МИР СВЯЗИ

Р. МОРЕЛОС-САРАГОСА

Искусство помехоустойчивого кодирования. Методы, алгоритмы, применение

Перевод с английского
В. Б. Афанасьева

Рекомендовано ИППИ РАН
в качестве учебного пособия
для студентов, обучающихся
по направлениям подготовки
"Прикладные математика и физика"
и "Телекоммуникации"

ТЕХНОСФЕРА
Москва
2005

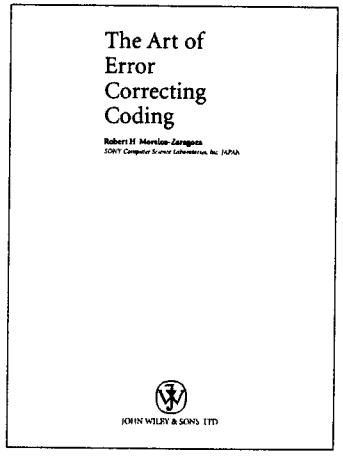
Р. Морелос-Сарагоса
Искусство помехоустойчивого кодирования.
Методы, алгоритмы, применение.
Москва:
Техносфера, 2005. – 320с. ISBN 5-94836-035-0

Новейшее пособие по теории и практике цифровой связи, не имеющее аналогов в литературе на русском языке.

Наиболее активно идеи помехоустойчивого кодирования внедряются в системах мобильной связи и в магистральных высокоскоростных линиях. Быстрое распространение Интернета и средств Мультимедиа стимулирует применение кодов, исправляющих ошибки, для защиты банков данных огромной емкости от случайных или преднамеренных искажений.

Помимо классических алгоритмов декодирования блоковых и сверточных кодов детально рассмотрены современные идеи декодирования с «мягким решением» и итеративного декодирования.

Идеальное учебное пособие для студентов программистских и связных специальностей, инженеров-разработчиков и практиков.



ISBN 5-94836-035-0
ISBN 0471 49581 6 (англ.)

Содержание

ПРЕДИСЛОВИЕ АВТОРА

Эта книга появилась в результате общения с коллегами из академии и промышленности по всему миру, присылавших сотни электронных писем с вопросами по теории и применению кодов, исправляющих ошибки (ECC). Больше всего вопросов поступало от инженеров и научных работников по компьютерным системам, которым необходимо было выбрать, реализовать или промоделировать конкретные схемы кодирования. Все вопросы высвечивались на ECC интернет-странице, которая предварительно была создана в лаборатории проф. Имаи в Институте технических исследований Университета Токио в начале 1995. Читатель сразу заметит отсутствие в тексте теорем и доказательств. Мой подход состоит в том, чтобы обучать основам на простых примерах. Ссылки на теоретические работы делаются по мере надобности. Эта книга создавалась как справочное пособие для аспирантов и специалистов, интересующихся изучением основных методов и применением ECC. Компьютерные программы, реализующие основные алгоритмы кодирования и декодирования практических схем кодирования, доступны на интернет-сайте

http://the-art-of-ecc.com

Повсюду в тексте этот сайт упоминается как ECC интернет-сайт. Эта книга уникальна тем, что основные идеи помехоустойчивого кодирования вводятся с помощью простых иллюстративных примеров. Компьютерные программы, написанные на языке программирования C и доступные на ECC интернет-сайте, позволяют продемонстрировать реализацию и применение в системах кодовой модуляции основных алгоритмов кодирования и декодирования важнейших кодовых схем, таких как сверточные коды, коды Хемминга и БЧХ, коды Рида-Соломона и турбо коды. Материал книги сосредоточен на основных алгоритмах для анализа и реализации кодов,

исправляющих ошибки (ECC). Существует обширная теория помехоустойчивого кодирования, прикосновение к которой осуществляется через ссылки к соответствующему материалу. Известно много хороших книг, посвященных теории кодирования, как например [LC], [MS], [PW], [Blah], [Bos], [Wic]. Читатели могут пролистать их прежде чем обратиться к материалам этой книги. В каждой ее главе рассматриваются базовые идеи какой-либо из схем кодирования или декодирования на простых и легко проверяемых численных примерах, вместо погружения в детали теории, лежащей в основании этих идей. Повсюду в книге даются основные инструменты, необходимые для оценки вероятности ошибки (эффективности) конкретным схем кодирования для некоторых основных моделей каналов передачи сигналов. Поддержка материала этой книги программами на ECC интернет-сайте делает ее уникальной.

Книга посвящена искусству помехоустойчивого кодирования в том смысле, что она ориентирована на проблемы выбора, реализации и моделирования алгоритмов кодирования и декодирования кодов, исправляющих ошибки. Книге организована следующим образом. В первой главе дается введение в базовые концепции исправления ошибок и технику кодирования и декодирования. В главе 2 рассматриваются важные и, вместе с тем, простые для понимания классы кодов, такие как коды Хемминга, Голея и Рида-Маллера. В главе 3 изучаются циклические коды и важное семейство кодов Боуза-Чоудхури-Хоквингема (БЧХ). Здесь дается введение в арифметику конечных полей и классические алгоритмы декодирования, такие как алгоритмы Берлекэмпа-Мэсси, Евклида и Питерсона-Горенстейна-Цирлера (PGZ), с простыми для понимания и проверки примерами. Глава 4 посвящена кодам Рида-Соломона и исправлению ошибок и стираний. Дается подробное исследование возможных алгоритмов вместе с примерами их работы. В главе 5 вводятся двоичные сверточные коды. Материал этой главы концентрируется на понимании основной структуры этих кодов вместе с объяснением алгоритма Витерби в Хеммингoвой метрике. Вводятся и обсуждаются важнейшие проблемы реализации. В главе 6 даются некоторые способы модифицирования отдельного кода и комбинирования нескольких кодов вместе с простыми примерами. Глава 7 посвящена алгоритмам мягкого декодирования. Некоторые из них, как например, мягкое декодирование по упорядоченным статистикам, еще не нашли должного отражения в литературе. В главе 8 представлена уникальная трактовка турбо кодов (параллельного и последовательного типа) и блоковых кодов-произведений с точки зрения теории кодирования. В этой же главе исследуются коды с малой плотностью проверок на четность. Для всех этих классов кодов описываются основные алгоритмы декодирования с простыми примерами. Наконец, глава 9 посвящена весьма эффективной технике комбинирования кодов, исправляющих ошибки, с цифровой модуляцией. Рассматриваются остроумные алгоритмы их декодирования. Книга включает всеобъемлющую библиографию для читателей, которые хотят знать больше о красивой теории. Я надеюсь, что эта книга станет ценным и необходимым инструментом как для студентов, так и для практиков, интересующихся этой интересной, увлекательной и никогда не кончающейся областью теории информации.

Я хотел бы выразить благодарность всем, кто повлиял на работу с этой книгой. Профессору Франциско Гарсиа Угалде, Национальный Университет Мехико, который ввел меня в увлекательный мир кодов, исправляющих ошибки. Часть этой книги основана на моей бакалаврской диссертации. Профессору Эдварду Бертраму, Университет Гаваи, за обучение меня основам абстрактной алгебры. Профессору Давиду Муньозу, Технологический институт высшей школы Монтерей, Мексика, за его доброту и поддержку. Профессорам Тадео Касами, Университет г. Хиросима, Тору Фудживара, Университет г. Осака, и Хидеки Имаи, Университет г. Токио, за поддержку моего пребывание в Японии в качестве приглашенного научного сотрудника-исследователя. Дану Лати и Эдвайту Могре,

корпорация больших интегральных схем, за бесконечные обсуждения проблем моделирования и возможность приобрести опыт реализации идей в интегральных схемах. Профессору Марку Фоссоуриеру, Гавайский Университет, за помощь. Моему коллеге доктору Мише Михалевичу, Лаборатория компьютерных систем, корпорация Сони, за объяснение связи между декодированием и криптоанализом. Я хотел бы также выразить искреннюю благодарность доктору Марио Токоро, Президенту лаборатории компьютерных систем корпорации Сони, и профессору Райю Коно, Национальный университет Йокагамы, за предоставленную мне возможность использования превосходного оборудования для написания этой книги. В частности, я хочу выразить бесконечную благодарность профессору Шу Лин, который находится теперь в Калифорнийском университете в Дависе, который поддерживал меня, пока я был аспирантом на Гаваях, и убедил меня продолжить исследования по этой увлекательной теме. Наконец, но ничуть не менее, я хочу поблагодарить студентов и коллег, которые все эти годы слушали лекции в Мексике, Японии и США.

Я посвящаю эту книгу памяти Ричарда Хемминга, Клода Шеннона и Густава Соломона, трем выдающимся ученым, которые оказали огромное влияние на жизнь и работу современного поколения людей.

Роберт Морелос-Сарагоса
Токио, Япония, Апрель 2002.

ПРЕДИСЛОВИЕ

В современных разработках систем цифровой связи и хранения данных все большее значение принимает теория информации. Наилучшей демонстрацией этого является возникновение и быстрое внедрение турбо кодов и блоковых кодов-произведений во многих спутниковых и беспроводных системах связи. Я с удовольствием рекомендую эту новую книгу, сделанную доктором Робертом Морелос-Сарагоса, всем, кто интересуется кодами, исправляющими ошибки, или их применением. В этой книге ключевые концепции помехоустойчивого кодирования (ECC) вводятся способом легким для восприятия. Материал книги хорошо структурирован и представлен с помощью простых примеров. Это вместе с компьютерными программами, доступными на интернет-сайте, является новым подходом в обучении основным методам, используемым в разработках и применении кодов, исправляющих ошибки.

Одна из лучших особенностей этой книги является то, что она дает естественное введение в принципы построения и методы декодирования турбо кодов, кодов с низкой плотностью проверок на четность (LDPC) и кодов-произведений, основанное на алгебраическом кодировании для каналов. В этом контексте турбо коды рассматриваются как перфорированные коды-произведения. С помощью простых примеров идеи и структуры, использованные в конструкциях и итеративном декодировании кодов-произведений, представлены здесь единым (unparalleled) образом. Подробное исследование алгебраических процедур исправления ошибок и стираний кодами Рида-Соломона также заслуживает упоминания. Автору удалось сделать хорошее введение в принципы построения некоторых важных классов кодовой модуляции, сочетающей помехоустойчивое кодирование с цифровой модуляцией.

Я уверен, что профессиональные инженеры и научные работники примут эту книгу и как хороший учебник, и как

полезный справочник. Корпоративный ECC интенет-сайт предоставляет уникальную поддержку, которая вряд ли имеется где-нибудь еще. Кстати, этот интернет-сайт родился в моей лаборатории в университете Токио в 1995 г. в то время, когда здесь работал доктор Морелос-Сарагоса. До июня 1997 он выполнил много хороших работ в качестве моего компаньона-исследователя и написал много высококачественных статей. Роберт вежлив, скромен, трудолюбив и всегда приветлив. В заключении, я настоятельно рекомендую *Искусство помехоустойчивого кодирования* как замечательное введение и справочное пособие по принципам построения и применения кодов, исправляющих ошибки.

Профессор Хидеки Имаи
Университет Токио
Токио, Япония, Апрель 2002

Глава 1

ВВЕДЕНИЕ

История кодов, исправляющих ошибки, началась с изобретения кодов Хемминга [Ham] почти одновременно с появлением основополагающей работы Шеннона [Sha]. Чуть позже были предложены коды Голея [Gol]. Эти классы кодов являются оптимальными. Понятие оптимальности кода мы рассмотрим в последующих разделах.

На Рисунке 1 показана блок-схема канонической цифровой системы передачи или хранения информации. Это знаменитый *Рисунок 1*, с которого начинаются почти все книги по теории помехоустойчивого кодирования (в дальнейшем будет использоваться авторское сокращение ECC со значением – *помехоустойчивое кодирование, или кодирование с исправлением ошибок, или коды, исправляющие ошибки*). Источник и получатель информации обычно включают какое-либо кодирование (преобразование) источника, соответствующее природе информации. Кодирующее устройство (кодер канала) системы помехоустойчивого кодирования получает информационные символы от источника и добавляет к ним избыточные символы таким образом, чтобы могла быть исправлена большая часть ошибок, возникающих в процессе модуляции сигналов, их передачи по каналу с шумом и демодуляции.

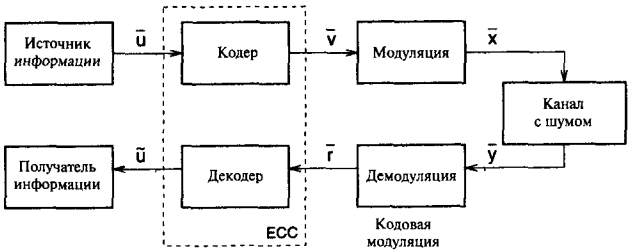


Рис. 1. Каноническая цифровая система связи.

Обычно предполагается, что в канале связи отсчеты аддитивного шумового процесса прибавляются к модулированным символам (рассматривается узкополосное комплексное представление сигналов). Отсчеты шума предполагаются независимыми от источника символов. Эту модель сравнительно легко исследовать. Она легко позволяет включить каналы с гауссовым шумом (АБГШ), каналы с общими Релеевскими замираниями, а также двоичный симметричный канал (ДСК). На приемной стороне декодирующее устройство (декодер канала) использует избыточные символы для исправления ошибок, внесенных каналом связи. В режиме обнаружения ошибок декодер ведет себя как кодер полученного из канала сообщения и проверяет совпадение вычисленных избыточных символов с принятыми.

В классической теории кодов, исправляющих ошибки, комплекс, включающий модулятор, демодулятор и шумовую среду распространения сигналов, называется *дискретным каналом без памяти* с входом **v** и выходом **r**. Примером такого канала является система передачи двоичных сигналов по каналу с АБГШ (аддитивным белым гауссовым шумом), который моделируется как *двоичный симметричный канал* с вероятностью ошибки (или вероятностью перехода) p, равной вероятности ошибки на бит для двоичного сигнала в АБГШ,

$$p = Q\left(\sqrt{\frac{2E_b}{N_0}}\right) \tag{1.1}$$

где

$$Q(x) = \frac{1}{\sqrt{2\pi}}\int_x^\infty e^{-z^2/2}dz,\ x \geq 0, \tag{1.2}$$

является гауссовой Q-функцией[1] и E_b / N_0 есть отношение сигнал-шум (SNR) на бит. Этот случай будет исследован ниже в этой главе.

[1] Заметим, что в стандартном определении функции ошибок $Q(x) = \frac{1}{2}\mathrm{erfc}\,(x/\sqrt{2})$

В 1974 Мэсси [Mas3] предложил рассматривать помехоустойчивое кодирование (ECC) и модуляцию как единое целое, известное в современной литературе как *кодовая модуляция* (*coded modulation*) (Часто используется и несколько более точный термин – *сигнально кодовая конструкция*). Совместное конструирование кода и множества сигнальных точек обеспечивает более высокую эффективность и больший (*энергетический*) *выигрыш от кодирования* (*coding gain*)[2], чем последовательное применение ECC и модуляции. В этой книге рассмотрены некоторые методы комбинирования кодирования и модуляции, включая: *решетчатую кодовую модуляцию* (TCM – *trellis-coded modulation*) [Ung1] и *многоуровневую кодовую модуляцию* (MCM – *multilevel coded modulation*) [IH]. В системах кодовой модуляции «мягкое решение» (*soft decision*) на выходе канала вводится непосредственно в декодер и обрабатывается им. Напротив, в классической системе с помехоустойчивым кодированием в декодер вводится «жесткое решение» (*hard decision*) демодулятора.

Коды, исправляющие ошибки, можно комбинировать различными способами. Примером *последовательного каскадирования* (*serial concatenation*, т.е. каскадная конструкция в классическом смысле) является следующая конструкция. Многие годы наиболее популярной каскадной схемой ECC была комбинация внешнего кода Рида-Соломона (РС), промежуточного перемежения (или перемешивания – *interleaving*) и внутреннего сверточного кода. Эта конструкция была использована во многих приложениях от систем космической связи до цифровых широковещательных систем телевидения высокой четкости. Основная идея состоит в том, что пакеты ошибок, которые появляются на выходе декодера сверточного кода с мягким решением, могут быть разбиты на мелкие части с помощью интерливинга и затем полностью исправлены де-

[2] Выигрыш от кодирования определен как разность между отношениями сигнал-шум системы с кодированием и системы без кодирования при одинаковой скорости (и одинаковой вероятности ошибки).

кодером кода РС. Коды Рида-Соломона являются недвоичными кодами, каждый символ которых состоит из нескольких двоичных бит. Эти коды способны исправлять многократные пакеты ошибок. Преимуществом каскадной конструкции является то, что для нее требуются раздельные декодеры внутреннего и внешнего кодов вместо одного, но очень сложного декодера для каскадного кода в целом.

В книге изучаются разные типы систем с ECC. Сначала рассматриваются базовые кодовые конструкции и алгоритмы их декодирования в Хемминговом пространстве (т.е. работающие с битами). Затем, во второй части книги, вводятся алгоритмы декодирования с мягким решением для передачи двоичных сигналов, работающие в Евклидовом пространстве. В системах этого типа необходимая мощность передаваемых сигналов снижается на 2 дБ/бит по сравнению с декодерами в Хемминговом пространстве (с жестким решением). Рассматриваются декодеры с мягким решением нескольких типов. При этом основное внимание уделяется собственно алгоритмам (тому, как они работают), а не теоретическим вопросам (почему они работают). Наконец, последняя часть книги посвящена комбинированию кодов и перемежителей в конструкциях с итеративным декодированием, а также комбинированию помехоустойчивого кодирования и дискретной модуляции.

1.1. Кодирование для исправления ошибок: Основные положения

Все коды, исправляющие ошибки, основаны на одной общей идее: для исправления ошибок, которые могут возникнуть в процессе передачи или хранения информации, к ней добавляется некоторая избыточность. По основной схеме (используемой на практике), избыточные символы дописываются вслед за информационными, образуя кодовую последовательность или *кодовое слово* (*codeword*). В качестве иллюстрации на Рисунке 2 показано кодовое слово, сформированное процедурой кодирования *блокового кода* (*block code*). Такое кодирование называют *систематическим* (*systematic*). Это означает, что информационные символы всегда появляются на первых k позициях кодового слова. Символы на оставшихся n-k позициях являются различными функциями от информационных символов, обеспечивая тем самым избыточность, необходимую для обнаружения или исправления ошибок. Множество всех кодовых последовательностей называют *кодом, исправляющим ошибки* (*error correcting code*), и обозначают через C.

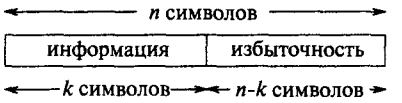


Рис. 2. Систематическое блоковое кодирование для исправления ошибок.

