовых символов). Более того, потребуем, чтобы матрица $G_0 = G(0)$ имела полный ранг.

Определение 9.5. *Реализуемая передаточная матрица $G(D)$ размерности $k \times n$ такая, что $G(0)$ имеет ранг k, называется порождающей матрицей сверточного кода.*

Поясним подробнее требование полноты ранга матрицы $G(0)$.

Если ранг $G(0)$ меньше k, то линейными преобразованиями можно привести $G(D)$ к таком виду, что в $G(0)$ будут нулевые строки. Соответствующие строки матрицы $G(D)$ будут иметь общим делителем моном D^i. Это означает, что часть информационных символов начинает участвовать в формировании кодовых слов

с некоторой задержкой, которая может быть устранена без всякого ущерба для характеристик кода.

Продолжая аналогию с линейными блоковыми кодами, введем понятие систематического сверточного кодера.

Определение 9.6. *Порождающая матрица называется систематической, если она имеет вид*

$$G(D) = \left(I_k \;\vdots\; R(D) \right),$$

где I_k — единичная матрица порядка k, а $R(D)$ — матрица из рациональных функций от D.

Из того факта, что ранг порождающей матрицы равен k, вытекает следующее утверждение.

Теорема 9.2. *Всякая порождающая матрица слабо эквивалентна систематической порождающей матрице.*

Доказательство аналогично соответствующему утверждению для линейных блоковых кодов.

Определение 9.7. *Порождающие матрицы, элементы которых — полиномы (а не дробно-рациональные функции), называют полиномиальными.*

Теорема 9.3. *Всякий сверточный код может быть задан полиномиальной порождающей матрицей.*

Доказательство. Пусть $G(D) = \{f_{ij}(D)/q_{ij}(D), i = 1, ..., k, j = 1, ..., n\}$. Обозначим $q(D) = \text{НОК}(\{q_{ij}(D)\})$. Матрица $G'(D) = q(D)G(D)$ — полиномиальная и, по определению, эквивалентна $G(D)$. □

Получается, что при поиске хороших сверточных *кодов* можно ограничиться рассмотрением только полиномиальных порождающих матриц. Это не означает, что изучение рациональных матриц не имеет смысла. Свойства *кодера* также важны по многим причинам, кодер хорошего кода может оказаться плохим или хорошим.

Например, как мы увидим, среди кодеров есть такие, использование которых нежелательно — *катастрофические* кодеры.

Как мы уже знаем из предыдущей главы, порождающая матрица называется *катастрофической*, если существует информационная последовательность бесконечного веса, которой соответствует последовательность конечного веса.

Вопрос о том, является ли кодер катастрофическим, связан с существованием полиномиального «обратного кодера», т.е. полиномиальной матрицы размера $n \times k$, которая позволит восстановить информационный вектор-полином умножением кодового вектора-полинома на эту матрицу. Если вектор конечного веса умножается на полиномиальную матрицу, то вес результата будет конечным. Поэтому катастрофический кодер не может иметь полиномиального обратного кодера. Существование полиномиальной псевдообратной матрицы является достаточным условием некатастрофичности кодера.

Отсюда видим, что систематический кодер не может быть катастрофическим, т.к. для него обратный кодер тривиален. Теорема 9.2 подсказывает, что для нахождения некатастрофического кодера данного кода достаточно привести его порождающую матрицу к систематической форме. Однако полученный таким образом кодер может быть не самым лучшим, например, с точки зрения сложности кодирования и декодирования.

Более формальное решение задачи определения катастрофичности кодера, включающее необходимые и достаточные условия, будет дано ниже после обсуждения вопроса о построении минимального кодера.

9.2. Смитова форма и существование правой обратной матрицы

Приведем без доказательства известную теорему линейной алгебры о приведении матрицы к нормальной форме Смита (1861).

Теорема 9.4. *Полиномиальная матрица $G(D)$ над полем $GF(q)$ ранга r может быть представлена в виде*

$$G(D) = A(D)\Gamma(D)B(D), \tag{9.9}$$

где $A(D)$ и $B(D)$ — полиномиальные квадратные матрицы порядка k и n соответственно, с определителями, равными 1, $\Gamma(D)$ — матрица размера $k \times n$ вида

$$\Gamma(D) = \begin{pmatrix} \gamma_1(D) & & & & & & & \\ & \gamma_2(D) & & & & & & \\ & & \ddots & & & & & \\ & & & \gamma_r(D) & & & & \\ & & & & 0 & & & \\ & & & & & \ddots & & \\ & & & & & & 0 & \cdots & 0 \end{pmatrix}. \tag{9.10}$$

Диагональные элементы $\gamma_i(D)$ называются инвариантными многочленами, $\gamma_i(D)$ являются делителями $\gamma_j(D)$ при $j > i$, $j \le r$.

Кроме того, обозначим через $\Delta_i(D)$ наибольший общий делитель всех миноров порядка i матрицы $G(D)$, тогда

$$\gamma_1(D) = \Delta_1(D), \quad \gamma_i(D) = \frac{\Delta_i(D)}{\Delta_{i-1}(D)}, \quad i = 1, 2, ..., k. \tag{9.11}$$

Доказательство теоремы можно найти в пособиях по линейной алгебре. Применительно к сверточным кодам конструктивные доказательства, включающие алгоритм приведения к смитовой форме, имеются в книге [25] и статье Форни [72]. В приложении приведена МАТЛАБ-программа приведения к смитовой форме. Эта программа следует алгоритму из работы [72].

Название «инвариантные многочлены» объясняется тем, что для всех эквивалентных матриц эти многочлены одинаковы, в отличие от матриц $A(D)$ и $B(D)$.

Пусть теперь $G(D)$ — полиномиальная матрица ранга r над полем $GF(q)$ и $q(D)$ — наименьшее общее кратное знаменателей элементов $G(D)$. Расширение теоремы Смита на рациональную матри-

цу $G(D)$ получается разложением по Смиту полиномиальной матрицы $q(D)G(D)$

$$q(D)G(D) = A(D)\Gamma_q(D)B(D).$$

Отсюда делением на $q(D)$ получаем

$$G(D) = A(D)\Gamma(D)B(D),$$

где $\Gamma(D) = \Gamma_q(D)/q(D)$,

$$\Gamma(D) = \begin{pmatrix} \frac{\alpha_1(D)}{\beta_1(D)} & & & & & & & & \\ & \frac{\alpha_2(D)}{\beta_2(D)} & & & & & & & \\ & & \ddots & & & & & & \\ & & & \frac{\alpha_r(D)}{\beta_r(D)} & & & & & \\ & & & & 0 & & & & \\ & & & & & \ddots & & & \\ & & & & & & 0 & \cdots & 0 \end{pmatrix}, \qquad (9.12)$$

инвариантные факторы $\frac{\alpha_i(D)}{\beta_i(D)}$ (после сокращения общих делителей числителя и знаменателя) обладают следующими свойствами:

- $\alpha_i(D)$ являются делителями $\alpha_j(D)$ при $j > i$, $j \le r$;
- $\beta_i(D)$ являются делителями $\beta_j(D)$ при $j < i$, $i \le r$.

Определение 9.8. *Правой обратной матрицей для заданной порождающей матрицы $G(D)$ назовем матрицу $G^{-1}(D)$ такую, что*

$$G(D)G^{-1}(D) = I_k.$$

Поскольку мы условились о том, что $G(D)$ имеет полный ранг, псевдообратная матрица существует всегда, но она может быть нереализуемой.

Определение 9.9. *Правой псевдообратной матрицей для заданной порождающей матрицы $G(D)$ назовем матрицу $\tilde{G}(D)$ такую, что*

$$G(D)\tilde{G}(D) = D^L I_k,$$

где $L \ge 0$ — неотрицательное целое число.

Теорема 9.5. *Необходимое и достаточное условие некатастрофичности кодера, заданного порождающей матрицей $G(D)$, — существование полиномиальной псевдообратной матрицы $\tilde{G}(D)$.*

Доказательство. Если полиномиальная $\tilde{G}(D)$ существует, то для некоторого $L \geq 0$ из $\boldsymbol{u}(D)G(D) = \boldsymbol{v}(D)$ следует

$$\boldsymbol{u}(D)D^L = \boldsymbol{v}(D)\tilde{G}(D).$$

Это означает, что по кодовому слову конечного веса однозначно (с задержкой L) восстанавливается информационный полином на входе кодера, и этот полином также имеет конечный вес.

Положим, напротив, что $\tilde{G}(D)$ содержит рациональные элементы. Тогда найдется полином (не моном) $q(D)$ такой, что $q(D)\tilde{G}(D)$ — полиномиальная матрица и

$$\boldsymbol{u}(D)q(D)D^L = q(D)\boldsymbol{v}(D)\tilde{G}(D) = z(D).$$

Если кодовое слово $\boldsymbol{v}$(D) имеет конечный вес, то и $z(D)$ имеет конечный вес, а

$$\boldsymbol{u}(D)D^L = z(D)/q(D)$$

имеет бесконечный вес, т.е. кодер — катастрофический. Таким образом, существование полиномиальной псевдообратной матрицы не только достаточное, но и необходимое условие некатастрофичности кодера. □

Теорема Смита поможет ответить на вопрос о существовании полиномиальной обратной или псевдообратной матрицы.

Теорема 9.6. *Рациональная порождающая матрица $G(D)$ имеет полиномиальную правую псевдообратную матрицу в том и только в том случае, если в ее разложении Смита $\alpha_k(D)$ представляет собой моном.*

Полиномиальная порождающая матрица $G(D)$ имеет полиномиальную правую псевдообратную матрицу в том и только в том случае, если в ее разложении Смита $\gamma_k(D)$ — моном.

В частности, полиномиальные правые обратные матрицы существуют, если $\alpha_k(D)$ ($\gamma_k(D)$) — мономы нулевой степени (ненулевые константы поля $GF(q)$).

Доказательство. Матрица $\tilde{G}(D)$ может быть записана как

$$\tilde{G}(D) = D^L B^{-1}(D)\Gamma^{-1}(D)A^{-1}(D), \quad L > 0. \tag{9.13}$$

Обратные матрицы к $A(D)$ и $B(D)$ существуют (т.к. их определители равны единице) и полиномиальные (по формуле обращения матриц). Запишем Γ^{-1} (для случая рациональной порождающей матрицы) в форме

$$\Gamma^{-1}(D) = \begin{pmatrix} \frac{\beta_1(D)}{\alpha_1(D)} & & & & \\ & \frac{\beta_2(D)}{\alpha_2(D)} & & & \\ & & \ddots & & \\ & & & \frac{\beta_k(D)}{\alpha_k(D)} & \\ & & & 0 & \\ & & & \vdots & \\ & & & 0 & \end{pmatrix}. \tag{9.14}$$

Если $\alpha_k(D) = a_k D^L$, $a_k \in GF(q)$, то $D^L/\alpha_i(D)$ — мономы при всех $i \le k$. Поэтому $D^L\Gamma^{-1}(D)$ — полиномиальная. Мы доказали, что все три матрицы в правой части (9.13) — полиномиальные, тем самым закончено доказательство достаточности для рациональных матриц. Для случая полиномиальной матрицы достаточность доказывается аналогично.

Докажем необходимость, т.е. что из существования полиномиальной обратной матрицы следует, что $\alpha_k(D)$ ($\gamma_k(D)$) — моном. По предположению,

$$D^L B^{-1}(D)\Gamma^{-1}(D)A^{-1}(D) = \tilde{G}(D) -$$

полиномиальная матрица. Полиномиальной будет также матрица

$$D^L\Gamma^{-1}(D) = \ B(D)\tilde{G}(D)A(D)\ .$$

Чтобы это условие выполнялось, дроби $D^L/\alpha_i(D)$ (соответственно, $D^L/\gamma_i(D)$) должны быть мономами. Значит, знаменатели должны быть также мономами степени не больше L, что и требовалось доказать. □

Следствие 9.7. *(Условие Месси–Саина [93]). Правая псевдообратная матрица для полиномиальной матрицы $G(D)$ существует, если и только если наибольший общий делитель всех ее миноров порядка k — моном переменной D.*

Доказательство. Это условие напрямую вытекает из представления инвариантных множителей через миноры в теореме 9.4 и теоремы 9.6. □

9.3. Минимальная базовая порождающая матрица сверточного кода

Мы установили, что практический интерес представляют только такие порождающие матрицы, для которых существуют правые псевдообратные.

Определение 9.10. *Порождающая матрица называется* базовой, *если она полиномиальная и имеет правую обратную матрицу.*

Такое сужение класса порождающих матриц мотивируется задачей отыскания кодеров с минимальной задержкой среди всех кодеров, порождающих код. Важно, чтобы сужение класса кодеров не вело к потере хороших кодов.

Теорема 9.8. *Всякая порождающая матрица имеет эквивалентную базовую порождающую матрицу.*

Доказательство. Для произвольной полиномиальной порождающей матрицы $G(D)$ выполним декомпозицию Смита. В качестве новой порождающей матрицы выберем $G_{\mathrm{b}}(D)$, составленную из первых k строк матрицы $B(D)$, иными словами

$$G_{\mathrm{b}}(D) = \begin{pmatrix} \boldsymbol{b}_1(D) \\ \boldsymbol{b}_2(D) \\ \ldots \\ \boldsymbol{b}_k(D) \end{pmatrix},$$

где $\boldsymbol{b}_i(D)$ — строки $B(D)$. Матрица $G_{\mathrm{b}}(D)$ эквивалентна $G(D)$, поскольку $T(D) = A(D)\mathrm{diag}(\gamma_1(D), \gamma_2(D), ..., \gamma_k(D))$ невырождена и

$G(D) = T(D)G_{\mathrm{b}}(D)$. С другой стороны, $G_{\mathrm{b}}(D)$ имеет правую обратную, поскольку в ее смитовой декомпозиции все инвариантные факторы будут равны единице. Следовательно, $G_{\mathrm{b}}(D)$ — базовая матрица, эквивалентная $G(D)$. □

Кодер, задаваемый базовой матрицей, мы также называем *базовым*.

Упражнение 9.1. Докажите, что базовый кодер некатастрофичен.

Подсказка: чтобы убедиться в существовании полиномиальной правой обратной матрицы, воспользуйтесь представлением матрицы $G_1(D)$ в нормальной форме Смита.

Упражнение 9.2. Докажите, что две базовых матрицы $G_1(D)$ и $G_2(D)$ эквивалентны тогда и только тогда, когда найдется невырожденная полиномиальная $T(D)$ такая, что $G_1(D) = T(D)G_2(D)$ и определитель $T(D)$ равен единице.

Решение: если такая матрица найдется, то эквивалентность имеет место по определению. Если матрицы эквивалентны и обе имеют полиномиальные правые обратные, то найдутся две невырожденные матрицы $S(D)$ и $T(D)$ такие, что $G_1(D) = T(D)G_2(D)$ и $G_2(D) = S(D)G_1(D)$. Отсюда $T(D) = G_2(D)G_1^{-1}(D)$, $S(D) = G_1(D)G_2^{-1}(D)$, причем $S(D)$ и $T(D)$ — полиномиальные по предположению о том, что $G_1(D)$ и $G_2(D)$ — базовые. Перемножив тождества для $S(D)$ и $T(D)$, получим $T(D)S(D) = I_k$. Определитель произведения матриц равен произведению определителей. Поскольку оба определителя — полиномы, $\det(T(D)) = 1$. □

Теорема 9.5 не только утверждает существование базовой матрицы, но и указывает способ ее построения. Наш следующий шаг — поиск среди базовых матриц матрицы с наименьшей сложностью, которая определяется длиной кодового ограничения

$$\nu = \max_{i=1,\ldots,k} \{\nu_i\},$$

где

$$\nu_i = \max_{j=1,\ldots,n} \deg(g_{ij}(D)).$$

Определение 9.11. *Кодер называется минимальным базовым, если он имеет наименьшую возможную длину кодового ограничения среди всех базовых кодеров данного кода. Соответствующая порождающая матрица называется минимальной базовой.*

Введем обозначение G_h для $k \times n$ матрицы с элементами из $GF(q)$, в которой ненулевые элементы являются коэффициентами при D^{ν_i} в тех позициях каждой строки, где степень $g_{ij}(D)$ совпадает с ν_i — наибольшей степенью элементов строки.

Теорема 9.9. *Пусть $G(D)$ — базовая с кодовым ограничением ν. Тогда следующие утверждения эквивалентны.*

A. $G(D)$ — минимальная базовая.

B. Максимальная степень μ среди всех $k \times k$ миноров $G(D)$ равна кодовому ограничению ν.

C. Матрица G_h имеет полный ранг k.

Доказательство. Докажем сначала, что утверждения B и C эквивалентны.

Представим $G(D)$ в виде

$$G(D) = \begin{pmatrix} D^{\nu_1} & & & \\ & D^{\nu_2} & & \\ & & \ddots & \\ & & & D^{\nu_k} \end{pmatrix} G_h + \hat{G}(D) = \Lambda(D) G_h + \hat{G}(D), \tag{9.15}$$

где $\hat{G}(D)$ не содержит элементов степени ν_i в строке с номером i. Поскольку G_h содержит только константы поля, степени миноров $G(D)$ не превышают $\nu = \sum_{i=1}^{k} \nu_i$ и определяются первым слагаемым в правой части (9.15). Если G_h имеет полный ранг k, то найдется минор степени $\mu = \nu$. Следовательно, из C следует B. Точно также, из разложения (9.15) следует, что при неполном ранге G_h в матрице $\Lambda(D) G_h$ не найдется минора степени ν. Тем самым, если имеет место B, то верно и C.

Докажем теперь, что из утверждения A следует C.

Без потери общности предположим, что $\nu_1 \leq \nu_2 \leq ... \leq \nu_k$. Если предположить, что C неверно, то найдется линейная комбинация строк G_h, такая, что одна из строк, например, последняя окажется равной нулю. Та же линейная комбинация, но с коэффициентами в виде мономов соответствующих степеней, позволит получить эквивалентный кодер, но наибольшая степень последней строки G_h будет меньше ν_k и кодовое ограничение кода в целом станет меньше, что противоречит предположению о минимальности. Таким образом, для минимального кодера матрица G_h невырождена.

Докажем теперь, что из утверждения B следует A. Предположим, что $\mu = \nu$ для $G(D)$, и рассмотрим матрицу $G'(D)$, эквивалентную $G(D)$. Тогда $G'(D) = T(D)G(D)$ для некоторой полиномиальной $T(D)$ с единичным определителем. Заметим, что все миноры $G'(D)$ будут такими же, как и миноры $G(D)$ для всех $T(D)$, поскольку $\det(T(D)) = 1$. Из разложения (9.15) следует, что $\nu \geq \mu$. Поэтому $G(D)$, имеющая кодовое ограничение $\nu = \mu$, минимальна.

Поскольку утверждения B и C эквивалентны, а A эквивалентно им обоим, теорема доказана. □

Теорема 9.9 указывает прямой путь к построению минимального кодера для данного кода. Если есть некоторый кодер, заданный порождающей матрицей, нужно сначала найти базовый кодер (привести его к полиномиальной форме, вычислить матрицу $B(D)$ в декомпозиции Смита и взять ее первые k строк в качестве эквивалентной порождающей матрицы). Следующий шаг состоит в проверке полноты ранга матрицы максимальных степеней. Если ранг неполон, то нужно линейными преобразованиями матрицы уменьшать степени строк до тех пор, пока ранг G_h не станет полным.

Полученная матрица будет минимальной базовой для данного кода. Подчеркнем, что минимальность тут понимается только в смысле числа ячеек памяти кодера. Некоторая другая (например, систематическая) форма кодера может иметь некоторые практические преимущества. Кроме того, поскольку минимальная базовая матрица не единственна, разные ее (эквивалентные) представления будут иметь разные характеристики в смысле, например, вероятности ошибки на бит или сложности решетки (см. главу 4).

Если задачей является построение кодера с минимальной сложностью решетки (минимальной сложностью декодирования по алгоритму Витерби), то после приведения к минимальной базовой форме достаточно привести матрицу к минимальной спэновой форме, точно также как это было описано применительно к блоковым кодам в главе 4.

9.4. Проверочная матрица, систематическая форма и дуальный код

Естественным путем нахождения проверочной матрицы кажется способ, использованный для блоковых кодов, а именно через приведение матрицы к систематической форме. Однако, как мы увидим, декомпозиция Смита дает не только минимальную базовую матрицу, но и информацию о проверочной матрице кода.

Пусть $B(D)$ — полиномиальная матрица размера $n \times n$ с единичным определителем, найденная в результате декомпозиции Смита заданной матрицы $G(D)$. Следуя доказательству теоремы 9.8, находим, что верхние k строк $B(D)$ образуют базовую матрицу $G'(D)$, эквивалентную $G(D)$. Поэтому имеет место равенство

$$B(D) = \begin{pmatrix} G'(D) \\ B_0(D) \end{pmatrix},$$

где $B_0(D)$ — матрица размера $(n-k) \times k$ ранга $n-k$. Обратная к $B(D)$ матрица имеет вид

$$B^{-1}(D) = \begin{pmatrix} G'(D)^{-1} & B_1(D) \end{pmatrix},$$

где $B_1(D)$ — матрица размера $(n-k) \times k$ ранга $n-k$, а $G'(D)^{-1}$ — правая псевдообратная к $G'(D)$. Она существует, поскольку $G'(D)$ — базовая.

Перемножая $B(D)$ и $B^{-1}(D)$, получаем

$$I_n = B(D)B^{-1}(D) = \begin{pmatrix} I_k & G'(D)B_1(D) \\ B_0(D)G'(D)^{-1} & B_0(D)B_1(D) \end{pmatrix}.$$

Отсюда заключаем, что

$$G'(D)B_1(D) = 0,$$

т.е., $B_1(D)$ — транспонированная проверочная матрица для кода, задаваемого $G'(D)$, а, следовательно, и $G(D)$. Больше того, проверочную матрицу $H(D) = B_1^{\mathrm{T}}(D)$ можно рассматривать как порождающую матрицу кода, который мы вправе назвать дуальным к рассматриваемому коду.

Теорема 9.10. *Если порождающая матрица $G(D)$ задает код, эквивалентный минимальному базовому коду с кодовым ограничением ν, то транспонированная проверочная матрица $H^{\mathrm{T}}(D)$ может быть найдена как последние правые $n-k$ столбцов матрицы $B^{-1}(D)$ в смитовой декомпозиции $G(D)$, причем кодовое ограничение минимального кодера для $H(D)$ равно ν.*

Доказательство. Первое из двух утверждений уже доказано, и установлено тождество

$$B^{-1}(D) = \Big(G'(D)^{-1} \,\Big|\, H^{\mathrm{T}}(D) \Big). \tag{9.16}$$

В матрицах $G'(D)$ и $H(D)$ выделим квадратные подматрицы G'_{00} и H_{11} порядков k и $n-k$ соответственно, и представим матрицы в виде

$$G'(D) = \Big(G'_{00}(D) \,\Big|\, G'_{01}(D) \Big), \quad H(D) = \Big(H_{10}(D) \,\Big|\, H_{11}(D) \Big).$$

Нетрудно убедиться в справедливости тождества

$$\left(\begin{array}{c|c} G'_{00}(D) & G'_{01}(D) \\ \hline 0 & I_{n-k} \end{array} \right) B^{-1}(D) = \left(\begin{array}{c|c} I_k & 0 \\ \hline H_{10}(D) & H_{11}(D) \end{array} \right).$$

Вычислив определители матриц слева и справа от знака равенства, помня, что $\det(B(D)) = 1$, получаем

$$\det(G'_{00}(D)) = \det(H_{11}(D)).$$

Переставляя столбцы матрицы $G(D)$, получим равенства между ее различными минорами порядка k и соответствующими минорами порядка $n-k$ матрицы $H(D)$. Таким образом, совпадают и максимальные степени миноров, а, значит, по теореме 9.9, совпадают и кодовые ограничения прямого и дуального кодов. $\square$

Для кодов с низкой или высокой скоростью размеры $G(D)$ и $H(D)$ сильно разнятся. Поэтому естественно желание из двух способов описания кода выбрать то, которое имеет меньшую сложность. Теорема 9.10 утверждает, что сложности, измеряемые как кодовое ограничение (сумма по строкам максимальных степеней элементов строк или суммарная длина регистров кодера), одинаковы.

Напомним, что подобное утверждение мы получили для минимальных решеток блокового кода, построенных по порождающей и проверочной матрице. Больше того, мы знаем, что все минимальные решетки блокового кода совпадают с точностью до изоморфизма. Подход к анализу сложности решеток, использованный в главе 4, также можно было использовать для получения результата, сформулированного в теореме 9.10.

Еще один путь получения проверочной матрицы кода основан на приведении порождающей матрицы к систематическому виду.

Для всякой порождающей матрицы $G(D)$ можно найти перестановку столбцов, при которой квадратная подматрица $G_0(D)$ из первых k столбцов имеет полный ранг. Соответствующий код либо совпадает, либо эквивалентен исходному, поэтому без потери общности будем считать, что это свойство имеет место. Умножив $G(D)$ слева на обратную к $G_0(D)$, получим эквивалентную матрицу в систематической форме:

$$G(D) = \left(I_k \,\vdots\, R(D) \right). \tag{9.17}$$

Подматрица $R(D)$ в общем случае состоит из рациональных элементов. Учитывая некоторые практические преимущества систематических кодеров (например, некатастрофичность), можно решать задачу поиска наилучших кодов только среди полиномиальных систематических. Оказывается, что найденные при таком ограничении коды намного уступают по своим характеристикам (например, по свободному расстоянию) кодам общего вида.

По аналогии с блоковыми кодами, из (9.17) получаем проверочную матрицу

$$H(D) = \left(-R^{\mathrm{T}}(D) \,\vdots\, I_{n-k} \right), \tag{9.18}$$

в общем случае рациональную. Эквивалентную полиномиальную матрицу можно получить, домножив строки $H(D)$ на наименьшее

общее кратное элементов строк. Если полученная матрица не будет минимальной, можно найти эквивалентную ей минимальную базовую матрицу с помощью декомпозиции Смита.

Выводы

Резюмируем основные результаты анализа алгебраических свойств сверточных кодов.