1.1.1. Блоковые и сверточные коды

В соответствии с тем, как вводится избыточность в сообщение, коды, исправляющие ошибки, могут быть разделены на два класса: блоковые и сверточные (*convolutional code*) коды. Обе схемы кодирования нашли практическое применение. Исторически сверточные коды имели преимущество главным образом потому, что для них известен алгоритм декодирования Витерби с мягким решением и в течение многих лет существовала убежденность в том, что блоковые коды невозможно декодировать с мягким решением. Однако последние достижения в теории и конструировании алгоритмов декодирования с мягким решением для линейных блоковых кодов помогли рассеять это убеждение. Более того, наилучшие ECC, известные сегодня (в начале двадцать первого века), являются блоковыми кодами (*нерегулярными кодами с низкой плотностью проверок — irregular low-density parity-check codes*).

При блоковом кодировании каждый блок информационных символов обрабатывается независимо от других. Другими

словами, блоковое кодирование является операцией без памяти в том смысле, что кодовые слова не зависят друг от друга. Выход сверточного кодера, напротив, зависит не только от информационных символов в данный момент, но и от предыдущих символов на его входе или выходе. Чтобы упростить объяснения, мы начнем с изучения структурных свойств блоковых кодов. Многие из этих свойств являются общими для обоих типов кодов.

Следует заметить, что на самом деле блоковые коды обладают памятью, если рассматривать кодирование как побитовый процесс в пределах кодового слова. Сегодня это различие между блоковыми и сверточными кодами кажется все менее и менее ясным, особенно после недавних достижений в понимании *решетчатой* (*trellis*) структуры блоковых кодов и *кольцевой* (*tail-biting*) структуры некоторых *сверточных* кодов.

Дополнение переводчика. *Название кольцевые сверточные коды еще не окончательно утвердилось в отечественной литературе, иногда tail-biting convolutional codes называют циклически замкнутыми.*

Специалисты по сверточным кодам иногда рассматривают блоковые коды как «коды с меняющейся во времени структурой решетки» (*time-varying trellis structure*). Аналогично, специалисты в области блоковых кодов могут рассматривать сверточные коды как «коды с регулярной структурой решетки».

1.1.2. Хеммингово расстояние, Хемминговы сферы и корректирующая способность

Рассмотрим двоичный код C, исправляющий ошибки. Если не все из 2^n возможных двоичных векторов длины n будут передаваться по каналу связи, то этот код может обладать свойством помехоустойчивости. В действительности, код C является подмножеством n-мерного двоичного векторного пространства $V_2 = \{0, 1\}^n$ таким, что элементы этого подмножества максимально удалены друг от друга.

В двоичном пространстве V_2 *расстояние* определяется как число позиций, на которых два вектора не совпадают. Пусть $\mathbf{x}_1=(x_{1,0}, x_{1,1},..., x_{1,n-1})$ и $\mathbf{x}_2=(x_{2,0}, x_{2,1},..., x_{2,n-1})$ два вектора в V_2. Тогда *Хеммингово расстояние* между векторами $\mathbf{x}_1$ и $\mathbf{x}_2$, обозначаемое как $d_H(\mathbf{x}_1, \mathbf{x}_2)$, равно

$$d_H(\mathbf{x}_1, \mathbf{x}_2)=\left|\left\{i : x_{1,i} \neq x_{2,i},\ \ 0 \le i < n\right\}\right| \qquad (1.3)$$

где через $|A|$ обозначено число элементов в множестве A.

Для заданного кода C *минимальное Хеммингово расстояние*, d_{min}, определяется как минимум Хеммингова расстояния по всем возможным парам различных кодовых слов,

$$d_{\min} = \min_{\mathbf{v}_1, \mathbf{v}_2 \in C} \left\{ d_H(\mathbf{v}_1, \mathbf{v}_2) | \mathbf{v}_1 \neq \mathbf{v}_2 \right\} \qquad (1.4)$$

Повсюду в книге обозначение (n, k, d_{min}) используется для параметров блокового кода длины n, который используется для кодирования сообщений длины k, и имеет минимальное Хеммингово расстояние d_{min}. Предполагается, что число кодовых слов этого кода равно $|C|=2^k$.

Пример 1. Простейшим примером является код-повторение длины 3. Каждый информационный бит повторяется три раза. Таким образом, сообщение «0» кодируется вектором (000), а сообщение «1», вектором (111). Так как два вектора различаются в трех позициях, Хеммингово расстояние между ними равно 3. На Рисунке 3 показано графическое представление этого кода. Трехмерное двоичное векторное пространство соответствует $2^3 = 8$ вершинам трехмерного единичного куба. Хеммингово расстояние между кодовыми словами (000) и (111) равно числу вершин, через которые проходит соединяющий их путь. Это эквивалентно числу координат, которые необходимо изменить, чтобы преобразовать (000) в (111) и наоборот. Таким образом, $d_H = ((000),(111)) = 3$. Так как в этом коде только два кодовых слова, то $d_{min} = 3$.

Двоичное векторное пространство V_2 обычно называют *Хеммингловым пространством*. Пусть $\mathbf{v}$ кодовое слово кода C. *Хеммингловой сферой* $S_t(\mathbf{v})$ радиуса t с центром в точке $\mathbf{v}$ является

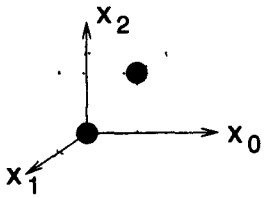


Рис. 3. (3,1,3) код-повторение в трехмерном векторном пространстве.

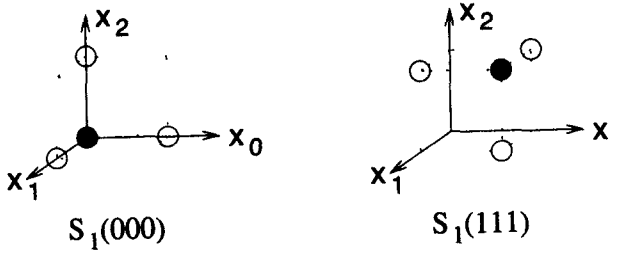


Рис. 4. Хемминговы сферы радиуса $t = 1$, окружающие кодовые слова (3,1,3) двоичного кода-повторения.

множество векторов (точек) в V_2 на расстоянии меньше или равном t от центра $\mathbf{v}$,

$$S_t(\mathbf{v}) = \{\mathbf{x} \in C \mid d_H(\mathbf{x}, \mathbf{v}) \le t\} \tag{1.5}$$

Заметим, что число слов (число векторов) в $S_t(\mathbf{v})$ равно

$$|S_t(\mathbf{v})| = \sum_{i=0}^{t} \binom{n}{i} \tag{1.6}$$

Пример 2. На Рисунке 4 показаны Хемминговы сферы радиуса $t = 1$, окружающие кодовые слова (3,1,3) двоичного кода-повторения.

Заметим, что Хемминговы сферы для этого кода не пересекаются, т.е. в пространстве V_2 нет векторов (или вершин в единичном трехмерном кубе), принадлежащих одновременно и $S_1(000)$, и $S_1(111)$. В результате, если изменить любую одну позицию кодового слова $\mathbf{v}$, то получится вектор, который, тем

не менее, останется внутри Хемминговой сферы с центром в $\mathbf{v}$. Эта идея является принципиальной для понимания и определения корректирующей способности кода C.

Корректирующей способностью (*error correcting capability*) t кода C называют наибольший радиус Хемминговой сферы $S_t(\mathbf{v})$ для всех кодовых слов $\mathbf{v} \subseteq C$ такой, что для любых различных пар $\mathbf{v}_i, \mathbf{v}_j \subseteq C$ соответствующие им Хемминговы сферы не пересекаются, т.е.

$$t = \max_{\mathbf{v}_i, \mathbf{v}_j \in C} \{l \mid S_l(\mathbf{v}_i) \cap S_l(\mathbf{v}_j) = 0, \mathbf{v}_i \neq \mathbf{v}_j\} \tag{1.7}$$

Это соответствует более распространенному определению

$$t = \lfloor (d_{\min} - 1)/2 \rfloor \tag{1.8}$$

где $\lfloor x \rfloor$ обозначена целая часть x, т.е. целое число меньше или равное x.

Заметим, что для определения минимального кодового расстояния произвольного блокового кода C в соответствии с (1.4), необходимо вычислить все $2^k(2^k - 1)$ расстояний между различными парами кодовых слов. Это практически невозможно даже для сравнительно коротких кодов, например, с $k = 50$. Одним из важных преимуществ *линейных* блоковых кодов является то, что для вычисления d_{min} достаточно знать только *Хемминговы веса* $2^k - 1$ ненулевых кодовых слов.

1.2. Линейные блоковые коды

Как уже упоминалось выше, построение хорошего кода означает поиск в V_2 подмножества элементов в наибольшей степени удаленных друг от друга. Это очень трудная задача. Более того, если даже это сделано, то остается неясным *как назначить кодовые слова информационным сообщениям.*

Линейный код (т.е. множество кодовых слов) является векторным подпространством в пространстве V_2. Это означает, что

операция кодирования может быть представлена умножением на матрицу. В терминах цифровой электроники простое кодирующее устройство может быть построено на элементах «исключающее ИЛИ», «И» и триггерах типа D. В этой главе операциями сложения и умножения в двоичном векторном пространстве являются «исключающее ИЛИ» (сложение по модулю 2) и «И», соответственно. Таблицы сложения и умножения двоичных элементов имеют следующий вид:

a	b	$a+b$	ab
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

1.2.1. Порождающая и проверочная матрицы

Пусть C двоичный линейный (n, k, d_{min}) код. Так как C есть k-мерное подпространство, то оно имеет базис, например, $(\mathbf{v}_0, \mathbf{v}_1,..., \mathbf{v}_{k-1})$ такой, что любое кодовое слово $\mathbf{v} \in C$ может быть записано как линейная комбинация элементов этого базиса:

$$\mathbf{v} = u_0\mathbf{v}_0 + u_1\mathbf{v}_1 + ... + u_{k-1}\mathbf{v}_{k-1} \tag{1.9}$$

где $u_i \in \{0, 1\}$, $0 \le i < k$. Уравнение (1.9) может быть записано в матричной форме через *порождающую матрицу* $\mathbf{G}$ и *вектор-сообщение* $\mathbf{u} = (u_0, u_1,..., u_{k-1})$:

$$\mathbf{v} = \mathbf{uG} \tag{1.10}$$

где

$$\mathbf{G} = \begin{pmatrix} \mathbf{v}_0 \\ \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_{k-1} \end{pmatrix} = \begin{pmatrix} v_{0,0} & v_{0,1} & \cdots & v_{0,k-1} \\ v_{1,0} & v_{1,1} & \cdots & v_{1,k-1} \\ \vdots & \vdots & \ddots & \vdots \\ v_{k-1,0} & v_{k-1,1} & \cdots & v_{k-1,k-1} \end{pmatrix} \tag{1.11}$$

Так как C является k-мерным векторным пространством в V_2, то существует $(n-k)$-мерное *дуальное пространство* (*dual space*) $C^{\perp}$, которое порождается строками матрицы $\mathbf{H}$, называемой *проверочной матрицей* (*parity-check matrix*), такой, что $\mathbf{GH}^{\top}=\mathbf{0}$, где через $\mathbf{H}^{\top}$ обозначена транспонированная матрица $\mathbf{H}$. Заметим, в частности, что любое кодовое слово $\mathbf{v} \in C$ удовлетворяет условию

$$\mathbf{vH}^{\mathrm{T}} = \mathbf{0} \tag{1.12}$$

Как будет показано в разделе 1.3.2 уравнение (1.12) является фундаментальным для *декодирования линейных кодов*.

Линейный код $C^{\perp}$, который генерируется матрицей $\mathbf{H}$, является двоичным линейным $(n, n-k, d^{\perp}_{min})$ кодом, который называют обычно *дуальным* коду C.

1.2.2. Вес как расстояние

Как упоминалось в разделе 1.1.2, линейные коды отличаются тем, что для определения минимального расстояния кода достаточно знать минимум Хеммингова веса ненулевых кодовых слов. Ниже этот факт будет доказан. Определим *вес Хемминга* $wt_H(\mathbf{x})$ вектора, $\mathbf{x} \in V_2$, как число ненулевых элементов в $\mathbf{x}$. Из определения Хеммингова расстояния ясно, что $wt_H(\mathbf{x}) = d_H(\mathbf{x}, \mathbf{0})$. Для двоичного линейного кода C получаем

$$d_H(\mathbf{v}_1, \mathbf{v}_2) = d_H(\mathbf{v}_1 + \mathbf{v}_2, \mathbf{0}) = wt_H(\mathbf{v}_1 + \mathbf{v}_2) \tag{1.13}$$

Наконец, из свойства линейности кода имеем $\mathbf{v}_1 + \mathbf{v}_2 \in C$. Отсюда следует, что минимальное расстояние кода C равно и может быть вычислено как минимальный вес по всем $2^k - 1$ ненулевым кодовым словам. Эта задача существенно проще, чем полный перебор по всем парам кодовых слов, хотя и остается очень сложной даже для кодов среднего размера (или размерности k).

1.3. Кодирование и декодирование линейных блоковых кодов

1.3.1. Кодирование с помощью матриц G и H

Равенство (1.10) определяет по существу правило кодирования для линейного блокового кода, которое может быть использовано непосредственно. Если кодирование должно быть систематическим, то произвольная порождающая матрица **G** линейного блокового (n, k, d_{min}) кода C может быть преобразована к *систематической форме* $\mathbf{G}_{sys}$ (иначе, к канонической форме) с помощью элементарных операций и перестановок столбцов матрицы. Матрица $\mathbf{G}_{sys}$ состоит из двух подматриц: $k \times k$ единичной матрицы, обозначаемой $\mathbf{I}_k$, и $k \times (n-k)$ проверочной подматрицы **P**. Таким образом,

$$\mathbf{G}_{sys} = (\mathbf{I}_k \mid \mathbf{P}) \tag{1.14}$$

где

$$\mathbf{P} = \begin{pmatrix} p_{0,0} & p_{0,1} & \cdots & p_{0,n-k-1} \\ p_{1,0} & p_{1,1} & \cdots & p_{1,n-k-1} \\ \vdots & \vdots & \ddots & \vdots \\ p_{k-1,0} & p_{k-1,1} & \cdots & p_{k-1,n-k-1} \end{pmatrix} \tag{1.15}$$

Так как $\mathbf{G}\mathbf{H}^{\top} = \mathbf{0}$, то отсюда следует, что систематическая форма проверочной матрицы имеет вид

$$\mathbf{H}_{sys} = (\mathbf{P}^{\top} \mid \mathbf{I}_{n-k}) \tag{1.16}$$

Пример 3. Рассмотрим двоичный линейный (4,2,2) код с порождающей матрицей

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

Перестановкой второго и четвертого столбцов преобразуем эту матрицу в систематическую форму

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

Таким образом, проверочная подматрица равна

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

Интересно отметить, что в данном случае выполняется соотношение[3] $\mathbf{P} = \mathbf{P}^{\top}$. Из (1.16) следует, что систематическая форма проверочной матрицы имеет вид

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

В дальнейшем будет использовано обозначение $\mathbf{u} = (u_0, u_1, \ldots, u_{k-1})$ для информационного сообщения и обозначение $\mathbf{v} = (v_0, v_1, \ldots, v_{n-1})$ для соответствующего кодового слова кода C.

Если параметры C таковы, что $k<(n-k)$ или, эквивалентно, *скорость кода* $k/n < 1/2$, то кодирование с помощью порождающей матрицы более экономно по числу логических операций. В этом случае

$$\mathbf{v} = \mathbf{u}\mathbf{G}_{sys} = (\mathbf{u}, \mathbf{v}_p) \tag{1.17}$$

где $\mathbf{v}_p = \mathbf{u}\mathbf{P} = (v_k, v_{k+1}, \ldots, v_{n-1})$ представляет проверочную часть кодового слова.

Однако, если $k > (n-k)$ или $k/n > 1/2$, тогда кодирование с помощью проверочной матрицы **H** требует меньшего количества вычислений. Этот вариант кодирования основан на уравнении (1.12) $(\mathbf{u}, \mathbf{v}_p)\mathbf{H}^{\top} = \mathbf{0}$. Проверочные позиции $(v_k, v_{k+1}, \ldots, v_{n-1})$ вычисляются следующим образом:

$$v_j = u_0 p_{0,j} + u_1 p_{1,j} + \ldots + u_{k-1} p_{k-1,j}, \quad k \le j < n. \tag{1.18}$$

Можно сказать, что элементами систематической формы проверочной матрицы являются коэффициенты проверочных уравнений, из которых вычисляются проверочные символы.

[3] В этом случае код называют *самодуальным*. См. Раздел 2.2.3 .

Этот факт будет использован в Разделе 8.3 при обсуждении кодов с низкой плотностью проверок.

Пример 4. Рассмотрим двоичный линейный (4,2,2) код из примера 3. Пусть сообщение и кодовые слова обозначены $\mathbf{u} = (u_0, u_1)$ и $\mathbf{v} = (v_0, v_1, v_2, v_3)$, соответственно. Из уравнения (1.18) получаем

$$v_2 = u_0 + u_1$$
$$v_3 = u_0$$

Соответствие между $2^2 = 4$ двух битными сообщениями и кодовыми словами имеет следующий вид:

$$\begin{array}{cc} (00) & (0000) \\ (01) & (0110) \\ (10) & (1011) \\ (11) & (1110) \end{array} \qquad (1.19)$$

1.3.2. Декодирование по стандартной таблице

В этом разделе представлена процедура декодирования, которая находит кодовое слово **v**, ближайшее к принятому с искажениями слову $\mathbf{r} = \mathbf{v} + \mathbf{e}$, где *вектор ошибок* $\mathbf{e} \in \{0, 1\}^n$ создан двоичным симметричным каналом (ДСК) в процессе передачи кодового слова. Модель ДСК показана на Рисунке 5. По предположению переходная вероятность p (или параметр ДСК) меньше 1/2.

Стандартной таблицей (или *стандартной расстановкой*) [Sle] для двоичного линейного (n, k, d_{min}) кода C называется таблица всех возможных принятых из канала векторов **r**, организованная таким образом, что может быть найдено ближайшее к **r** кодовое слово **v**.

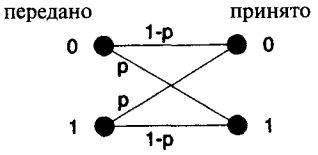


Рис. 5. Модель двоичного симметричного канала.

Таблица 1. Стандартная таблица двоичного линейного блокового кода.

s	$\mathbf{u}_0 = 0$	$\mathbf{u}_1$	...	$\mathbf{u}_{2^k-1}$
0	$\mathbf{v}_0$	$\mathbf{v}_1$	...	$\mathbf{v}_{2^k-1}$
$\mathbf{s}_1$	$\mathbf{e}_1$	$\mathbf{e}_1 + \mathbf{v}_1$	...	$\mathbf{e}_1 + \mathbf{v}_{2^k-1}$
$\mathbf{s}_2$	$\mathbf{e}_2$	$\mathbf{e}_2 + \mathbf{v}_1$	...	$\mathbf{e}_2 + \mathbf{v}_{2^k-1}$
⋮	⋮	⋮	⋱	⋮
$\mathbf{s}_{2^{n-k}-1}$	$\mathbf{e}_{2^{n-k}-1}$	$\mathbf{e}_{2^{n-k}-1} + \mathbf{u}_{2^k-1}$	...	$\mathbf{e}_{2^{n-k}-1} + \mathbf{v}_{2^k-1}$

Стандартная таблица содержит $2^n - k$ строк и $2^k + 1$ столбцов. Правые 2^k столбцов таблицы содержат все вектора из пространства $V_2 = \{0, 1\}^n$.

Для описания процедуры декодирования необходимо ввести понятие *синдрома*. Определение синдрома произвольного слова из V_2 следует из уравнения (1.12),

$$\mathbf{s} = \mathbf{r}\mathbf{H}^{\mathrm{T}} \qquad (1.20)$$

где **H** проверочная матрица кода C. Покажем, что синдром является индикатором вектора ошибок. Предположим, что кодовое слово $\mathbf{v} \in C$, переданное по ДСК, принято как $\mathbf{r} = \mathbf{v} + \mathbf{e}$. Синдром принятого слова равен

$$\mathbf{s} = \mathbf{r}\mathbf{H}^{\mathrm{T}} = (\mathbf{v}+\mathbf{e})\mathbf{H}^{\mathrm{T}} = \mathbf{e}\mathbf{H}^{\mathrm{T}} \qquad (1.21)$$

Таким образом, вычисление синдрома можно рассматривать как *линейное преобразование* вектора ошибок.

Процедура построения стандартной таблицы

1. Правые столбцы первой строки заполним кодовыми словами кода C, начиная с нулевого кодового слова. В первую ячейку первого столбца запишем нулевой синдром. Положим $j = 0$.
2. Для $j = j + 1$, найдем слово $\mathbf{e}_j \in V_2$ минимального веса Хемминга, не являющееся кодовым и не включенное в предыдущие строки таблицы. Соответствующий синдром

$\mathbf{s}_j = \mathbf{e}_j\mathbf{H}^T$ запишем в первый (крайний левый) столбец j-ой строки. В оставшиеся 2^k ячеек этой строки запишем суммы $\mathbf{e}_j$ и содержимого первой строки (т.е. кодового слова).

3. Повторяем шаг 2 процедуры, пока все вектора из V_2 не окажутся включенными в таблицу. Эквивалентно, повторяем шаг 2 пока $j < 2^n - k$, иначе Стоп.

Пример 5. Стандартная таблица двоичного линейного (4,2,2) кода:

s	00	01	10	11
00	0000	0110	1011	1101
11	1000	1110	0011	0101
10	0100	0010	1111	1001
01	0001	0111	1010	1100

Декодирование с помощью стандартной таблицы выполняется следующим образом. Пусть $\mathbf{r} = \mathbf{v} + \mathbf{e}$ принятое слово. Найдем это слово в таблице и возьмем в качестве результата декодирования сообщение $\mathbf{u}$, записанное в верхней (первой) ячейке того столбца, в котором лежит принятое слово $\mathbf{r}$. По идее, этот процесс требует хранения в памяти всей таблицы и поиска в ней заданного слова.

Однако, возможно некоторое упрощение процедуры декодирования, если заметить, что все элементы одной и той же строки имеют один и тот же синдром. Каждая строка $\mathbf{Row}_i$, $0 \le i < 2^n - k$, этой таблицы представляет *смежный класс* кода C, а именно, $\mathbf{Row}_i = \{\mathbf{e}_i + \mathbf{v} | \mathbf{v} \in C\}$. Вектор $\mathbf{e}_i$ называется *лидером смежного* класса.

Синдром всех элементов i-ой строки равен

$$\mathbf{s}_i = (\mathbf{e}_i + \mathbf{v})\mathbf{H}^T = \mathbf{e}_i\mathbf{H}^T \qquad (1.22)$$

и *не зависит* от конкретного значения кодового слова $\mathbf{v} \in C$. Упрощенная процедура декодирования состоит в следующем: вычислить синдром принятого слова $\mathbf{r} = \mathbf{e}_j + \mathbf{v}$

$$\mathbf{s}_j = (\mathbf{e}_j + \mathbf{v})\mathbf{H}^T = \mathbf{e}_j\mathbf{H}^T$$

и найти его в левом столбце стандартной таблице;

взять лидер смежного класса $\mathbf{e}'_j$ из второго столбца той же строки и прибавить его к принятому слову, получив ближайшее к принятому $\mathbf{r} = \mathbf{e}'_j + \mathbf{v}'$ кодовое слово $\mathbf{v}'$.

Таким образом, вместо таблицы $n \times 2^n$ бит для декодирования достаточно использовать таблицу лидеров смежных классов $n \times 2^{n-k}$ бит.

Пример 6. Снова рассмотрим двоичный линейный (4, 2, 2) код из Примера 3. Положим, что передано было кодовое слово (0110) , а принято слово (0010). Тогда синдром равен

$$\mathbf{s} = \mathbf{r}\mathbf{H}^T = (0010)\begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} = (10)$$

Находим в стандартной таблице лидер смежного класса (0100) и получаем декодированное кодовое слово (0010)+(0100)=(0110). Одиночная ошибка в слове исправлена! Это может показаться странным, так как минимальное кодовое расстояние равно 2 и, согласно условию (1.8), исправление однократных ошибок невозможно. Однако объяснение этому может быть найдено в стандартной таблице (Пример 5). Заметим, что третья строка таблицы содержит два различных двоичных вектора веса 1. Это означает, что только три из возможных четырех одиночных ошибок могут быть исправлены. В Примере 6 дана одна из исправляемых ошибок.

Оказывается, что данный (4, 2, 2) код является простейшим примером линейного кода с *неравной защитой от ошибок* (*linear unequal error protection* – LUEP код) [Wv, Van]. Данному LUEP коду соответствует *разделяющий вектор* (3,2), который показывает, что минимальное кодовое расстояние равно трем, если различаются первые биты сообщений и равно двум, если различаются вторые биты сообщений.

В случае *систематического* кодирования рассмотренная выше процедура находит оценку переданного сообщения на первых k позициях декодированного слова. Это может быть

одной из возможных причин применения систематического кодирования.

1.3.3. Хемминговы сферы, области декодирования и стандартная таблица

Стандартная таблица предоставляет удобный способ объяснения понятий Хемминговой сферы и корректирующей способности линейного кода C, введенной в Разделе 1.1.2.

Из конструкции стандартной таблицы видно, что j-ый столбец из 2^k правых столбцов таблицы, обозначаемый $\mathbf{Col}_j$, $1 \le 2^k$, содержит кодовое слово $\mathbf{v}_j \in C$ и множество 2^{n-k} слов, ближайших к нему по Хеммингову расстоянию, т.е.

$$\mathbf{Col}_j = \left\{ \mathbf{v}_j + \mathbf{e}_i \mid \mathbf{e}_i \in \mathbf{Row}_i,\ 0 \le i < 2^{n-k} \right\} \qquad (1.23)$$

Каждый столбец (1.23) представляет собой *область декодирования* j-ого кодового слова в Хемминговом пространстве. Таким образом, если по ДСК передано кодовое слово $\mathbf{v}_j \in C$ и принятое слово $\mathbf{r}$ принадлежит столбцу $\mathbf{Col}_j$, то оно будет успешно декодировано в переданное слово $\mathbf{v}_j$.

Граница Хемминга

Множество столбцов $\mathbf{Col}_j$ и корректирующая способность t кода C связаны между собой через Хеммингову сферу $S_t(\mathbf{v}_j)$ следующим образом: двоичный линейный (n, k, d) код C имеет корректирующую способность t, если каждая область декодирования $\mathbf{Col}_j$ содержит Хеммингову сферу радиуса t, т.е. $S_t(\mathbf{v}_j) \subseteq \mathbf{Col}_j$.

Учитывая, что каждая область декодирования содержит 2^{n-k} слов, и, используя уравнение (1.6), получаем знаменитую *границу Хемминга*

$$\sum_{i=0}^{t} \binom{n}{i} \le 2^{n-k}. \qquad (1.24)$$

Граница Хемминга имеет несколько комбинаторных интерпретаций. Вот одна из них:

Число синдромов, 2^{n-k}, должно быть больше или равно числу исправляемых комбинаций ошибок, $\sum_{i=0}^{t} \binom{n}{i}$.

Пример 7. Двоичный код (3,1,3) имеет порождающую матрицу $\mathbf{G} = (111)$ и проверочную матрицу

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

Соответственно, стандартная таблица имеет вид:

s	0	1
00	000	111
11	100	011
10	010	101
01	001	110

Четыре вектора во втором столбце таблицы (т.е. лидеры смежных классов) являются элементами Хемминговой сферы $S_1(000)$, показанной на Рисунке 4. Этот столбец содержит все векторы длины 3 и веса 1 или меньше. Аналогично, третий столбец (правый) содержит все элементы $S_1(111)$. Для этого кода граница Хемминга выполняется с равенством.

Блоковые коды, удовлетворяющие границе (1.24) с равенством, называются *совершенными кодами*. Нетривиальными совершенными кодами являются следующие:

- двоичные $(2^m - 1, 2^m - m - 1, 3)$ коды Хемминга,
- недвоичные $((q^m - 1)/(q - 1), (q^m - 1)/(q - 1) - m - 1, 3)$ коды Хемминга, $q > 2$,
- коды-повторения $(n,1,n)$,
- коды с проверкой на четность $(n,n\text{-}1,2)$,
- двоичный (23,12,7) код Голея и
- троичный (11,6,5) код Голея.

Расширенные, т.е. дополненные общей проверкой на четность, коды Хемминга и Голея тоже совершенны.

Для недвоичных кодов граница Хемминга имеет вид:

$$\sum_{i=0}^{t}\binom{n}{i}(q-1)^i \le q^{n-k} \qquad (1.25)$$

1.4. Распределение весов и вероятность ошибки

При выборе конкретной схемы кодирования очень важно иметь представление об ее помехоустойчивости. Известны несколько характеристик помехоустойчивости систем с исправлением ошибок. В этом разделе вводятся оценки для линейных кодов и трех базовых моделей каналов: модель ДСК, модель с аддитивным белым гауссовым шумом (АБГШ) и модель канала с общими Релеевскими замираниями.

1.4.1. Распределение весов и вероятность необнаруженной ошибки в ДСК.

Распределение весов $W(C) = \{A_i,\ 0 \le i \le n\}$ кода C, исправляющего ошибки, определено как совокупность $n + 1$ целых A_i, где A_i – количество кодовых слов Хеммингова веса i.

В следующем ниже разделе выводится оценка вероятности необнаруженной ошибки линейного кода в ДСК. Заметим, прежде всего, что вес $wt(\mathbf{v})$ слова $\mathbf{v}$ равен Хеммингову расстоянию до нулевого слова, т.е. $wt(\mathbf{v}) = d_H(\mathbf{v},\mathbf{0})$. Напомним также, что Хеммингово расстояние между кодовыми словами $\mathbf{v_1}$ и $\mathbf{v_2}$ равно весу их разности,

$$d_H(\mathbf{v}_1,\mathbf{v}_2) = d_H(\mathbf{v}_1+\mathbf{v}_2,\mathbf{0}) = wt(\mathbf{v}_1+\mathbf{v}_2) = wt(\mathbf{v}_3)$$

где из линейности кода следует, что $\mathbf{v}_3 \in C$.

Вероятность необнаруженной ошибки $P_u(C)$ равна вероятности того, что принятое из канала связи слово отличается от переданного, но имеет нулевой синдром, т.е.

$$\mathbf{s} = (\mathbf{v}+\mathbf{e})\mathbf{H}^{\mathrm{T}} = \mathbf{e}\mathbf{H}^{\mathrm{T}} = \mathbf{0} \Leftrightarrow \mathbf{e} \in C$$

Таким образом, вероятность того, что синдром принятого ненулевого слова равен нулю, есть вероятность того, что вектор ошибок совпадает с одним из ненулевых кодовых слов.

В ДСК вектор ошибок веса i возникает с вероятностью, равной вероятности того, что i символов приняты с ошибкой, а остальные n-i приняты правильно. Обозначим вероятность этого события через $P(\mathbf{e}, i)$. Тогда

$$P(\mathbf{e},i) = p^i(1-p)^{n-i}$$

Для того чтобы возникла необнаруженная ошибка, вектор ошибок должен быть ненулевым кодовым словом. Имеется A_i кодовых слов веса i в кодовом множестве C. Следовательно,

$$P_u(C) = \sum_{i=d_{\min}}^{n} A_i P(\mathbf{e},i) = \sum_{i=d_{\min}}^{n} A_i p^i(1-p)^{n-i} \qquad (1.26)$$

Формула (1.26) дает точное значение $P_u(C)$ в ДСК. К сожалению, для большинства кодов, имеющих практическое значение, распределение весов неизвестно. В таких случаях, можно использовать тот факт, что число кодовых слов веса i меньше (или равно) общего числа слов веса i в двоичном векторном пространстве V_2. Следовательно, справедлива следующая верхняя граница:

$$P_u(C) \le \sum_{i=d_{\min}}^{n}\binom{n}{i}p^i(1-p)^{n-i} \qquad (1.27)$$

Дополнение переводчика. *На самом деле допустима и более сильная оценка:*

$$P_u^*(C) \approx 2^{-d_{\min}+1}P_u(C)$$

В уравнении $\mathbf{e}\mathbf{H}^{\mathrm{T}} = \mathbf{0}$ *матрица* $\mathbf{H}$ *имеет ранг* $\rho = d_{min} - 1$ *и, по меньшей мере,* ρ *неизвестных элементов любого вектора ошибок* $\mathbf{e}$ *определяются однозначно. Следовательно, средняя вероятность того, что произвольный вектор ошибок совпадает с некоторым кодовым словом, не превосходит* $2^{-\rho}$.

Формулы (1.26) и (1.27) полезны в системах, использующих помехоустойчивое кодирование только для обнаружения ошибок таких, как системы связи с обратной связью и автома-

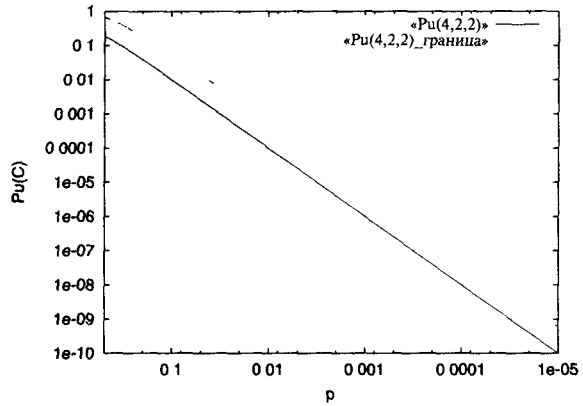


Рис. 6. Точное значение и верхняя граница вероятности необнаруженной ошибки для двоичного линейного (4,2,2) кода в ДСК.

тическим запросом (ARQ) на повторную передачу сообщения с обнаруженными ошибками. Оценки помехоустойчивости для случая, когда кодирование используется для исправления ошибок, выводятся в следующем ниже разделе.

Пример 8. Для двоичного линейного (4,2,2) кода из Примера 4 $W(C)$=(1,0,1,2,0). С помощью (1.26) находим

$$P_u(C) = p^2(1-p)^2 + 2p^3(1-p)$$

На Рисунке 6 показана зависимость $P_u(C)$ вместе с правой частью границы (1.27).

1.4.2. Границы вероятности ошибки в ДСК, каналах с АБГШ и с замираниями

Целью этого раздела является введение в базовые модели каналов связи, которые будут рассматриваться в книге, и вывод формул для оценки помехоустойчивости линейных кодов. Первым рассматривается ДСК.

Модель ДСК

Для двоичного линейного кода процедура декодирования с помощью стандартной таблицы состоит в выборе кодового слова,

ближайшего к принятому слову. Ошибка декодирования возникает всякий раз, когда принятое слово оказывается вне правильной области декодирования.

Обозначим L_i число лидеров смежных классов веса i в стандартной таблице линейного кода C. *Вероятность правильного декодирования* равна вероятности того, что вектор ошибок совпадает с одним из лидеров смежных классов,

$$P_c(C) = \sum_{i=0}^{\ell} L_i p^i (1-p)^{n-i} \tag{1.28}$$

где ℓ это максимальный вес лидера смежного класса **e**. Для совершенных кодов $\ell = t$,

$$L_i = \binom{n}{i}, 0 \le i \le t$$

и из границы Хемминга (1.24) следует, что

$$\sum_{i=0}^{\ell} L_i = \sum_{i=0}^{\ell} \binom{n}{i} = 2^{n-k}$$

В общем случае для двоичных кодов выражение (1.28) дает нижнюю границу $P_c(C)$, так как существует хотя бы один лидер смежного класса веса более t.

Вероятность неправильного декодирования $P_e(C)$ или *вероятность ошибки декодирования* равна вероятности того, что вектор ошибок принадлежит дополнению множества исправляемых ошибок, т.е. $P_e(C) = 1 - P_c(C)$. Из (1.28) получаем,

$$P_e(C) = 1 - \sum_{i=0}^{\ell} L_i p^i (1-p)^{n-i} \tag{1.29}$$

Наконец, учитывая обсуждение оценки (1.28), получаем верхнюю границу

$$P_e(C) \le 1 - \sum_{i=0}^{t} \binom{n}{i} p^i (1-p)^{n-i} \tag{1.30}$$

которую можно записать и в следующем виде

$$P_e(C) \le \sum_{i=t+1}^{n} \binom{n}{i} p^i (1-p)^{n-i} \tag{1.31}$$

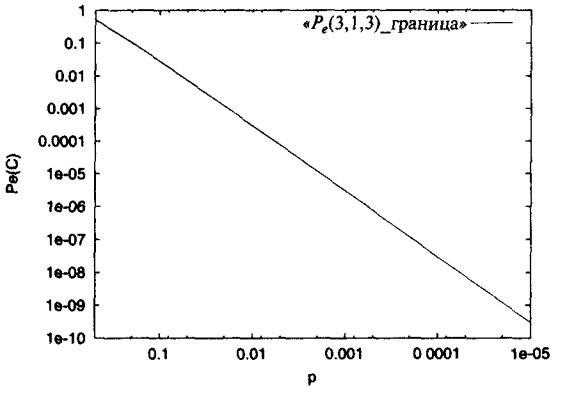


Рис. 7. Вероятность ошибки декодирования для двоичного (3,1,3) кода.