- Сверточный код со скоростью $R = k/n$ может быть задан линейной постоянной схемой с памятью, имеющей k входов и n выходов.
- Линейная постоянная схема может быть описана передаточной функцией, задаваемой матрицей с рациональными или полиномиальными элементами.
- Передаточная функция является порождающей матрицей сверточного кода, если матрица коэффициентов при нулевой степени формальной переменной имеет ранг k.
- Всякий кодер эквивалентен полиномиальному кодеру, систематическому кодеру.
- Для того чтобы кодер не был катастрофическим, достаточно существования полиномиальной правой псевдообратной матрицы по отношению к порождающей матрице кода.
- Для данного кода полиномиальный кодер, для которого существует полиномиальный правый обратный кодер, называется базовым. Базовый кодер не может быть катастрофическим.
- Декомпозиция Смита позволяет найти базовый и минимальный базовый кодер эквивалентный данному.
- Базовый кодер является минимальным базовым, если матрица коэффициентов при максимальных степенях строк невырождена.

- Кодовое ограничение минимального базового кодера равно максимальной степени его миноров порядка k.
- Кодовое ограничение дуального кода, задаваемого проверочной матрицей, совпадает с кодовым ограничением исходного кода.
- Решетка с минимальной сложностью может быть построена по приведенной к минимальной спэновой форме порождающей матрице минимального базового кодера.

Приложение. МАТЛАБ-программа декомпозиции Смита

Полиномам в программах сопоставляются целые числа. Двоичным разрядам числа соответствуют коэффициенты полиномов. Поэтому степень полиномов ограничена числом 52 — максимальной разрядностью целых чисел в МАТЛАБе. Если в процессе вычислений промежуточные результаты выходят за пределы точности, то этот факт обнаруживается (см. скрипт control) и программа прерывает работу.

Программа 9.1. Приведение матрицы к канонической форме Смита

```
function [A,D,B,MSF]=smith_form(G)
% Represents G in the form  A*D*B
% where A and B have det=1 and D is diagonal matrix
% Computes Min span form MSF of equivalent code

% Инициализация
[k,n]=size(G);  A=eye(k);   B=eye(n); D=G;
% Основной цикл
for h=1:k % iterations
% Step 1. Find nonzero and put it on diagonal
    i=h;
    while D(h,i)==0 %// i<=n,
        i=i+1;
        if i>n, error('rank(G)<k'); end;
    end;
    if i>h
 % Permute i-th column with h-th column
```

```
        t=D(:,h);  D(:,h)=D(:,i);   D(:,i)=t;
% Modify B (permute rows)
        t=B(i,:); B(i,:)=B(h,:); B(h,:)=t;
    end;
   control;
   flag=1;
    while flag==1
        flag=0;
% Обнуление h-й строки
        for i=h+1:n     % along row
           if D(h,i)~=0,
              % Kill nonzero
              a=D(h,h); b=D(h,i); % b to be killed
              % solve ax+by=d
              [d,x,y]=gcd_bin_pol(a,b);
              % find a/d, b/d;
              ad=div_bin_pol(a,d);
              bd=div_bin_pol(b,d);
              % Modify D
              D(h,h)=d; D(h,i)=0;  % h-th row
              for j=h+1:k  % other rows
                 c=D(j,h); g=D(j,i);
                 % cx+gy =
                 D(j,h)=bitxor( mul_bin_pol(c,x), ...
                                mul_bin_pol(g,y));
                 % (cb+ag)/d
                 D(j,i)=bitxor( mul_bin_pol(c,bd), ...
                                mul_bin_pol(g,ad));
              end;
              % Modify B
              % replace h-th row by lin comb
              for j=1:n  % along row
                 % B(h,:)=B(h,:)*a/d+B(i,:)*b/d
                 t=bitxor( mul_bin_pol(B(h,j),ad), ...
                                mul_bin_pol(B(i,j),bd));
                 % B(i,:)=B(h,:)*y+B(i,:)*x
                 B(i,j)=bitxor( mul_bin_pol(B(h,j),y), ...
                                mul_bin_pol(B(i,j),x));
                 B(h,j)=t;
              end;
              control;
           end; % if D(h,i)~=0
        end;  % along row
 % Обнуление h-го столбца
        d=D(h,h);
```

```
        for j=h+1:k
            c=D(j,h);
            if c>=d
                [q,r]=div_bin_pol(c,d);
                % A(:,h)=A(:,h)+A(:,j)*q
                for i=1:k
                    A(i,h)=bitxor(A(i,h), mul_bin_pol(A(i,j
                        ),q));
                end;
                D(j,h)=r;
            end;
        end;
        % Check the column
        col_sum=sum(D(h+1:k,h));
        if col_sum~=0,
            flag=1; % more iterations
            i=h+1;
            while D(i,h)==0 %|| i<=n,
                i=i+1;
            end;
            % Permute i-th row with h-th row
            t=D(h,:);  D(h,:)=D(i,:);   D(i,:)=t;
            % Modify A (permute columns)
            t=A(:,i); A(:,i)=A(:,h); A(:,h)=t;
        end;  %col_sum~=0,
    end; % while flag
    control;
end;
% Min span form
MSF=min_basic(B(1:k,1:n));
```

Программа 9.2. Проверка

```
T=mul_matr_bin(A,mul_matr_bin(D,B));
if (sum(sum(abs(T-G)))~=0),
        disp('error'),disp(h);
end;
```

Программа 9.3. Умножение двоичных полиномов

```
function x=mul_bin_pol(a,b)
x=0;
while b>0
    t=mod(b,2);
    if t>0, x=bitxor(x,a); end;
    b=bitshift(b,-1);
    a=bitshift(a,1);
end;
```

Программа 9.4. Деление двоичных полиномов

```
function [q,r]=div_bin_pol(a,b)

if (b>a), q=0; r=b; return; end;
 q=0;
while a>=b
   t=b; d=1;
   r=bitxor(a,t);
   while r>t || r>a
       t=bitshift(t,1);
       r=bitxor(a,t);
       d=bitshift(d,1);
   end;
   a=r;
   q=q+d;
end;
r=a;
```

Программа 9.5. Расширенный алгоритм Евклида для двоичных полиномов

```
function [d,x,y]=gcd_bin_pol(r,s)
% returns the GCD d of r and s
% and a pair of numbers x and y such that rx+sy=d=GCD

x=1; y=0;
```

```
u=0; v=1;
if s==0, d=r;  return; end
while s~=0
    q=div_bin_pol(r,s);
    t=x; x=u; u=bitxor(t, mul_bin_pol(q,u));
    t=y; y=v; v=bitxor(t, mul_bin_pol(q,v));
    t=s;
    s=bitxor(r, mul_bin_pol(q,s));
    r=t;
end
d=r;
```

Программа 9.6. Формирование минимальной базовой матрицы

```
function MSF=min_basic(G)
MSF=G;
[k,n]=size(MSF);
flag=1;  % 1 means that more iterations are possible
while flag==1,
    flag=0;
    Starts=ones(1,k);
    for i=1:k % over rows
        % Find starts
        a=bitand(MSF(i,:),1);
        while a(Starts(i))==0,
            Starts(i)=Starts(i)+1;
        end;
    end;
    % Sort rows in order of starts
    [Starts,v]=sort(Starts);
    MSF=MSF(v,:);
    % Kill identical starts
    % (new bad starts can appear)
    for i=1:k-1
        for j=i+1:k
            if Starts(i)==Starts(j)
                MSF(j,:)=bitxor(MSF(i,:),MSF(j,:));
                flag=1;
            end;
        end;
    end;
end;
flag=1;  % 1 means that more iterations are possible
while flag==1,
    flag=0;
    % Avoid identical ends
```

```
    Ends=ones(1,k);
    Maxdegs=zeros(1,k);
    for i=1:k
        % find max degree for row
        t=1;
        while sum(MSF(i,:)>=t)>0
            t=bitshift(t,1); Maxdegs(i)=Maxdegs(i)+1;
        end;
        t=bitshift(t,-1);
        a=bitand(MSF(i,:),t);
        % find end of row
        Ends(i)=n;
        while a(Ends(i))==0,
            Ends(i)=Ends(i)-1;
        end;
    end;
    % Kill identical ends
    for i=1:k-1
        for j=i+1:k
            if Ends(i)==Ends(j)
                flag=1;
                d=Maxdegs(i)-Maxdegs(j);
                if (d>=0)
                    MSF(i,:)=bitxor(MSF(i,:), ...
                         bitshift(MSF(j,:),d));
                else
                    MSF(j,:)=bitxor(MSF(j,:), ...
                              bitshift(MSF(i,:),-d));
                end
            end
        end
    end
end;
```

10. Длинные коды из коротких кодов

Для обеспечения высокой эффективности кодирования корректирующие коды должны иметь большую длину, при этом с ростом длины кода не должна уменьшаться доля корректируемых ошибок. Алгебраические коды, такие как коды БЧХ или Рида–Соломона, не решают этой проблемы.

Плодотворная идея, появившаяся на свет еще в 1950-х — построение длинных кодов на основе комбинаций хороших коротких кодов. Революционными решениями на этом пути были каскадные коды Форни (1966), обобщенные каскадные коды Блоха–Зяблова (1976) и турбо-коды (1993).

Каскадные коды, в частности, турбо-коды, с успехом используются в настоящее время в самых различных приложениях. Воплощение турбо-кодов в решение практических задач связи потребовало астрономических усилий инженеров и исследователей. Накопленный опыт был востребован в 2000-е годы при построении современных систем на основе кодов с малой плотностью проверок на четность (МППЧ-кодов). МППЧ-кодам посвящена отдельная глава.

Несомненные преимущества каскадных кодов — низкая сложность построения, сравнительно низкая сложность декодирования, возможность исправления пакетов ошибок.

10.1. Итеративные коды

Все рассматриваемые в данной главе методы кодирования являются частными случаями представленной на рис. 10.1 общей схемы.

Рис. 10.1. Общая схема итеративного или каскадного кодера

Информационная последовательность разбивается на блоки длины, соответствующей размерности первого из двух линейных кодов. Каждый блок кодируется, кодовые слова поступают в перемежитель и, уже в качестве информационных последовательностей второго кода, они поступают на вход второго кодера. Последовательность на его выходе является кодером итеративного (каскадного) кода.

Еще более общая схема могла бы содержать больше двух каскадов кодирования. Такие схемы тоже используются, но, принимая во внимание, что каждый новый каскад уменьшает скорость кода, чаще всего используют двухкаскадную схему, показанную на рис. 10.1. Другое естественное обобщение — комбинирование нескольких разных кодов на каждой ступени кодирования.

Хотя эффективность кода в целом сильно зависит от конкретного выбора каждого из блоков, многие особенности итеративного кодирования можно проследить на примере простейшей схемы, показанной на рис. 10.2.

Для построения *кода-произведения* или *итеративного кода* нужны два компонентных кода: (n_1, k_1) и (n_2, k_2). Обозначим минимальные расстояния этих кодов через d_1 и d_2.

Сначала k_2 блоков по k_1 информационных символов записываются в виде таблицы с k_1 строками и k_2 столбцами. Каждый столбец кодируется первым кодом, в результате получаем k_2 столбцов длины n_1. Столбцами служат кодовые слова первого кода. Затем каждая строка длины k_2 кодируется вторым кодом. В результате получаем кодовое слово длины $n = n_1 n_2$.

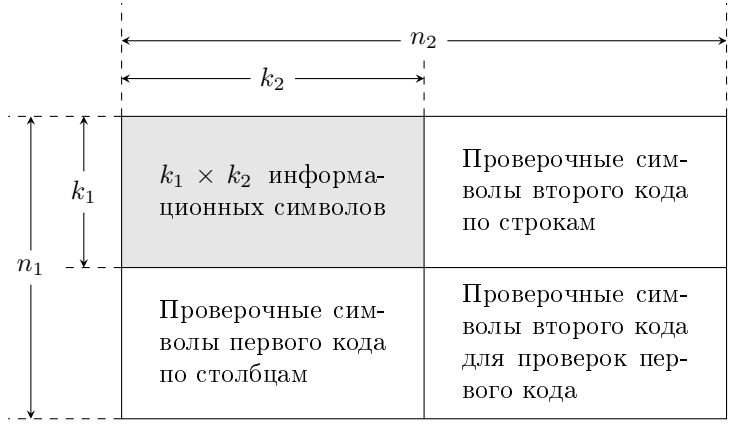


Рис. 10.2. Прямое произведение кодов

Понятно, что каждый кодовый символ представляет собой линейную комбинацию информационных символов. Поэтому имеем линейный (n_1n_2, k_1k_2)-код со скоростью

$$R = R_1R_2,$$

где R_1 и R_2 обозначают скорости компонентных кодов.

Упражнение 10.1. Докажите, что порождающая матрица кода-произведения представляет собой кронекеровское произведение порождающих матриц компонентных кодов. Напомним, что кронекеровское произведение $C = A \otimes B$ матриц $A = \{a_{ij}\}$ и $B = \{b_{ij}\}$ представляет собой матрицу, состоящую из блоков $a_{ij}B$.

Подсчитаем минимальное расстояние кода-произведения.

Если хотя бы один из информационных символов отличен от нуля, то по меньшей мере d_1 строк в таблице, приведенной на рис. 10.2, будут ненулевыми. Вес каждой ненулевой строки не меньше d_2. Поэтому для минимального расстояния кода-произведения имеет место неравенство $d \geq d_1d_2$. В действительности имеет место равенство, поскольку всегда можно найти кодовое слово веса d_2 второго кода и повторить его в строках с номерами, соответствующими номерам ненулевых позиций некоторого слова веса d_1. Итак,

имеем

$$d = d_1 d_2.$$

Этот результат нельзя назвать оптимистическим, поскольку и скорость кода-произведения, и отношение минимального расстояния к длине меньше, чем для компонентных кодов. Тем не менее, это ухудшение характеристик во многих случаях можно принять как адекватную плату за простоту описания кода. Кроме того, простота структуры кода позволяет надеяться на возможность его декодирования с малой вычислительной сложностью.

Рассмотрим декодирование кода-произведения в ДСК.

Прямолинейное решение задачи декодирования состоит в том, что принятая из канала последовательность (сумма кодового слова и матрицы ошибок) декодируется сначала по строкам, потом по столбцам.

Упражнение 10.2. Укажите комбинацию ошибок минимального веса, приводящую к ошибке декодирования при использовании декодирования по строкам и столбцам.

Решение данной задачи приводит к выводу о том, что число гарантированно исправляемых ошибок при таком декодировании равно

$$t = (t_2 + 1)(t_1 + 1) - 1 = \left\lfloor \frac{d_1 + 1}{2} \right\rfloor \left\lfloor \frac{d_2 + 1}{2} \right\rfloor - 1,$$

где $t_i = \left\lfloor \frac{d_i - 1}{2} \right\rfloor$, $i = 1, 2$, — кратности ошибок, исправляемых компонентными кодами. Поскольку расстояние кода-произведения равно произведению $d = d_1 d_2$, потенциальная кратность исправляемых ошибок составляет

$$t = \left\lfloor \frac{d_1 d_2 - 1}{2} \right\rfloor, \tag{10.1}$$

что почти вдвое больше числа ошибок, исправляемых при простом декодировании.

Более рациональным подходом к решению задачи представляется декодирование по минимуму обобщенного расстояния (МОР)

(см. параграф 7.5). Иными словами, декодер, как и прежде, выполняет декодирование по строкам, но затем по результатам декодирования назначает надежности результатам декодирования строк, например, пропорциональные числу исправленных ошибок.

Декодер столбцов сначала пробует просто декодировать столбцы, затем стирает 2 наименее надежных символа каждого столбца, и т.д. в соответствии с алгоритмом декодирования по МОР.

Анализ такого декодирования не прост (см., например, Зяблов, 1973). Попытки придумать наихудшее для такого декодера расположение ошибок наводят на (неверную!) мысль о том, что декодер исправляет все комбинации ошибок кратности порядка $d_1 d_2/2$, т.е. реализует потенциальную корректирующую способность кода. На самом деле, корректирующая способность чуть меньше.

Естественное развитие такого подхода — после декодирования строк вернуться к декодированию столбцов. Остановимся подробнее на этой возможности.

С точки зрения практического применения модель ДСК не очень интересна. Для гауссовского канала декодирование по алгоритму БКДР дает возможность оценить надежность каждого принятого на первом этапе решения и затем эффективно использовать эти оценки при выполнении второго этапа. На втором этапе можно снова применить БКДР и т.д. Эта продуктивная идея, примененная к произведению двух систематических сверточных кодов, была успешно разработана в 1993 году. Коды получили название «турбо-коды», а декодирование называют итеративным турбодекодированием или принципом распространения доверия.

Отметим важную особенность кодов-произведений. Если ошибки группируются, например, по строкам, то количество исправляемых ошибок может быть очень большим. Например, может быть исправлена комбинация из $d_2/2$ пакетов по n_1 ошибок, что намного превышает гарантированное их минимальным расстоянием число исправляемых ошибок $d_1 d_2/2$. Иными словами, коды-произведения эффективны в каналах с сильным группированием ошибок (с пакетами ошибок).

Кроме того, равномерно распределенные по всему кодовому слову ошибки довольно большого веса также могут быть исправлены. Это означает, что истинная способность кода-произведения исправ-

лять случайные ошибки существенно выше, чем оценка, вытекающая из величины минимального расстояния кода.

10.2. Каскадные и обобщенные каскадные коды

Как и для итеративного кода, кодирование выполняется в соответствии со схемой, приведенной на рис. 10.1, с той разницей, что один из кодов, назовем его внешним, выбирается недвоичным. Пусть это будет код длины n_2 с k_2 информационными символами над полем $GF(2^{k_1})$. Внутренний код — двоичный (n_1, k_1)-код. Обозначим через d_1 и d_2 минимальные расстояния внешнего и внутреннего кодов.

Сначала производится кодирование k_2 двоичных информационных блоков длины k_1 внешним кодом. Полученные n_2 символов кодового слова, каждый из которых — двоичная последовательность длины k_1, кодируется внутренним кодом. В итоге имеем кодовое слово длины $n_1 n_2$. Множество таких кодовых слов представляет собой *каскадный код*. Структура кодового слова показана на рис. 10.3.

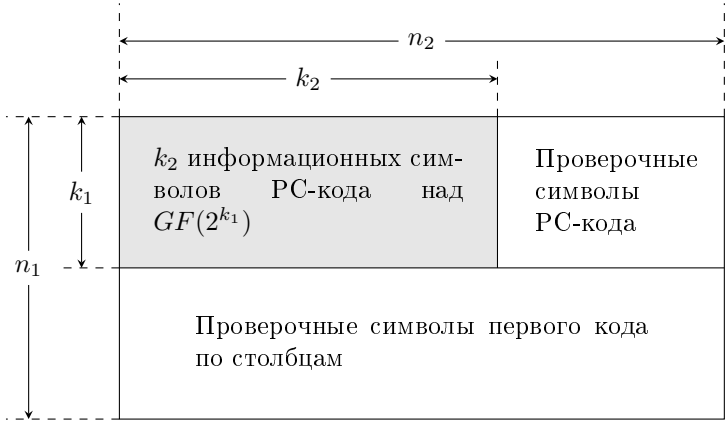


Рис. 10.3. Каскадный код

Упражнение 10.3. Докажите, что минимальное расстояние d и скорость R каскадного кода равны

$$\begin{aligned} d &= d_1 d_2, \\ R &= R_1 R_2. \end{aligned}$$

R_1 и R_2 — скорости компонентных внутреннего и внешнего кодов.

Характеристики каскадного кода, на первый взгляд, в точности такие же, как и характеристики итеративного кода. Принципиальное различие в том, что построение хороших кодов над большими алфавитами проще, чем над двоичным алфавитом. В частности, в нашем распоряжении есть класс кодов Рида–Соломона с минимальным расстоянием

$$d_2 = n_2 - k_2 + 1$$

при $n_2 \leq 2^{k_1} - 1$. (Выбрав в качестве внешнего кода расширенный код Рида–Соломона, можно получить длину $n_2 \leq 2^{k_1} + 1$.)

На основе кода Рида–Соломона в качестве внешнего кода получаем каскадный код с относительным расстоянием

$$\frac{d}{n} = \frac{n_2 - k_2 + 1}{n_2} \times \frac{d_1}{n_1}. \tag{10.2}$$

Из этой границы можно получить асимптотическую границу существования кодов с заданным относительным расстоянием. В качестве внутреннего кода можно выбрать код, удовлетворяющий асимптотической границе Варшамова–Гилберта (см. главу 3), согласно которой при достаточно большой длине n_1 существует код со скоростью

$$R_1 \geq 1 - h\left(\frac{d_1}{n_1}\right), \quad h(x) = -x \log_2 x - (1-x)\log_2(1-x).$$

Комбинируя его с РС-кодом, из (10.2), получаем асимптотическую нижнюю границу для относительного расстояния $\delta(R)$

$$\delta(R) = \frac{d}{n} \geq \left(1 - \frac{R}{R_1}\right) h^{-1}(1 - R_1), \tag{10.3}$$

где $h^{-1}(\cdot)$ обозначает обратную функцию по отношению к двоичной энтропии $h(\cdot)$. В этой границе скорость внутреннего кода R_1 играет роль параметра, который нужно выбрать так, чтобы максимизировать достижимое расстояние при заданной скорости. Поэтому окончательная формула для границы нормированного расстояния имеет вид

$$\delta(R) \geq \max_{R_1 \in [R,1]} \left(1 - \frac{R}{R_1}\right) h^{-1}(1 - R_1). \qquad (10.4)$$

Ее часто называют границей Зяблова.

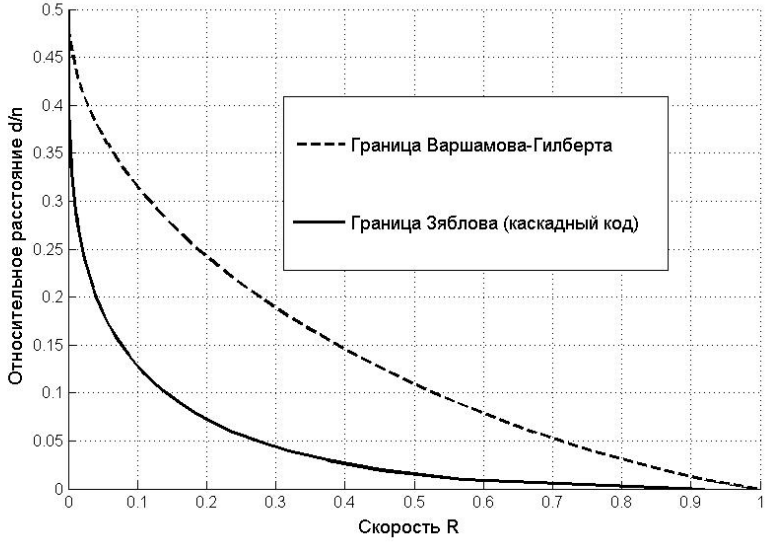


Рис. 10.4. Граница Варшамова–Гилберта и асимптотическая граница на относительное расстояние каскадных кодов

На рис. 10.4 приведено сравнение достижимой корректирующей способности произвольных линейных кодов (граница Варшамова–Гилберта) и каскадных кодов. Хотя достижимое расстояние каскадных кодов заметно проигрывает границе Варшамова–Гилберта, оно остается положительным при всех скоростях кодов. При этом код

имеет простую структуру и относительно простое декодирование (с неэкспоненциально растущей с длиной кода сложностью).

Еще раз подчеркнем, что реальная помехоустойчивость каскадных кодов заметно выше, чем гарантируемая их минимальным расстоянием.

Еще один шаг вперед по пути построения длинных и эффективных корректирующих кодов — обобщенные каскадные коды (ОКК) [31].

Поясним идею, лежащую в основе ОКК.

Начнем с того, что при использовании внешнего РС-кода над полем $GF(2^m)$ не обязательно выбирать в качестве внутреннего кода код с $k_2 = m$ информационными символами. Можно, например, выбрать k_2 кратным m, $k_2 = hm$, а на второй ступени кодирования формировать h кодовых слов РС-кода. За счет увеличения длины внутреннего кода мы выиграем по надежности решений на первом этапе декодирования, при этом сложность системы в целом увеличится незначительно.

Следующий шаг — более тонкий. Допустим, что наборы по m бит внутреннего кода имеют разную степень защиты от ошибок. Тогда разумно для каждого «слоя» из m-битовых блоков использовать свой РС-код, выбирая код с бóльшим расстоянием для менее защищенных информационных символов.

Заметим теперь, что значения m для различных слоев не обязательно должны быть одинаковыми.

При конструировании кода мы предположили, что символы внутреннего кода не одинаково защищены от ошибок. Как можно добиться этого? Одно из решений задачи использует конструкцию циклического (например, БЧХ) кода, порождающий многочлен которого представляет собой произведение нескольких минимальных многочленов, например,

$$g(x) = M_1(x)M_3(x)...M_{2t-1}(x),$$

где t — число исправляемых кодом ошибок, а $M_i(x)$ — минимальный многочлен, корнем которого является t-я степень примитивного элемента. В таком коде, манипулируя числом сомножителей в порождающем многочлене, получаем набор вложенных кодов с разными расстояниями. Для них можно получить соответствую-

щие порождающие матрицы, каждая из которых получается дописыванием строк к предыдущей матрице. Эти дописываемые строки предполагаются приведенными к систематической форме таким образом, что первые кодовые символы совпадают с информационными символами первого кода, следующая группа кодовых символов совпадает с информационными символами второго подкода, при условии, что первые информационные символы равны нулю и т.д. (см. пример 10.1) ниже.

Мы приходим к конструкции, показанной на рис. 10.5.

Рис. 10.5. Обобщенный каскадный код

Сначала выбирается (n, k)-двоичный код с минимальным расстоянием d и с несколькими (в данном случае двумя) уровнями вложения. Минимальное расстояние первого подкода (n, m_1) обозначим через d_{11}, второго подкода $(n, m_1 + m_2)$, обозначим через d_{12}. В примере на рис. 10.5 ступени всего две, поэтому $m_1 + m_2 = k$, $d_{12} = d$.

В конструкции используется несколько кодов Рида–Соломона. В данном случае мы используем два кода: (n_2, h_1) над полем $GF(2^{m_1})$ с расстоянием $d_{21} = n_2 - h_1 + 1$ и код (n_2, h_2) над $GF(2^{m_1})$ с расстоянием $d_{22} = n_2 - h_2 + 1$.

Кодирование происходит следующим образом.

1. Первые h_1 блоков по m_1 информационных символов кодируются РС-кодом первой ступени и записываются в первые m_1 строк таблицы.

2. Следующие h_2 блоков по m_2 информационных символов кодируются РС-кодом второй ступени и записываются в следующие m_2 строк таблицы.

3. Выполняется кодирование по столбцам, тем самым заполняются остальные строки таблицы.

Подсчитаем минимальное расстояние кода.

Предположим сначала, что в первой группе информационных символов были ненулевые, а во второй не было. Тогда по меньшей мере d_{21} столбцов будут иметь вес не менее d_{11}.

Предположим теперь, что во второй группе были ненулевые информационные символы. Независимо от того, какими были символы первой группы, по меньшей мере d_{22} столбцов будут иметь вес не менее d_{12}.

Обобщая эти рассуждения на произвольное число компонентных кодов и подсчитывая число информационных символов, получаем следующий результат.

Теорема 10.1. *Число информационных символов и минимальное расстояние ОКК порядка s удовлетворяет следующим соотношениям*

$$k = \sum_{i=1}^{s} h_i m_i; \tag{10.5}$$

$$d = \min_{i\in\{1,..,s\}} \{d_{1i} d_{2i}\}. \tag{10.6}$$

Пример 10.1. В качестве порождающей матрицы двоичного кода выберем

$$G = \left(\begin{array}{ccccccc} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ \hdashline 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{array}\right).$$

Первые три строки образуют порождающую матрицу (7,3)-кода с минимальным расстоянием $d_{11} = 4$, а вся матрица — (7,6)-код с минимальным расстоянием $d_{12} = 2$.

В качестве внутренних кодов выберем РС-коды над полем $GF(2^3)$: (7,5) с расстоянием $d_{21} = 3$, и (7, 2) с $d_{22} = 6$. Итоговый ОКК — (49,21)-код с минимальным расстоянием 12.

Если в качестве внутренних кодов выбрать расширенные РС-коды над полем $GF(2^3)$: (9,6) с расстоянием $d_{21} = 4$, и (9, 2) с $d_{22} = 8$, получим (63,24)-код с минимальным расстоянием 16. Оба кода — коды с наибольшим расстоянием среди известных линейных кодов соответствующей длины и размерности.

Отметим, что среди лучших известных кодов довольно много кодов из класса ОКК.

Важная идея, лежащая в основе обобщенных каскадных кодов, — использование вложенных двоичных кодов с различной корректирующей способностью — получила дальнейшее развитие во многих других приложениях. Наиболее важное из них, пожалуй, построение кодов над аналоговыми сигналами. Мы рассмотрим эти вопросы в параграфе 10.4.

10.3. Турбо-коды

Появление на свет турбо-кодов в 1993 г. [46, 45] было неоднозначно воспринято специалистами в области теории информации и кодирования. Было непривычно, что авторы, Берру, Главье и Тхитамайшима, в своих работах не формулируют точных результатов, не доказывают строгих утверждений о свойствах кодов. Они просто приводят конкретную конструкцию кода, алгоритм декодирования и кривые зависимости вероятности ошибки от характеристик канала. Скептицизм усиливался тем, что сложность декодера казалась довольно большой, и практическая ценность результатов на тот момент времени была далеко не очевидной.

Довольно скоро, в 1998 году, авторы турбо-кодов Берру, Главье и Тхитамайшима были удостоены золотой медали в честь 50-летнего юбилея теории информации за большой вклад в развитие технологий. Еще через несколько лет турбо-коды и алгоритм их де-

кодирования почти в первозданном виде стали частью стандартов связи, они используются, например, для передачи данных в стандарте мобильной связи WiMAX [118].

Поясним конструкцию турбо-кода схемой, показанной на рис. 10.6. Выберем два линейных систематических кода. Информационные символы кодируются первым кодером и они же, но «перемешанные» устройством перемежения П, кодируются вторым кодером. На выход турбо-кодера поступают три потока данных: набор информационных символов и два набора проверочных символов, причем часть проверочных символов может быть «выколота» для получения нужной скорости кода в целом.

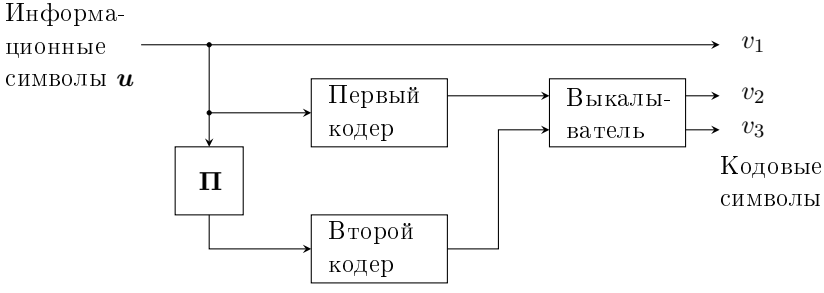


Рис. 10.6. Схема кодера турбо-кода

Обратим внимание на то, что формирование кодового слова очень похоже на кодирование кода-произведения (см. рис. 10.2). Отличие состоит в том, что проверочные символы правого нижнего прямоугольника (проверочные символы кодирования проверок первого кода) не передаются. Это изменение, конечно, повышает скорость кода, и при этом существенно снижает минимальное расстояние кода.

Схема турбо-кодера проста, но непонятно, за счет чего следует ожидать хорошей корректирующей способности. Дело как раз в том, что код, действительно, получается довольно плохим с точки зрения минимального расстояния. Если бы речь шла о декодировании (жестком или мягком) по максимуму правдоподобия, то турбо-код не конкурировал бы с более эффективными конструкциями алгебраических или каскадных кодов. Важным (точнее, един-

ственным) преимуществом турбо-кода является возможность декодирования с линейной по длине кода сложностью. Благодаря этой возможности, можно выбрать длину кода практически любой, на практике от сотен бит до десятков тысяч бит, и достичь рекордных уровней надежности, немыслимых при использовании многих других классов кодов.

Предположим, что мы имеем простой декодер для каждого из двух компонентных кодов в отдельности. Потребуем от компонентного декодера, чтобы он не только принимал решения о переданном кодовом слове, но и вычислял надежности этих решений. Применив такой декодер к символам первого кода, мы, возможно, исправим часть ошибок. За исправлением оставшихся ошибок обратимся к декодеру второго кода, который воспользуется невостребованными первым декодером проверочными символами второго кода. Возможно, ошибок станет еще меньше, и тогда можно еще раз попытаться исправить их первым кодом, и т.д.

Как мы знаем, декодирование с мягкими решениями эффективнее декодирования с жесткими решениями, поэтому мы выбираем в качестве декодеров SISO-декодеры, т.е. декодеры с мягким входом и мягким выходом (см. главу 4). Чтобы сэкономить на сложности реализации, выберем одинаковые компонентные коды. Однако если комбинации ошибок будут одинаково опасны для обоих кодов, то схема будет неэффективной. Проблема решается введением перемежителя. Если как следует перемешать информационные биты, то «плохая» комбинация ошибок (близкая к одному из кодовых слов) для одного из кодов с большой вероятностью будет безвредной для другого кода, и наоборот.

Для понимания работы схемы в целом нужно еще иметь в виду, что скорость компонентных кодов заметно выше скорости кода в целом. Если скорость кода близка к пропускной способности канала, то компонентные коды работают со скоростью выше пропускной способности. Поэтому вероятность ошибки каждого конкретного кода на первых итерациях будет заведомо большой. Важно, чтобы при этом в результате декодирования компонентного кода не происходило увеличения числа ошибок (размножения ошибок). Если компонентный код будет иметь большое минимальное расстояние d, то ошибка в пользу неправильного слова приведет к появле-

нию примерно $d/2$ дополнительных ошибочных символов. Отсюда приходим к парадоксальному заключению, что компонентный код должен быть достаточно плохим в смысле минимального расстояния.

Теперь, когда понятны эвристические предпосылки конструкции, осталось выбрать ее компоненты: код (или коды) и перемежитель. Авторы турбо-кодов остановили свой выбор на усеченных сверточных кодах с маленькой длиной кодового ограничения.

10.3.1. Выбор компонентных кодов

Сформулируем требования к компонентным кодам турбо-кода:

- Код должен иметь линейную по длине сложность SISO-декодирования.
- Код должен быть представлен в систематической форме.

Почти безальтернативным выбором в этих ограничениях является выбор усеченных сверточных кодов с небольшой длиной кодового ограничения. Требование возможности SISO-декодирования выполняется применением алгоритма БКДР. При скорости кода $R = 1/2$ систематическая форма представления кода может быть получена выбором кодов с полиномиальными порождающими матрицами вида $(1, g_2(D))$ либо с рациональными матрицами вида $(1, g_2(D)/g_1(D))$. Пример кодера кода с генераторами (в восьмеричной форме) (1, 7/5) приведен на рис. 10.7.

Этот код получен приведением к систематической форме генераторов кода (5,7). Следовательно, свободное расстояние и спектр будут такими же, как и для кода (5,7). Коды с той же длиной кодового ограничения, выбираемые из числа кодов в полиномиальной форме, имеют заметно более плохие характеристики.

В оригинальной работе по турбо-кодам [46, 45] после перебора по всем кодам с малой длиной кодового ограничения авторы выбрали код (1, 21/37) с ограничением 4. В стандартах UMTS и LTE для мобильных сетей поколения G3 рекомендован код (1, 13/15) с ограничением 3. В стандартах DVB-RCS (Return Channel over Satellite) и WiMAX (IEEE 802.16) кодовое ограничение тоже 3, но используется более сложный код, названный duo-binary кодом. Мы

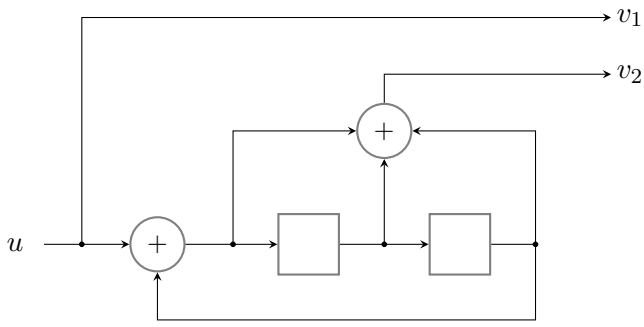


Рис. 10.7. Рекурсивный систематический сверточный кодер кода с порождающей матрицей $\big(1, (1+D+D^2)/(1+D)\big)$

будем переводить это название как *двойной* код. В стандартной терминологии теории кодирования этот код — просто сверточный код со скоростью $R = 2/4$ с порождающей матрицей

$$G(D) = \begin{pmatrix} 1 & 0 & \frac{1+D^2+D^3}{1+D+D^3} & \frac{1+D^3}{1+D+D^3} \\ 0 & 1 & \frac{1+D+D^2+D^3}{1+D+D^3} & \frac{1+D^2}{1+D+D^3} \end{pmatrix}. \qquad (10.7)$$

Одна из возможных реализаций (далеко не очевидная) показана на рис. 10.8.

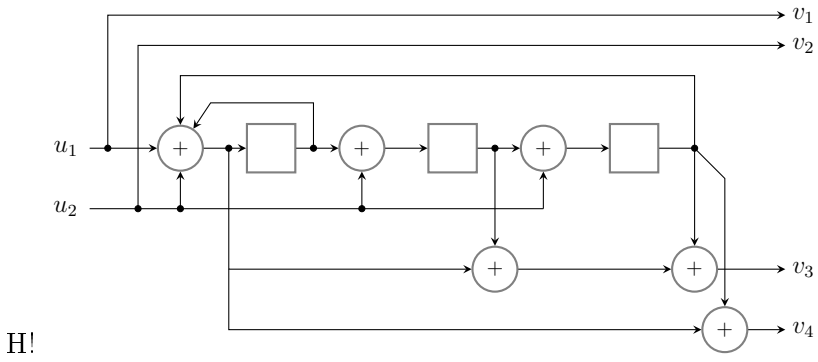
Уменьшение вдвое (с 16 до 8) числа состояний кодера, а значит и сложности декодирования, — существенный аргумент при выборе кода для мобильных устройств с учетом ограничений на сложность и потребление энергии.

10.3.2. Турбо-декодирование

Схема декодера турбо-кода показана на рис. 10.9.

Входами декодера являются зашумленные информационные символы и избыточные символы двух систематических кодеров. Более точно, на входе наблюдаются соответствующие логарифмы отношения правдоподобия L_u, L_{c1} и L_{c2}. Два МАП-декодера, работающих по алгоритму БКДР (см. параграф 4.6), пересчитывают оценки вероятностей информационных символов.

Первым обработку данных выполняет первый декодер. Новые оценки надежностей поступают на сумматор (точнее, вычитатель),



Н!

Рис. 10.8. Двойной рекурсивный кодер

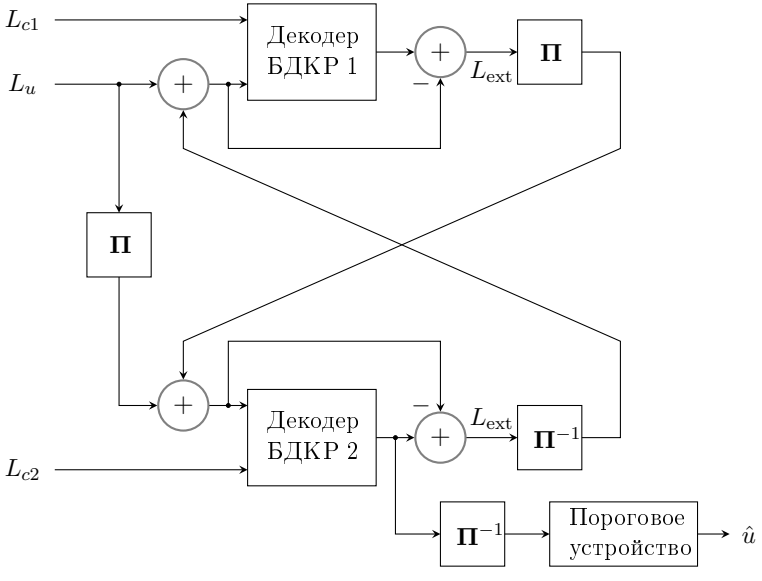


Рис. 10.9. Декодер турбо-кода

роль которого прояснится позже. Пересчитанные оценки, переставленные перемежителем П, с тем, чтобы восстановить порядок, в котором символы обрабатывались вторым кодером, суммируются с синхронно переставленными входными информационными символами и поступают на вход второго кодера.

Суммирование надежностей на входе второго кодера кажется разумным. По результатам работы первого кодера надежности некоторых символов выросли, другие, подозрительные на наличие ошибок, наоборот, уменьшились и некоторые, возможно, даже поменяли знак. Суммируя новые значения со входными, повышаем шансы на то, что второй декодер исправит еще какую-то часть ошибок.

Более формально, результатом работы декодера БКДР для информационного символа u_t является его логарифм отношения правдоподобия

$$\begin{aligned} L_t &= \log \frac{\Pr(u_t = 1, \boldsymbol{y})}{\Pr(u_t = 0, \boldsymbol{y})} = \\ &= \log \frac{\Pr(\boldsymbol{y}|u_t = 1)}{\Pr(\boldsymbol{y}|u_t = 0)} + \log \frac{\Pr(u_t = 1)}{\Pr(u_t = 0)} = \\ &= L_{\text{ext}} + L_t^{'}, \end{aligned} \tag{10.8}$$

где $\boldsymbol{y}$ — последовательность на выходе канала, $\Pr(u_t = u)$ — априорная вероятность символа, $L_t^{'}$ — логарифм отношения правдоподобия на входе декодера, а L_{ext} — приращение логарифма отношения правдоподобия, полученное в результате декодирования. За этим приращением закрепилось название «extrinsic information» (внешняя или добавленная информация).

Вернемся к рис. 10.9. В соответствии с (10.8), априорные отношения правдоподобия $L_t^{'}$ на выходе первого декодера в отсутствие сумматора сложились бы на входе второго декодера с априорными входными значениями и были бы учтены дважды. Чтобы избежать этого, на выходе первого декодера вклад априорной информации вычитается, и только добавленная информация L_{ext} после перемежителя поступает на вход второго кодера. Аналогично, с выхода второго кодера только внешняя информация передается на вход первого кодера, и т.д.

На практике используется до 10 итераций, после которых окончательные мягкие решения поступают на пороговое устройство, формирующее окончательные двоичные жесткие решения в соответствии со знаками мягких решений.

10.3.3. Практическая реализация

Кодер и декодер описаны пока не полностью: нужно выбрать способ реализации перемежителя. Хотя теоретически почти любой случайно выбранный перемежитель окажется достаточно хорошим, от выбора перемежителя зависит и помехоустойчивость и сложность реализации. Вторая проблема, которую нам придется затронуть, — способ усечения сверточных кодов (напомним, что алгоритм БКДР предполагает применение к блоковым кодам). И еще одна проблема, которую мы обсудим в этом параграфе, — проблема точности вычислений. В оригинальном алгоритме БКДР операции выполняются над вещественными числами, а результат вычисляется с помощью вычисления логарифма. Поскольку в большинстве процессоров и интегральных схем допустимы только целочисленные операции, алгоритм БКДР должен быть адаптирован к условиям реального применения.

Перемежитель

Перечислим основные требования, которые предъявляются к перемежителю в схеме турбо-кодера. Эти требования трудно точно формализовать, поскольку влияние тех или иных параметров на характеристики системы связи устанавливается, в основном, компьютерным моделированием.

- Хорошие перемешивающие свойства. Можно, например, потребовать, чтобы символы, первоначально отстоящие друг от друга на S позиций, после перемешивания отстояли друг от друга не менее, чем на S позиций (так называемый S-random interleaver).

- Отсутствие циклов: расстояние между любыми двумя позициями должно изменяться после перестановки. В главе 11, посвященной кодам с малой плотностью проверок на четность (МППЧ), мы остановимся подробнее на представлении кодов графами. Короткие циклы в графе соответствуют комбинациям ошибок, трудно корректируемым при использовании итеративных алгоритмов. Причина состоит в том, что работа двух декодеров перестает быть независимой, одна и та же

информация используется многократно, что приводит к дополнительным ошибкам.

- Сложность реализации. При аппаратной реализации желательно, чтобы адреса записи и считывания вычислялись по простым формулам, а не хранились в виде таблиц.

- Бесконфликтность (contention-free). Это требование появилось относительно недавно в связи с широким распространением параллельных вычислений. Схема и алгоритм турбодекодирования проигрывают в этом отношении другому классу итеративно-декодируемых кодов — КМПЧ-кодам. Распараллеливание между по меньшей мере двумя процессорами можно реализовать, организовав одновременную работу двух БКДР-декодеров, при условии, что они никогда не обращаются к одним и тем же ячейкам памяти одновременно. Перемежители, удовлетворяющие этому условию, получили название *бесконфликтных*.

В стандарте UMTS перемежитель выполнен в виде прямоугольной таблицы, которая сначала заполняется построчно, затем выполняются перестановки внутри строк и внутри столбцов. Считывание выполняется по столбцам.

В более новом стандарте LTE учтено требование бесконфликтности. Пусть M — число параллельно работающих процессоров. Достаточное условие бесконфликтности на окне длины W для перестановки ψ — следующее [101, 96]

$$\left\lfloor \frac{\psi(u_1 W + n u)}{W} \right\rfloor \neq \left\lfloor \frac{\psi(u_2 W + n u)}{W} \right\rfloor \qquad (10.9)$$

для любых $0 \leq \nu < W$ и $0 \leq u_1, u_2 < M$, $u_1 \neq u_2$. Это условие должно выполняться как для прямой перестановки π, так и для обратной перестановки π^{-1} перемежителя.

Условию (10.9) удовлетворяет перемежитель на основе квадратичного полинома (Quadratic Permutation Polynomial (QPP), [101, 96])

$$\pi(i) = (f_1 i + f_2 i^2) \mod K,$$

где K обозначает длину информационного блока, константы f_1 и f_2 зависят от длины блока, f_1 взаимно просто с K, и все простые делители K являются делителями f_2.

Выбор конкретных значений параметров f_1 и f_2 влияет на вид обратной перестановки, которая, с учетом ограничений на сложность реализации, должна описываться полиномом как можно меньшей степени. В стандарте LTE эта задача решена для всех значений K, предусмотренных стандартом.

Усечение кодов

Мы рассматривали обычное усечение и циклическое усечение (см. главу 8). Оба способа применяются на практике. Укажем на особенности их использования по отношению к составляющим кодам турбо-кода.

При обычном усечении кодов, заданных полиномиальными (не дробно-рациональными) генераторами с кодовым ограничением μ, на вход кодера в конце подаются μ нулевых информационных символов. В результате в конце работы кодер приходит в нулевое состояние. В случае циклического усечения начальное состояние не нулевое, оно определяется информационными символами, и в конце пакета вместо нулевых символов подаются те же символы, что были в кодере в начале работы. Конечное состояние совпадает с начальным. Обычное усечение, в отличие от циклического, приводит к некоторой потере скорости. Эта потеря несущественна при кодировании блоков большой длины.

Особенность рекурсивных кодеров состоит в том, что состояние кодера не определяется однозначно последними поступившими на вход символами. Благодаря обратной связи каждый информационный символ влияет на все последующие символы. Тем не менее, мы всегда можем добиться нужного состояния кодера, если каждый из ν последних бит на его входе будем выбирать с учетом символов, формируемых в цепи обратной связи.

Еще одна особенность усечения кодов с схеме турбо-кодера состоит в том, что последними символами двух кодеров служат разные символы битовой последовательности. Это означает, что нужно заранее вычислить место расположения последних символов в

памяти перемежителя и продумать порядок кодирования таким образом, чтобы оба кодера оказывались в нужном состоянии на последнем такте работы.

Ограничение на точность вычислений

Рассмотрим проблему точности вычислений при выполнении алгоритма БКДР. Искомые функции правдоподобия формируются как логарифмы отношения сумм произведений. Перемножаемые числа — вероятности, представляющие собой экспоненты логарифмов отношений правдоподобия, поэтому, по сути, задача сводится к многократному вычислению функций вида

$$f(x,y) = \log(e^x + e^y).$$

В процессорах с плавающей точкой вычисления выполнялись бы с использованием разложения в степенной ряд Тэйлора. Для использования в мобильных устройствах и при высоких требованиях к скорости вычислений такой поход не может быть использован. Чтобы упростить задачу, используют приближенное представление $f(x,y)$ в виде

$$f(x,y) \approx \max\{x,y\} + e^{-|y-x|}.$$

Эту задачу можно решить с помощью таблицы экспоненциальной функции. Если использование большого числа обращений к памяти нежелательно, можно использовать кусочно-линейную аппроксимацию экспоненты.

Возможна еще более грубая аппроксимация

$$f(x,y) \approx \max\{x,y\} + C,$$

где

$$C = \begin{cases} 0, & |y-x| \geq 1.5; \\ 0.5, & |y-x| < 1.5. \end{cases}$$

Наконец, самая простая аппроксимация имеет вид

$$f(x,y) \approx \max\{x,y\}.$$

Версия декодера БКДР, построенная на такой аппроксимации, называется Max-Log MAP декодером. Считается, что потери помехоустойчивости при использовании Max-Log MAP декодера вместо точных вычислений составляют примерно 0.25 дБ.

Подводя итоги параграфа, посвященного турбо-кодам, отметим, что на современном этапе развития теории кодирования и программно-аппаратных средств реализации устройств кодирования и декодирования турбо-коды успешно решают широкий круг задач передачи дискретной информации. Хотя изначально они рассматривались как длинные коды, позволяющие приблизиться к пределу Шеннона, их успешно применяют для передачи коротких пакетов сообщений, начиная примерно от 200 бит.

При скорости кодов $R > 1/2$, особенно при большой длине блоков, турбо-коды все больше вытесняются кодами с малой плотностью проверок на четность, которые мы рассмотрим в главе 11.

10.4. Кодированная модуляция

В предыдущих главах мы рассматривали коды над конечными полями, в основном, двоичные коды. Переносчиками сигналов в канале связи, как известно, служат модулированные гармонические колебания (электрические или электромагнитные волны). Параметры фрагментов гармонических колебаний, такие как амплитуда, частота и фаза, принимают некоторые дискретные значения, зависящие от передаваемых сообщений. Приемник вычисляет оценки этих параметров и тем самым восстанавливает передаваемую информацию.

Такая упрощенная модель мало отражает достаточно изощренные способы формирования сигналов (методы модуляции) и методы демодуляции в современных системах связи. Эти вопросы лежат далеко за рамками теории кодирования. Формальная модель, которую достаточно рассматривать с точки зрения применения кодов, предполагает, что в системе связи входом модулятора, а значит, выходом кодера, являются не абстрактные символы, а числа, вещественные или комплексные. Соответственно, выходом канала являются те же числа, но подвергшиеся некоторым искажениям.

Для достижения высокой спектральной эффективности используется многопозиционная модуляция, т.е. приходится различать много дискретных значений параметров сигналов. Содержанием данного раздела курса является краткое обсуждение задачи защиты от ошибок в таких системах связи. Оказывается, что эта «недвоичная» задача имеет простое и эффективное решение с помощью двоичных кодов.

10.4.1. Коды и сигналы

Привычными методами модуляции для систем связи XX века были амплитудная, частотная и фазовая, их названия указывают на параметры гармонического сигнала, изменением которых передается информация. В современной технике связи, как в радио-, так и в проводных каналах, основным ресурсом является диапазон частот. Сегодня конкурентами в борьбе за место в стандартах будущих поколений являются OFDM (orthogonal frequency devision multiplexing, т.е. мултиплексирование на основе ортогонального частотного разделения) и FBMC (filter bank multicarier, т.е. многочастотное разделение с помощью банков фильтров). Вместе эти два способа организации передачи данных объединяются под названием «multicarrier data transmission» (многочастотные системы передачи данных), основой которых является разбиение доступной полосы частот на большое число (например, 2^{10}) частотных подканалов и использование спектрально-эффективного кодирования в каждом подканале.