Эти границы удовлетворяются с равенством только для совершенных кодов (когда и граница Хемминга удовлетворяется с равенством).

Пример 9. На Рисунке 7 показана зависимость $P_e(C)$ по оценке (1.31) от переходной вероятности ДСК p для двоичного (совершенного) кода-повторения (3,1,3).

Модель канала с АБГШ

Пожалуй, наиболее важной моделью для систем цифровой связи является модель канала с аддитивным белым гауссовым шумом (АБГШ – additive white Gaussian noise (AWGN)). В этом разделе выводятся оценки вероятности ошибки декодирования и вероятности ошибки на бит для линейных кодов в канале с АБГШ. Хотя аналогичные выражения оказываются справедливыми и для сверточных кодов, они будут выведены в последующих разделах, вместе с обсуждением декодирования с «мягким решением» по алгоритму Витерби. Следующие ниже результаты содержат необходимые инструменты для оценки помехоустойчивости двоичных систем кодирования в *гауссовом канале*.

Рассмотрим двоичную систему передачи сигналов, в которой кодовые символы {0,1} отображаются в действительные

числа {+1,–1}, соответственно, как показано на Рисунке 8. В дальнейшем, вектора имеют размерность n и обозначение $\mathbf{x} = (x_0, x_1, \ldots, x_{n-1})$. Условная функция плотности вероятности (ф.п.в.) последовательности $\mathbf{y}$ на выходе канала при условии, что на его входе передавалась последовательность $\mathbf{x}$, равна

$$p(\mathbf{y}\,|\,\mathbf{x}) = p_{\mathbf{n}}(\mathbf{y}-\mathbf{x}) = \prod_{i=0}^{n-1} \frac{1}{\sqrt{\pi N_0}} \exp^{\frac{(y_i - x_i)^2}{N_0}} \qquad (1.32)$$

где $p_{\mathbf{n}}(\mathbf{n})$ есть ф.п.в. n статистически независимых и одинаково распределенных (i.i.d.) отсчетов шума, каждый из которых имеет Гауссово распределение с нулевым средним и дисперсией, раной $N_0/2$. Величина N_0 называется односторонней спектральной плотностью мощности шума. Легко показать, что *декодирование по максимуму правдоподобия* (м.п.) линейного кода в таком канале соответствует выбору последовательности $\mathbf{x}'$, минимизирующей *квадрат Евклидова расстояния* между принятой последовательностью $\mathbf{y}$ и $\mathbf{x}'$, т.е.

$$D^2(\mathbf{x},\mathbf{y}) \triangleq \sum_{i=0}^{n-1} (x_i - y_i)^2 \qquad (1.33)$$

см. [WJ], [Wil], [BM]. Следует заметить, что декодер, использующий (1.33) как *метрику*, называется *декодером с мягким решением* не зависимо от того, используется или нет принцип максимума правдоподобия. Методы декодирования с мягким решением рассматриваются в Главе 7.

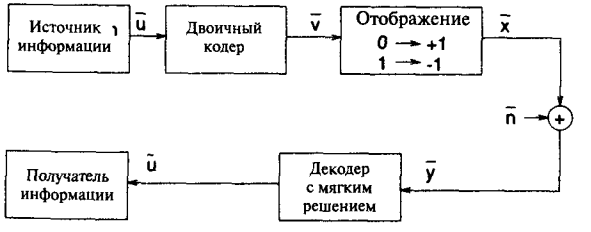


Рис. 8. Система двоичной передачи с кодированием по каналу с АБГШ.

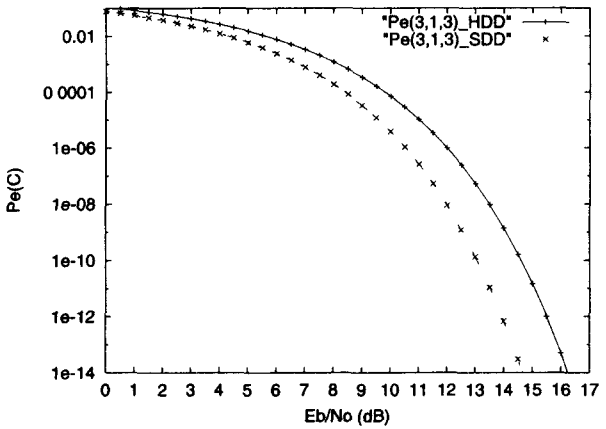


Рис. 9. Вероятность ошибки декодирования для жесткого декодирования (Pt(3,1,3)_HDD) и мягкого декодирования (Pe(3,1,3)_SDD) двоичного (3,1,3) кода при передаче двоичных сигналов в канале с АБГШ.

Вероятность ошибки для МП декодирования, $P_e(C)$, равна вероятности того, что при передаче последовательности x вектор шума оказался таким, что принятая последовательность $\mathbf{y} = \mathbf{x} + \mathbf{n}$ ближе к другой кодовой последовательности $\mathbf{x}'' \in C$, $\mathbf{x}'' \neq \mathbf{x}$. Для линейного кода можно предположить, что передается нулевое кодовое слово. Тогда вероятность $P_e(C)$ может быть ограничена сверху с помощью *границы объединения* [Cla] и распределения весов $W(C)$ следующим образом:

$$P_e(C) \leq \sum_{w=d_{\min}}^{n} A_w Q\left(\sqrt{2wR\frac{E_b}{N_0}}\right) \tag{1.34}$$

где $R = k/n$ скорость кода, E_b/N_0 *отношение энергии сигнала на бит к мощности шума* или (SNR на бит) и $Q(x)$ определено в (1.2).

На Рисунке 9 показаны оценки вероятности ошибки для жесткого декодирования (1.30) и для мягкого декодирования (1.34) для двоичного (3,1,3) кода. *Декодирование с жестким решением* (или *жесткое декодирование*) означает, что декодер

для ДСК использует выход двоичного демодулятора. Эквивалентный ДСК имеет переходную вероятность равную [Pro, WJ]

$$p = Q\left(\sqrt{2R\frac{E_b}{N_0}}\right)$$

Заметим, что в данном частном случае, обе оценки вероятности ошибки являются точными, т.е. не оценками сверху, так как используется совершенный код, содержащий только два кодовых слова. Рисунок 9 показывает также, что мягкое декодирование обеспечивает большую эффективность, чем жесткое декодирование, в том смысле, что одинаковое значение $P_e(C)$ при меньшей мощности передачи сигналов. Разность (в дБ) между соответствующими отношениями *SNR* на бит обычно называют *выигрышем от кодирования*.

В [FLR] показано, что для *вероятности ошибки на бит*, обозначаемой $P_b(C)$ двоичного *систематического* кода при передаче двоичных сигналов по каналу с АБГШ, справедлива следующая верхняя граница:

$$P_b(C) \leq \sum_{w=d_{\min}}^{n} \frac{wA_w}{n} Q\left(\sqrt{2wR\frac{E_b}{N_0}}\right) \tag{1.35}$$

Эта граница справедлива только для систематического кода. Кроме того, результаты в [FLR] показывают, что *систематическое кодирование минимизирует вероятность ошибки на бит*. Это означает, что систематическое кодирование не только желательно, но и оптимально в рассматриваемом выше смысле.

Пример 10. Рассмотрим двоичный линейный (6,3,3) код с порождающей и проверочной матрицами

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}, \qquad \mathbf{H} = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

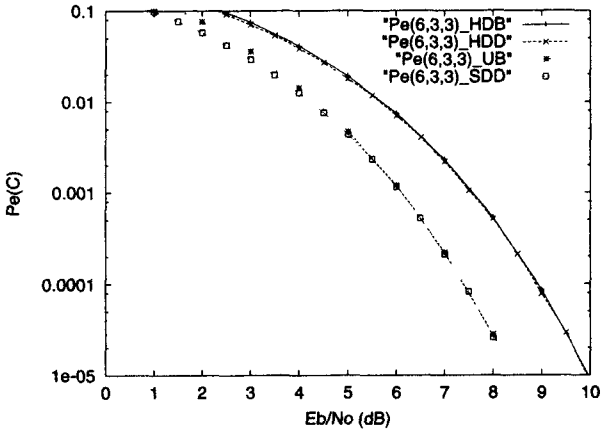


Рис. 10. Моделирование и границы объединения для двоичного (6,3,3) кода при передаче двоичных сигналов в канале с АБГШ.

соответственно. Распределение весов этого кода равно $W(C) = \{1,0,0,4,3,0,0\}$, которое может быть проверено непосредственным вычислением для всех кодовых слов $\mathbf{v} = (\mathbf{u}, \mathbf{v}_p)$:

u	v
000	000
001	101
010	011
011	110
100	110
101	011
110	101
111	000

В этом частном случае, МП декодирование состоит в вычислении квадрата Евклидова расстояния по формуле (1.33) между принятым словом и каждым из восьми возможных кодовых слов. В качестве решения выбирается слово с наименьшим расстоянием. На Рисунке 10 показаны результаты моделирования и границы объединения для жесткого и мягкого декодирования по максимуму правдоподобия для передачи двоичных сигналов в канале с АБГШ.

Модель канала с общими Релеевскими замираниями.
Другой важной моделью канала является модель с общими Релеевскими замираниями. Замирания сигналов происходят в системах беспроводной связи в виде меняющихся во времени искажений передаваемых сигналов. В этой книге рассматриваются только *общие замирания* (*flat fading*). Термин «общие» (или «гладкие») означает, что канал не является частотно-селективным, т.е. передаточная функция канала в полосе пропускания равна константе [BM, WJ, Pro].

Модель канала с покомпонентными мультипликативными искажениями показана на Рисунке 11. Мультипликативные искажения представлены случайным вектором α размерности n, компоненты которого являются независимыми, одинаково распределенными случайными величинами (i.i.d.), α_i, $0 \le i < n$, имеющими плотность вероятности Релея,

$$p_{\alpha_i}(\alpha_i) = \alpha_i e^{-\alpha_i^2/2}, \alpha_i \ge 0 \qquad (1.36)$$

При такой плотности вероятностей множителей среднее значение отношения сигнал-шум (SNR) на бит равно E_b/N_0 (как и для АБГШ без замираний), так как второй момент коэффициентов замирания равен $E\{\alpha_i^2\} = 1$.

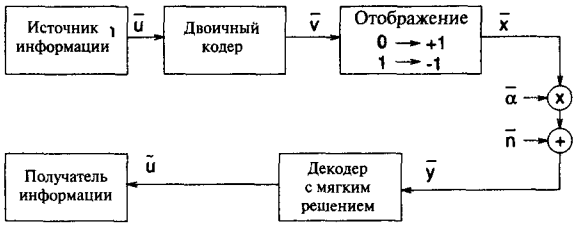


Рис. 11. Передача двоичных кодированных сообщений в канале с гладкими Релеевскими замираниями

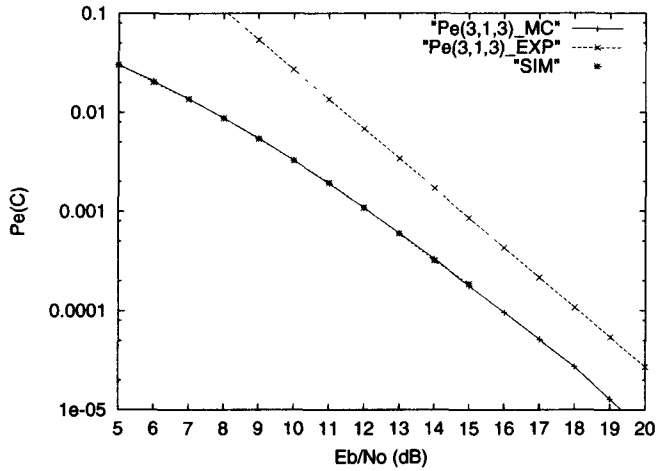


Рис. 12. Двоичная передача по Релеевскому каналу. Результаты моделирования (SIM), оценка границы объединения методом Монте-Карло ($P_e(3,1,3)\_MC$) и граница Чернова ($P_e(3,1,3)\_EXP$) для двоичного (3.1,3) кода.

Оценки эффективности двоичных линейных кодов в канале с общими Релеевскими замираниями находятся из оценок условной вероятности ошибки декодирования, $P_e(C|\boldsymbol{\alpha})$, или условной вероятности ошибки на бит, $P_b(C|\boldsymbol{\alpha})$. Безусловные вероятности ошибки находятся интегрированием условных вероятностей с весами α_i, имеющими плотность вероятности (1.36).

Условные вероятности ошибки идентичны безусловным в канале с АБГШ. Существенное различие имеется только в аргументах функции $Q(x)$, которые теперь взвешены на коэффициенты замирания α_i. Рассматривая передачу двоичных кодированных сообщений при отсутствии (внешней) информации о *состоянии канала*, находим, что

$$P_e(C\,|\,\boldsymbol{\alpha}) \le \sum_{w=d_{\min}}^{n} A_w Q\left(\sqrt{\frac{1}{w}\Delta^2{}_w 2R\frac{E_b}{N_0}}\right) \qquad (1.37)$$

где

$$\Delta_w = \sum_{i=1}^{w} \alpha_i \qquad (1.38)$$

Окончательно, вероятность ошибки декодирования при передаче двоичных сигналов в канале с Релеевскими замираниями получается как математическое ожидание относительно случайной величины Δ_w,

$$P_e(C) \le \sum_{w=d_{\min}}^{n} A_w \int_0^{\infty} Q\left(\sqrt{\frac{1}{w}\Delta^2{}_w 2R\frac{E_b}{N_0}}\right) p_{\Delta_w}(\Delta_w)\, d\Delta_w \qquad (1.39)$$

Известны несколько методов оценивания выражения (1.39). Один из них состоит в применении метода *Монте-Карло численного интегрирования* с использованием следующей аппроксимации:

$$P_e(C) \le \frac{1}{N}\sum_{l=1}^{N}\sum_{w=d_{\min}}^{n} A_w Q\left(\sqrt{2\Delta_w(\ell)R\frac{E_b}{N_0}}\right) \qquad (1.40)$$

где $\Delta_w(\ell)$ равно сумме квадратов w независимых одинаково распределенных (i.i.d.) случайных величин с Релеевской плотностью вероятностей (1.38), выданных на ℓ-ом обращении к компьютерной программе генерации, и N достаточно большое целое число, зависящее от ожидаемого диапазона значений $P_e(C)$. Хорошим правилом, которое следует запомнить, является следующее: величина N должна быть, по меньшей мере, в 100 раз больше обратной величины $P_e(C)$. (См. [Jer], стр. 500–503)

Другим методом является экспоненциальная аппроксимация сверху функции Q (см. [WJ], стр. 82-84), которая позволяет проинтегрировать выражение или воспользоваться *границей Чернова*. Этот подход хоть и несколько ухудшает результат, зато дает замкнутое выражение ([Wil], стр. 526, и [BM], стр. 718):

$$P_e(C) \le \sum_{w=d_{\min}}^{n} A_w \frac{1}{\left(1+\frac{RE_b}{N_0}\right)^w} \qquad (1.41)$$

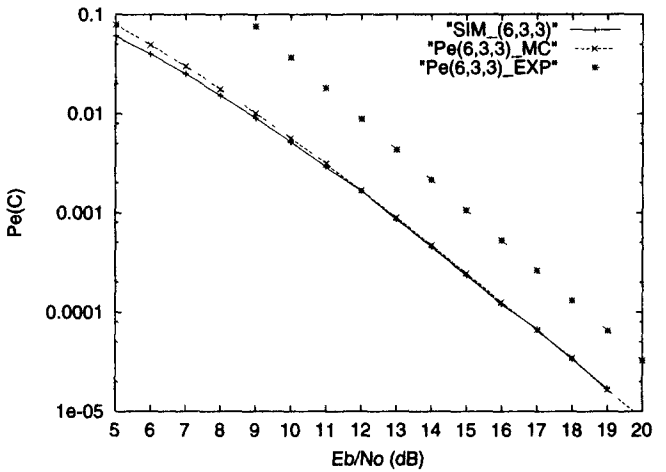


Рис. 13. Двоичная передача по Релеевскому каналу. Результаты моделирования (SIM_(6,3,3)),оценка границы объединения методом Монте-Карло (P_e(6,3,3)_MC) и граница Чернова (P_e(6,3,3)_EXP) для двоичного (6,3,3) кода.

Граница (1.41) полезна в случаях, когда достаточно знать первое приближение в оценке помехоустойчивости кода.

Пример 11. На Рисунке 12 показаны результаты компьютерного моделирования двоичного (3,1,3) кода в канале с общими Релеевскими замираниями. Заметим, что интегрирование методом Монте-Карло дает точное значение помехоустойчивости кода, так как граница (1.40) содержит только один член. Заметим также, что граница Чернова дает результат, смещенный почти на 2дБ, относительно результата моделирования при отношении сигнал-шум на бит $E_b/N_0 > 18$ дБ.

Пример 12. На Рисунке 13 показаны результаты компьютерного моделирования двоичного (6,3,3) кода из примера 10 в Релеевском канале. В этом случае граница объединения теряет точность при малых значениях E_b/N_0 из-за присутствия дополнительных членов в формуле (1.40). Как и раньше, граница

Рис. 14. Общая структура жесткого декодера для линейных блоковых кодов для ДСК.

Чернова проигрывает около 2 дБ в отношении сигнал – шум на бит при $E_b/N_0>18$ дБ.

1.5 Общая структура жесткого декодера для линейных кодов

В этом разделе проводится итоговое обсуждение структуры декодера с жестким решением. На Рисунке 14 показана упрощенная блок-схема процесса декодирования. Так как обсуждается жесткое решение, то решения демодулятора поступают на вход декодера, рассчитанного на работу с ДСК.

Обозначим $\mathbf{v} \in C$ переданное кодовое слово. На вход декодера подается принятое искаженное слово $\mathbf{r} = \mathbf{v} + \mathbf{e}$. Процедура декодирования состоит из следующих шагов:

- Вычисляется синдром $\mathbf{s} = \mathbf{r}\,\mathbf{H}^T$. Согласно свойству линейного кода синдром является линейным преобразованием вектора ошибок, возникшего в канале,

$$\mathbf{s} = \mathbf{e}\mathbf{H}^T \tag{1.42}$$

- Для вычисленного синдрома $\mathbf{s}$ найти наиболее вероятный вектор ошибок $\mathbf{e}$ и вычесть его (по модулю два в двоичном случае) из принятого вектора.