Число подканалов выбирается равным степени двойки, поскольку модуляции и демодуляция выполняются с помощью быстрого дискретного преобразования Фурье. Высокая эффективность системы в целом достигается за счет того, что скорость передачи в каждом подканале выбирается с учетом спектральной плотности мощности шума в соответствующем диапазоне частот. Тем самым задача построения эффективной системы связи сводится к спектрально-эффективному кодированию для каждой отдельной несущей частоты.

При фиксированной несущей частоте сигнала модуляции подлежат его фаза и амплитуда. Иными словами, сигнал как функция

времени записывается в форме

$$s(t) = \sum_{i=-\infty}^{\infty} \left(\sqrt{2}a_i \cos\omega_0 t + \sqrt{2}b_i \sin\omega_0 t\right) g(t - iT). \qquad (10.10)$$

В этой формуле индекс i — номер интервала времени, T — длительность одного сигнального интервала ($1/T$ — скорость передачи, измеренная как число сигналов, переданных за единицу времени), ω_0 — несущая частота, $g(t)$ — огибающая элементарного сигнала, (a_i, b_i) — числа из дискретного множества, представляющего собой сигнальный алфавит.

Вид огибающей $g(t)$ не играет роли в дальнейшем рассмотрении. Сам сигнал может иметь длительность, бóльшую, чем интервал T, теоретически может быть даже бесконечным, но сдвиги сигналов во времени на величину T должны быть ортогональны друг другу и образовывать ортонормированный базис пространства сигналов. Важно также, чтобы спектр сигнала $g(t)$ не был шире отведенной для данного частотного подканала полосы частот W (точнее говоря, сигнал должен быть ортогонален сигналам, передаваемым на других несущих частотах).

Сигналы вида (10.10), определяемые набором $\{(a_i, b_i)\}$, называют в общем случае сигнальным множеством, а при определенных ограничениях, которые мы укажем позже, — *сигналами квадратурной амплитудной* модуляции (КАМ, QAM). Числа (a_i, b_i) называют синфазной и квадратурной составляющей, либо значениями вещественной и мнимой части сигнала (это соответствует экспоненциальной форме записи гармонического сигнала в соответствии с формулами Эйлера).

Уточним, что формула (10.10) описывает сигнал в частотном подканале OFDM, но не в подканале FBMC. При использовании FBMC используется только косинусная составляющая, при этом спектральная эффективность эквивалентная OFDM (даже несколько выше) достигается за счет того, что формируется дополнительная косинусная составляющая, сдвинутая на полпериода по отношению к первой. Такую модуляцию называют КАМ со смещением (offset QAM). С точки зрения кодирования нет необходимости различать КАМ и КАМ со смещением. Достаточно считать, что

кодовыми символами являются точки двумерного евклидова пространства (в дальнейшем будут рассматриваться пространства более высокой размерности).

Примеры сигнальных множеств приведены на рис. 10.10. Множество на рис. 10.10, *а*, представляет собой набор сигналов четверичной амплитудной модуляции (4-АМ). В этом случае используется только одна из двух ортогональных составляющих сигнала. Далее, на рис. 10.10, *б*, следует другое множество, состоящее из четырех так называемых биортогональных сигналов. Этот конкретный набор можно интерпретировать как 4-КАМ или как набор из четырех сигналов фазовой модуляции (4-ФМ). На остальных рисунках показаны ФМ и КАМ сигналы с более высокой кратностью модуляции. Черные точки соответствуют сигналам. Белые точки приведены для пояснения способа построения сигнальных множеств требуемого размера.

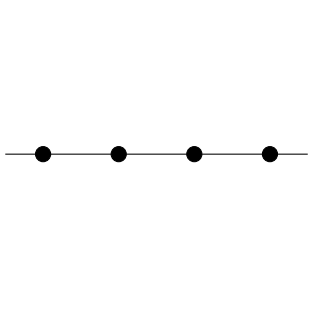
В канале с АБГШ вероятность ошибки для двух сигналов, отстоящих друг от друга на евклидовом расстоянии d_{E}, вычисляется по формуле, аналогичной (1.7)

$$P_{e2} = Q\left(\frac{d_{\mathrm{E}}}{\sqrt{2N_0}}\right) = Q\left(\frac{d_{\mathrm{E}}}{2\sqrt{E}}\sqrt{\frac{E}{N_0}}\right), \qquad (10.11)$$

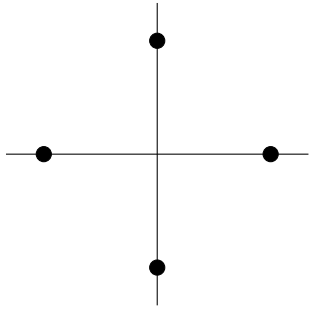
где N_0 — спектральная плотность мощности шума, E. Отсюда видно, что при фиксированном отношении сигнал/шум E/N_0 предпочтительны сигнальные множества, для которых велико отношение квадрата минимального попарного расстояния между сигналами к средней энергии сигналов d_{E}^2/E.

Упражнение 10.4. Считая все сигналы равновероятными, подсчитайте отношение квадрата минимального попарного расстояния между сигналами к средней энергии сигналов для сигнальных множеств *а*) и *б*) на рис. 10.10. Какое из двух множеств предпочтительнее?

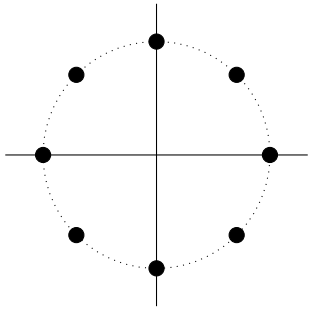
При заданной полосе W максимальная скорость передачи в сигналах в единицу времени определяется так называемой скоростью Найквиста и соответствует длине сигнального интервала $T = 1/(2W)$ (см. главу 1). Обозначим через M мощность множества



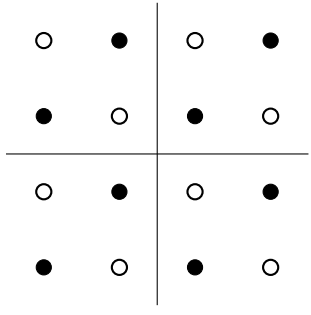
а) 4-АМ



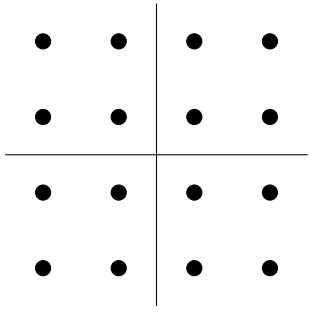
б) 4-КАМ



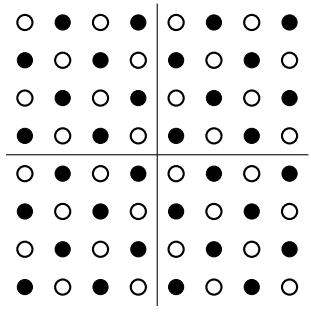
в) 8-ФМ



г) 8-КАМ



д) 16-КАМ



е) 32-КАМ

Рис. 10.10. Примеры сигнальных множеств

разрешенных пар $\{(a_i, b_i)\}$, т.е. *кратность* КАМ. Это означает, что в полосе шириной W Гц скорость передачи не превысит

$$\log_2 M/T = 2W \log_2 M \text{ бит/с},$$

что определяет максимальную возможную спектральную эффективность (1.9)

$$\beta = 2 \log_2 M \text{ бит/с/Гц}.$$

Конечно, при наличии шума необходимо использование избыточных кодов, при этом спектральная эффективность уменьшается.

Поучительные графики достижимой спектральной эффективности в зависимости от отношения сигнал/шум на бит приведены на рис. 10.11. Пропускная способность канала (предел Шеннона) достигается при бесконечном алфавите и гауссовском распределении вероятностей на входных значениях. Практические системы кодирования строятся на основе линейных кодов, поэтому сигнальные точки выбираются кодером практически с одинаковой вероятностью. Принимая во внимание это ограничение, пересчитана предельно достижимая спектральная эффективность для 4, 16 и 64 сигнальных точек. Это соответствует применению двоичных кодов в сочетании с 4-КАМ, 16-КАМ, 64-КАМ соответственно.

Из приведенных графиков заключаем, что при использовании двоичных кодов уточненная минимальная энергия на бит при скорости $R = 1/2$ ($\beta = 1$) равна 0.19 дБ (без учета ограничения на входной алфавит она оценивалась как 0 дБ , см. главу 1). Для достижения спектральной эффективности $\beta = 1.5$ нужен двоичный код со скоростью $R = 3/4$. Тогда согласно графику потребуется не менее 1.62 дБ. Альтернативным решением является использование 16-КАМ, что позволит передавать информацию при 1.04 дБ. Предел Шеннона составляет 0.86 дБ, следовательно теоретические потери от использования неоптимальных сигналов 16-КАМ не превышают 0.2 дБ. Попытка обойтись двоичными кодами обходится намного дороже, примерно в $1.62 - 0.86 = 0.76$ (дБ) на каждый переданный по каналу бит.

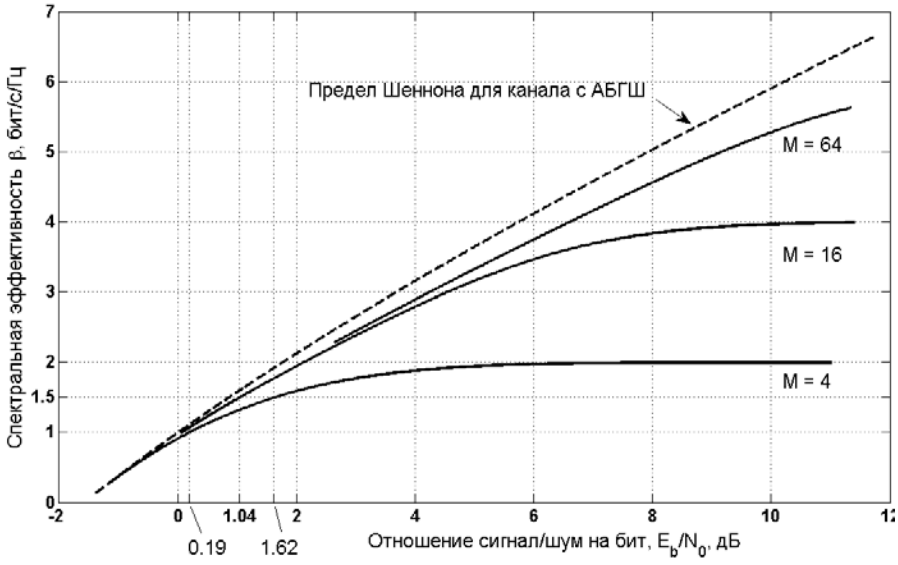


Рис. 10.11. Спектральная эффективность как функция отношения сигнал/шум на бит при различных значениях кратности модуляции

Из этих примеров понятно, что для достижения предельной спектральной эффективности нужны хорошие коды над сигнальными алфавитами больших размеров.

Прежде чем перейти к построению кодов, рассмотрим модель демодулятора сигналов, описываемых формулой (10.10). На выходе канала с АБГШ наблюдается зашумленная версия сигнала

$$r(t) = s(t) + n(t), \tag{10.12}$$

где $n(t)$ — шум со спектральной плотностью мощности $N_0/2$. Одна из возможных реализаций приемника для сигнала одной несущей показана на рис. 10.12.

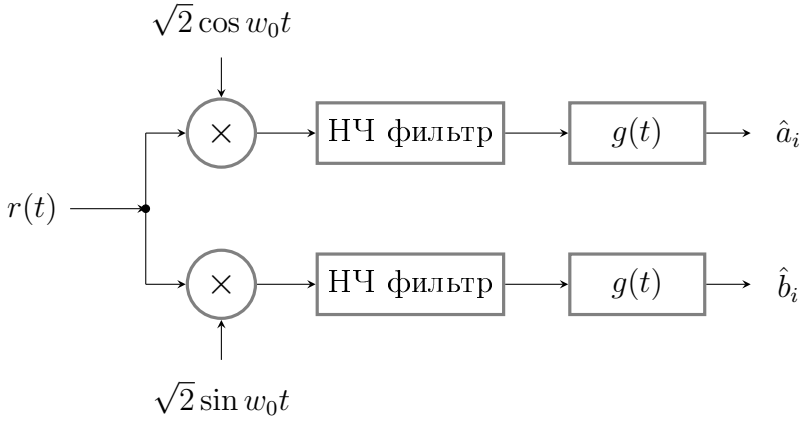


Рис. 10.12. Демодулятор сигналов КАМ

Сначала, с помощью умножения на синфазную и квадратурную гармоники с последующей фильтрацией, происходит выделение оценок низкочастотных составляющих

$$r_{\rm c}(t) = \sum_{i=-\infty}^{\infty} a_i g(t-iT) + n_{\rm c}(t); \tag{10.13}$$

$$r_{\rm s}(t) = \sum_{i=-\infty}^{\infty} b_i g(t-iT) + n_{\rm s}(t). \tag{10.14}$$

Поскольку $\{g(t-iT)\}$ образует ортонормированный базис, коэффициенты разложения $\{a_i\}$ и $\{b_i\}$ можно оценить, вычисляя проекции на сдвиги импульса вида $g(t)$. Эта операция выполняется с помощью показанных на схеме согласованных фильтров. Выходом фильтров служат оценки

$$r_{ci} = a_i + n_{ci}; \tag{10.15}$$

$$r_{si} = b_i + n_{si}. \tag{10.16}$$

В реальных многочастотных системах все каналы обрабатываются совместно с помощью методов полифазной фильтрации, основанных на применении быстрого преобразования Фурье.

В силу линейности и ортонормированности выполняемых приемником преобразований АБГШ непрерывного времени в формуле (10.12) преобразуется в пары дискретных отсчетов (n_{ci}, n_{si}), с гауссовским распределением, нулевым математическим ожиданием и дисперсией $N_0/2$. Пара (r_{ci}, r_{si}) интерпретируется как точка в сигнальной плоскости, а оценки координат сигнальной точки $(\hat{a}_i, \hat{b}_i)$ вычисляются как координаты ближайшей сигнальной точки в евклидовом пространстве.

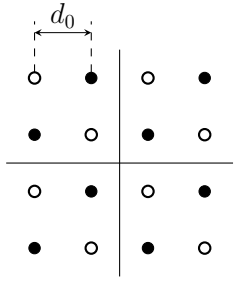
Как и в случае каналов с дискретными алфавитами, прием каждого сигнала в отдельности неэффективен. Мы должны построить достаточно длинные коды, словами которых будут последовательности пар сигналов. В кодах должна быть предусмотрена некоторая избыточность для возможности исправления ошибок, а декодер должен совместно обрабатывать всю принятую последовательность для вынесения решения о кодовом слове.

Зададим любое отображение информационных последовательностей на множество КАМ сигналов. Конкретное правило отображения, даже безызбыточное, определяет код, обладающий естественной неравномерной защищенностью информационных символов. Рассмотрим, например сигналы 16-КАМ, показанные на рис. 10.13. Все множество точек (белых и черных) содержит 16 точек, отстоящих друг от друга на расстоянии не меньше d_0, этим точкам можно сопоставить информационные последовательности из четырех бит. Разбиение этого множества на два подмножества по 8 точек (белых и черных) порождает подмножества с минимальным расстоянием d_1. На рис. 10.13, *б*, показано множество черных точек, разбитое, в свою очередь, на два подмножества с минимальным расстоянием d_2. Дальнейшее разбиение дает пары точек на расстоянии d_3.

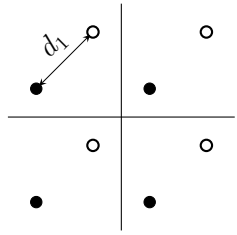
Если выбор подмножества при каждом разбиении описывать двоичным символом (например, 0 — черное, 1 — белое подмножество), то получим нумерацию точек, в которой инверсия одного из битов в номере соответствует переходу в подмножество, отстоящее на расстоянии d_i, где i — номер бита. Пример нумерации, обладающей такими свойствами, показан на рис. 10.13, *г*.

Расстояния d_i связаны соотношениями

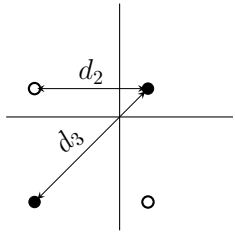
$$d_1^2 = 2d_0^2;\ d_2^2 = 4d_0^2;\ d_3^2 = 8d_0^2; \tag{10.17}$$



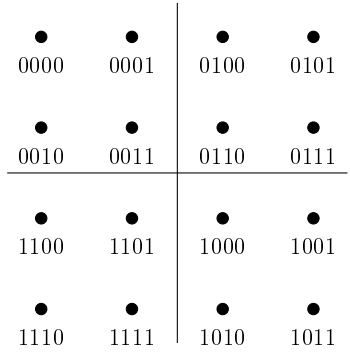
а) 16-КАМ = (8-КАМ) + (8-КАМ)



б) 8-КАМ = (4-КАМ) + (4-КАМ)



в) 4-КАМ = (2-КАМ) + (2-КАМ)



г) 16-КАМ

Рис. 10.13. Разбиение сигнального множества на подмножества

Нетрудно видеть, что эта конструкция — идеальный пример вложенных кодов, которые были востребованы для построения обобщенных каскадных кодов в параграфе 10.2. Идея совмещения разбиения сигнальных множеств на подмножества с последующим применением каскадных и обобщенных каскадных кодов принадлежит В.В. Гинзбургу [33]. Более подробное описание конструкций кодов приведено в [36].

Блоковые коды над сигнальными множествами весьма эффективны, но, также как и блоковые коды над конечными алфавитами, проигрывают сверточным аналогам при заданных ограничениях на сложность декодирования в каналах с мягкими решениями. Во многих современных стандартах применяют сверточные коды над сигнальными алфавитами, полученные как развитие конструкции Унгербоека [108] (1982), рассматриваемой в следующем параграфе.

Если говорить о приоритетах в разработке принципов кодовой модуляции, то правильно считать, что два класса кодов, блоковые коды Гинзбурга и сверточные коды Унгербоека появились на свет одновременно и независимо друг от друга. Первый черновик работы Унгербоека был представлен в редакцию в 1977 году, но был отправлен автору на доработку, которая длилась более 5 лет. Коды Гинзбурга докладывались на конференциях задолго до его журнальной публикации 1984 года.

10.4.2. Сигнально-кодовые конструкции

В этом параграфе мы опишем простую конструкцию сверточных кодов над сигнальными множествами. Наше описание не повторяет в точности описание Унгербоека [108], но для большинства конкретных примеров коды получаются эквивалентными.

Общий вид схемы, включающей как частный случай кодер Унгербоека, показан на рис. 10.14. Предполагается, что сигнальное множество КАМ содержит $M = 2^{m+1}$ точек. В каждый элемент времени i на вход системы поступает блок $\boldsymbol{u}_i$ из m двоичных информационных символов. Сверточный код со скоростью $R_{\mathrm{c}} = m/(m+1)$ и с некоторым кодовым ограничением ν преобразует последовательность блоков $\boldsymbol{u}_1$, $\boldsymbol{u}_2$,... длины m в избыточную последовательность $\boldsymbol{v}_1$, $\boldsymbol{v}_2$,... блоков длины $m+1$. Далее следует устройство, отображающее каждый блок $\boldsymbol{v}_i$ в соответствующий КАМ-сигнал (a_i, b_i).

Точное описание модуляционного кода требует указания, во-первых, сверточного кода, во вторых — правила отображения. Прежде чем мы конкретизируем выбор кода, поясним, какой и за счет чего можно получить выигрыш.

Заметим, что скорость передачи составляет m бит/сигнал. Такую же скорость передачи можно получить без кодирования, отоб-

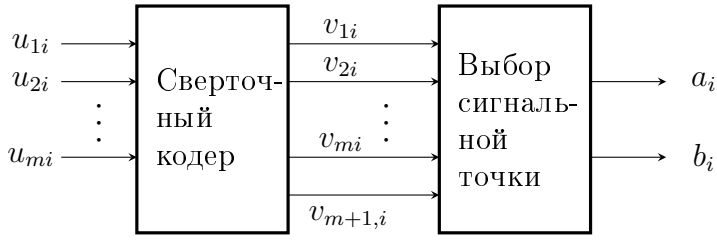


Рис. 10.14. Кодер системы с кодированной модуляцией на основе сверточного кода

ражая блоки $\boldsymbol{u}$ длины m прямо на точки множества из $M' = 2^m$. При одинаковой средней энергии такое множество будет иметь бóльшее минимальное расстояние, чем множество из $M = 2^{m+1}$, применяемое при наличии кодирования. В каждый конкретный момент времени может быть использована любая из $M > M'$ точек, поэтому вероятность неправильного приема одного сигнала в системе с кодированием больше, чем в системе без кодирования. Однако, благодаря кодированию, далеко не любая последовательность сигналов является «кодовой». Вероятность ошибки — вероятность принятия решения в пользу неправильного кодового слова. Эта вероятность при правильном выборе кода будет намного меньше, чем вероятность ошибки в системе без кодирования.

Воспользуемся теперь тем фактом, что не все биты в двоичной записи номера сигнальной точки одинаково критичны к шуму в канале. Рисунок 10.13 и формула (10.17) показывают, что разница в защищенности битов очень велика: квадрат следующего расстояния вдвое больше предыдущего. Напомним, что энергетический выигрыш кодирования при больших отношениях сигнал/шум определяется кодовым расстоянием (см. (8.27)). Если сравниваются два кода с расстояниями (или свободными расстояниями) d_a и d_b, то, поскольку энергия пропорциональна квадрату евклидовой метрики, энергетический выигрыш одного кода по отношению к другому оценивается как

$$\eta = 10 \lg \frac{d_a^2}{d_b^2} \text{ дБ}. \tag{10.18}$$

Это означает, что каждый следующий бит защищен от шума на

$10 \lg 2 = 3.01$ дБ сильнее предыдущего. Отсюда следует, что нет необходимости защищать избыточными кодами те биты, которые обладают достаточной надежностью без кодирования.

Пример схемы, учитывающей эту возможность, приведен на рис. 10.15. В данном частном случае предполагается, что защищены избыточностью два младших бита. Обратимся к примеру КАМ на рис. 10.13, *г*. Нетрудно убедиться простым перебором по всем сигналам, что любые два сигнала с номерами, отличающимися в младшем бите, отстоят друг от друга на расстояние не меньше d_0, а различие в двух младших разрядах соответствует расстоянию не меньше $d_1 = \sqrt{2}d_0$. При фиксированных двух защищенных младших битах расстояние между сигналами, отличающимся в одном из двух старших битах, не меньше $d_2 = 2d_0$.

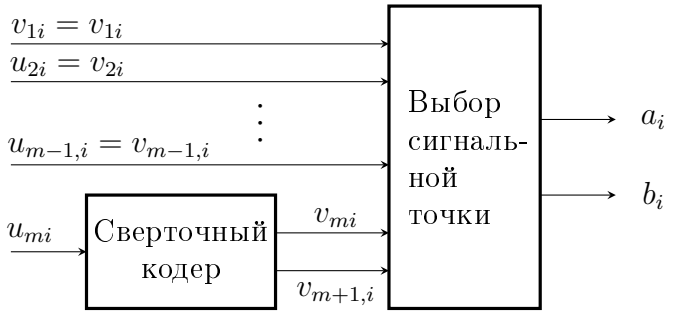


Рис. 10.15. Кодер системы с кодированной модуляцией с разделением битов на кодируемые и некодируемые

Предположим, что требуемая скорость передачи составляет 3 бита на один сигнал КАМ. При такой скорости можно воспользоваться 8-КАМ, получим минимальное расстояние между сигнальными точками равное d_1. Если добиться с помощью кода, чтобы ошибок в младших разрядах не было, то схема с кодированием обеспечит расстояние d_2. Следовательно, предельный энергетический выигрыш для этого примера

$$\eta = 10 \log_{10} \frac{d_2^2}{d_1^2} = 3.01 \text{ дБ}.$$

Предположим теперь, что для кодирования выбран сверточный

код со скоростью $R_c = 1/2$ с кодовым ограничением 1 с генераторами $(1, 1 + D)$ (в двоично-восьмеричной форме (1,3)). Свободное расстояние кода в метрике Хэмминга равно 3 и соответствует слову минимального веса (10 11). Нетрудно догадаться, что две ближайшие друг к другу сигнальные последовательности отличаются по меньшей мере в двух КАМ-сигналах, причем в одном сигнале не меньше чем на d_0 (подблок 10) и другом сигнале на d_1 (подблок 11). Итак, евклидово расстояние всего кода не меньше

$$d_f = \min\{\sqrt{d_0^2 + d_1^2}, d_2\} = \sqrt{3}d_0.$$

Энергетический выигрыш по отношению к 8-КАМ равен

$$\eta = 10 \log_{10} \frac{3d_0}{2d_0} = 1.76 \text{ дБ}.$$

Подсчитаем выигрыш, который можно получить при использовании кода с кодовым ограничением 2. В коде (5,7) слову минимального веса 5 соответствует последовательность (11 01 11), которой соответствует евклидово расстояние

$$d_f = \min\{\sqrt{d_1^2 + d_0^2 + d_1^2}, d_2\} = 2d_0.$$

Энергетический выигрыш равен 3.01 дБ, следовательно, использование более мощного сверточного кода в данной конструкции нецелесообразно.

Если потребуется бóльший выигрыш при той же скорости, можно применить код со скоростью $R_c = 2/3$, защищающий 3 бита из четырех. Изучив на рис. 10.13, *г*, все пары сигналов, отличающиеся только в первом бите, убеждаемся, что ошибка в незащищенном бите соответствует расстоянию d_3, следовательно потенциальный выигрыш в сравнении с некодированной 8-КАМ равен 6 дБ. Согласно подсчетам Унгербоека [108], кодовое ограничение сверточного кода должно быть не меньше 7. Нужно отметить, что ни один из примеров классической работы [108] не был заметно улучшен, несмотря на то, что прогресс в области вычислительной техники дает намного больше возможностей для поиска хороших кодов.

Рассмотрим кратко декодирование сверточных модуляционных кодов.