Несмотря на то, что большинство практических декодеров не реализуют процедуру декодирования так, как она сформулирована выше, имеет смысл рассматривать процедуру жест-

кого декодирования как метод решения уравнения (1.42). Заметим, что любой метод решения этого уравнения является методом декодирования. Например, можно попытаться решать это (ключевое) уравнение с помощью псевдо-обратной матрицы $(\mathbf{H}^{\mathrm{T}})^{+}$ матрицы $\mathbf{H}^{\mathrm{T}}$ такой, что $\mathbf{H}^{\mathrm{T}}(\mathbf{H}^{\mathrm{T}})^{+} = \mathrm{I}_n$ и для которой результат декодирования

$$\mathbf{e} = \mathbf{s}(\mathbf{H}^{\mathrm{T}})^{+} \tag{1.42}$$

имеет *наименьший вес Хемминга*. Как легко бы это не казалось, задача эта очень сложна. Мы вернемся к этому соображению при обсуждении методов декодирования кодов БЧХ и Рида-Соломона.

Глава 2

КОДЫ ХЕММИНГА, ГОЛЕЯ И РИДА-МАЛЛЕРА

В этой главе изучаются важные примеры линейных блоковых кодов. Эти примеры помогут глубже понять принципы исправления ошибок и хорошие алгоритмы декодирования. Коды Хемминга являются, по-видимому, наиболее известным классом блоковых кодов, за исключением, может быть, только кодов Рида-Соломона. Как упоминалось в Главе 1, коды Хемминга являются оптимальными в том смысле, что они требуют минимальную избыточность при заданной длине блока для исправления одной ошибки. Двоичные коды Голея это единственный нетривиальный пример оптимального кода, исправляющего тройные ошибки (другими примерами оптимальных кодов являются коды-повторения и коды с одной проверкой на четность). Коды Рида-Маллера представляют собой очень элегантную комбинаторную конструкцию с простым декодированием.

2.1. Коды Хемминга

Напомним (формула (1.12)), что любое кодовое слово $\mathbf{v}$ линейного $(n,k,d_{\min})$ кода C удовлетворяет уравнению

$$\mathbf{v}\mathbf{H}^{\mathrm{T}} = \mathbf{0} \tag{2.1}$$

Полезная интерпретация этого уравнения состоит в следующем: *максимальное число линейно независимых столбцов проверочной матрицы* $\mathbf{H}$ *кода* C *равно* $d_{\min}-1$.

В двоичном случае, для $d_{\min} = 3$, из (2.1) следует, что сумма любых двух столбцов проверочной матрицы не равна нулевому вектору. Положим, что столбцы $\mathbf{H}$ являются двоичными векторами длины m. Имеется всего $2^m - 1$ ненулевых различных

столбцов. Следовательно, длина двоичного кода, исправляющего одиночную ошибку, удовлетворяет условию

$$n \le 2^m - 1$$

Это неравенство в точности совпадает с границей Хемминга (1.24) для кода длины n, с $n - k = m$ проверками и исправлением $t = 1$ ошибок. Соответственно, код, удовлетворяющий этой границе с равенством, известен как код *Хемминга*.

Пример 13. Для m=3 мы получаем (7,4,3) код Хемминга с проверочной матрицей

$$\mathbf{H} = \begin{pmatrix} 1110100 \\ 0111010 \\ 1101001 \end{pmatrix}$$

В конце этого раздела будут даны алгоритмы кодирования и декодирования в форме, удобной для реализации и моделирования. В последующих разделах книги приводятся эффективные алгоритмы *мягкого декодирования*, которые могут быть использованы в каналах с непрерывным выходом, таких как каналы с АБГШ или Релеевскими замираниями.

Как уже отмечалось, проверочная матрица кода Хемминга обладает тем свойством, что все ее столбцы различны. Если возникает одиночная ошибка на j-ой позиции, $1 \le j \le n$, то синдром искаженного принятого слова равен j-ому столбцу матрицы $\mathbf{H}$. Обозначим $\mathbf{e}$ вектор ошибок, добавленный к кодовому слову в процессе его передачи по ДСК, и предположим, что все его компоненты равны нулю за исключением j-ой позиции, $e_j = 1$. Тогда синдром принятого слова равен

$$\mathbf{s} = \mathbf{r}\mathbf{H}^{\mathrm{T}} = \mathbf{e}\mathbf{H}^{\mathrm{T}} = \mathbf{h}_j \qquad (2.2)$$

где $\mathbf{h}_j$ представляет j-ый столбец $\mathbf{H}$.

2.1.1. Процедуры кодирования и декодирования

Из уравнения (2.2) следует, что, если столбцы проверочной матрицы рассматривать как двоичное представление целых чисел, то значение синдрома равно номеру искаженной (ошибочной) позиции. Эта идея лежит в основе алгоритмов кодирования и декодирования, представленных ниже. Запишем столбцы проверочной матрицы в виде двоичного представления номера (от 1 до n) позиции кодового слова в возрастающем порядке. Обозначим эту матрицу через $\mathbf{H}^*$. Очевидно, что матрице $\mathbf{H}^*$ соответствует эквивалентный код Хемминга с точностью до перестановки позиций кодового слова.

Напомним, что проверочная матрица в систематической форме содержит единичную подматрицу $\mathbf{I}_{n-k}$ размера $(n-k) \times (n-k)$ как показано в (1.16). Очевидно, что в матрице $\mathbf{H}^*$ столбцы единичной подматрицы $\mathbf{I}_{n-k}$ (т.е. столбцы веса один) размещаются на позициях с номерами, равными степени 2, т.е. 2^l, $l = 0,1,\ldots, m$.

Пример 14. Пусть $m = 3$. Тогда систематическая (каноническая) проверочная матрица может быть задана в следующем виде

$$\mathbf{H} = \begin{pmatrix} 1101100 \\ 1011010 \\ 0111001 \end{pmatrix}$$

Матрица $\mathbf{H}^*$ заданная двоичным представлением целых чисел от 1 до 7 (младший разряд записывается в верхней строке) имеет вид:

$$\mathbf{H}^* = \begin{pmatrix} 1010101 \\ 0110011 \\ 0001111 \end{pmatrix}$$

где матрица $\mathbf{I}_3$ содержится в столбцах 1, 2 и 4.

В общем случае, для $(2^m - 1,\ 2^m - 1 - m)$ кода Хемминга и данного (арифметического) порядка столбцов единичная матрица $\mathbf{I}_m$ содержится в столбцах проверочной матрицы с номерами 1, 2, 4, ..., 2^{m-1}.

Кодирование

Процедура кодирования следует из уравнения (1.18). При вычислении проверочных символов p_j для всех $1 \le j \le m$ проверяются номера столбцов и те столбцы, номера которых не являются степенью 2, сопоставляются информационным позициям слова. Соответствующие информационные символы включаются в процесс вычисления проверок. Такая процедура кодирования в чем-то сложнее обычной процедуры для систематического (канонического) кода Хемминга. Однако, соответствующая ей процедура декодирования очень проста. Для некоторых приложений этот подход может оказаться наиболее привлекательным, так как обычно декодирование должно быть очень быстрым.

Декодирование

Если кодирование выполнялось в соответствии с матрицей **H***, то декодирование оказывается очень простым. Синдром (2.2) равен номеру позиции, в которой произошла ошибка! После вычисления синдрома s , который рассматривается как целое число s, ошибка исправляется по правилу

$$v_s = v_s + 1, \qquad (2.2)$$

где выполняется сложение по модулю 2 (т.е. 0+0=0, 1+1=0, 0+1=1, 1+0=1).

Программа hamming.c на домашней странице ECC реализует именно этот вариант кодирования и декодирования двоичных кодов Хемминга.

Дополнение переводчика. *Следует заметить, что в расширенном толковании оба варианта кода Хемминга в Примере 14 являются систематическими, так как в обоих случаях известен фиксированный набор позиций кодового слова, на которых значения символов кодового слова всегда совпадают с соответствующими информационными символами. Другими словами, любые перестановки символов кодового слова оставляют код систематическим (в расширенном смысле).*

Пример 14 интересен еще и тем, что $\mathbf{H^*=PH}$*, т.е. проверочной матрице* $\mathbf{H^*}$*, полученной перестановкой некоторых позиций, соответствует, как оказывается, линейное преобразование исходной матрицы, где преобразующая матрица имеет вид*

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

Таким образом, проверочная матрица заданного кода может иметь много различных представлений. Если $\mathbf{eH}^T = \mathbf{0}$*, то и* $\mathbf{eH}^T\mathbf{P}^T = \mathbf{0}$*, где* $\mathbf{P}$ *любая невырожденная 3x3 матрица. Если* $\mathbf{eH}^T = s$*, то и* $\mathbf{eH}^T\mathbf{P}^T = \mathbf{sP}^T = s^*$*. Следовательно, для заданного линейного кода можно найти такое линейное преобразование его проверочной матрицы, которое окажется наиболее удобным для реализации, без каких либо изменений самого кода. Эта возможность будет использована, например, при декодировании циклических кодов БЧХ.*

Из примера 14 следует еще одно свойство линейных кодов. Пользуясь линейным преобразованием проверочной матрицы можно «назначить» проверочными любые (n-k) позиций кодового слова, если только соответствующие им столбцы матрицы являются линейно независимыми. Это означает, что по оставшимся k позициям можно однозначно восстановить все кодовое слово. Такие наборы позиций называются информационными совокупностями. Множество информационных совокупностей используется в процедурах декодирования под общим названием «ловля ошибок».

2.2. Двоичный код Голея

Голей [Gol] обнаружил, что

$$\sum_{i=0}^{3} \binom{23}{i} = 2^{11}$$

Это равенство позволяет предположить возможность существования совершенного двоичного (23, 12, 7) кода с $t = 3$, т.е. кода, способного исправлять до трех ошибок в словах длиной

23 символа. В своей статье Голей привел порождающую матрицу такого двоичного кода, исправляющего до трех ошибок.

Из-за относительно небольшой длины (23) и размерности (12), а также небольшого числа проверок (11), кодирование и декодирование двоичного (23,12,7) кода Голея может быть выполнено табличным методом. Программа golay23.c на домашней странице ECC использует таблицу объема 16K×23 бит для кодирования и 8K×23 бит для декодирования.

2.2.1 Кодирование

Табличное (LUT, *look-up-table*) кодирование реализуется с помощью просмотра таблицы, содержащей список всех $2^{12} = 4096$ кодовых слов, которые пронумерованы непосредственно информационными символами. Обозначим **u** информационный вектор размерности 12 бит и **v** соответствующее кодовое слово (23 бита). Табличный кодер использует таблицу, в которой для каждого информационного вектора (12 бит) вычислен и записан синдром (11 бит). Синдром берется из таблицы и приписывается справа к информационному вектору.

Операция LUT является взаимно однозначным отображением из множества векторов **u** на множество векторов **v**, которое может быть записано в виде

$$\mathbf{v} = \mathrm{LUT}(\mathbf{u}) = (\mathbf{u}, get\_syndrome(\mathbf{u},\mathbf{0})) \qquad (2.3)$$

В реализации табличного кодера учтены упрощения, которые следуют из циклической природы кода Голея. Его *порождающий полином*[1] имеет вид

$$g(x) = x^{11} + x^{10} + x^6 + x^5 + x^4 + x^2 + 1$$

или в *шестнадцатиричной счисления* системе C75. Этот полином используется в процедуре "get_syndrome", заданной уравнением (2.3).

[1] Определение порождающего полинома дано в Главе 3.

2.2.2. Декодирование

Напомним, что задачей декодера (Глава 1, Рисунок 14) является оценка наиболее вероятного (т.е. обладающего минимальным Хемминговым весом) вектора ошибок **e** по принятому вектору **r**.

Процедура построения табличного LUT-декодера состоит в следующем:

1. Выписать все возможные вектора ошибок **e**, вес Хемминга которых не превышает трех;
2. Для каждого вектора ошибок вычислить соответствующий синдром **s**=get_syndrome(**e**);
3. Записать в таблицу для каждого значения **s** соответствующий ему вектор **e**, так что

$$\mathrm{LUT}(\mathbf{s}) = \mathbf{e}$$

Исправление до трех ошибок в принятом искаженном слове **r** с помощью LUT-декодера может быть представлено следующим образом:

$$\mathbf{v}'' = \mathbf{r} \oplus \mathrm{LUT}(\mathrm{get\_syndrome}(\mathbf{r}))$$

где $\mathbf{v}''$ – исправленное кодовое слово.

2.2.3. Арифметическое декодирование расширенного (24,12,8) кода Голея.

В этом разделе рассматривается процедура декодирования расширенного (24,12,8) кода Голея C_{24}, реализующая арифметический алгоритм декодирования [Wic, VO]. Этот алгоритм использует строки и столбцы подматрицы **B** проверочной матрицы $\mathbf{H}=(\mathbf{B},\mathbf{I}_{12})$. Заметим, что (24,12,8) код Голея, эквивалентный с точностью до перестановки отдельных позиций коду C_{24}, может быть построен добавлением общей проверки на четность к кодовым словам (23,11,7) кода Голея.

Двенадцать строк подматрицы **B**, обозначенные как $\mathbf{row}_i$, $1 \le i \le 12$, имеют следующий вид в шестнадцатеричной системе:

```
0x7ff, 0xee2, 0xdc5, 0xb8b, 0xf16, 0xe2d,
0xc5b, 0x8b7, 0x96e, 0xadc, 0xdb8, 0xb71
```

Важно отметить, что подматрица **B** проверочной матрицы кода C_{24} удовлетворяет равенству $\mathbf{B} = \mathbf{B}^{\mathrm{T}}$, т.е. транспонированная и исходная матрицы совпадают. Это означает, что код C_{24} является *самодуальным* кодом. *Дуальным* кодом называют код, использующий проверочную матрицу исходного кода в качестве порождающей. Свойства самодуальных кодов не рассматриваются в этой книге. Интересующиеся этой темой читатели найдут все необходимое в [MS, Wic].

В программе golay24.c кодирование реализуется рекуррентно, по правилу (1.18). Как и раньше, через $\mathrm{wt}_H(\mathbf{x})$ обозначен вес Хемминга вектора **x**. Процедура декодирования [Wic, VO] состоит в выполнении следующей последовательности шагов:

1. Вычислить синдром $\mathbf{s} = \mathbf{r}\mathbf{H}^{\top}$.
2. Если $\mathrm{wt}_H(\mathbf{s}) \le 3$, то исправляющий вектор равен $\mathbf{e} = (\mathbf{s}, \mathbf{0})$, перейти к шагу 8.
3. Если $\mathrm{wt}_H(\mathbf{s} + \mathbf{row}_i) \le 2$, для $1 \le i \le 12$, то исправляющий вектор равен $\mathbf{e} = (\mathbf{s} + \mathbf{row}_i, \mathbf{x}_i)$, где $\mathbf{x}_i$ вектор длины 12, содержащий 1 в i-ой координате и нули в остальных. Перейти к шагу 8.
4. Вычислить **sB**.
5. Если $\mathrm{wt}_H(\mathbf{sB}) \le 3$, то исправляющий вектор равен $\mathbf{e} = (\mathbf{0}, \mathbf{sB})$, перейти к шагу 8.
6. Если $\mathrm{wt}_H(\mathbf{sB} + \mathbf{row}_i) \le 2$, для некоторого i, $1 \le i \le 12$, то исправляющий вектор равен $\mathbf{e} = (\mathbf{x}_i, \mathbf{sB} + \mathbf{row}_i)$, где $\mathbf{x}_i$ вектор длины 12, содержащий 1 в i-ой координате и нули в остальных. Перейти к шагу 8.
7. Если ни одно из условий шагов 2 – 6 не было удовлетворено, то вектор **r** содержит неисправимое сочетание ошибок и устанавливается флаг «отказ от декодирования». Конец процедуры.
8. Вычислить $\mathbf{c} = \mathbf{r} + \mathbf{e}$. Конец процедуры.

Дополнение переводчика. *Эта процедура имеет очень простое объяснение. Кодовое слово длиной 24 символа состоит из информационной части (12) бит и проверочной части (12) бит. Если все ошибки находятся только в проверочной части, то вес синдрома равен числу ошибок. Если этот вес не превышает трех, то декодер принимает на шаге 2 процедуры декодирования однозначное решение о том, что вектор ошибок совпадает с синдромом.*

На шаге 3 проверяется предположение о том, что одна (именно одна) ошибка находится на i-ой позиции в информационной части принятого слова. Для правильного выбора i прибавление $\mathbf{row}_i$ *к синдрому компенсирует вклад этой ошибки в синдром, после чего вес измененного синдрома будет равен 2 или 1. На основании этого декодер принимает однозначное решение о том, что исправляющий вектор равен конкатенации (соединению) вектора* $\mathbf{x}_i$ *с измененным синдромом. В противном случае, число ошибок возрастет, вес синдрома не удовлетворит условию. Разумеется, может оказаться, что какое-то другое слово расположено на расстоянии не более 2 от измененного слова при условии, что в принятом слове больше трех ошибок.*

На шагах 4 – 6 процедуры выполняются такие же проверки в предположении, что, либо все ошибки, либо все кроме одной, находятся в информационной части принятого слова. Для этого используется следующее линейное преобразование проверочной матрицы:

$$\boldsymbol{H}^* = \boldsymbol{HB} = (\boldsymbol{I}_{12}, \boldsymbol{B}),$$

которому соответствует преобразование синдрома, равное $\boldsymbol{sB}$.

Справедливость этого преобразования следует из равенства $\boldsymbol{BB} = \boldsymbol{I}_{12}$, *которое легко можно проверить. Другими словами, с помощью этого преобразования, с точки зрения декодера, меняются местами информационные и проверочные позиции кодового слова, хотя никакие изменения с самим кодом не происходят.*

Рассмотренная идея использована в некоторых алгоритмах декодирования, близкого к максимуму правдоподобия. В наиболее точном и полном виде теоретическое обоснование ее дано в [Evs]*

2.3. Двоичные коды Рида-Маллера

Двоичные коды Рида-Маллера (РМ) составляют семейство кодов, исправляющих ошибки, с простым декодированием,

основанным на *мажоритарной логике*. Кроме того, известно, что коды этого семейства имеют очень простые и хорошо структурированные решетки [LKFF]. Более подробно решетки линейных блоковых кодов обсуждаются в Главе 7.

Известно элегантное определение двоичных РМ кодов, основанное на двоичных полиномах (или Булевых функциях). Согласно этому определению, РМ коды становятся близкими к кодам БЧХ и РС, которые входят в класс *полиномиальных кодов*[2].

2.3.1. Булевы полиномы и РМ коды

Этот раздел полностью основан на материалах [MS]. Обозначим $f(x_1, x_2, \ldots, x_m)$ *Булеву функцию* от m двоичных переменных $x_1, x_2, \ldots, x_m$. Известно, что такие функции легко представить с помощью *таблицы истинности*. Таблица истинности содержит список значений функции f для всех 2^m комбинаций значений ее аргументов. Все обычные Булевы операции (такие как «и», «или») могут быть представлены как Булевы функции.

Пример 15. Рассмотрим функцию $f(x_1, x_2)$, заданную следующей таблицей истинности:

x_2	0	0	1	1
x_1	0	1	0	1
$f(x_1, x_2)$	0	1	1	0

Тогда

$$f(x_1, x_2) = (x_1 \,\&\, \mathrm{NOT}(x_2)) \cup (\mathrm{NOT}(x_1) \,\&\, x_2)$$

Ассоциируем с каждой Булевой функцией f двоичный вектор f длины 2^m, составленный из значений данной функции для всех возможных комбинаций значений m ее аргументов. В последнем примере f=(0110), где принято соглашение о лексикографическом (иначе, арифметическом) упорядочении значений аргументов функции, таком, что x_1 представляет младший разряд, а x_m – старший разряд.

[2] Полиномиальные коды представлены в разделе 3.4.

Заметим, что Булева функция может быть записана прямо по таблице истинности с помощью *дизъюнктивной нормальной формы* (ДНФ). В терминах ДНФ любая Булева функция может быть записана как сумма[3] 2^m элементарных функций: 1, x_1, $x_2,\ldots,x_m$, $x_1 x_2,\ldots, x_1x_m,\ldots, x_1x_2\ldots x_m$, такой что

$$\mathbf{f} = \mathbf{1} + a_1\mathbf{x}_1 + a_2\mathbf{x}_2 + \ldots + a_m\mathbf{x}_m + a_{12}\mathbf{x}_1\mathbf{x}_2 + \ldots + a_{12\ldots m}\mathbf{x}_1\mathbf{x}_2\ldots\mathbf{x}_m \qquad (2.4)$$

где вектор **1** добавлен для того, чтобы учесть свободный член (степени 0). В примере 15 $\mathbf{f}=\mathbf{x}_1+\mathbf{x}_2$.

Двоичный $(2^m, k, 2^m - r)$ РМ код, обозначенный $\mathrm{PM}_{r,m}$, определяется как множество векторов, ассоциированных со всеми Булевыми функциями степени до r, включительно, от m переменных. Код $\mathrm{PM}_{r,m}$ называют также кодом РМ r-ого порядка длины 2^m. Размерность $\mathrm{PM}_{r,m}$ кода, как легко может быть показано, равна

$$k = \sum_{i=0}^{r} \binom{m}{i}$$

Это число равно числу способов, которыми могут быть построены полиномы степени r или меньше от m переменных.

С учетом уравнения (2.4) строками порождающей матрицы $\mathrm{PM}_{r,m}$ кода являются вектора, ассоциированные с k Булевыми функциями, которые могут быть записаны как полиномы степени r или меньше от m переменных.

Пример 16. Код РМ первого порядка длины 8, $\mathrm{PM}_{1,3}$, является двоичным (8.4,4) кодом, который может быть построен из Булевых функций степени 1 от 3 переменных: $\{1, x_1, x_2, x_3\}$. Таким образом,

$\mathbf{1}=$	1	1	1	1	1	1	1	1
$\mathbf{x}_1=$	0	0	0	0	1	1	1	1
$\mathbf{x}_2=$	0	0	1	1	0	0	1	1
$\mathbf{x}_3=$	0	1	0	1	0	1	0	1

Порождающая матрица $\mathrm{PM}_{1,3}$ кода имеет, следовательно, вид

[3] «сумма» понимается как исключительное «ИЛИ» (XOR), а «произведение» понимается как логическое «И».

$$\mathbf{G} = \begin{pmatrix} \mathbf{1} \\ \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \tag{2.5}$$

Заметим, что код $\mathrm{PM}_{1,3}$ может быть построен также и из кода Хемминга (7,4,3) добавлением общей проверки на четность. Расширенный код Хемминга и $\mathrm{PM}_{1,3}$ код могут отличаться только порядком позиций (столбцов).

Дуальные коды кодов РМ также являются кодами РМ

Можно показать, что $\mathrm{PM}_{m-r-1,m}$ код дуален коду $\mathrm{PM}_{r,m}$. Другими словами, порождающая матрица $\mathrm{PM}_{m-r-1,m}$ кода может быть использована как проверочная матрица $\mathrm{PM}_{r,m}$ кода.

2.3.2. Конечные геометрии и мажоритарное декодирование.

Определение кодов РМ может быть дано и в терминах *конечных геометрии. Евклидова геометрия*, $EG(m,2)$, размерности m над $GF(2)$ содержит 2^m *точек*, которые представляют собой все двоичные вектора длины m. Заметим, что столбцы матрицы, образованной последними тремя строками порождающей матрицы $\mathrm{PM}_{1,3}$ кода, см. Пример 16, представляют собой 8 точек $EG(3,2)$. Удалением нулевой точки это множество точек преобразуется в *проективную геометрию* $PG(m\text{-}1,2)$. Читателю рекомендуется монография [LC] с превосходным изложением конечных геометрий и теории кодов Рида-Маллера. Коды над конечными геометриями являются по существу обобщением кодов РМ[4].

Связь между кодами и конечными геометриями можно объяснить следующим образом. Возьмем $EG(m,2)$. Столбцы матрицы $(\mathbf{x}^{\mathrm{T}}_1\ \mathbf{x}^{\mathrm{T}}_2 \cdots \mathbf{x}^{\mathrm{T}}_m)^{\mathrm{T}}$ (где T – операция транспонирования матрицы) рассматриваются как координаты точек геометрии $EG(m,2)$. Тогда имеет место взаимно однозначное соответствие между компонентами двоичного вектора длины 2^m и точками

[4] См раздел 3 4

$EG(m,2)$. Заданный двоичный вектор длины 2^m ассоциируется с подмножеством точек $EG(m,2)$. В частности, подмножество $EG(m,2)$ может быть ассоциировано с двоичным вектором $\mathbf{w} = (w_1, w_2, \cdots, w_n)$ длины $n = 2^m$, если интерпретировать значение его координат $w_i = 1$ как выбор точки. Другими словами, $\mathbf{w}$ является *вектором инцидентности* (совпадений).

Теперь двоичный код Рида-Маллера можно определить следующим образом: кодовыми словами $\mathrm{PM}_{r,m}$ кода являются вектора инцидентности всех подпространств (т.е. линейных комбинаций точек) размерности $m\text{-}r$ в $EG(m,2)$ (Теорема 8 в [MS]). Из этого определения следует, что число кодовых слов минимального веса $\mathrm{PM}_{r,m}$ кода равно

$$A_{2^m,r} = 2^r \prod_{i=0}^{m-r-1} \frac{2^{m-i}-1}{2^{m-r-i}-1} \tag{2.6}$$

Код, который получается после удаления (перфорации) координат, соответствующих условию $x_1 = x_2 = \ldots = x_m = 0$, из всех кодовых слов $\mathrm{PM}_{r,m}$ кода является двоичным *циклическим* $\mathrm{PM}^*_{r,m}$ кодом. Число слов минимального веса циклического РМ кода равно

$$A^*_{2^m,r,1} = \prod_{i=0}^{m-r-1} \frac{2^{m-i}-1}{2^{m-r-i}-1} \tag{2.7}$$

Декодирование РМ кодов может быть выполнено на основе *мажоритарной логики* (МЛ). Идея мажоритарного декодирования состоит в следующем. Как известно, проверочная матрица порождает 2^{n-k} проверочных уравнений. Построение МЛ декодера сводится к выбору такого подмножества проверочных уравнений, чтобы решение о значении кодового символа на определенной позиции формировалось по большинству «голосов», причем каждый «голос» связан с одним из проверочных уравнений. В качестве иллюстрации рассмотрим код $\mathrm{PM}_{1,3}$ из Примера 16.

Пример 17. Пусть вектор

$$\mathbf{v} = \mathbf{u}\mathbf{G} = (v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8)$$

является кодовым словом кода $\mathrm{PM}_{1,3}$. Как уже упоминалось ранее, выражение (2.5) определяет порождающую, а также и проверочную матрицу $\mathrm{PM}_{1,3}$ кода, поскольку в данном случае $r = 1$ и $m - r - 1 = 1$ и, следовательно, код является *самодуальным.* Все возможные ненулевые линейные комбинации (всего 15) проверочных уравнений (строк проверочной матрицы **H**) имеют вид:

$$\begin{aligned}
v_1 + v_2 + v_3 + v_4 + v_5 + v_6 + v_7 + v_8 &= 0 \\
v_5 + v_6 + v_7 + v_8 &= 0 \\
v_3 + v_4 + v_7 + v_8 &= 0 \\
v_2 + v_4 + v_6 + v_8 &= 0 \\
v_1 + v_2 + v_3 + v_4 &= 0 \\
v_1 + v_2 + v_5 + v_6 &= 0 \\
v_1 + v_3 + v_5 + v_7 &= 0 \\
v_3 + v_4 + v_5 + v_6 &= 0 \\
v_2 + v_4 + v_5 + v_7 &= 0 \\
v_2 + v_3 + v_6 + v_7 &= 0 \\
v_1 + v_2 + v_7 + v_8 &= 0 \\
v_1 + v_3 + v_6 + v_8 &= 0 \\
v_1 + v_4 + v_5 + v_8 &= 0 \\
v_2 + v_3 + v_5 + v_8 &= 0 \\
v_1 + v_4 + v_6 + v_7 &= 0
\end{aligned} \qquad (2.8)$$

Читателю предлагается проверить, что сумма $v_i + v_j$ любой пары кодовых символов v_i и v_j появляется точно в четырех уравнениях. Более того, в уравнениях, которые содержат одинаковую сумму $(v_i + v_j)$, любые другие суммы пар появляются не более одного раза (т.е. не более чем в одном уравнении). Такие проверочные уравнения называют *ортогональными* относительно пары символов v_i и v_j.

Покажем теперь способ исправления одиночной ошибки. Пусть в результате передачи кодового слова **v** по ДСК получен вектор

$$\mathbf{r} = \mathbf{v} + \mathbf{e} = (r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8)$$

Предположим, что ошибка, которая должна быть исправлена, находится в v_5. Построим процедуру мажоритарного декодирования для данного случая.

Выберем два уравнения, включающие сумму $v_i + v_5$, $i \neq 5$, и два других уравнения, содержащие сумму $v_j + v_5$, $j \neq 5$, $j \neq j$. Примем, например, $i = 3$ и $j = 4$. Имеется четыре проверки ортогональные сумме $v_3 + v_5$. Выберем любые две из них. Выполним тоже самое для суммы $v_4 + v_5$. Ассоциированные с выбранными проверками синдромы обозначим S_1 и S_2 для суммы $v_3 + v_5$, а также S_3 и S_4 для суммы $v_4 + v_5$. В результате имеем:

$$\begin{aligned}
S_1 &\triangleq r_1 + r_3 + r_5 + r_7 \\
S_2 &\triangleq r_3 + r_4 + r_5 + r_6 \\
S_3 &\triangleq r_2 + r_4 + r_5 + r_7 \\
S_4 &\triangleq r_1 + r_4 + r_5 + r_8
\end{aligned} \qquad (2.9)$$

Так как вектор **v** является кодовым словом $\mathrm{PM}_{1,3}$ кода, то система уравнений (2.9) эквивалентна следующей

$$\begin{aligned}
S_1 &\triangleq e_1 + e_3 + e_5 + e_7 \\
S_2 &\triangleq e_3 + e_4 + e_5 + e_6 \\
S_3 &\triangleq e_2 + e_4 + e_5 + e_7 \\
S_4 &\triangleq e_1 + e_4 + e_5 + e_8
\end{aligned} \qquad (2.10)$$

Поскольку проверки S_1, S_2, S_3, S_4 ортогональны относительно $e_3 + e_5$ и $e_4 + e_5$, то может быть построена новая пара уравнений ортогональных относительно e_5.

$$\begin{aligned}
S_1' &\triangleq e_3' + e_5' \\
S_2' &\triangleq e_4' + e_5'
\end{aligned} \qquad (2.11)$$

где e_j', $j = 3, 4, 5$, представляют мажоритарные оценки, полученные из уравнений (2.9). Уравнения (2.11) ортогональны относительно e_5' и, следовательно, значение e_5' может быть получено голосованием. Например, в данном случае это может быть результат операции «И» с входами S_1' и S_2'.

Предположим, что передавалось слово **v** = (11110000) и было принято слово **r** = (11111000). Тогда из (2.10) получаем

$$
\begin{aligned}
S_1 &= r_1 + r_3 + r_5 + r_7 = 1 \\
S_2 &= r_3 + r_4 + r_5 + r_6 = 1 \\
S_3 &= r_2 + r_4 + r_5 + r_7 = 1 \\
S_4 &= r_1 + r_4 + r_5 + r_8 = 1
\end{aligned}
\qquad (2.12)
$$

Таким образом, оба терма $e_3 + e_5$ и $e_4 + e_5$ оцениваются равными «1». Из (2.11) получаем оценку $e_5 = 1$ и оценку исправляющего вектора **e** = (00001000). Окончательно, оценка переданного слова равна **v** = **r** + **e** = (11110000). Таким образом, продемонстрирован *двух шаговый* способ мажоритарного исправления одной ошибки на пятой позиции.

В предыдущем примере было показано как исправляется одна ошибка в заданной позиции для кода $\mathrm{PM}_{1,3}$. Аналогичная процедура может быть применена к любой позиции принятого слова. Следовательно, могут быть получены и все восемь мажоритарных оценок для каждого символа.

В общем случае код $\mathrm{PM}_{r,m}$ может быть декодирован $(r+1)$ – шаговой мажоритарной процедурой декодирования с исправлением любой из возможных комбинаций случайных ошибок веса $\lfloor (2^{m-2} - 1) / 2 \rfloor$ или меньше [MS, LC].

Дополнительно заметим, что циклический код $\mathrm{PM}^*_{r,m}$ декодируется несколько проще. В циклическом коде[5] C, если $(v_1, v_2, \ldots, v_n)$ любое кодовое слово, то и его *циклический сдвиг* $(v_n, v_1, \ldots, v_{n-1})$ тоже кодовое слово. Отсюда следует, что, если некоторая позиция может быть декодирована по мажоритарному методу, то и все n позиций будут декодированы тем же самым способом с использованием циклического сдвига.

Пример 18. В этом примере рассматривается декодер циклического $\mathrm{PM}^*_{1,3}$ кода, который совпадает с двоичным (7,4,3) кодом Хемминга. Для того, чтобы получить проверочные уравнения циклического кода из проверок $\mathrm{PM}_{1,3}$ кода, достаточно удалить во всех уравнениях координату v_1, для которой $x_1 = x_2 = \ldots = x_m$. После удаления v_1 изменим нумерацию позиций $\mathrm{PM}^*_{1,3}$ кода следующим образом:

[5] См. раздел 3.

$$(v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8) \rightarrow (v_1, v_2, v_3, v_4, v_5, v_6, v_7)$$

Как и раньше, можно построить мажоритарный декодер для произвольной позиции (например, пятой), учитывая семь линейно независимых проверочных уравнений:

$$
\begin{aligned}
v_1 + v_2 + v_3 + v_5 &= 0 \\
v_2 + v_3 + v_4 + v_6 &= 0 \\
v_3 + v_4 + v_5 + v_7 &= 0 \\
v_1 + v_4 + v_5 + v_6 &= 0 \\
v_2 + v_5 + v_6 + v_7 &= 0 \\
v_1 + v_3 + v_6 + v_7 &= 0 \\
v_1 + v_2 + v_4 + v_7 &= 0
\end{aligned}
\qquad (2.13)
$$

По аналогии с предыдущим примером находим, что синдромы S_1 и S_2 ортогональны относительно символов v_4 и v_5, а S_2 и S_3 ортогональны относительно v_5 и v_6:

$$
\begin{aligned}
S_1 &= e_3 + e_4 + e_5 + e_7 \\
S_2 &= e_1 + e_4 + e_5 + e_6 \\
S_3 &= e_2 + e_5 + e_6 + e_7
\end{aligned}
\qquad (2.14)
$$

Используя промежуточные оценки сумм $v_4 + v_5$ и $v_5 + v_6$, получаем дополнительно два ортогональных уравнения относительно v_5:

$$
\begin{aligned}
S_1' &= e_4' + e_5' \\
S_2' &= e_5' + e_6'
\end{aligned}
\qquad (2.15)
$$

где e_j', $j = 4, 5, 6$, обозначены мажоритарные оценки, полученные на предыдущем шаге. Этот алгоритм представлен на Рисунке 15 в виде логической схемы.

Эта схема работает следующим образом. Начальное состояние семи ячеек регистра памяти устанавливается нулевым. Предположим, что принятое слово содержит ошибку в позиции с номером i, $1 \le i \le 7$. На каждом такте содержимое регистра памяти сдвигается вправо на одну позицию. Время в тактах представлено на схеме нижним индексом.

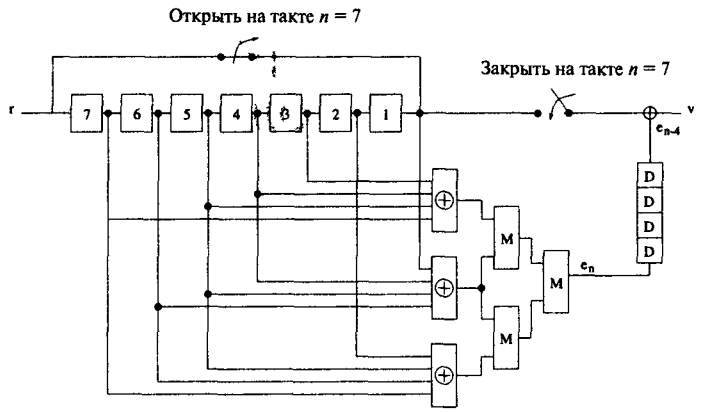


Рис. 15. Мажоритарный декодер циклического кода PM*(1,3)

Рассмотрим случай $i = 1$. Это означает, что ошибка находится в первой позиции принятого слова. Через три такта эта ошибка переместится в пятую ячейку регистра (v_5). Выход мажоритарной логической схемы примет состояние $e_n = 1$. Еще через четыре такта (на седьмом такте) первый принятый символ попадет на выход декодера и будет исправлен. Рассмотрим теперь случай $i = 7$. Через девять тактов ошибка будет обнаружена и выход мажоритарной схемы примет значение $e_n = 1$. Спустя четыре такта (т.е. на тринадцатом такте) последний символ поступит на выход декодера и будет исправлен. Декодер, который здесь рассматривается, имеет задержку 13 тактов. Через 13 тактов содержимое регистра стирается и начинается обработка нового принятого слова.

В следующей главе рассматриваются циклические коды и обширное семейство БЧХ кодов.