Как и для сверточных кодов над конечными полями, подходящим алгоритмом декодирования является алгоритм Витерби. Чтобы применить его, нужно построить описание кода с помощью решетки и сформулировать правило вычисления метрик путей в решетке кода.

«Классическое» решетчатое описание кода, приведенное в [108], задается решеткой с параллельными ребрами. Узлы решетки соответствуют состояниям сверточного кодера, переходы между узлами решетки определяются кодированными битами, параллельные ребра соответствуют всевозможным значениям некодированных битов, для которых возможен данный переход из состояния в состояние. Для уменьшения сложности практической реализации можно упростить описание кода.

Рассмотрим пример, иллюстрирующий вычисление метрик для одного такта работы декодера Витерби, т.е. для вычисления метрик ребер на одном ярусе решетки. На рис. 10.16 через $\boldsymbol{r}$ обозначена точка, соответствующая принятому на данном такте сигналу КАМ, через d_{ij}, $i,j = 1,2$, — евклидовы расстояния до ближайших точек решетки. Мы знаем, что декодирование в евклидовой метрике в канале с АБГШ эквивалентно декодированию по максимуму правдоподобия. Отсюда видно, что для декодирования младших битов номера сигнальной точки можно временно игнорировать старшие биты и тогда декодирование по максимуму правдоподобия можно выполнять практически точно так же, как в обычном канале с АБГШ. После того, как для данного ребра будет принято решение относительно младших битов, декодер сформирует решение о старших битах. В данном случае скорее всего (но совсем не обязательно!) победителем станет точка $\boldsymbol{s}_{11}$, и тогда результатом декодирования будет индекс 0011.

Отметим, что для декодирования нужны не расстояния, а их квадраты, причем эти квадраты расстояний вычисляются как суммы квадратов расстояний по каждой координате. Детальный анализ показывает, что вычисление метрик ребер не требует операций умножения.

Упражнение 10.5. Сформулируйте правило вычисления метрик декодера Витерби для кода над сигнальным алфавитом 16-КАМ.

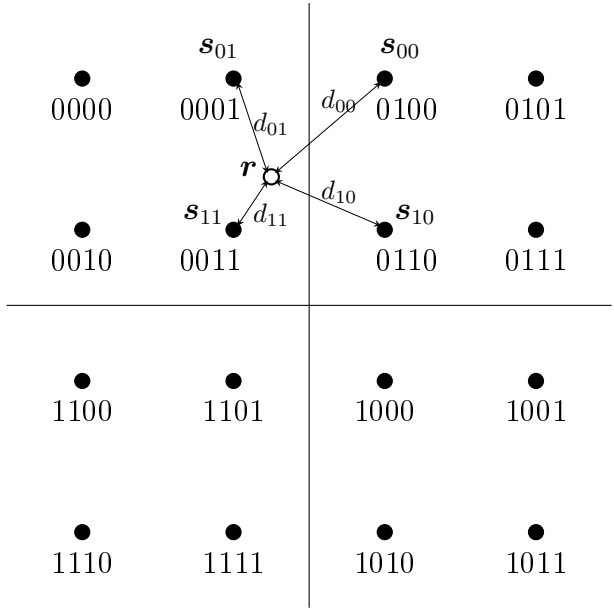


Рис. 10.16. Вычисление метрик при декодировании сверточных модуляционных кодов

Схема на рис. 10.15 предполагает использование одного избыточного бита на каждый двумерный сигнал КАМ. Для некоторых приложений такое снижение спектральной эффективности является слишком большим. Чтобы уменьшить его, можно рассматривать не одну, а несколько пар сигналов совместно. Соответственно, вместо одного дополнительного бита на двумерный сигнал, применяя код со скоростью $R_c = 3/4$, затратим один дополнительный бит на два двумерных сигнала, иначе говоря, на четырехмерный сигнал. Коды для такой многомерной модуляции построены Веем [114]. Эти коды стали основой стандартов проводной связи V.34, ADSL. Кроме энергетического выигрыша, многомерные коды Вея обеспечивают так называемую инвариантность к повороту (rotational invariance), что делает их устойчивыми по отношению к сбоям синхронизации фазы.

Еще одно, почти очевидное, обобщение конструкции Унгербоека состоит в том, что биты индексов сигнальных точек могут быть

распределены в несколько битовых потоков и к ним можно применить корректирующие коды, избыточность которых согласована с надежностью передачи битов [82]. Потенциально такая конструкция эффективнее кодов Унгербоека или Вея, но, из-за более высокой сложности, она не нашла до настоящего времени практического применения.

10.4.3. Кодированная модуляция с перемешиванием битов

Мы рассмотрели несколько классов весьма эффективных кодов над сигнальными алфавитами. Эти коды допускают относительно простое декодирование по максимуму правдоподобия с помощью алгоритма Витерби. Можно ли на этом пути приблизиться к пределам Шеннона на скорость передачи при заданной вероятности ошибки? Ответ, к сожалению, отрицательный. На практике декодер Витерби может быть реализован только для относительно небольших длин кодовых ограничений (обычно 4–5, в любом случае, не больше 10). При этом системы на основе сверточных модуляционных кодов или каскадные коды на их основе при требуемой вероятности ошибки порядка 10^{-5} затрачивают на передачу каждого бита не меньше 2–2.5 дБ сверх предела Шеннона.

Почти сразу после «турбо-революции» середины 1990-х годов возникла идея применить турбо-коды, а позднее и коды с малой плотностью проверок на четность, для построения модуляционных кодов. Наиболее плодотворной оказалась идея совмещения итеративно-декодируемых кодов с так называемой *кодированной модуляцией с перемешиванием битов* (КМПБ) (bit-interleaved coded modulation, BICM) [58].

Блок-схема системы с КМПБ показана на рис. 10.17. Блок информационных символов $\boldsymbol{u}$ поступает на вход кодера (например, кодера турбо-кода), где преобразуется в кодовое слово $\boldsymbol{v}$. Далее следует перемежитель, выполняющий псевдослучайную перестановку π. В КАМ-модуляторе с числом сигналов $M = 2^m$ кодовые символы разбиваются на подблоки длины $M = 2^m$ и отображаются на точки сигнального множества. Способ отображения важен для обеспечения эффективности системы в целом, ниже мы опишем его подроб-

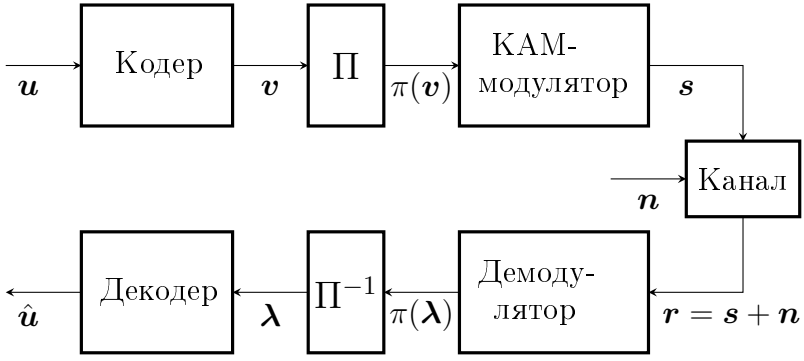


Рис. 10.17. Структурная схема системы на основе кодированной модуляции с перемешиванием битов

нее. На приемной стороне зашумленная последовательность сигналов поступает в демодулятор, формирующий мягкие решения по каждому биту индекса сигнала КАМ. Последовательность оценок значений отношений логарифмов правдоподобия символов кодовых слов обозначена на схеме через $\boldsymbol{\lambda}$. Выходом демодулятора является переставленная последовательность $\pi(\boldsymbol{\lambda})$. После обратной перестановки последовательность $\boldsymbol{\lambda}$ поступает на вход декодера, который вычисляет оценку $\hat{\boldsymbol{u}}$ переданной информационной последовательности $\boldsymbol{u}$.

Заметим, что, хотя в системе используется двоичный линейный код, после отображения кодовых блоков на точки сигнального множества код становится нелинейным. Отсюда следует, что вероятность ошибки может зависеть от передаваемой последовательности и может быть большой для некоторых информационных последовательностей. Смысл перестановки (перемешивания) двоичных символов кодовых слов состоит в том, чтобы устранить такую зависимость.

Эффективность КМПБ определяется отображением кодовых подблоков на сигнальные точки и правилом вычисления оценок надежностей двоичных кодовых символов. Рассмотрим сначала отображение или, что то же самое, правило назначения индексов сигнальным точкам. Идея состоит в применении кода Грэя.

Код Грэя — безызбыточный код длины m объема $M = 2^m$, со-

стоящий из всевозможных последовательностей длины m, упорядоченных таким образом, что каждое слово отличается от соседей не более, чем в одной позиции. Общее правило построения кода Грэя произвольной длины можно найти в учебниках по комбинаторике и алгоритмам, например, в [1]. Коды небольших длин легко построить «вручную».

Смысл использования кода Грея в системе КМПБ состоит в том, что типичная ошибка при передаче сигнала КАМ переводит сигнал в соседнюю точку. При последующем использовании двоичного кода желательно, чтобы такая ошибка искажала как можно меньше кодовых символов. Пример такой индексации для 16-КАМ показан на рис. 10.18.

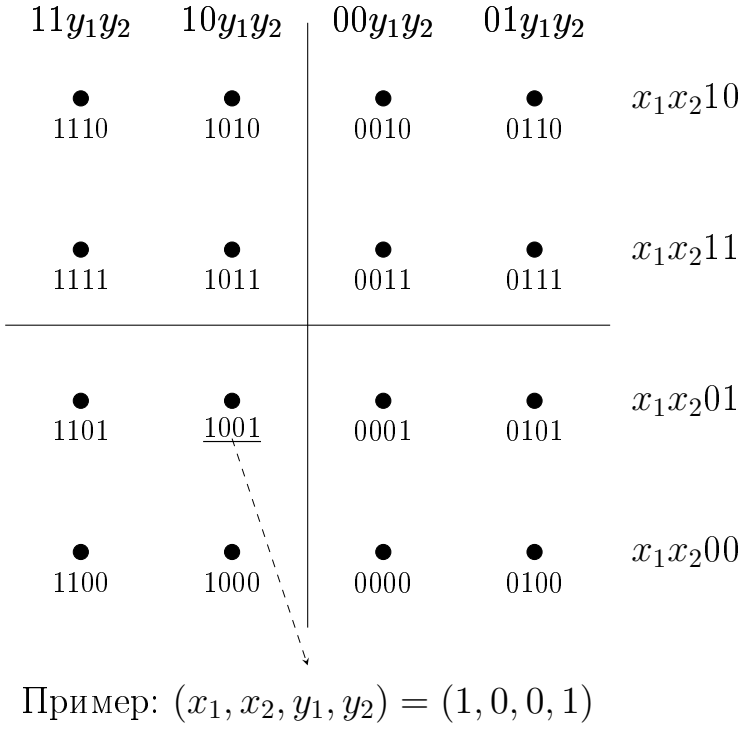


Рис. 10.18. Отображение кодовых символов на сигналы 16-КАМ с использованием кода Грея

Для построения отображения использованы два кода Грэя длины 2. Индекс из четырех битов составляется из двух частей. Первая — код Грэя координаты точки по оси X, вторая часть — код Грэя координаты по оси Y. Прямой проверкой убеждаемся, что переход в соседнюю точку соответствует изменению одного бита в индексе. Это свойство не выполнялось для отображения, выбранного для модуляционного кодирования сверточными кодами на рис. 10.16. В том отображении соседствовали точки с индексами, отличающимися в трех битах.

Рассмотрим теперь вычисление надежностей кодовых символов. Обозначим через $p(\boldsymbol{r}|\boldsymbol{s})$ двумерное условное распределение на точках плоскости при передаче сигнальной точки $\boldsymbol{s}$. Поскольку положение сигнальной точки M-КАМ, $M = 2^m$, определяется индексом или подблоком $\boldsymbol{c} = (c_1, c_2, ..., c_m)$, то вместо $p(\boldsymbol{r}|\boldsymbol{s})$ можно рассматривать вероятности $p(\boldsymbol{r}|\boldsymbol{c})$. Надежность элемента c_i задается логарифмом отношения правдоподобия

$$\lambda_i = \log \frac{p(\boldsymbol{r}|c_i = 1)}{p(\boldsymbol{r}|c_i = 0)} = \log \frac{p(\boldsymbol{r}, c_i = 1)}{p(\boldsymbol{r}, c_i = 0)}. \qquad (10.19)$$

Последнее равенство верно, поскольку при использовании линейных кодов мы считаем значения кодовых символов 0 и 1 равновероятными. Совместные вероятности в этой формуле вычисляются как суммы вида

$$p(\boldsymbol{r}, c_i = a) = \sum_{\boldsymbol{c}:\ c_i = a} p(\boldsymbol{r}, \boldsymbol{c}). \qquad (10.20)$$

В правой части (10.20) имеем $M/2 = 2^{m-1}$ слагаемых, каждое из которых — экспоненциальная функция квадрата евклидова расстояния между принятым сигналом-вектором $\boldsymbol{r}$ и сигнальной точкой, определяемой комбинацией $\boldsymbol{c}$. Реализация такого правила вычисления надежностей требует выполнения арифметических операций с плавающей точкой и вычисления экспонент и логарифмов.

Можно упростить вычисления, используя те же способы, что и при декодировании турбо-кодов. В частности, можно оставить в сумме (10.20) одно наибольшее слагаемое. При последующей подстановке вероятности в (10.19) отпадет необходимость в вычислении логарифмов и экспонент. Разумеется, такое упрощение приведет к

некоторой погрешности в вычислениях и может сказаться на предельной эффективности кодирования.

В целом, предельная спектральная эффективность КМПБ ниже спектральной эффективности кодированной модуляции на основе сверточных кодов. Типичная разница составляет порядка 0.2 дБ, что является адекватной платой за возможность применения длинных кодов с итеративным декодированием. В итоге КМПБ является почти безальтернативным кандидатом на использование в телекоммуникационных стандартах будущих поколений для тех систем связи, где требуется предельно высокая эффективность использования спектра сигнала.

Задачи

1. Можно ли улучшить минимальное расстояние итеративного кода, если для кодирования по строкам и столбцам использовать вместо одного фиксированного кода несколько разных кодов с одинаковой скоростью?

2. В продолжение предыдущей задачи: можно ли получить выигрыш по расстоянию, если при кодировании использовать наборы эквивалентных кодов? Можно ли добиться положительного эффекта, изменяя способ перемежения символов после кодирования первым кодом?

3. Как повлияют модификации кода, указанные в первых двух задачах на эффективность простого декодирования итеративного кода по строкам и по столбцам?

4. Приведите примеры кодов, полученных итерацией циклических кодов длины 7. Сравните их с лучшими известными кодами.

5. Приведите примеры каскадных кодов, получаемых с помощью РС-кодов длин 7,..., 15. Сравните их с лучшими известными кодами.

6. Приведите примеры кодов, полученных итерацией циклических кодов длины 7. Сравните их с лучшими известными кодами.

7. Приведите примеры обобщенных каскадных кодов на основе циклических кодов длины 15. Сравните их с лучшими известными кодами.

11. Коды с малой плотностью проверок на четность

Коды с малой плотностью проверок на четность (МППЧ-коды) были введены Галлагером в 1962 году [76, 5], примерно полвека назад. Не только сами коды, но и метод декодирования, описанный в диссертации и статье Галлагера, являются прототипом кодов и методов декодирования, ставших объектом огромного интереса теоретиков и практиков кодирования в последнее десятилетие.

Интерес к МППЧ-кодам возобновился на рубеже веков благодаря работе Мак-Кея [92], в которой было показано, что их характеристики близки к характеристикам турбо-кодов, при том, что декодирование проще и хорошо поддается распараллеливанию.

В данной главе мы рассмотрим алгоритм декодирования МППЧ-кодов, обсудим некоторые классы кодов и приведем их характеристики.

11.1. Проверочная матрица МППЧ-кода

Линейный двоичный код может быть задан его проверочной матрицей H. Теоремы случайного кодирования говорят, что если элементы матрицы выбраны случайно и независимо, то с большой вероятностью параметры кода будут хорошими (удовлетворять границе Варшамова–Гилберта на минимальное расстояние).

Очевидные проблемы на пути воплощения случайных кодов в практику связи — высокая сложность кодирования и декодирования для матриц произвольного вида.

Идея Галлагера, которую он подкрепил анализом с использованием метода случайного кодирования, состоит в том, чтобы выбрать матрицу H разреженной: в ней должно быть мало единиц, строки и столбцы должны содержать мало общих элементов. Не вдаваясь в детали математического анализа МППЧ-кодов, попытаемся пояснить, как требование малой плотности влияет на характеристики (минимальное расстояние) кода и на сложность его декодирования.

Прежде всего, заметим, что для получения хорошей асимптотической оценки минимального расстояния важно, чтобы линейные комбинации столбцов образовывали последовательности случайных независимых символов. Это свойство может быть выполнено при плотности единиц проверочной матрицы много меньшей, чем $1/2$.

Чтобы понять, как плотность единиц в H связана со сложностью декодирования, рассмотрим случай ДСК. Предположим, что по принятой из канала последовательности вычислен синдром $\boldsymbol{s}$ и его вес Хэмминга равен $w(\boldsymbol{s})$. При низкой плотности матрицы H можно ожидать, что вес синдрома растет пропорционально числу ошибок, и что декодировать можно методом проб и ошибок, пытаясь подобрать последовательность столбцов H, при сложении которых с синдромом вес синдрома уменьшается от попытки к попытке. Похожую стратегию декодирования называют декодированием подбрасыванием монеты (coin flipping). Интуиция подсказывает, что при низкой плотности единиц можно рассчитывать на декодирование с линейной или почти с линейной сложностью, и что при высокой плотности единиц в проверочной матрице такой метод заведомо неэффективен.

Практически более значимый алгоритм, рассмотренный в следующем параграфе, внешне далек от подбрасывания монеты, но он тоже неявно использует возможность судить о влиянии конкретного столбца проверочной матрицы на синдром независимо от других столбцов. Основанием для таких независимых решений служит то, что столбцы не имеют или почти не имеют общих единиц.

Определение 11.1. *Двоичный линейный (n, k)-код, заданный проверочной матрицей H, называется (J, K)-регулярным кодом с малой плотностью проверок на четность, если каждый столбец матрицы H содержит ровно J единиц, а каждая строка — K единиц.*

В этом определении подразумевается, что числа J и K малы по сравнению с длиной и размерностью кода. Если рассматривается последовательность кодов с фиксированной скоростью $R = k/n$ и с возрастающей длиной n и размерностью k, то параметры J и K остаются неизменными.

В основополагающих работах Галлагера рассматривались регулярные коды. В исследованиях последних лет, напротив, много внимания уделяется *иррегулярным* МППЧ-кодам, для которых фиксируется не вес столбцов и строк, а наборы весов, из которых разрешено выбирать значения в соответствии с некоторыми предопределенными вероятностями.

Считается установленным факт, что среди иррегулярных кодов есть коды, позволяющие приблизиться весьма близко к пропускной способности канала. К сожалению, гипотезы, принятые при доказательстве таких результатов (методом эволюции плотностей), могут выполняться только при астрономически больших длинах кодов (см. параграф 11.4.3). При приемлемых для практических приложений длинах кодов регулярные и иррегулярные коды конкурируют друг с другом.

Почти без потери общности проверочная матрица МППЧ-кода (регулярного или иррегулярного) может быть записана в виде

$$H = \begin{pmatrix} H_1 \\ H_2 \\ \vdots \\ H_J \end{pmatrix},$$

где подматрицы H_i содержат не более одной единицы в каждом столбце. Галлагер изучал ансамбль МППЧ-кодов, в котором полосы, начиная со второй, являются перестановками столбцов первой полосы.

Следующий шаг на пути придания матрице H некоторой структуры состоит в разбиении полос проверочной матрицы на подматрицы, т.е.

$$H = \begin{pmatrix} H_{11} & H_{12} & \cdots & H_{1L_1} \\ H_{21} & H_{22} & \cdots & H_{2L_1} \\ \vdots & \vdots & \vdots & \vdots \\ H_{J1} & H_{J2} & \cdots & H_{JL_1} \end{pmatrix}, \tag{11.1}$$

где $L_1 \geq K$, а матрицы H_{ij} содержат не больше одной единицы в каждой строке и каждом столбце. Можно дополнительно сузить класс кодов, допустив использование в качестве H_{ij} только нулевых либо перестановочных матриц. Такое множество проверочных матриц является основным объектом изучения в классе регулярных МППЧ-кодов.

Заметим, что при построении или случайном выборе матриц, удовлетворяющих указанным выше ограничениям на структуру, очень часто оказывается, что ранг матрицы H меньше числа строк r. В этом случае скорость кода оказывается несколько больше «конструктивной» скорости, т.е. в общем случае для кода с матрицей H размера $r \times n$ скорость кода удовлетворяет неравенству $R \geq 1 - r/n$.

Пример 11.1. Проверочная матрица

$$H = \left(\begin{array}{cc:cc:cc:cc} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ \hdashline 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ \hdashline 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \end{array}\right)$$

задает $(J = 3.K = 4)$-регулярный МППЧ-код. Плотность единиц в данном конкретном случае велика, но ее можно сделать меньше, если в качестве подматриц выбрать перестановочные матрицы более высокого порядка. Поскольку строки линейно зависимы, строки с номерами 4 и 6 можно удалить, скорость кода $R = 1/2 > 1 - 6/8$.

11.2. Декодирование по принципу распространения доверия

Нам предстоит по известным значениям компонент синдрома для символов, входящих в соответствующие этим компонентам проверки, принимать решение об их надежности в зависимости от надежности остальных символов. Для решения этой непростой задачи нам понадобятся решения простых вспомогательных задач.

Лемма 11.1. *Пусть в последовательности из n двоичных независимых символов вероятность единицы в любой позиции равна p. Тогда вероятность четного числа единиц в последовательности равна*

$$P_{\text{even}} = \frac{1 + (q-p)^n}{2}, \quad q = 1 - p. \tag{11.2}$$

Доказательство. Вероятность появления i единиц равна $\binom{n}{i} p^i q^{n-i}$. Поэтому

$$\begin{aligned} P_{\text{even}} + P_{\text{odd}} &= (q+p)^n = 1; \\ P_{\text{even}} - P_{\text{odd}} &= (q-p)^n. \end{aligned}$$

Полусумма этих соотношений дает (11.2). □

Усложним задачу, предположив, что вероятности единиц на разных позициях могут быть разными.

Лемма 11.2. *Пусть в последовательности из n двоичных независимых символов вероятность единицы в позиции i равна p_i. Тогда вероятность четного числа единиц в последовательности равна*

$$P_{\text{even}} = \frac{1 + \prod_{i=1}^{n} (q_i - p_i)}{2}, \quad q_i = 1 - p_i. \tag{11.3}$$

Доказательство. Рассмотрим произведение

$$\prod_{i=1}^{n} (q_i - p_i).$$

Если записать это произведение в виде суммы, то из общего числа 2^n слагаемых отрицательные слагаемые будут соответствовать вероятностям последовательностей с нечетным числом единиц,

положительные — с четным числом единиц. Заметив это и повторив те же действия, что и в доказательстве леммы 11.1, получим (11.3). □

Предположим, что по каналу с двоичным входом и дискретным выходом для передачи сообщений используется (J, K)-регулярный МППЧ-код, заданный проверочной матрицей. Предположения о регулярности кода и дискретности выхода канала на самом деле не нужны, мы принимаем их только для простоты записи (чтобы иметь постоянными пределы суммирования в суммах и записывать вероятности вместо плотностей).

Рассмотрим одно из проверочных соотношений, вычисленных по принятой последовательности. Предположим, что в число K его ненулевых позиций входит позиция с номером i. Кодовый символ, соответствующий этой позиции, можно выбрать равным 0 или 1. Для того чтобы проверка выполнялась (была равна нулю), в зависимости от выбора $x_i = 0$ или 1, мы должны предположить суммарное число единиц (ошибок) на остальных $K-1$ позициях, накрываемых проверкой, соответственно, четным или нечетным.

Обозначим через $\boldsymbol{y}$ принятую последовательность, через p_i — вероятность единицы на позиции с номером i, $p_i = \Pr(x_i = 1 \,|\boldsymbol{y})$, вычисленную по $\boldsymbol{y}$; под S понимается событие, состоящее в том, что проверка выполняется. То, что жесткие решения должны удовлетворять проверкам, определяемым проверочной матрицей кода, дает возможность пересчитать апостериорные вероятности p_i, в новые вероятности $\Pr(x_i = 0 \,|\boldsymbol{y}, S)$.

Теорема 11.3. *Если результаты различных проверок, в которые входит данный символ — независимые события, а p_{jh}, $h = 1,\ldots, K-1$, $j = 1,\ldots,J$ — вероятность единицы для h-го символа j-й проверки при заданной последовательности $\boldsymbol{y}$, то*

$$\frac{\Pr(x_i = 0 \,|\boldsymbol{y}, S)}{\Pr(x_i = 1 \,|\boldsymbol{y}, S)} = \frac{1-p_i}{p_i} \prod_{j=1}^{J} \frac{1 + \prod_{h=1}^{K-1}(1-2p_{jh})}{1 - \prod_{h=1}^{K-1}(1-2p_{jh})}. \tag{11.4}$$

Доказательство. По формуле апостериорной вероятности запишем

$$\Pr(x_i = 0 \,|\boldsymbol{y}, S) = \frac{\Pr(S \,|\boldsymbol{y}, x_i = 0)\Pr(x_i = 0 \,|\boldsymbol{y})}{\Pr(S|\boldsymbol{y})}. \tag{11.5}$$

Пусть S_j представляет собой событие, состоящее в том, что j-я из J проверок выполнена. В силу независимости проверок

$$\Pr(S\,|\boldsymbol{y}, x_i = 0) = \prod_{j=1}^{J} \Pr(S_j\,|\boldsymbol{y}, x_i = 0)\,. \tag{11.6}$$

При $x_i = 0$ проверка выполняется, если сумма остальных символов, входящих в проверку, четна. Это дает возможность использовать лемму 11.2 для вычисления сомножителей в правой части (11.6).

Записав формулы аналогичные (11.5) и (11.6) для $x_i = 1$ после подстановки вычисленных условных вероятностей $\Pr(x_i = 0\,|\boldsymbol{y}, S)$ и $\Pr(x_i = 1\,|\boldsymbol{y}, S)$ в дробь в левой части (11.4), получаем требуемый результат. □

Обсудим теорему.

Сформулированное утверждение является основой для итеративной процедуры декодирования, поскольку все вероятности в правой части мы считаем известными, и по ним находим одну из вероятностей, которая будет востребована при подсчете апостериорных вероятностей (точнее, их отношений) для других символов.

Нетрудно заметить, что в формулировке теоремы есть подводный камень — предположение о независимости проверок. Поскольку все вероятности p_{jh} в правой части относятся к несовпадающим позициям кодового слова, и мы рассматриваем канал без памяти, то предположение кажется реалистичным. С другой стороны, значения 0 и 1 в кодовых символах не могут быть выбраны произвольно — некоторые ограничения накладываются кодом. С этой точки зрения может показаться неубедительной ссылка на лемму 11.2 в доказательстве теоремы. Еще одна, более серьезная, проблема состоит в том, что после применения теоремы для получения оценки одного из символов эта оценка появится в правой части при вычислении оценок других символов. Это означает, что вычисленная оценка будет влиять на несколько сомножителей в произведении по j, т.е. предположение о независимости результатов проверки вызывает сомнения.

Мы вернемся к этим проблемам позже при анализе характеристик кодов с применением графов Таннера. Перечисленные контр-

аргументы не мешают применять теорему 11.3, осознавая, что соотношение (11.4) на самом деле — приближенное и выполняется с тем большей точностью, чем менее зависимы проверки, т.е. чем ниже плотность единиц в проверочной матрице кода.

Чтобы сделать формулы удобнее для применения на практике, переформулируем их в терминах логарифмов отношений правдоподобия. Для этого введем обозначение α для знака логарифма отношения правдоподобия и β для его абсолютной величины, т.е. при вероятности единицы равной $p = 1 - q$ получаем

$$\ln \frac{1-p}{p} = \ln \frac{q}{p} = \alpha\beta. \tag{11.7}$$

(В соответствии с обозначениями предыдущих глав, следует логарифмом отношения правдоподобия считать $\ln(p/q)$. В данной главе мы следуем общепринятым для МППЧ-кодов обозначениям, введенным Галлагером.) Введем функцию

$$f(\beta) = \ln \frac{e^{\beta}+1}{e^{\beta}-1}, \quad \beta > 0.$$

Заметим, что

$$f(\beta) = -\ln \frac{e^{\beta/2} - e^{-\beta/2}}{e^{\beta/2} + e^{\beta/2}} = -\ln \tanh(\beta/2), \tag{11.8}$$

где

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} = \frac{e^{2x}-1}{e^{2x}+1}$$

гиперболический тангенс.

Из (11.7) и (11.8) получаем

$$1 - 2p = \frac{q/p - 1}{q/p + 1} = \tanh\left(\frac{\alpha\beta}{2}\right) = e^{-\alpha f(\beta)}.$$

Эти тождества позволяют переписать (11.4), затратив некоторые усилия, в форме

$$\alpha_i'\beta_i' = \alpha_i\beta_i + \sum_{j=1}^{J-1}\left[\left(\prod_{h=1}^{K-1}\alpha_{jh}\right) f\left(\sum_{h=1}^{K-1} f(\beta_{jh})\right)\right], \tag{11.9}$$

где $\alpha'_i\beta'_i$ и $\alpha_i\beta_i$ — знаки и модули логарифмов отношения правдоподобия i-го символа до и после модификации, учитывающей знаки и надежности символов, входящих в общие проверки с этим символом. Суммирование и перемножение в (11.9) выполняется так, что $\alpha_i\beta_i$ участвует только в виде первого слагаемого, именно поэтому значение индексов ограничено $J-1$ по столбцам и $K-1$ по строкам.

Алгоритм декодирования теперь понятен: при инициализации нужно вычислить все $\alpha_i\beta_i$ по принятым из канала мягким решениям. Затем поочередно для всех символов нужно выполнять вычисления по формуле (11.9), тем самым обновляя знаки и надежности символов. Окончательно решение принимается после выполнения заданного числа итераций либо когда синдром, вычисленный по жестким решениям, окажется равным нулю.

Этот алгоритм примерно одинаково часто называют алгоритмом распространения доверия (belief propagation или BP) либо алгоритм суммирования произведений (sum-product algorithm или SPA).

Для программной реализации вычислений по формуле (11.9) следует ввести массив ненулевых индексов строк V размера $r \times K$ и массив ненулевых индексов столбцов C размера $n \times J$. Для вре́менного хранения слагаемых в квадратных скобках в сумме по j вводим вспомогательный массив Z, а для хранения величин $\alpha_{jh}\beta_{jh}$) используем массив L. Массивы Z и L могут быть выбраны размера равного числу ненулевых элементов проверочной матрицы, т.е. линейного по длине кода. Для простоты записи алгоритма и программы мы выбрали Z и L того же размера, что и проверочная матрица H.

Полностью декодирование по принципу распространения доверия представлено в виде алгоритма 11.1 и реализовано на МАТЛАБе в виде программы 11.1. Обратите внимание, что вход программы декодирования нормализован так, что значения входной последовательности равны логарифмам отношений правдоподобия кодовых символов при передаче противоположными сигналами по каналу с гауссовским шумом.

Алгоритм 11.1. Алгоритм распространения доверия

Input: Логарифмы отношений правдоподобия кодовых символов $\boldsymbol{y} = (y_1, ..., y_n)$, максимальное число итераций ITER, массивы ненулевых позиций столбцов C_i $i = 1, ..., n$, и строк V_j, $j = 1, ..., r$

Output: Жесткие решения $\boldsymbol{x} = (x_1, ..., x_n)$

`1. Инициализация`

for $i = 1$ **to** n **do**
  for $j = 1$ **to** r **do**
    $L(j,i) = Z(j,i) = 0;$
  end
end

`2. Основной цикл`

for $iter = 1$ **to** $ITER$ **do**
  for $i = 1$ **to** n **do**
    for $j = 1$ **to** r **do**

$$L(j,i) = y_i + \sum_{h \in C_i, h \neq j} Z(h,i)$$

    end
  end
  for $i = 1$ **to** n **do**
    for $j = 1$ **to** J **do**

$$\alpha_{hj} = \operatorname{sign}(L(h,j)),$$

$$Z(j,i) = \left(\prod_{h \in V_j, h \neq i} \alpha_{hj} \right) f \left(\sum_{h \in V_j, h \neq i} f(|L(h,j)|) \right)$$

    end
  end
  for $i = 1$ **to** n **do**

$\hat{y}_i = y_i + \sum_{h \in C_i} Z(h,i)$ `% Мягкие решения`

$x_i = \hat{y}_i < 0$ `% Жесткие решения`

  end
  $\boldsymbol{s} = \boldsymbol{x}H.$ `% Синдром`
  if $\boldsymbol{s} = 0$ **then return** $\boldsymbol{x}$
  `% Декодирование закончено`
end
return $\boldsymbol{x}$;

Заметим, что вычисление произведения переменных α не требует умножения чисел. Таким образом, единственная затратная операция — вычисление функции логарифма гиперболического тангенса $f(\cdot)$. При реализации в реальных системах эти вычисления выполняются таблично либо применяются аппроксимации (см. параграф 10.3.3).

Программа 11.1. Декодирование по принципу распространения доверия

```
function [hard,soft,iter]=bp_decod(y,V,C,H, MAXITER)
% Вход:
%        y  - последовательность на выходе канала
%        V - ненулевые позиции проверок по строкам
%        C - ненулевые позиции проверок по столбцам
%        H - проверочная матрица
%        MAXITER -- максимальное число итераций
% Выход
%        hard  - жесткие решения
%        soft - мягкие решения
%        iter - число итераций до принятия решения.
% Пример вызова
%   [V,C]=bp_prepare(H);      % подготовка массивов
%   noise=sigma*randn(1,n); % формирование шума
%   out=(2*code_word-1)+noise;   % выход канала;
%   out=-2*out/sigma^2:       % нормализация
%   [hard, soft, iter]=bp_decod(out, V, C, H);

% Инициализация
if nargin==4,  MAXITER=50; end;
[r,n]=size(H);
Z=zeros(r,n);
L=zeros(r,n);
hard=y<0;
soft=y;
syndrome=mod(hard*H',2);

iter=0;
while (iter<MAXITER)&& (sum(syndrome)~=0)
    % Обработка столбцов H
    for i=1:n
        L(C(i,:),i)=y(i)+sum(Z(C(i,:),i))-Z(C(i,:),i);
    end;
   % Обработка строк
    for i=1:n
```

```
        for j=C(i,:)
            a=prod(sign(L(j,V(j,:))))*sign(L(j,i));
            b=sum(logtanh(L(j,V(j,:))))-logtanh(L(j,i));
            Z(j,i)=max(min(a*logtanh(b),19.07),-19.07);
        end;
    end;
    % Формирование результата
    soft=y+sum(Z);
    hard=soft<0;
    syndrome=mod(hard*H',2);
    iter=iter+1;
end;

% Функция логарифма арктангенса
function y=logtanh(x)
t=exp(abs(x));
y=log((t+1)./(t-1));

 % Подпрограмма подготовки массивов проверок
function [V,C]=bp_prepare(H)
% Вход:       H - проверочная матрица
% Выход       V - ненулевые позиции строк
%                 C - ненулевые позиции столбцов
[r,n]=size(H);
c=max(sum(H));    % макс. число единиц по столбцам
v=max(sum(H'));   % макс. число единиц по строкам
V=zeros(r,v);
C=zeros(n,c);
for i=1:r
    f=find(H(i,:));
    V(i,1:length(f))=f;
end;
for i=1:n
    f=find(H(:,i));
    C(i,1:length(f))=f;
end;
```

11.3. Графы Таннера и характеристики МППЧ-кодов

Особенность МППЧ-кодов состоит в том, что они представляют интерес только в совокупности с алгоритмом декодирования по опи-

санному в предыдущем параграфе принципу распространения доверия. Главная характеристика обычного линейного кода — минимальное расстояние. Оно характеризует способность кода гарантированно исправлять определенное число ошибок. Применительно к МППЧ-кодам минимальное расстояние уже не является важной характеристикой, поскольку даже для кодов, расстояние которых известно, его величина не говорит о том, сколько ошибок может быть исправлено, хорош ли код и лучше ли он того, расстояние которого меньше.

Связь МППЧ-кодов с графами и связь их характеристик с циклами определенного графа была замечена еще Галлагером. Большой шаг в понимании этой связи, способствовавший прогрессу в теоретическом исследовании МППЧ, связан с работой Таннера [106].

Чтобы описать граф, нужно указать множество вершин и множество ребер. Напомним, что множество узлов двудольного графа состоит из двух подмножеств. Ребра связывают узлы одного подмножества только с узлами другого, и не существует ребер, связывающих узлы внутри подмножества.

Определение 11.2. *Графом Таннера линейного кода, заданного проверочной матрицей* $H = \{h_{ij}\}, i = 1, ..., r, j = 1, ..., n$, *называется двудольный граф, одно множество узлов которого соответствует множеству проверочных соотношений (проверочные узлы), другое — множеству кодовых символов (символьные узлы). Символьный узел* c_i *связан ребром с некоторым проверочным узлом* s_j, *если* i*-й кодовый символ входит в* j*-ю проверку, иными словами, если элемент* h_{ij} *проверочной матрицы не равен нулю.*

Пример 11.2. Рассмотрим проверочную матрицу кода Хэмминга (7,4)

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

Соответствующий граф Таннера приведен на рис. 11.1. Номера проверочных и кодовых узлов соответствуют номерам кодовых символов и проверочных символов.

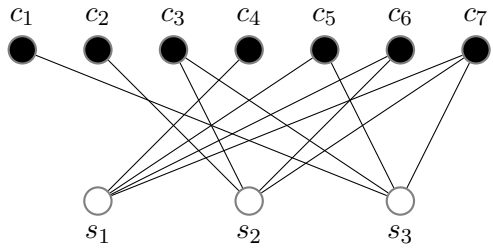


Рис. 11.1. Граф Таннера для кода примера 11.2

Предыдущий пример показывает, что граф Таннера можно построить для произвольного кода. Графы Таннера для МППЧ-кодов имеют специфические особенности.

Пример 11.3. Рассмотрим МППЧ ($J = 2, K = 3$) код с проверочной матрицей

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

Скорость кода равна $3/6$, поскольку строки проверочной матрицы линейно зависимы. Граф Таннера показан на рис 11.2. Заметим, что в каждый символьный узел входит 2 ребра, в каждый проверочный узел — 3 ребра.

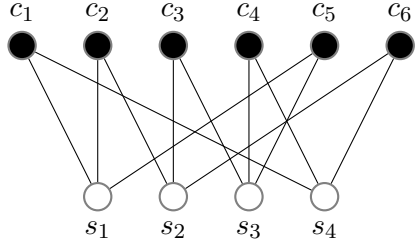


Рис. 11.2. Граф Таннера для кода примера 11.3

Представление МППЧ-кода графом Таннера наводит на альтернативную интерпретацию декодирования по методу распространения доверия.

До начала декодирования символьным (черным) узлам сопоставлены значения логарифмов отношения правдоподобия, вычисленные по отсчетам на выходе демодулятора.

В соответствии с алгоритмом 11.1, при подсчете величин $Z(\cdot,\cdot)$ в проверочных узлах происходит обработка информации о надежности связанных с ними символьных узлов, а при подсчете $L(\cdot,\cdot)$ в символьных узлах обрабатывается информацию о надежности проверочных узлов.

Поэтому такое декодирование часто называют *алгоритмом обмена сообщениями (message passing algorithm)*.

Вернемся к коду примера 11.3 и проследим, как распространяется информация о надежности одного из символов, скажем, c_1. Соответствующее дерево показано на рис. 11.3.

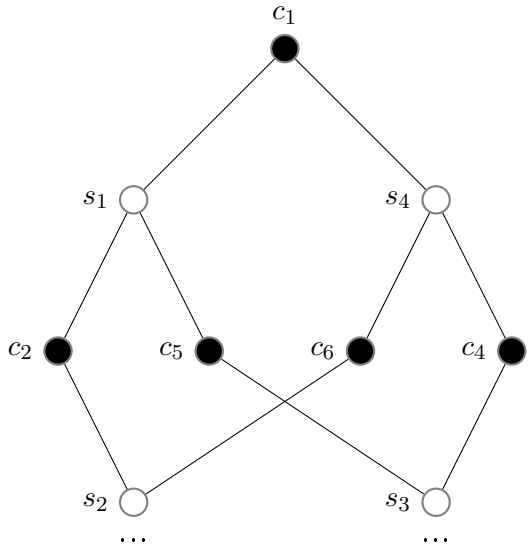


Рис. 11.3. Дерево распространения сообщений для кода примера 11.3

Напомним, что в условиях теоремы 11.3 результаты проверок полагались независимыми. Мы видим, что на первой и второй итерациях декодирования декодер вычисляет оценки вероятностей для узла c_1 по непересекающимся подмножествам символов, но на третьей итерации одна и та же информация о надежности s_2 будет присутствовать в оценках надежностей для узлов s_1 и s_4. Условия теоремы будут нарушены.

Мы видим из этого примера, что эффективность работы алгоритма связана с длиной циклов в графе Таннера. Чем длиннее циклы, тем для большего числа итераций выполняется гипотеза о независимости проверок.

В теории графов длина кратчайшего цикла графа называется его *обхватом* (girth).

Определение 11.3. *Обхватом МППЧ-кода называется обхват g его графа Таннера.*

Для кода из примера 11.3 перебором по множеству циклов находим его обхват $g = 6$.

Кажется естественным при декодировании по алгоритму распространения доверия рассматривать обхват МППЧ-кодов как характеристику, эквивалентную минимальному расстоянию линейных кодов при их декодировании по максимуму правдоподобия. К сожалению, обхват МППЧ-кода в меньшей степени характеризует его эффективность, чем минимальное расстояние характеризует обычный линейный код. Тем не менее, многие исследователи рассматривают именно обхват как критерий для сравнения кодов.

Практика построения и моделирования МППЧ-кодов показывает, что коды с $g < 6$ заведомо неэффективны.

Упражнение 11.1. Докажите самостоятельно следующие простые свойства обхвата кода.

Свойство 11.1. *Для любого МППЧ-кода обхват g — четное число.*

Свойство 11.2. *Если 2 столбца матрицы H совпадают более, чем в одной позиции, то $g = 4$.*

Свойство 11.3. *Если любые 2 столбца матрицы H совпадают не более чем в одной ненулевой позиции, то $g \geq 6$.*

Свойство 11.4. *С увеличением длины кода n его обхват растет не быстрее, чем* $\log n$.

Подсказка: для того, чтобы убедиться в справедливости свойства 11.4, подсчитайте число вершин на ярусе $g/2$ в дереве распространения сообщений (см. рис. 11.3).

Более точный подсчет числа вершин в дереве распространения сообщений графа Таннера МППЧ-кода приводит к следующей оценке.

Теорема 11.4. *(Граница Мура). Длина (J, K)-регулярного МППЧ-кода с обхватом g удовлетворяет неравенству*

$$n \geq J(K-1)\frac{[(J-1)(K-1)]^{\lfloor g/4 \rfloor} - 1}{(J-1)(K-1)-1} + 1. \qquad (11.10)$$

Доказательство. Для доказательства нужно подсчитать, сколько символьных узлов охватывает дерево распространения сообщений на каждом ярусе. Длина кода должна быть больше, чем суммарное число узлов на $g/2$ ярусах дерева. Применив формулу для суммы геометрической прогрессии, получим искомое неравенство. □

Скорость регулярного кода

$$R \geq 1 - J/K.$$

Регулярные коды с $J = 2$ недостаточно эффективны. Поэтому выбирают $J \geq 3$. Пусть, например, нужен код со скоростью $1/2$. Тогда минимально возможные значения параметров кода: $J = 3$, $K = 6$. Если мы поставим перед собой задачу построения кода с обхватом $g = 12$, то в соответствии с теоремой 11.4 длина кода не может быть меньше 1666. Интересно отметить, что кратчайший из существующих кодов имеет длину 1836 [51]. Этот пример показывает, что хорошие МППЧ-коды не могут быть слишком короткими.

В следующем параграфе мы рассмотрим некоторые методы построения МППЧ-кодов.

11.4. Построение МППЧ-кодов

Для кодов с малой плотностью проверок на четность характерно то, что случайный выбор проверочной матрицы дает очень неплохие результаты. Именно на таких примерах была осознана высокая эффективность этого класса кодов. По сравнению с линейными кодами общего вида описание кодов с малой плотностью проще, поскольку его проверочная матрица очень разреженная и достаточно хранить только расположение ненулевых элементов. Тем не менее, задача поиска хороших кодов с некоторой комбинаторной структурой чрезвычайно важна по следующим причинам:

- Для неструктурированного кода задача кодирования недопустимо сложна. Порождающая матрица нерегулярна, имеет, вообще говоря, высокую плотность единиц, ни сохранить ее, ни выполнить кодирование с ее помощью невозможно.

- При схемной реализации декодирования крайне нежелательно иметь для каждого символа кодового слова свою систему проверочных соотношений. Желательно, чтобы матрица обладала симметрией, позволяющей пересчитывать проверочные соотношения по простому правилу.

Сначала мы рассмотрим самый простой для анализа и реализации класс кодов — квазициклические коды. Эти коды широко применяются на практике. Обзор других способов построения МППЧ-кодов приведен в конце параграфа.

11.4.1. Квазициклические МППЧ-коды

Нетрудно придумать короткий МППЧ-код с заданными параметрами (J, K). Например, (3,6)-регулярный код может иметь проверочную матрицу

$$H = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}. \qquad (11.11)$$

Конечно, сказать, что плотность проверок на четность для этого кода мала, можно только с большой натяжкой. Однако из этого

кода можно получить длинный МППЧ-код с тем же количеством единиц в строках и столбцах, заменив единицы в этой матрице на матрицы специального вида.

Введем обозначение $\mathcal{P}_M^i$ для квадратной перестановочной матрицы порядка M, полученной из единичной матрицы циклическим сдвигом ее элементов на i позиций вправо. Например,

$$\mathcal{P}_3^0 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}; \quad \mathcal{P}_3^1 = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}; \quad \mathcal{P}_3^2 = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}.$$

Мы в дальнейшем будем опускать нижний индекс, если величина M известна либо ее конкретное значение не важно.

Напомним, что матрица, каждая строка которой — сдвиг на одну позицию предыдущей строки, называется *циркулянтной матрицей* или *циркулянтом*. Следовательно, матрицы $\mathcal{P}^i$ — циркулянты.

Линейный код называют квазициклическим, если для некоторого m циклический сдвиг любого кодового слова на m позиций — кодовое слово.

Переупорядочением позиций кодового слова можно представить порождающую и проверочную матрицу квазициклического (n, k)-кода в виде блочной матрицы из квадратных блоков-циркулянтов. Поэтому из произвольной проверочной матрицы размера $r \times n$, $r = n - k$, после подстановки $\mathcal{P}_M^{\mu_{ij}}$ вместо ненулевых элементов h_{ij} матрицы H мы получаем проверочную матрицу размера $rM \times nM$ квазициклического (nM, kM)-кода (на самом деле число информационных символов может быть чуть больше, если строки проверочной матрицы линейно зависимы).

Например, из матрицы (11.11) получим новую матрицу

$$H = \begin{pmatrix} \mathcal{P}^{\mu_{11}} & \mathcal{P}^{\mu_{12}} & \dots & \mathcal{P}^{\mu_{16}} \\ \mathcal{P}^{\mu_{21}} & \mathcal{P}^{\mu_{22}} & \dots & \mathcal{P}^{\mu_{26}} \\ \mathcal{P}^{\mu_{31}} & \mathcal{P}^{\mu_{32}} & \dots & \mathcal{P}^{\mu_{36}} \end{pmatrix}. \tag{11.12}$$

В общем случае матрица квазициклического МППЧ состоит из нулевых блоков и блоков вида $\mathcal{P}^i$. Граф Таннера исходного кода иногда называют протографом. Поэтому квазициклические коды

с матрицей, полученной подстановкой $\mathcal{P}^i$, рассматривают как подкласс кодов над протографами [107].

Попробуем связать обхват нового кода с обхватом исходного кода (кода-прототипа).

Рассмотрим сначала общую задачу построения графа с большим обхватом из исходного (базового) графа. Для этого в теории графов используется так называемый метод *лифтинга* или *расширения*.

Пусть $\mathcal{G}_\text{B} = \{\mathcal{E}_\text{B}, \mathcal{V}_\text{B}\}$ — базовый граф с множеством вершин $\{\mathcal{V}_\text{B}\}$ и множеством ребер $\{\mathcal{E}_\text{B}\}$. Введем в рассмотрение аддитивную группу $\Gamma = \{\gamma\}$ (в нашем случае это будет множество вычетов по модулю положительного целого числа M). Сопоставим каждому ребру базового графа некоторый элемент группы $\Gamma = \{\gamma\}$. Тем самым получим *размеченный граф* или *граф напряжений* (voltage graph).

Определение 11.4. *Расширенным графом называется граф* $\mathcal{G} = \{\mathcal{E} \in \mathcal{E}_\text{B} \times \Gamma, \mathcal{V}_\text{B} \times \Gamma\}$, *в котором две вершины* (v, γ) *и* (v', γ') *связаны ребром* e *в том случае, если вершины* v *и* v' *связаны ребром в базовом графе* $\mathcal{G}_\text{B} = \{\mathcal{E}_\text{B}, \mathcal{V}_\text{B}\}$ *и этому ребру присвоена метка (напряжение)* $\gamma - \gamma'$.

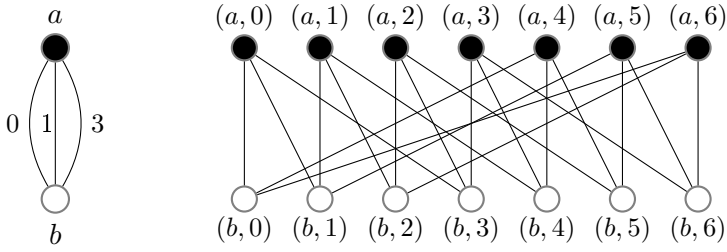
Заметим, что в этом определении знак метки имеет значение.

Идея расширения графа станет ясной на следующем примере.

Пример 11.4. Предположим, что матрица инцидентности графа с двумя вершинами и тремя ребрами имеет вид

$$H_\text{B} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}.$$

Каждому ребру соответствует столбец матрицы H. В качестве группы Γ выберем кольцо вычетов по модулю 7. Размеченный базовый граф показан на рис. 11.4, *а*. Разметим матрицу, назначив веса ребрам, как это показано на рисунке. Размеченному графу удобно сопоставить матрицу $H_\text{B}(D)$, в которой степеням формальной переменной D соответствуют метки ребер. Поскольку ребру соответствуют две единицы столбца, а нам нужно указать приращение напряжения на ребре, одной из единиц можно присвоить значение показателя степени равное нулю. В нашем случае получаем



а) Базовый граф *б*) Расширенный граф

Рис. 11.4. Пример базового и расширенного графа

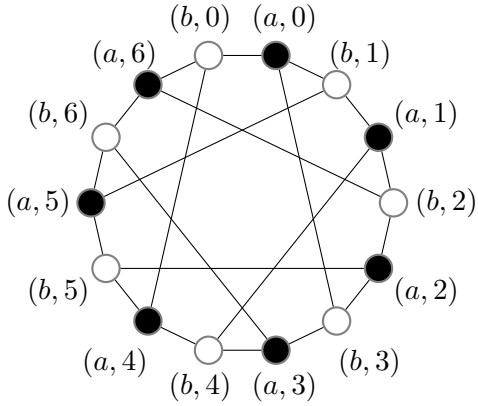


Рис. 11.5. Граф Хивуда

$$H_{\text{B}}(D) = \begin{pmatrix} D^0 & D^0 & D^0 \\ D^0 & D^1 & D^3 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & D & D^3 \end{pmatrix}. \tag{11.13}$$

Расширенный граф показан на рис. 11.4, *б*. Другая форма этого графа приведена на рис. 11.5. Полученный граф известен в комбинаторике как граф Хивуда — регулярный граф степени 3 с обхватом 6 с наименьшим возможным числом вершин.