Глава 3

ДВОИЧНЫЕ ЦИКЛИЧЕСКИЕ КОДЫ И КОДЫ БЧХ

Цель этой главы состоит в том, чтобы дать минимальный набор понятий, необходимых для понимания конструкции циклических кодов и эффективной реализации их алгоритмов кодирования и декодирования. Кроме того, в этой главе дается введение в двоичные коды БЧХ. Коды Боуза-Чоудхури-Хоквингема (БЧХ) относятся к семейству *циклических кодов*, обладающих четкой алгебраической структурой, которая существенно упрощает процедуры кодирования и декодирования. Двоичные БЧХ коды с минимальным расстоянием 3, известные также как коды Хемминга, имели широкое применение в компьютерных сетях и устройствах памяти из-за простого и быстрого кодирования и декодирования. Кроме того, укороченные (48, 36, 5) БЧХ коды использованы в Американской сотовой системе с временным разделением каналов (TDMA, стандарт IS-54).

3.1. Двоичные циклические коды.

Циклические коды составляют класс кодов, исправляющих ошибки, кодирование и декодирование которых основано на полиномиальном представлении. Простая реализация этих кодов использует регистры сдвига и логические схемы. В этом разделе обсуждаются фундаментальные понятия в области циклических кодов.

3.1.1. Порождающий и проверочный полиномы.

Обозначим C линейный блоковый (n, k) код. Пусть $\mathbf{u}$ *сообщение* и $\mathbf{v}$ соответствующее ему *кодовое слово* кода C. Циклические

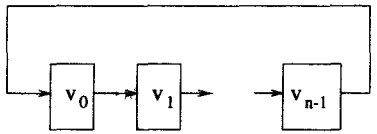


Рис. 16 Циклический регистр сдвига

коды обладают свойствами, которые делают их удобными для аппаратурной (микросхемной) реализации. Сопоставим каждому кодовому слову $\mathbf{v}$ полином $\mathbf{v}(x)$:

$$\mathbf{v} = (v_0, v_1, \cdots, v_{n-1}) \rightarrow \mathbf{v}(x) = v_0 + v_1 x + \cdots v_{n-1} x^{n-1}$$

Переменная x служит индикатором относительного положения элемента v_i в кодовом слове в виде терма (монома) $v_i x^i$ полинома $v(x)$.

Линейный блоковый код C является циклическим тогда и только тогда, когда любой *циклический сдвиг любого кодового слова оказывается другим (или тем же самым) кодовым словом*, т.е.

$$\mathbf{v} = (v_0, v_1, \cdots, v_{n-1}) \rightarrow v(x) = v_0 + v_1 x + \cdots + v_{n-1} x^{n-1}$$

В полиномиальном представлении циклический сдвиг на одну позицию, обозначенный $v^{(1)}(x)$, соответствует умножению на x по модулю $(x^n - 1)$,

$$v(x) \in C \Leftrightarrow v^1(x) = xv(x) \bmod(x^n - 1) \in C$$

Операция циклического сдвига реализуется на *регистре сдвига*, показанном на Рисунке 16.

Пример 19. Рассмотрим случай $n = 7$. Циклический сдвиг на одну позицию вектора $\mathbf{v} = (0101011)$ равен $\mathbf{v}^{(1)} = (1010101)$. В полиномиальном представлении и двоичной арифметике получаем:

$$\begin{aligned} v(x) &= x + x^3 + x^5 + x^6, \\ v^{(1)}(x) &= xv(x) = x^2 + x^4 + x^6 + x^7 \bmod(x^7 + 1) \\ &= x^2 + x^4 + x^6 + x^7 + (x^7 + 1) \\ &= 1 + x^2 + x^4 + x^6 \end{aligned}$$

3.1.2. Порождающий многочлен

Важным свойством циклических кодов является то, что все кодовые слова-полиномы кратны одному фиксированному полиному $g(x)$, который называется *порождающим полиномом* кода. Этот полином (многочлен), как и всякий другой, задается своими *корнями*, которые обычно называют *нулями кода*. Легко показать, что порождающий полином $g(x)$ является делителем бинома $(x^n - 1)$. (По аналогии с целыми числами «$a(x)$ делит $b(x)$» (иначе $a(x)|b(x)$), если $b(x) = a(x)\, q(x)$.) Таким образом, для того, чтобы найти некоторый порождающий многочлен, надо знать разложение бинома $(x^n - 1)$ на *неприводимые множители* $\phi_j(x), j = 1, 2, \ldots, l$,

$$(x^n - 1) = \phi_1(x)\phi_2(x)\cdots\phi_l(x) \tag{3.1}$$

Заметим, что в двоичной арифметике операции $a + b$ и $a - b$ (по модулю 2) дают одинаковый результат. Так как ниже рассматриваются только двоичные коды или коды над конечными полями характеристики два, т.е. использующие двоичную арифметику, то в дальнейшем мы не будем различать эти операции (т.е. знаки «+» и «–»).

Из вышесказанного следует, что

$$g(x) = \prod_{j \in J \subset \{1,2,\ldots,l\}} \phi_j(x) \tag{3.2}$$

Пример 20. На множестве двоичных многочленов, т.е. полиномов с коэффициентами из $Z_2 = \{0,1\}$, бином $x^7 - 1$ имеет следующее разложение,

$$x^7 + 1 = (x+1)(x^3 + x + 1)(x^3 + x^2 + 1)$$

Приведем примеры циклических кодов длины 7.

- Двоичный циклический (7,4,3) код Хемминга с порождающим полиномом $g(x) = x^3 + x + 1$.
- Двоичный циклический (7,6,2) код с проверкой на четность порождается полиномом $g(x) = (x + 1)$.

• Дуальный коду Хемминга двоичный (7,3,4) код (*код максимальной длины*) имеет порождающий многочлен $g(x) = (x + 1)(x^3 + x + 1)$.

3.1.3. Кодирование и декодирование двоичных циклических кодов.

Размерность двоичного циклического (n, k) кода равна

$$k = n - \deg[g(x)]$$

где deg[.] есть степень аргумента. Так как циклический код C является линейным кодом, то любое множество k линейно независимых векторов (кодовых слов) может быть выбрано в качестве порождающей матрицы кода. В частности, двоичные векторы, ассоциированные с многочленами $g(x)$, $xg(x)$, ..., $x^{k-1}g(x)$, линейно независимы. Эти векторы могут быть использованы в качестве строк порождающей матрицы кода C. В этом случае реализуется *несистематическое* кодирование. Другими словами, сообщение не появляется в неизмененном виде на каких-либо позициях кодового слова.

Пример 21. Рассмотрим циклический (7,4,3) код Хемминга с порождающим полиномом $g(x) = x^3 + x + 1 \Leftrightarrow (1101)$. Порождающая матрица этого кода имеет вид:

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

В другом варианте проверочная часть порождающей матрицы циклического кода может быть построена с помощью следующих полиномов:

$$x^{n-1} \bmod g(x),$$
$$\vdots$$
$$x^{n-k+1} \bmod g(x),$$
$$x^{n-k} \bmod g(x).$$

С их помощью реализуется *систематическое* кодирование, показанное в примере ниже.

Пример 22. Пусть C циклический (7,4,3) код Хемминга с порождающим многочленом $g(x) = x^3 + x + 1$. Тогда имеем

$$x^6 \bmod(x^3 + x + 1) = x^2 + 1,$$
$$x^5 \bmod(x^3 + x + 1) = x^2 + x + 1,$$
$$x^4 \bmod(x^3 + x + 1) = x^2 + x,$$
$$x^3 \bmod(x^3 + x + 1) = x + 1.$$

Следовательно, систематическая порождающая матрица кода C имеет вид:

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

Пусть $u(x)$ представляет кодируемое сообщение. Кодирование циклического кода может быть систематическим или несистематическим в зависимости от того, что именно делается с сообщением:

• Несистематическое кодирование

$$v(x) = u(x)g(x) \tag{3.3}$$

• Систематическое кодирование

$$v(x) = x^{n-k}u(x) + [x^{n-k}u(x) \bmod g(x)] \tag{3.4}$$

3.1.4. Проверочный полином.

Полином h(x), который может быть ассоциирован с проверочной матрицей циклического кода, называется проверочным полиномом. Порождающий и проверочный полиномы связаны следующим соотношением

$$g(x)h(x) = x^n + 1 \tag{3.5}$$

Если известен порождающий полином, то проверочный полином легко вычисляется как $h(x) = (x^n + 1) / g(x) = h_0 + h_1x + \ldots + h_kx^k$. Проверочную матрицу кода C легко построить, исполь-

зуя в качестве строк n-k-1 циклических сдвигов проверочного полинома,

$$h^j(x) = x^j h(x) \bmod (x^n - 1),\ j = 0,1,\ldots,n-k-1$$

$$\mathbf{H} = \begin{pmatrix} h_0 & h_1 & h_2 & \cdots & h_k & 0 & 0 & \cdots & 0 \\ 0 & h_0 & h_1 & h_2 & \cdots & h_k & 0 & \cdots & 0 \\ 0 & 0 & h_0 & h_1 & \cdots & \cdots & h_k & \cdots & 0 \\ 0 & 0 & 0 & \ddots & 0 & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & \cdots & \cdots & h_k \end{pmatrix} \qquad (3.6)$$

Пример 23. Циклический (7,4,3) код Хемминга с порождающим многочленом $g(x)=x^3+x+1$ имеет проверочный многочлен $h(x) = (x^7 + 1) / (x^3+x+1) = x^4 + x^2 + x + 1$. Проверочная матрица этого кода имеет, например, следующий вид:

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

Так же как и для линейных кодов, систематическое кодирование циклического кода можно реализовать как решение уравнения

$$v(x)h(x) = 0 \bmod (x^n - 1)$$

Рассмотрим следующее правило систематического кодирования [PW, LC]. Предположим, что код имеет скорость $k/n \le 0{,}5$. Пусть сообщение представлено многочленом $u(x) = u_0 + u_1 x + \ldots + u_{k-1}x^{k-1}$, степень которого меньше k. Пусть $v(x)$ кодовое слово кода C, соответствующее информационному многочлену $u(x)$. На первом шаге $v_\ell = u_\ell$, $\ell = 0, 1, \ldots, k-1$.

Из циклической природы этого кода следует, что проверочные символы кода v_ℓ, $\ell = k, k+1,\ldots, n-1$, могут быть вычислены рекурсивно с помощью проверочного уравнения

$$v_\ell = \sum_{j=0}^{\ell-1} v_j h_{(\ell-k),j},\ \ell = k, k+1,\ldots,n-1 \qquad (3.7)$$

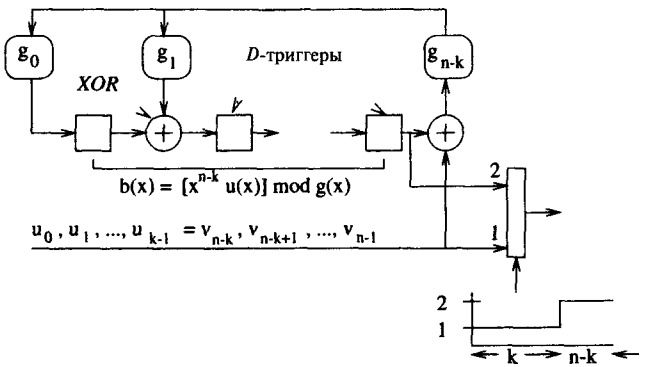


Рис. 17. Устройство систематического кодирования делением на $g(x)$.

где $h_{(\ell-k),j}$ есть j-ый элемент $(\ell - k)$-ой строки матрицы (3.6).

В случае высокой скорости кода, $k/n > 0{,}5$, кодирование с помощью деления $x^{n-k}u(x)$ на порождающий полином эффективнее. В любом случае, кодовое слово получается в систематической форме, k первых символов которого совпадают с символами сообщения, а последние n-k являются проверочными символами.

На Рисунке 17 показана структурная схема кодера двоичного кода с порождающим полиномом $g(x)$. Первые k тактов переключатель (правая нижняя часть схемы) находится в положении 1, а информационные символы передаются в канал связи и одновременно вводятся в схему умножения на x^{n-k} и деления на порождающий многочлен $g(x)$. За эти k тактов в регистре сдвига вычисляется остаток от деления, после чего переключатель переводится в положение 2 и содержимое регистра передается в канал.

Дуальные циклические коды и последовательности максимальной длины

По аналогии с линейными кодами, *дуальным кодом* циклического кода C, порождаемого полиномом $g(x)$, является циклический код $C^\perp$, порождаемый полиномом $h(x)$. Важный

класс циклических кодов, словами которого являются все сдвиги *последовательности максимальной длины* (MLS), дуален циклическому коду Хемминга, [PW]. Множество сдвигов MLS является $(2^m - 1, m, 2^{m-1})$ циклическим кодом, который порождается полиномом $g(x)=(x^n - 1)/p(x)$, где $p(x)$ *примитивный полином*[1]. В дальнейшем этот код будем называть *MLS-код.*

3.1.5. Укороченные циклические коды и CRC коды

Существует много практических задач, в которых требуются коды, исправляющие ошибки, с простыми процедурами кодирования и декодирования. Однако существующие конструкции не всегда имеют нужную длину, размерность и минимальное расстояние.

В этом абзаце приведен пример обращения к автору по электронной почте: «*Нам нужна простая FEC/ECC схема для обнаружения/исправления двоичных одиночных ошибок в информационном блоке длиной 64 символа. Задача состоит в том, чтобы найти или выбрать (ECC) схему кодирования для исправления однобитовой ошибки, используя не более 8 избыточных символов, т.е. при максимальной длине кода 72 (64 информационных и 8 проверочных) символа*».

Так как 72 не является числом вида $2^m - 1$, то, очевидно, не существует подходящего циклического кода, среди тех, с которыми мы уже познакомились. Одним из возможных решений может быть использование циклического (127, 120, 3) кода Хемминга, укороченного до необходимой размерности 64. Этот вариант дает укороченный (71, 64, 3) код Хемминга[2].

[1] 1 Определение примитивного полинома дано в разделе 3.2.1.

[2] В этом примере введено понятие об укороченных кодах. Исторически, (72,64,4) код, исправляющий одну и обнаруживающий две ошибки (SEC/DED), построенный укорочением кода Хемминга с добавлением общей проверки на четность, был предложен для вычислительной машины ИБМ 360 ([Hsia] и Глава 16 в [LC]).

Укорочение (в обсуждаемом здесь варианте) сводится к отбрасыванию информационных позиций исходного кода. Пусть s есть число неиспользуемых информационных символов, которое называют *глубиной* (*длиной*) *укорочения*. Пусть C циклический (n, k, d) код. Укороченное сообщение получается за счет фиксированной установки нулевых значений в некоторых (произвольных) информационных позициях. Остальные k-s позиций могут принимать произвольные значения. Без потери общности, можем считать, что старшие позиции сообщения устанавливаются в нулевые состояния. Тогда $u(x) = u_0 + u_1x + \ldots + u_{k-1-s}x^{k-1-s}$. Данное сообщение преобразуется систематическим кодером в кодовое слово,

$$v(x) = x^{n-k}u(x) + (x^{n-k}u(x) \bmod g(x))$$

степень которого не превышает n-s-1. Таким образом, укороченный код C_s является линейным $(n\text{-}s, k\text{-}s, d_s)$ кодом с кодовым расстоянием $d_s \geq d$. В общем случае укороченный код не остается циклическим кодом.

Пример 24. Пусть C циклический (7,4,3) код Хемминга с порождающим полиномом $1 + x + x^3$. Новый код, полученный из C установкой в нулевое состояние двух старших информационных разрядов, имеет два информационных символа и три проверочных, вычисляемых кодером кода C. Множество полученных кодовых слов образует укороченный линейный (5,2,3) код.

Фундаментальное свойство укороченных циклических кодов C_s состоит в том, что могут быть использованы те же самые кодеры и декодеры, хотя эти коды и не сохраняют устойчивость к циклическому сдвигу. Для компьютерного моделирования гораздо проще дополнять слова нулями на старших позициях и использовать те же самые алгоритмы кодирования и декодирования, которые обсуждаются в этой книге. Этот способ (дополнения нулями) широко используется в микросхемной реализации кодов Рида-Соломона. Очевидно, что нули на старших позициях сообщения не должны включаться в кодовое слово. Более того, декодер модифицируется так, что при-

нятое слово $r(x)$ умножается на x^{n-k+s} вместо умножения на x^{n-k} по модулю $g(x)$ в обычном декодере. Дополнительные материалы о модификации структур кодеров и декодеров для укороченных кодов можно найти в [PW, LC, Wie] и других публикациях.

Другим возможным решением может быть попытка построения других классов циклических кодов с требуемыми параметрами. Интересными классами таких кодов, которые не рассматриваются в этой книге, являются *не примитивные* коды БЧХ [PW], евклидово-геометрические (EG) и проективно-геометрические (PG) коды [LC]. Еще одна возможность состоит в применении *недвоичных* циклических кодов в двоичном представлении, таких как коды Рида-Соломона, рассматриваемые в следующей главе. Двоичное отображение РС кодов обладает дополнительно способностью исправления многократных пакетов ошибок. Дополнительная информация имеется в главе 4.

CRC коды

Один из наиболее популярных стандартов помехоустойчивого кодирования известен как *избыточные циклические коды для обнаружения ошибок* (CRC коды). Эти циклические коды используются для обнаружения ошибок в блоках данных. CRC коды имеют длину $n \le 2^m - 1$. Обычно CRC коды имеют порождающий полином вида $(1 + x)g(x)$, где $g(x)$ порождающий полином кода Хемминга. Обычные значения m равны 12, 16 и 32. Выбор порождающего полинома зависит от допустимой *вероятности необнаруженной ошибки*, которая определяется распределением (спектром) весов кода. Вычисление вероятности необнаруженной ошибки эквивалентно определению спектра весов кода. Эта задача остается исключительно трудной. Несмотря на 50 лет существования и развития теории кодирования имеется лишь незначительный прогресс, опубликованный в [FKKL, Kaz]. Ниже приведен список наиболее популярных CRC кодов (или CRC полиномов).

Код	m	$g(x)$
CRC-12	12	$x^{12} + x^{11} + x^3 + x^2 + x + 1$
CRC-16	16	$x^{16} + x^{15} + x^2 + 1$
CRC-CCITT	16	$x^{16} + x^{15} + x^5 + 1$
CRC-32	32	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

3.2. Общий алгоритм декодирования циклических кодов

Пусть $r(x) = v(x) + e(x)$, где $e(x)$ *полином ошибок*, ассоциированный с вектором ошибок ДСК. Тогда синдром (*синдромный полином*) имеет вид

$$s(x) \triangleq r(x) \bmod g(x) = e(x) \bmod g(x) \tag{3.8}$$

На Рисунке 18 показана обобщенная структура декодера циклического кода. Синдром $s(x)$ используется для определения полинома ошибок $e(x)$. Так как циклический код является, прежде всего, линейным кодом, то эта структура может рассматриваться как вариант «стандартной таблицы» для циклических кодов.