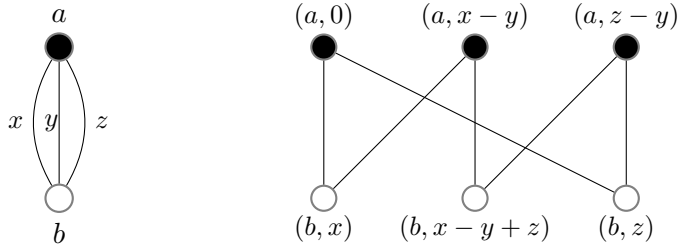
В данном примере мы из графа с обхватом $g_{\text{B}} = 2$ с помощью удачной разметки получили расширенный граф с обхватом $g = 6$. Не составляет труда убедиться в том, что матрица инцидентности

нового графа получается из (11.13) заменой формальной переменной D на матрицу $\mathcal{P}_M$, $M = 7$.

$$H_{\mathrm{B}}(D) = \begin{pmatrix} I_7 & I_7 & I_7 \\ I_7 & \mathcal{P}_7^1 & \mathcal{P}_7^3 \end{pmatrix}, \tag{11.14}$$

где I_M обозначает единичную матрицу порядка M.

Теорема 11.5. *Циклу в расширенном графе соответствует цикл с нулевой суммой напряжений в базовом графе.*



а) Базовый граф *б*) Цикл в расширенном графе

Цикл с нулевой суммой в базовом графе:

$$a \xrightarrow{x} b \xrightarrow{-y} a \xrightarrow{z} b \xrightarrow{-x} a \xrightarrow{y} b \xrightarrow{-z} a$$

Рис. 11.6. Иллюстрация к теореме 11.5

Доказательство. Поясним суть теоремы и ее доказательства простым примером на рис. 11.6. Заметим, что при подсчете суммы напряжений цикла метки ребер складываются с учетом направления движения. В нашем примере мы для определенности складываем при движении «вниз» — с плюсом, «вверх» — с минусом.

Доказательство для общего случая оставляем читателю в качестве упражнения. □

Заметим, что в примере, показанном на рис. 11.6, никакая разметка не может обеспечить обхват больше чем 6, поскольку цикл длины 6 с нулевой суммой существует при любых x, y и z.

Определение 11.5. *Цикл в базовом графе называется сбалансированным, если каждое ребро встречается четное число раз, причем цикл проходит по ребру равное числ раз в каждом направлении.*

Непосредственно из определения вытекает следующее утверждение.

Теорема 11.6. *Обхват расширенного графа не превышает минимальной длины сбалансированного цикла в базовом графе.*

Более сложное утверждение приведем без доказательства (см. [32]).

Теорема 11.7. *Существует разметка, при которой обхват расширенного графа равен минимальной длине сбалансированного цикла в базовом графе.*

Трехкратное увеличение, демонстрируемое приведенным выше примером, как оказывается, — достижимый и при этом максимальный выигрыш, который можно получить от расширения графа.

Теорема 11.8. *Существует разметка, при которой обхват расширенного графа, полученного из базового графа с обхватом $g_{\text{в}}$, удовлетворяет неравенству*

$$g \geq 3g_{\text{в}}.$$

Доказательство. Вершины кратчайшего сбалансированного цикла образуют подграф, содержащий по меньшей мере два цикла, имеющих общую вершину или общие ребра. Обозначим через n_1, n_2 и n_0 число ребер, принадлежащих только первому из двух циклов, только второму циклу и число общих ребер для двух циклов. Имеют место неравенства

$$\begin{aligned} n_1 + n_0 &\geq g_{\text{в}}; \\ n_2 + n_0 &\geq g_{\text{в}}; \\ n_1 + n_2 &\geq g_{\text{в}}, \end{aligned}$$

суммируя которые, имеем

$$2(n_0 + n_1 + n_2) \geq 3g_{\text{в}}.$$

Поскольку выражение в левой части — длина кратчайшего цикла в расширенном графе, утверждение теоремы следует из теоремы 11.7. □

Теорема 11.9. *Если в базовом графе имеются два цикла длины $g_{\text{в}}$, перекрывающихся в $g_{\text{в}}/2$ ребрах, то при любой разметке обхват расширенного графа удовлетворяет неравенству*

$$g \leq 3g_{\text{в}}.$$

Упражнение 11.2. Докажите теорему 11.9.

Приведенные выше утверждения о расширенных графах имеют место, в частности, для графов Таннера. Почти непосредственно из теорем для графов общего вида приходим к оценкам обхватов квазициклических-МППЧ-кодов на основе перестановочных матриц.

Свойство 11.5. *Если проверочная матрица базового кода содержит подматрицы вида*

$$\begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} \quad \textit{или} \quad \begin{matrix} 1 & 1 \\ 1 & 1, \\ 1 & 1 \end{matrix}$$

то построенный на его основе квазициклический код будет иметь обхват не больше 12.

Свойство 11.6. *Если базовый код имеет обхват $g_{\text{в}}$, то на его основе можно построить квазициклический код с обхватом $3g_{\text{в}}$.*

В общем случае квазициклический МППЧ-код со скоростью $R = b/c$, как регулярный, так и иррегулярный, можно описать полиномиальной матрицей размера $b \times c$ вида

$$H(D) = \begin{pmatrix} h_{11}(D) & h_{12}(D) & \dots & h_{1c}(D) \\ h_{21}(D) & h_{22}(D) & \dots & h_{2c}(D) \\ \vdots & \vdots & \ddots & \vdots \\ h_{(c-b)1}(D) & h_{(c-b)2}(D) & \dots & h_{(c-b)c}(D) \end{pmatrix}, \qquad (11.15)$$

где $h_{ij}(D)$ — либо нули, либо мономы, т.е. $h_{ij}(D) = D^{w_{ij}}$, степени w_{ij} — неотрицательные целые числа.

Матрицу $H(D)$ можно рассматривать как полиномиальную проверочную матрицу сверточного кода. Тогда максимальная степень мономов $m = \max_{i,j} w_{ij}$ определяет память дуального сверточного кода. Циклическим усечением кода (см. параграф 8.6.2) на длине M блоков получаем квазициклический (Mc, Mb) МППЧ-код.

Другим способом получения двоичного представления МППЧ-кода из полиномиальной формы (11.15) служит замена элементов D^w соответствующими степенями $\mathcal{P}^w$ перестановочной матрицы $\mathcal{P}$. Как уже обсуждалось, эти две формы порождают эквивалентные коды.

Итак, описание квазициклического МППЧ-кода задается парой матриц, базовой матрицей

$$B = H(D)|_{D=1}$$

и матрицей степеней

$$W = \begin{pmatrix} w_{11} & w_{12} & \dots & w_{1c} \\ w_{21} & w_{22} & \dots & w_{2c} \\ \vdots & \vdots & \ddots & \vdots \\ w_{(c-b)1} & w_{(c-b)2} & \dots & w_{(c-b)c} \end{pmatrix}.$$

11.4.2. Кодирование

Для линейных кодов общего вида сложность кодирования считается пренебрежимой по сравнению со сложностью декодирования. Сложность декодирования МППЧ-кодов примерно линейна с длиной кода. В то же время, его порождающая матрица. вообще говоря, не обладает специальными свойствами (например, малой плотностью ненулевых элементов), упрощающими кодирование. При непосредственной реализации кодирования сложность кодирования может быть квадратичной функцией длины кода и тогда она намного превысит сложность декодирования.

При некоторых ограничениях на вид проверочной матрицы кодирование может быть сделано очень простым. Описанный ниже

подход предложен в [95] и нашел практическое применение, например, в стандарте WiMAX.

Проверочная матрица квазициклического МППЧ-кода выбирается в виде

$$H(D) = \Big(H_{\rm bd}(D) \;\vdots\; \boldsymbol{h}_0(D) \;\vdots\; H_{(c-b)\times b}(D) \Big), \tag{11.16}$$

где первая часть $H_{\rm bd}$ представляет собой бидиагональную матрицу размера $(c-b)\times(c-b-1)$, $\boldsymbol{h}_0(D)$ — столбец с тремя ненулевыми элементами, из которых два идентичны, и $H_{(c-b)\times b}(D)$ может быть произвольной мономиальной матрицей размера $(c-b)\times b$. Соответствующая матрица H двоичного (Mc, Mb)-кода получается подстановкой матрицы-циркулянта $\mathcal{P}$ порядка M вместо формальной переменной D.

Например, матрица с параметрами $b=4$, $c=8$ может иметь вид

$$H(D) = \begin{pmatrix} D^0 & 0 & 0 & D^{\alpha} & D^{w_{15}} & D^{w_{16}} & D^{w_{17}} & D^{w_{18}} \\ D^0 & D^0 & 0 & D^0 & D^{w_{25}} & D^{w_{26}} & D^{w_{27}} & D^{w_{28}} \\ 0 & D^0 & D^0 & 0 & D^{w_{35}} & D^{w_{36}} & D^{w_{37}} & D^{w_{38}} \\ 0 & 0 & D^0 & D^{\alpha} & D^{w_{45}} & D^{w_{46}} & D^{w_{47}} & D^{w_{48}} \end{pmatrix}. \tag{11.17}$$

Пусть информационная последовательность $\boldsymbol{u} = \big(\boldsymbol{u}_1\, \boldsymbol{u}_2 \ldots \boldsymbol{u}_b\big)$ записана в виде b блоков длины M бит. Кодовое слово запишем в виде

$$\boldsymbol{v} = \big(\boldsymbol{v}_1\, \boldsymbol{v}_2 \ldots \boldsymbol{v}_c\big) = \big(\boldsymbol{v}_1\, \boldsymbol{v}_2 \ldots \boldsymbol{v}_{c-b}\, \boldsymbol{u}_1\, \boldsymbol{u}_2 \ldots \boldsymbol{u}_b\big),$$

где первые $c-b$ проверочных блоков должны быть подобраны так, чтобы выполнялось тождество $\boldsymbol{v}H^{\rm T} = \boldsymbol{0}$ для заданных информационных блоков $\boldsymbol{u}$. (Вопреки традиции, мы поместили информационные символы не в начало, а в конец слова.)

Обозначим через $\boldsymbol{s} = \big(\boldsymbol{s}_1\, \boldsymbol{s}_2 \ldots \boldsymbol{s}_{(c-b)}\big)$ частичный синдром, полученный умножением последних (информационных) b блоков кодового слова на соответствующие столбцы проверочной матрицы. Заметим, что сумма строк матрицы H дает нули в первых $c-b-1$ блоках и единичную матрицу D^0 в блоке с номером $c-b$. Отсюда следует простая формула для проверочного блока $\boldsymbol{v}_{c-b}$

$$\boldsymbol{v}_{c-b} = \sum_{i=1}^{c-b} \boldsymbol{s}_i\,.$$

После этого остальные проверочные блоки могут быть вычислены рекурсивно:

$$\begin{aligned} \boldsymbol{v}_1 &= \boldsymbol{s}_1 + \boldsymbol{v}_{c-b}P^{\alpha} \\ \boldsymbol{v}_i &= \begin{cases} \boldsymbol{s}_i + \boldsymbol{v}_{i-1}, & \text{если} \quad h_{i,c-b} = 0 \\ \boldsymbol{s}_i + \boldsymbol{v}_{i-1} + \boldsymbol{v}_{c-b}, & \text{в противном случае} \end{cases} \end{aligned} \tag{11.18}$$

для $i = 2, ..., c - b$.

Вычислительная сложность такого кодирования — линейная функция параметра M.

11.4.3. Обзор конструкций МППЧ-кодов

Теория и практика МППЧ-кодов — динамично развивающаяся область на стыке теории информации и теории кодирования. В данном пособии нет возможности привести детальное описание даже самых основных претендентов на применение в перспективных системах связи. Приведенное ниже краткое словесное описание позволит познакомиться с терминологией и сориентироваться в литературных источниках.

Эволюция плотностей

Особенность теории МППЧ-кодов состоит в том, что наибольший интерес представляет их анализ в совокупности с итеративным декодированием по принципу распространения доверия. Фундаментальный шаг в этом направлении был сделан в работе [99], где был введен метод анализа МППЧ-кодов, названный *эволюцией плотностей распределения* (ЭП) (density evolution). Поясним кратко суть метода.

Рассмотрим случайный иррегулярный МППЧ-код с заданными распределением весов столбцов проверочной матрицы и распределением весов ее строк. Предположим, что для передачи двоичных сигналов используется канал с АБГШ с заданным отношением сигнал/шум. Поскольку код линейный, достаточно рассматривать передачу нулевого кодового слова.

Сформулированных предположений достаточно, чтобы, воспользовавшись соотношением (11.9), подсчитать распределение ве-

роятностей надежностей символов после первой итерации декодирования. Заметим, что аналитического способа решения этой задачи не существует. Нужно аппроксимировать и начальное (гауссовское), и искомое непрерывное распределение дискретными распределениями с достаточно большим числом дискретных значений.

Подсчитанное распределение вероятностей рассматривается как входное для второй итерации, и т.д. Повторяя эту процедуру, можно наблюдать изменение плотности распределения от итерации к итерации. Замечено, что при большом отношении сигнал/шум (SNR) распределение вырождается, бесконечно большое значение надежности символов получает вероятность, стремящуюся к единице. При очень низком отношении сигнал/шум вероятность отрицательных значений сохраняет положительные значения при любом числе итераций. Пороговое (наименьшее) значение SNR, при котором вероятность ошибки стремится к нулю, называется *порогом декодирования* (decoding threshold) для данного кода или ансамбля кодов. Эту величину можно интерпретировать как аналог пропускной способности канала при ограничении на класс алгоритмов декодирования.

В работе [99] приведены оптимальные распределения весов строк и столбцов для кодов со скоростью 1/2 при различных ограничениях на максимальный вес столбца. Результаты близки к пропускной способности канала и тем ближе к ней, чем больше максимальный вес столбца.

Это исследование было продолжено в работе [62]. Авторы показали, что надежная передача может быть осуществлена при затратах энергии на бит всего на 0.0045 дБ больше шенноновского предела (см. параграф 10.4.1). Этот результат, конечно, реализуется при сложности, лежащей далеко за пределами практической реализуемости.

Результаты, полученные методами ЭП, правильно воспринимать не как конструктивные решения, а скорее как теоремы кодирования. Дело в том, что этот метод анализа не учитывает циклов в графе Таннера соответствующего кода, а именно эти циклы при практически приемлемых длинах кодов в большой степени определяют надежность передачи информации. В частности, хорошо известно, что распределения весов столбцов и строк, минимизирую-

щие порог, весьма далеки от распределений, на которых минимизируется вероятность ошибки для кодов конечных длин.

Жадные алгоритмы построения МППЧ-кодов

Можно принять в качестве критерия качества кода обхват графа Таннера, построенного по проверочной матрице кода. Тогда задача построения кода сводится к поиску наилучших матриц с точки зрения этого критерия. Теоретически эта задача должна решаться методами комбинаторики, в частности, теории графов. К сожалению, известные комбинаторные конструкции (например, тройки Штайнера, графы Маргулиса, кубические графы и т.п.) обладают ограниченным набором параметров и, как правило, приводят к регулярным кодам. Для решения практических задач нужно строить проверочные матрицы произвольного размера и с заданным распределением весов столбцов. Естественное решение задачи — строить проверочную матрицу столбец за столбцом. Пошагово-оптимальные алгоритмы такого типа называют жадными алгоритмами.

Применительно к МППЧ-кодам первой успешной попыткой построения хороших кодов оказался алгоритм *последовательного добавления ребер* (ПДР) (progressive edge growing, PEG) [80]. Идея алгоритма состоит в добавлении единиц в матрицу H столбец за столбцом, начиная со столбцов меньшего веса. Каждая новая единица добавляется так, чтобы максимизировать длину появляющихся при этом циклов в графе Таннера.

Алгоритм ПДР порождает достаточно эффективные коды, но с ростом длины кодов отсутствие структуры порождает трудности в реализации кодирования и декодирования. Алгоритм ПДР был перенесен на квазициклические коды в работе [88]. Помимо этого, при последовательном построении проверочной матрицы можно учитывать другие критерии. На практике оправдал себя критерий ACE (approximated cycle extrinsic message degree), учитывающий не только длину цикла, но и количество связей с другими, не входящими в цикл столбцами. Этот критерий вычисляется как

$$\mathrm{ACE} = \min \sum (w_i - 2),$$

где w_i — веса столбцов цикла, суммирование выполняется по всем столбцам цикла и минимум отыскивается по всем циклам графа Таннера. Применение этого критерия в сочетании с ПДР рассматривалось в работе [113].

Еще один подход к жадному построению проверочных матриц квазициклических МПП-кодов рассматривается в [55]. Код задается в форме (11.15). На первом этапе строится базовая матрица B, а затем — матрица степеней W. Построение выполняется, начиная со столбцов малого веса. Накладывается ограничение на так называемый *профиль обхватов* — последовательность значений обхвата g_i, $i = 1, 2, ...$ для подматриц, состоящих из первых i столбцов. На каждом шаге поиска формируется список кандидатов. Если удается построить всю матрицу с заданным профилем, то лучший из кандидатов выбирается с помощью моделирования. Если нет, нужно ослабить требования и возобновить поиск сначала. Этот алгоритм позволил построить хорошие коды для решения ряда практических задач (см. параграф 11.5).

Коды с накоплением и повторением

Коды с накоплением и повторением (НП) (repeat-accumulate (RA) codes) появились еще в 1998 году как простая модель кодирования, удобная для теоретического анализа и в то же время позволяющая приблизиться к пропускной способности канала [67]. По мере нарастания интереса к МППЧ-кодам НП-коды стали рассматриваться как конкурентоспособный класс МППЧ-кодов.

Идея кодирования проста: каждый информационный символ сначала повторяется фиксированное число раз s, затем вся последовательность перемешивается и пропускается через сумматор-накопитель. Накопитель заменяет каждый элемент его суммой по модулю два со всеми предыдущими элементами. К кодовому слову дописывается последовательность информационных символов. Скорость такого кода равна

$$R = \frac{1}{s+1}.$$

Найдем представление НП-кода с помощью порождающей и проверочной матрицы. Описанная процедура кодирования реали-

зуется умножением информационной последовательности $\boldsymbol{m} = (m_1, m_2, ..., m_k)$ на матрицу вида

$$G = \left(R_s \Pi A \,\vdots\, I_k \right), \tag{11.19}$$

где I_k — единичная матрица порядка матрица k, R_s — матрица-повторитель

$$R_s = \underbrace{\left(I_k \,\vdots\, I_k \,\vdots \cdots \vdots\, I_k \right)}_{s \text{ матриц}},$$

через Π обозначена перестановочная матрица порядка sk, квадратная матрица A порядка sk описывает накопитель

$$A = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 0 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}. \tag{11.20}$$

Поскольку порождающая матрица (11.19) задана в систематической форме, находим проверочную матрицу

$$H = \left(I_{ks} \,\vdots\, A^{\mathrm{T}} (R_s \Pi)^{\mathrm{T}} \right). \tag{11.21}$$

Эквивалентную проверочную матрицу получим, умножив эту матрицу справа на бидиагональную матрицу

$$B = \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 \\ 1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 1 & 1 \end{pmatrix}. \tag{11.22}$$

В итоге приходим к проверочной матрице НП-кода в простой форме

$$H = \left(B \,\vdots\, (R_s \Pi)^{\mathrm{T}} \right). \tag{11.23}$$

Такой код обладает несколькими недостатками: низкая скорость, фиксированное распределение весов столбцов и строк, использование случайной перестановочной матрицы большого размера. Чтобы преодолеть эти недостатки и улучшить характеристики кодов, сначала были предложены иррегулярные НП коды (IRA

codes, [83]), затем в схему кодера был введен еще один накопитель, уже перед повторителем (ARA codes, [42]). В работе [117] построены структурированные НП-коды. В итоге, после всех усовершенствований, структура проверочной матрицы почти совпадает со структурой проверочной матрицы квазициклического МППЧ-кода с малой сложностью кодирования (11.16). Разница состоит в том, что столбец $\boldsymbol{h}_0(D)$ вместо веса 3 для квазициклического кода имеет вес 1 для НП-кода.

Эта разница, хотя и не может служить преимуществом кода, не слишком сильно ухудшает его характеристики. Примером практического применения структурированных иррегулярных НП-кодов является стандарт DVB-S2 (см. параграф 11.5).

МППЧ-коды из РС-кодов

Эмпирически установлено, что коды с обхватом 4 обладают плохими характеристиками, но среди кодов с обхватом 6 существуют коды с очень хорошими характеристиками. Поэтому задача построения МППЧ-кода может быть сформулирована как задача построения кода с обхватом не меньше 6.

Условие отсутствия циклов длины 4 состоит в том. что никакие два столбца проверочной матрицы не должны совпадать более чем в одной позиции. Следующая алгебраическая конструкция из работы [68] гарантирует выполнение этого условия.

Рассмотрим код Рида–Соломона над полем $\mathrm{GF}(q)$, $q = 2^m$, с минимальным расстоянием d с двумя информационными символами. Длина такого кода будет равна $n = d + 1$ и порождающая матрица может быть представлена в форме

$$G = \begin{pmatrix} g_1 & g_2 & \dots & g_d & 0 \\ 0 & g_1 & \dots & g_{d-1} & g_d \end{pmatrix}.$$

Множество из q^2-1 нетривиальных линейных комбинаций строк дает последовательности из d или $d+1$ чисел, совпадающие не более, чем в одной позиции.

Будем записывать ненулевые q-ичные элементы кодовых слов РС-кода как двоичные векторы длины $q-1$ с единственной единицей на позиции, определяемой номером элемента в поле $\mathrm{GF}(q)$

(способ нумерации не важен). Полученные двоичные векторы длины $n(q-1)$ совпадают не более, чем в одной позиции и, следовательно, могут служить набором, из которого можно выбирать столбцы проверочной матрицы МППЧ-кода.

На пути к реалистичным кодовым конструкциям нужно научиться формировать матрицы с заданным распределением весов столбцов и строк, обеспечить низкую сложность описания кодов и сложность кодирования. Получаемые в конечном результате коды на основе PC-кодов успешно конкурируют с другими конструкциями кодов.

Недвоичные МППЧ-коды

До сих пор мы рассматривали только двоичные МППЧ-коды. Возможность построения недвоичных кодов была исследована уже в основополагающей работе Галлагера [76, 5]. После переоткрытия МППЧ-кодов в 1990-е годы интерес к недвоичным кодам был стимулирован работой [65], в которой было показано, что недвоичные коды могут быть эффективнее двоичных.

Более точно, говоря о недвоичных МППЧ-кодах, по умолчанию подразумевают так называемые *двоичные образы* недвоичных кодов над полями характеристики 2. Строится МППЧ-код с проверочной матрицей размера $(c-b)\times c$ над полем $\mathrm{GF}(q)$, $q=2^m$. Двоичная информационная последовательность длины mb интерпретируется как последовательность длины b над полем $\mathrm{GF}(q)$. После кодирования получаем q-ичную кодовую последовательность длины c, из которой получаем двоичное кодовое слово длины mc.

Чтобы понять причины, по которым недвоичные коды могут быть лучше двоичных, напомним идею алгоритма декодирования двоичных МППЧ-кодов. Алгоритм распространения доверия на каждой итерации обрабатывает проверочные узлы, соответствующие строкам проверочной матрицы, затем символьные узлы, соответствующие столбцам. Каждая строка задает код с проверкой на четность для некоторым кодовых символов. Результат обработки строки — обновленные оценки надежностей символов, входящих в проверку. При обработке столбца мы выносим решение о кодовом символе по нескольким оценкам его надежности.

Код строк — код с проверкой на четность, имеет очень слабую корректирующую способность. Можно ожидать, что замена этого кода на более мощный позволит улучшить характеристики кода в целом.

Пусть теперь элементы строки — недвоичные символы. Обработка узла, как и прежде, — декодирование кода с проверкой на четность, но уже над полем $\mathrm{GF}(2^m)$. Если вес проверки был равен K, то вместо кода $(K, K-1)$ в двоичном случае мы имеем как бы $(Km, Km-m)$-код. Хотя код по-прежнему высокоскоростной, но имеет больше проверочных символов, что создает некоторый потенциал для получения энергетического выигрыша.

Единственное изменение в алгоритме декодирования состоит в том, что для вычисления надежностей q-ичных символов строк нужно воспользоваться алгоритмом БКДР (см. параграф 4.6). Решетка кода имеет q узлов на каждом ярусе и q ребер, конкурирующих в каждом узле. Таким образом, сложность реализации алгоритма имеет порядок q^2.

Способы уменьшения сложности — активная область исследований в настоящее время. Существенные шаги в этом направлении сделаны, в частности, в работе [66]. Тем не менее, относительно высокая сложность декодирования — по-прежнему основное препятствие для использования недвоичных кодов в реальных системах связи.

11.5. Коды для стандартов: результаты моделирования

Среди большого разнообразия практических задач, в которых находят применение коды с малой плотностью проверок на четность, мы рассмотрим два примера. Первый пример — стандарт WiMAX [118] для передачи данных в мобильных сетях связи. Второй пример — стандарт DVB-S2 [119], предназначенный для передачи сигналов цифрового телевидения высокого разрешения.

Эти два стандарта сильно отличаются друг от друга по характеристикам каналов связи и требованиям к качеству передачи. В обоих случаях мы имеем дело с миллионами потребителей конечных

устройств, что обостряет конкуренцию между разработчиками способов защиты данных от ошибок за место в будущих версиях стандартов.

Каналы стандарта WiMAX — это беспроводные каналы связи, характеризующиеся большой нестабильностью условий передачи. Подспорьем служит наличие возможности запроса на повторную передачу ненадежных блоков. Учитывая эти особенности, передача ведется короткими блокам от 576 до 2304 кодовых символов, с шагом по 24 символа. Скорости кодов выбираются из набора значений $\{1/2, 2/3, 3/4, 5/6\}$.

В текущей версии стандарта используются квазициклические коды с описанным выше алгоритмом кодирования. Базовая матрица для всех скоростей кодов имеет одинаковое число столбцов равное 24, число строк определяется скоростью кода. Для упрощения реализации одна и та же базовая матрица и одна и та же матрица степеней для всех длин кодов при фиксированной скорости, меняется только параметр усечения кода M (степени, превышающие M, заменяются остатками от деления на M). Матрица имеет бидиагональную форму, допускающую кодирование с малой сложностью (см. параграф 11.4.2).

Указанные ограничения на размер матриц, простоту реализации кодирования и совместимость по задержке декодирования несколько сужают область поиска хороших кодов.

Характеристики кодов длины 576 и 2304 со скоростью $R = 1/2$ для стандарта показаны на рис. 11.7. Через FER и SNR обозначены вероятность ошибки передачи кодового слова и отношение сигнал/шум на бит соответственно. При моделировании выполнялось до 50 итераций декодирования по алгоритму распространения доверия, каждая точка графика соответствует не менее 100 ошибочным событиям.

Поскольку стандартом предусмотрено использование обратной связи для переспроса блоков с обнаруженными ошибками, приемлемой величиной вероятности ошибки на блок считают $FER = 10^{-3}$. Из представленных на рисунке кривых следует, что стандартный код обеспечивает такой уровень надежности при SNR=2.8 и 1.7 дБ для длин 576 и 2304 соответственно. Оптимизация квазициклического кода [55] позволяет снизить требования к SNR примерно на

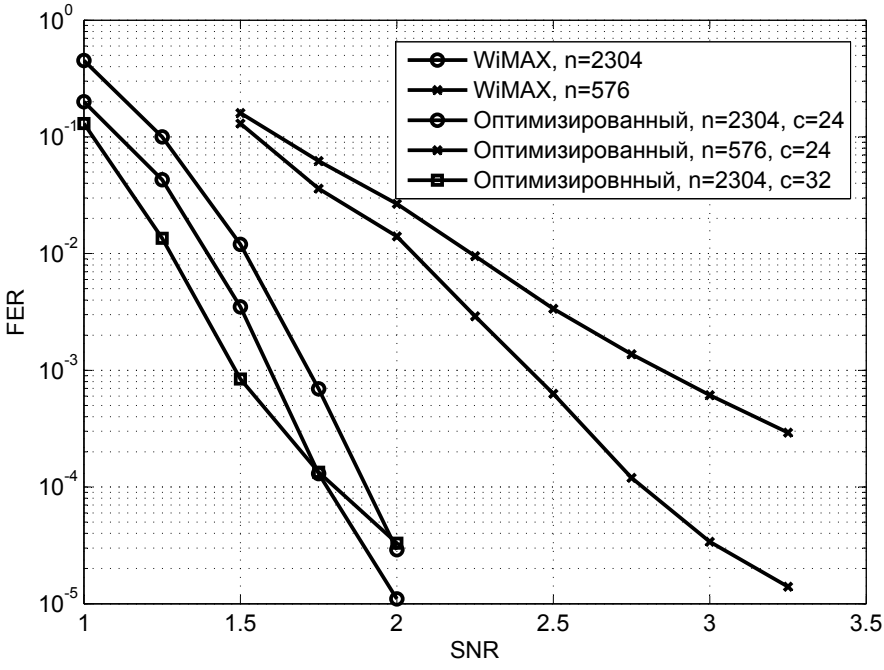


Рис. 11.7. Сравнение кодов стандарта WiMAX со скоростью $R = 1/2$ с другими МППЧ-кодами

0.5 и 0.15 дБ. Дополнительный выигрыш можно получить, увеличив размер базовой матрицы.

Рассмотрим теперь стандарт DVB-S2. В стандарте используется итерация МППЧ-кода с БЧХ-кодом. Для того чтобы упростить сравнение кодов, мы приводим на рис. 11.8 характеристики только МППЧ-кода. Кодер и декодер стандартного кода, принадлежащего классу кодов с накоплением и повторением, реализованы в пакете MATLAB, и именно этот пакет использовался для получения графиков.

Сплошные кривые на рисунке соответствуют вероятности ошибки на блок (FER), пунктирные — вероятности ошибки на бит (BER). Нужно отметить, что в данном приложении величина FER менее важна, чем BER, поскольку ошибки в отдельных двоичных симво-

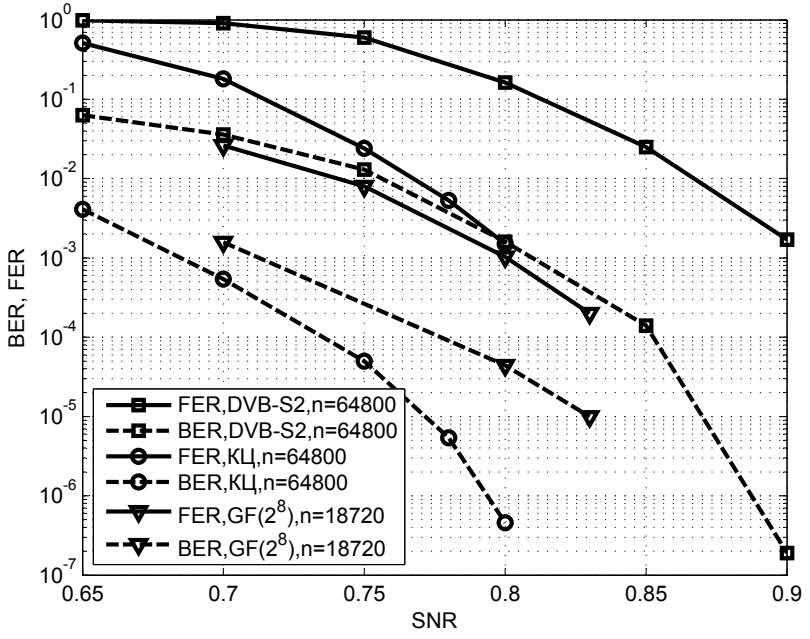


Рис. 11.8. Сравнение кода стандарта DVB-S2 длины 64800 со скоростью $R = 1/2$ с другими длинными МППЧ-кодами

лах, как правило, не приводят к разрушению всего кадра изображения, а лишь к искажению его небольшой части. Если таких ошибок немного, они могут не повлиять на качество воспроизведения телевизионного сигнала.

Стандартный код DVB-S2 позволяет надежно передавать информацию при отношении сигнал/шум на бит порядка 0.9 дБ. Оптимизированный квазициклический код [55] выигрывает по сравнению со стандартным кодом примерно 0.1 дБ. Интересно, что недвоичный МППЧ-код над полем $\mathrm{GF}(2^8)$ обеспечивает сопоставимое качество передачи при значительно меньшей длине блока. У сожалению, сложность его декодирования несколько выше, чем сложность декодирования двоичного МППЧ-кода.

Литература

Учебники и монографии

[1] Ахо А., Хопкрафт Ульман. *Построение и анализ вычислительных алгоритмов.*— М.: Мир, 1979.

[2] Блейхут Р. *Теория и практика кодов, контролирующих ошибки.*— М.: Мир, 1986.

[3] Витерби А. Д., Омура Д. К. *Принципы цифровой связи и кодирования.*— Радио и связь, 1982.

[4] Возенкрафт Д., Джекобс И. *Теоретические основы техники связи.*— М.: Мир, 1969.

[5] Галлагер Р. *Коды с малой плотностью проверок на четность.*— М.: Мир, 1966.

[6] Галлагер Р. *Теория информации и надежная связь.*— М.: Советское радио, 1974.

[7] Кларк Дж.,мл., Кейн Дж. *Кодирование с исправлением ошибок в системах цифровой связи.*— М.: Радио и связь, 1987.

[8] Колесник В. Д. *Кодирование при передаче и хранении информации (алгебраическая теория блоковых кодов).*— М.: Высшая школа, 2009.

[9] Колесник В. Д., Мирончиков Е. Т. *Декодирование циклических кодов.*— М.: Связь, 1968.

[10] Мак-Вильямс Ф. Дж., Слоэн Н. Дж. А. *Теория кодов, исправляющих ошибки.*— М.: Связь, 1979.

[11] Морелос-Сарагоса Р. *Искусство помехоустойчивого кодирования: Методы, алгоритмы, применения. Учебное пособие для студентов.*— Техносфера, 2005.

[12] Питерсон У., Уэлдон Э. *Коды, исправляющие ошибки.*— М.: Мир, 1976.

[13] Полтырев Г. Ш., Колесник В. Д. *Курс теории информации.*— М.: Наука, 1982.

[14] Соловьёва Ф. И. *Введение в теорию кодирования.*— Новосибирск: Изд-во НГУ, 2011.

[15] Сидельников В. М. *Теория кодирования.*— М.: Физматлит, 2008.

[16] Трифонов П. В. *Основы помехоустойчивого кодирования.*— СПб.: Изд-во Политехнического ун-та, 2011.

[17] Федоренко С. В. *Методы быстрого декодирования линейных блоковых кодов.*— СПб.: ГУАП, 2008.

[18] Форни Д. *Каскадные коды.*— М.: Мир, 1970.

[19] Шеннон К. *Работы по теории информации и кибернетике.*— М.: ИЛ, 1963.

[20] Berger T. *Rate-Distortion Theory.*— Wiley, 1971.

[21] Biglieri, E., Calderbank, R., Constantinides, A., Goldsmith, A., Paulraj, A., Poor, H. V. *MIMO wireless communications.*— Cambridge University Press, 2007.

[22] Biglieri E. *Coding for wireless channels.*— Springer, 2005.

[23] Bossert M. *Channel coding for telecommunications.*— John Wiley & Sons, Inc., 1999.

[24] McEliece R. *The theory of information and coding.*— Addison-Wesley, Reading, Mass., 1977.

[25] Johannesson R., Zigangirov K. Sh. *Fundamentals of convolutional coding.*— Wiley-IEEE press, 1999.

[26] Lin S., Costello D. *Error control coding.*— Pearson Higher Education, 2004.

[27] Moon T. K. *Error correction coding. Mathematical Methods and Algorithms.*— Jhon Wiley and Sons. 2005.

[28] Richardson T., Urbanke R. *Modern coding theory.*— Cambridge University Press, 2008.

[29] Yeung R. W. *Information theory and network coding.*— Springer, 2008.

Статьи

[30] Arikan E. Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels, *IEEE Transactions on Information Theory* 55(7) : 3051–3073, 2009.,

[31] Блох Э. Л., Зяблов В. В. Кодирование обобщенных каскадных кодов. *Проблемы передачи информации*, 10(3): 45–50, 1974.

[32] Бочарова И. Е., Кудряшов Б. Д., Сатюков Р. В. Сверточные и блоковые коды с малой плотностью проверок на основе графов. *Проблемы передачи информации*, 45(4):69–90, 2009.

[33] Гинзбург В. В. Многомерные сигналы для непрерывного канала. *Проблемы передачи информации*, 20(1):28–46, 1984.

[34] Влэдуц С. Г., Кацман Г. Л., Цфасман М. А. Модулярные кривые и коды с полиномиальной сложностью построения. *Проблемы передачи информации*, 20(1):47–55, 1984.

[35] Енгдал К., Зигангиров К. Ш. К теории низкоплотностных сверточных кодов. I", *Проблемы передачи информации*, 35(4): 12-–28, 1999.

[36] Зяблов В. В., Портной С. Л. Об одной системе модуляции и кодирования для гауссовского канала. *Проблемы передачи информации*, 23(3):18–26, 1987.

[37] Зяблов В. В., Сидоренко В. Р. Границы сложности декодирования линейных блоковых кодов с помощью решеток. *Проблемы передачи информации*, 29(3):3–9, 1993.

[38] Колесник В. Д., Мирончиков Е. Т. О спектрах линейных кодов. В сб. *Системы обработки и передачи информации*, Вып. 87, с. 54–58. ЛИАП, Ленинград, 1974.

[39] Крук Е. А. Граница для сложности декодирования линейных блоковых кодов. *Проблемы передачи информации*, 25(3):103–107, 1989.

[40] Кудряшов Б. Д. Декодирование блоковых кодов, получаемых из сверточных. *Проблемы передачи информации*, 26(2):18–26, 1990.

[41] Кудряшов Б. Д., Захарова Т. Г. Блоковые коды, получаемые из сверточных. *Проблемы передачи информации*, 25(4):98–102, 1989.

[42] Abbasfar A., Divsalar D., Yao K. Accumulate-repeat-accumulate codes. *IEEE Transactions on Communications*, 55(4):692–702, 2007.

[43] Bahl L. R, Cocke J., Jelinek E., Raviv J. Optimal decoding of linear codes for minimizing symbol error rate. *IEEE Transactions on Information Theory*, IT-20(1):284–287, Mar. 1974.

[44] Barg A. *Complexity Issues in Coding Theory*, Handbook of Coding Theory, North-Holland, Amsterdam, 1998, 649–754, 1998.

[45] Berrou C., Glavieux A. Near optimum error correcting coding and decoding: Turbo-codes. *IEEE Transactions on Communications*, 44(10):1261–1271, 1996.

[46] Berrou C., Glavieux A., Thitimajshima P. Near Shannon limit error-correcting coding and decoding: Turbo-codes (1). In *ICC '93 Geneva. Technical Program, Conference Record, IEEE International Conference on Communications, 1993*, volume 2, pages 1064–1070 vol.2, May 1993.

[47] Berrou C., Glavieux A., Thitimajshima P. Near Shannon limit error-correcting coding and decoding: Turbo-codes. (1). In *1993. ICC 93. Geneva. Technical Program, Conference Record, IEEE International Conference on Communications*, volume 2, pages 1064–1070. IEEE, 1993.

[48] Bocharova I., Kudryashov B. Rational rate punctured convolutional codes for soft-decision Viterbi decoding. *IEEE Transactions on Information Theory*, 43(4):1305–1313, 1997.

[49] Bocharova I. E., Handlery M., Johannesson R., Kudryashov B. D. A BEAST for prowling in trees. *IEEE Transactions on Information Theory*, 50(6):1295–1302, 2004.

[50] Bocharova I. E., Handlery M., Johannesson R., Kudryashov B. D. BEAST decoding of block codes obtained via convolutional codes. *IEEE Transactions on Information Theory*, 51(5):1880–1891, 2005.

[51] Bocharova I. E., Hug F., Johannesson R., Kudryashov B. D., Satyukov R. V. Searching for voltage graph-based LDPC tailbiting codes with large girth. *IEEE Transactions on Information Theory*, 58(4):2265–2279, 2012.

[52] Bocharova I. E., Johannesson R., Kudryashov B. D. Trellis complexity of short linear codes. *IEEE Transactions on Information Theory,* , 53(1):361–368, 2007.

[53] Bocharova I. E., Johannesson R., Kudryashov B. D., Lončar M. BEAST decoding for block codes. *European Transactions on Telecommunications*, 15(4):297–305, 2004.

[54] Bocharova I. E., Johannesson R., Kudryashov B. D., Stahl P, Tailbiting codes: Bounds and search results. *IEEE Transactions on Information Theory*, 48(1):137–148, 2002.

[55] Bocharova I E., Johannesson R., Kudryashov B. D. Combinatorial optimization for improving QC LDPC codes performance. In *2013 IEEE International Symposium on Information Theory Proceedings (ISIT)*, pages 2651–2655. IEEE, 2013.

[56] Bose R. C., Ray-Chaudhuri D. K. On a class of error correcting binary group codes. *Information and control*, 3(1):68–79, 1960.

[57] Boutros J. J., Patel K. Performance of optimal codes at finite length. 2006.

[58] Caire G., Taricco G., Biglieri E. Bit-interleaved coded modulation. *IEEE Transactions on Information Theory*, 44(3):927–946, 1998.

[59] Calderbank A R., Forney G. D. Jr, Vardy A. Minimal tail-biting trellises: The Golay code and more. *IEEE Transactions on Information Theory*, 45(5):1435–1455, 1999.

[60] Calderbank A. R., Rains E. M., Shor P. W ., Sloane N. J. Quantum error correction via codes over GF (4). *IEEE Transactions on Information Theory,* 44 (4): 1369–1387, 1998.

[61] Cedervall M. L., Johannesson R. A fast algorithm for computing distance spectrum of convolutional codes. *IEEE Transactions on Information Theory,* , 35(6):1146–1159, 1989.

[62] Chung S. Y., Forney G. D. Jr,, Richardson T. J., Urbanke R. On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit. *IEEE Communications Letters*, 5(2):58–60, 2001.

[63] Coffey J. T., Goodman R. M. The complexity of information set decoding. *IEEE Transactions on Information Theory*, 36(5):1031–1037, 1990.

[64] Cohen G. A nonconstructive upper bound on covering radius. *IEEE Transactions on Information Theory*, 29(3):352–353, 1983.

[65] Davey M., Mackay D. J. C. Low density parity check codes over GF(q). *IEEE Commun Lett.*, 2(6):165–167, 1998.

[66] Declercq D., Fossorier M. Decoding algorithms for nonbinary LDPC codes over GF(q). *IEEE Transactions on Communications*, 55(4):633–643, 2007.

[67] Divsalar D., Jin H., McEliece R. J. Coding theorems for "turbo-like" codes. In *Proceedings of the annual Allerton Conference on Communication control and Computing*, volume 36, pages 201–210. Univ. of Illinois, 1998.

[68] Djurdjevic I., Xu J., Abdel-Ghaffar K., Lin.S. A class of low-density parity-check codes constructed based on Reed-Solomon codes with two information symbols. *IEEE Communications Letters*, 7(7):317–319, July 2003.

[69] Dumer I. Suboptimal decoding of linear codes: partition technique. *IEEE Transactions on Information Theory,*, 42(6):1971–1986, 1996.

[70] Forney, G. D., Jr. On decoding BCH codes. *IEEE Transactions on Information Theory*, 11(4):549–557, Oct 1965.

[71] Forney, G. D., Jr. Generalized minimum distance decoding. *IEEE Transactions on Information Theory,*, 12(2):125 – 131, apr 1966.

[72] Forney, G. D., Jr. Convolutional codes I: Algebraic structure. *IEEE Transactions on Information Theory*, IT-16:720–738, Nov. 1970.

[73] Forney, G. D., Jr. Convolutional codes, II: Maximum likelihood decoding. *Information and control*, 25(4):222–266, 1974.

[74] Forney, G. D., Jr. Coset codes. II. Binary lattices and related codes. *IEEE Transactions on Information Theory*, 34(5):1152–1187, 1988.

[75] Fossorier M. P. C., Lin S. Soft-decision decoding of linear block codes based on ordered statistics. *Information Theory, IEEE Transactions on*, 41(5):1379–1396, 1995.

[76] Gallager R. Low-density parity-check codes. *IRE Transactions on Information Theory,*, 8(1):21–28, 1962.

[77] Grassl M. *Bounds on the minimum distance of linear codes and quantum codes.* Online available at http://www.codetables.de, 2007.

[78] Hagenauer J. Rate-compatible punctured convolutional codes (RCPC codes) and their applications. *IEEE Transactions on Communications*, 36(4):389–400, 1988.

[79] Hocquenghem A. Codes correcteurs d'erreurs. *Chiffres*, 2(2):147–56, 1959.

[80] Hu X.-Y., Eleftheriou E, Arnold D.-M. Regular and irregular progressive edge-growth Tanner graphs. *IEEE Trans. Inf, Theory*, 51(1):386–398, 2005.

[81] Igal S., Shlomo S. *Performance analysis of linear codes under maximum-likelihood decoding: a tutorial.* Now Publishers Inc, 2006.

[82] Imai H., Hirakawa S. A new multilevel coding method using error-correcting codes. *IEEE Transactions on Information Theory*, 23(3):371–377, 1977.

[83] Jin H., Khandekar A., McEliece R. Irregular repeat-accumulate codes. In *Proc. 2nd Int. Symp. Turbo codes and related topics*, pages 1–8, 2000.

[84] Kasami T. A Gilbert-Varshamov bound for quasi-cycle codes of rate 1/2. *IEEE Transactions on Information Theory*, 20(5):679–679, 1974.

[85] Kötter R., Médard M. An algebraic approach to network coding *IEEE/ACM Transactions on. Networking*, 11(5): 782–795, 2003.

[86] Lafourcade A., Vardy A. Lower bounds on trellis complexity of block codes. *Information Theory, IEEE Transactions on*, 41(6):1938–1954, 1995.

[87] Levitin L., Hartmann C. A new approach to the general minimum distance decoding problem: The zero-neighbors algorithm. *IEEE Transactions on Information Theory*, 31(3):378–384, 1985.

[88] Li Z., Kumar B.V.K.V. A class of good quasi-cyclic low-density parity check codes based on progressive edge growth graph. In *Conference Record of the Thirty-Eighth Asilomar Conference on Signals, Systems and Computers, 2004.*, volume 2, pages 1990–1994. IEEE, 2004.

[89] Li S. Y. R., Yeung R. W., Cai N. Linear network coding, *IEEE Transactions on Information Theory*, 49(2):371–381. 2003. T

[90] Litsyn S., Rains E.M., Sloane N. J. A., *Table of Nonlinear Binary Codes*, Online available at http://http://www.eng.tau.ac.il/ litsyn/tableand/index.html, 1999.

[91] Ma J. H., Wolf J. K. On tail-biting convolutional codes. *IEEE Transactions on Communications*, 34(2):104–111, February 1986.

[92] MacKay D. J. C. Good error-correcting codes based on very sparse matrices. *IEEE Transactions on Information Theory*, 45(2):399–431, 1999.

[93] Massey J. L., Sain M. K. Inverses of linear sequential circuits. *IEEE Transactions on Computers*, 100(4):330–337, 1968.

[94] Muder D. J. Minimal trellises for block codes. *IEEE Transactions on Information Theory*, 34(5):1049–1053, 1988.

[95] Myung S, Yang K., Kim J. Quasi-cyclic LDPC codes for fast encoding. *IEEE Transactions on Information Theory*, 51(8):2894–2901, 2005.

[96] Nimbalker A., Blankenship Y., Classon B., Blankenship T. K. ARP and QPP interleavers for LTE turbo coding. In *Wireless Communications and Networking Conference, 2008. WCNC 2008. IEEE*, pages 1032–1037. IEEE, 2008.

[97] Poltyrev G. Bounds on the decoding error probability of binary linear codes via their spectra. *IEEE Transactions on Information Theory*, 40(4):1284–1292, 1994.

[98] Reed I. S., Solomon G. Polynomial codes over certain finite fields. *Journal of the Society for Industrial & Applied Mathematics*, 8(2):300–304, 1960.

[99] Richardson T. J., Shokrollahi M.-A., Urbanke R. L. Design of capacity-approaching irregular low-density parity-check codes. *IEEE Transactions on Information Theory*, 47(2):619–637, 2001.

[100] Semenov S., Trofimov A. Convolutional codes and turbo-codes. *Modulation and Coding Techniques in Wireless Communications*, pages 161–205, 2011.

[101] Sesia S., Toufik I., Baker M. *LTE: the UMTS long term evolution.* Wiley Online Library, 2009.

[102] Shannon C. E. Probability of error for optimal codes in a Gaussian channel. *Bell System Technical Journal*, 38(3):611–656, 1959.

[103] Shannon C. E. A mathematical theory of communication. *Bell Syst. Tech. J.*, 27(1):379–423, 1948.

[104] Shao R. Y., Lin S., Fossorier M. P. C. Two decoding algorithms for tailbiting codes. *IEEE Transactions on Communications*, 51(10):1658–1665, 2003.

[105] Solomon G., Tilborg H. C. A. A connection between block and convolutional codes. *SIAM Journal on Applied Mathematics*, 37(2):358–369, 1979.

[106] Tanner R. M. A recursive approach to low complexity codes. *IEEE Transactions on Information Theory*, 27(5):533–547, 1981.

[107] Thorpe J. Low-density parity-check (LDPC) codes constructed from protographs *IPN progress report*, V. 42, no 154, pp. 42–154, 2003.

[108] Ungerboeck G., Channel coding with multilevel/phase signals. *IEEE Transactions on Information Theory*, 28(1):55–67, 1982.

[109] Valembois A., Fossorier M. P. C. Box and match techniques applied to soft-decision decoding. *IEEE Transactions on Information Theory*, 50(5):796–810, 2004.

[110] Valembois A., Fossorier M. P. C. Sphere-packing bounds revisited for moderate block lengths. *EEE Tranctions on Information Theory*, 50(12):2998–3014, 2004.

[111] Van de Meeberg L. A tightened upper bound on the error probability of binary convolutional codes with Viterbi decoding. *IEEE Transactions on Information Theory*, 20(3):389–391, 1974.

[112] Vardy A. Trellis structure of codes. *Handbook of coding theory*, 2:1989–2117, 1998.

[113] Vukobratovic D., Senk V. Generalized ACE constrained progressive edge-growth LDPC code design. *IEEE Communications Letters*, 12(1):32–34, 2008.

[114] Wei L.-F. Rotationally invariant convolutional channel coding with expanded signal space-Part II: Nonlinear codes. *IEEE Journal on Selected Areas in Communications*, 2(5):672–686, 1984.

[115] Wicker S. B., Bhargava V. K. *Reed-Solomon codes and their applications.* Wiley, 1999.

[116] Wolf J. Efficient maximum likelihood decoding of linear block codes using a trellis. *IEEE Transactions on Information Theory*, 24(1):76–80, 1978.

[117] Zhang Y., Ryan W. E. Structured IRA codes: performance analysis and construction. *IEEE Transactions on Communications*, 55(5):837–844, 2007.

Стандарты

[118] 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 and Corrigendum 1,"IEEE Std 802.16e-2005 and IEEE Std 802.16-2004/Cor 1-2005 (Amendment and Corrigendum to IEEE Std 802.16-2004), 2006.

[119] Digital Video Broadcasting (DVB); Second generation framing structure, channel coding and modulation systems for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications (DVB-S2), 2009.

[120] ITU-T Recommendation G.992.3. Asymmetric digital subscriber line transceivers 2 (ADSL2) (2005).

[121] ITU-T Recommendation V.34, A Modem Operating at Data Signalling Rates of Up to 28,800 bit/s for Use on the General Switched Telephone Network and on Leased Point-to-Point 2-Wire Telephone-Type Circuits, 1994.

Предметный указатель

Д

З

И

К

Л

М

О

П

P

C

Т

Ф

Ц

Ч

Ш

Э