Проблема декодирования равноценна поиску (неизвестного) полинома ошибок $e(x)$ по известному синдрому $s(x)$. Эти полиномы связаны уравнением (3.8), которое составляет основу *синдромного декодера* (известного также как декодер Меггита [Meg]) для циклического кода. Другой (но близкий) вариант декодера, реализующий алгоритм *ловли ошибок*, известный также как декодер Касами, проверяет совпадение синдрома с возможным вектором ошибок. Только очень немногие клас-

Рис. 18. Обобщенная структура декодера циклического кода.

сы кодов имеют такое относительно простое декодирование, как циклические коды Хемминга и Голея. Однако с увеличением корректирующей способности кода $t = \lfloor (d_{min} - 1)/2 \rfloor$ сложность декодера, основанного на комбинаторном обнаружении ошибок, становится чрезвычайно большой.

Предположим, что ошибка возникла на первой принятой позиции, т.е. $e(x) = x^{n-1}$. Соответствующий синдром равен $s(x) = x^{n-1} \bmod g(x)$. Если ошибка, искажающая заданную позицию, обнаруживается данным циклическим кодом, то могут быть обнаружены и ошибки на других позициях за счет циклических сдвигов и соответствующей коррекции синдрома. Синдромный декодер проверяет синдром для каждой позиции принятого слова и, если обнаруживается полином $x^{n-1} \bmod g(x)$, то символ на этой позиции исправляется.

Пример 25. В этом примере рассматривается декодирование циклического (7,4,3) кода Хемминга с порождающим многочленом $g(x) = x^3+x+1$. Схема вычисления синдрома показана на Рисунке 19. Принимаемые символы накапливаются в регистре сдвига и одновременно вводятся в схему деления на $g(x)$. После приема седьмого бита содержимое этого регистра сдвигается на один разряд в каждом такте, а схема деления модифицирует синдром и проверяет совпадение с полиномом

$$x^6 \bmod(1+x+x^3) = 1+x^2 \Leftrightarrow 101 \text{ (в двоичной записи)}$$

Как только на выходе схемы проверки появится 1, будет исправлена ошибка в позиции x^6. В тот же самый момент исправление вводится по обратной связи в схему деления и, тем самым, обнуляет остаток от деления. Нулевой остаток может рассматриваться как сигнал об успешном завершении декодирования. Проверка на нулевой остаток схемы деления позволяет обнаруживать некоторые аномалии по окончании процедуры декодирования.

Перейдем теперь к изучению циклических кодов, исправляющих многократные ошибки, для которых задача декодирования может рассматриваться как решение системы уравнений. По этой причине здесь необходимо знакомство с *полем*, в котором будут выполняться операции умножения, сложения

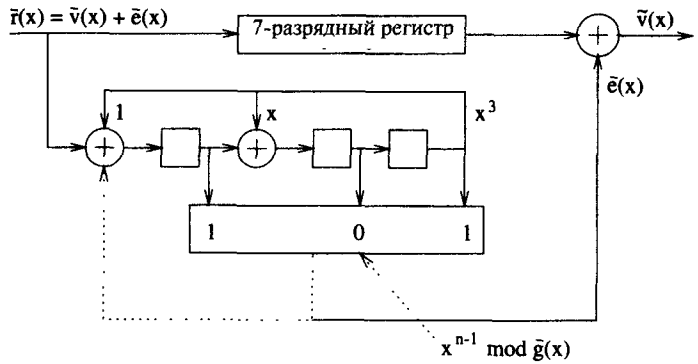


Рис. 19. Синдромный декодер двоичного циклического (7,4,3) кода Хемминга.

и деления. Циклические коды имеют хорошую алгебраическую структуру. Позднее будет показано, что эффективная реализация мощных алгоритмов декодирования достигается при использовании арифметики *конечного поля*, когда известно размещение корней порождающего многочлена кода.

Напомним, что порождающий полином всегда может быть представлен произведением двоичных неприводимых многочленов:

$$g(x) = \prod_{j \in J \subset \{1,2,\ldots,l\}} \phi_j(x)$$

Алгебраическая структура циклических кодов выражается, в частности, в возможности определения корней каждого из многочленов $\phi_j(x)$ в некотором *поле разложения* (которое является расширением поля, которому принадлежат коэффициенты многочлена). В интересующем нас случае поле разложения неприводимого многочлена является *полем Галуа*[3]. В литературе поля Галуа называют также *конечными полями*, имея в виду конечное число принадлежащих ему элементов. Стандартным обозначением является **GF**(q), где q число элементов поля.

[3] В память знаменитого французского математика Эвариста Галуа (1811–1832).

В общем случае число элементов поля есть степень простого числа, однако ниже будут рассматриваться только *поля характеристики* 2, когда $q = 2^m$.

Пример 26. В этом примере мы напомним читателям, что они хорошо знакомы с понятием «поле разложения». Рассмотрим поле действительных чисел. Известно, что в этом поле многочлен $x^2 + 1$ неприводим. Однако в поле *комплексных чисел* он раскладывается в произведение $(x + i)(x - i)$, где $i = \sqrt{-1}$. Таким образом, комплексное поле является полем разложения (т.е. расширением) поля вещественных чисел!

3.2.1. Арифметика GF(q)

Используя методы абстрактной алгебры [PW, LC], можно показать, что в двоичном поле любой многочлен степени m раскладывается над $\mathbf{GF}(2^m)$. Для данной книги достаточно ознакомиться с основами вычислений в конечных полях. Серьезным читателям настоятельно рекомендуется изучение абстрактной алгебры по хорошему учебнику[4].

Использование арифметики $\mathbf{GF}(2^m)$ в процедурах декодирования позволяет заменить сложные комбинационные схемы практичными процессорными архитектурами, пригодными для решения уравнения (3.8). Ниже приводятся инструменты, необходимые для решения систем уравнений, которые возникают при декодировании циклических кодов.

Важные свойства $\mathbf{GF}(2^m)$

Поле Галуа $\mathrm{GF}(2^m)$ изоморфно (с точностью до знака «+») линейному пространству $\{0, 1\}^m$. Другими словами, каждому элементу $\beta \in \mathrm{GF}(2^m)$ соответствует единственный m-мерный вектор $v_\beta \in \{0, 1\}^m$.

В конечном поле имеется *примитивный элемент* $\alpha \in \mathbf{GF}(2^m)$, степени которого порождают все ненулевые элементы поля, т.е. $\beta = \alpha^i \in \mathbf{GF}(2^m)$, $0 \le i \le 2^m - 2$. Элемент α является корнем

[4] Например, по [Her] или [LN*]

неприводимого двоичного полинома $p(x)$, $p(\alpha) = 0$. Наименьшее положительное целое n, для которого $\alpha^n = 1$, равно $2^m - 1$. Все элементы поля удовлетворяют уравнению $\beta^n = 1$, $n = 2^m - 1$ (Другими словами, все ненулевые элементы поля совпадают с корнями степени n из единицы).

Наименьшее положительное целое z, для которого $\beta^z = 1$, равно одному из делителей $2^m - 1$. Это число z называют *порядком* элемента. Примитивными элементами поля являются все такие элементы $\beta = \alpha^i$, $0 \le i \le 2^m - 2$, для которых i взаимно просто с числом $2^m - 1$.

Неприводимый многочлен называют *примитивным*, если его корни имеют максимальный порядок, т.е. $2^m - 1$. Все корни неприводимого многочлена имеют одинаковый порядок. *Периодом* многочлена называют наименьшее число z, при котором $p(x)$ делит бином $x^z - 1$, $z|(2^m - 1)$. Период неприводимого многочлена совпадает с порядком его корней.

Пример 27. Примитивный полином $p(x) = x^3 + x + 1$ порождает поле $\mathbf{GF}(2^3)$. Примитивный элемент поля обозначим α, $p(\alpha) = \alpha^3 + \alpha + 1 = 0$. В таблице показаны три способа представления элементов поля $\mathbf{GF}(2^3)$.

Степень	Полином	Вектор	Степень	Полином	Вектор
-	0	000	α^3	$1+\alpha$	011
1	1	001	α^4	$\alpha+\alpha^2$	110
α	α	010	α^5	$1+\alpha+\alpha^2$	111
α^2	α^2	100	α^6	$1+\alpha^2$	101

Для сложения элементов в $\mathbf{GF}(2^m)$ наиболее удобно векторное представление. При этом используются поразрядное сложение по модулю два. Однако для умножения в поле наиболее удобно степенное представление. В этом случае умножение сводится к сложению показателей степени элементов по модулю $2^m - 1$. Действительно, так как равенство $\alpha^n = 1$, $n = 2^m - 1$, выполняется для всех элементов поля, то $\alpha^{n+1} = \alpha$, $\alpha^{n+2} = \alpha^2$, и т.д. Таким же способом находим, что $\alpha^{-1} = \alpha^{-1+n} = \alpha^n - 1$.

В Примере 27 $\alpha^{-1} = \alpha^6$. В общем случае, обратный элемент $\beta^{-1} = \alpha^b$ элемента $\beta = \alpha^k$ определяется из сравнения $b + k \equiv 0 \bmod(2^m - 1)$. Таким образом, $b = 2^m - 1 - k$. Наконец, для реализации сложений в степенном представлении полезно использовать равенство $p(\alpha) = 0$. Например, в Примере 27 имеем $\alpha^3 = \alpha^3 + (\alpha^3 + \alpha + 1) = \alpha + 1$.

Полиномиальное представление может быть удобным, когда требуется реализация вычислений по модулю некоторого полинома. Примером этого может служить декодирование циклических кодов, где требовалось вычисление $x^i \bmod g(x)$.

Дополнение переводчика. *На самом деле, наиболее естественным представлением элементов конечного поля являются вычеты, т.е. остатки от деления по модулю неприводимого многочлена. В этом случае арифметика поля определяется как сложение, умножение и деление многочленов по неприводимому модулю. Если p(x) примитивный многочлен, то примитивным элементом поля вычетов по модулю p(x) является многочлен x. Иначе, все ненулевые элементы поля могут быть записаны как вычеты $x^i \bmod p(x)$, где $i = 0,1,\ldots,2^m - 2$.*

Дополнительного пояснения требует только операция деления, или точнее, вычисление обратного элемента по неприводимому модулю. Пусть f(x) некоторый (ненулевой) вычет, тогда его обратным элементом $\varphi(x) = (f(x))^{-1}$ называется решение сравнения

$$f(x)\phi(x) - g(x)p(x) = 1 \equiv 1 \bmod p(x)$$

Решение этого сравнения можно найти с помощью алгоритма Евклида для вычисления наибольшего общего делителя многочленов или вычисления подходящей дроби для отношения [p(x) / f(x)]. Подробности можно найти в книгах [Berl, Blah, MS, PW].

Таблицы логарифмов и анти-логарифмов (степеней)

Удобным способом реализации умножений и сложений в конечном поле является применение таблиц с различной интерпретацией их входного адреса. Эти таблицы позволяют легко

переходить от полиномиального к степенному представлению элементов поля и наоборот.

Таблица анти-логарифмов (точнее, степеней) $A(i)$ удобна для реализации сложения. Выходом таблицы является двоичный вектор, записанный как целое число, $A(i)$, соответствующее элементу α^i. Таблица логарифмов (или индексов) $L(i)$ удобна для реализации умножения в конечном поле. Эта таблица выдает значение показателя степени примитивного элемента альфа, $\alpha^{L(i)}$, соответствующего двоичному вектору, представленному целым числом i. Справедливо следующее равенство

$$\alpha^{L(i)} = A(i)$$

Рассмотрим пример использования таблиц для реализации арифметики конечного поля.

Пример 28. Рассмотрим поле $\mathbf{GF}(2^3)$ с порождающим многочленом $p(\alpha) = \alpha^3 + \alpha + 1$ и $\alpha^7 = 1$. Таблицы логарифмов и анти-логарифмов имеют вид:

Адрес входа i	Анти-логарифм $A(i)$	Логарифм $L(i)$
0	1	−1
1	2	0
2	4	1
3	3	3
4	6	2
5	7	6
6	5	4
7	0	5

Дополнение переводчика. *В приведенной таблице имеются две особых точки. Нулевой элемент конечного поля не может быть представлен степенью примитивного элемента. Так что в столбце «Анти-логарифм» не должно быть нулевого элемента поля, а в столбце «Логарифм» нулевому элементу поля не должно быть сопоставлено какое-либо число. Таким образом, при работе*

с этими таблицами требуется специальная проверка на нулевой элемент поля в результате вычислений или в исходных данных.

Рассмотрим вычисление элемента $\gamma = \alpha\,(\alpha^3 + \alpha^5)^3$ в векторной форме. Учитывая свойства поля GF(2^3), находим последовательно:

$$\alpha^3 + \alpha^5 = 110 \oplus 111 = 001 = \alpha^2, \gamma = \alpha(\alpha^2)^3 = \alpha^{1+6} = \alpha^7 = \alpha^0 = 1$$

С использованием таблиц эти вычисления выглядят следующим образом:

$$\gamma = A(L(A(3) \oplus A(5)) * 3 + 1) = A(L(3 \oplus 7) * 3 + 1) =$$
$$= A(L(4) * 3 + 1) = A(2 * 3 + 1) = A(7) = A(0) = 1.$$

Таблицы логарифмов и степеней (анти-логарифмов) используются для реализации арифметики конечного поля и **GF**(2^m). Соответствующие компьютерные программы доступны на ECC веб сайте для моделирования алгоритмов кодирования и декодирования кодов БЧХ и РС с арифметикой в **GF**(2^m). Сами алгоритмы рассматриваются в следующих ниже разделах.

Дополнительные свойства GF(2^m)

Минимальным многочленом $\phi_i(x)$ элемента α^i называется многочлен минимальной степени, корнем которого является данный элемент поля. Следующие свойства минимальных многочленов легко могут быть доказаны. Минимальный многочлен $\phi_i(x)$ элемента α^i имеет двоичные коэффициенты и является неприводимым над **GF**(2) = {0,1}. Более того, корнями этого многочлена являются α^i, α^{2i}, α^{4i}, …, α^{τ}, где $\tau = 2^k$ и k делит m. Эти элементы называются *сопряженными* с α^i элементами поля. Степени сопряженных элементов поля образуют *циклотомический смежный класс* [MS, стр.104, GG, стр. 391]:

$$C_i = \{i, 2i, 4i, \ldots, 2^{k-1} i\}$$

Очевидно, что степень минимального многочлена равна числу элементов (мощности) соответствующего циклотомического смежного класса, т.е.

$$\deg[\phi_i(x)] = |C_i|$$

Циклотомические смежные классы (называемые также циклическими множествами в [PW], стр. 209) обладают свойством расщепления множества вычетов целых чисел Z_n по модулю $n = 2^m - 1$. Таким образом, циклотомические смежные классы не пересекаются. Иначе, их пересечение $C_i \cap C_j = \varnothing$, является пустым множеством, а их объединение равно всему множеству, т.е. $\cup_i C_i = Z_n$.

Пример 29. Циклотомическими множествами по модулю 7 являются:

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

Примитивный элемент α поля **GF**(2^m) (как и все другие) удовлетворяет уравнению $\alpha^n = 1$. Все ненулевые элементы поля являются некоторой степенью примитивного элемента. Таким образом, бином степени $n = 2^m - 1$ имеет следующее разложение на неприводимые двоичные сомножители в двоичном поле:

$$(x^{2^m-1} + 1) = \prod_{j=0}^{M} \phi_{l_j}(x)$$

где M число циклотомических классов, и следующее полное разложение на сомножители первой степени в поле **GF**(2^m)

$$(x^{2^m-1} + 1) = \prod_{j=0}^{2^m-2} (x + \alpha^j) \qquad (3.9)$$

Важно отметить, что степень минимального многочлена $\phi_i(x)$ равна мощности (т.е. числу элементов множества) циклотомического класса C_i. Отсюда следует способ нахождения всех неприводимых делителей бинома ($x^n - 1$):

1. Найти все циклотомические классы по модулю $2^m - 1$.
2. Для каждого циклотомического класса C_s вычислить минимальный многочлен

$$\phi_s(x) = \prod_{i_s \in C_s} (x + a^{i_s}) \qquad (3.10)$$

Этот способ может быть использован для построения порождающих многочленов циклических кодов. Он используется в программе моделирования БЧХ кодов, доступной на ECC веб сайте, для построения порождающего многочлена кода, заданного его корнями.

Пример 30. Рассмотрим поле **GF**(2^3), порождаемое примитивным многочленом $p(x) = x^3 + x + 1$. Корни каждого из делителей бинома $x^7 + 1$ показаны в таблице. Читателю предлагается проверить, что произведения линейных множителей в (3.10) равно указанным в таблице двоичным многочленам.

C_s	Сопряженные элементы	Минимальный многочлен $\phi_s(x)$
$C_0 = \{0\}$	1	$\phi_0(x) = x + 1$
$C_1 = \{1,2,4\}$	$\alpha, \alpha^2, \alpha^4$	$\phi_1(x) = x^3 + x + 1$
$C_3 = \{3,6,5\}$	$\alpha^3, \alpha^6, \alpha^5$	$\phi_3(x) = x^3 + x^2 + 1$

3.3. Двоичные коды БЧХ

Коды БЧХ это двоичные коды, конструкция которых определяется заданием их нулей, т.е. корней порождающего их многочлена:

> *БЧХ код с кодовым расстоянием $d_{min} \geq 2t_d + 1$ является циклическим кодом, порождающий многочлен $g(x)$ которого имеет $2t_d$ последовательных корней в точках $\alpha^b, \alpha^{b+1}, \ldots, \alpha^{b+\delta}$, где $\delta = 2t_d - 1$.*

Таким образом, порождающий многочлен двоичного (n, k, d_{min}) кода БЧХ имеет вид

$$g(x) = \text{НОК}\{\phi_b(x), \phi_{b+1}(x), \ldots, \phi_{b+2t_d-1}(x)\}$$

а длина и размерность кода равны, соответственно,

$$n = \text{НОК}\{n_b, n_{b+1}, \ldots, n_{b+2t_d-1}\}$$

$$k = n - \deg[g(x)]$$

Двоичный БЧХ код, заданный таким образом, имеет *конструктивное кодовое расстояние* равное $2t_d + 1$. Однако, следует заметить, что истинное кодовое расстояние может быть больше конструктивного.

Пример 31. Над полем **GF**(2^3), порождаемым примитивным многочленом $p(x) = x^3 + x + 1$, для параметров $t_d = 1$ и $b = 1$ многочлен

$$g(x) = \text{НОК}\{\phi_1(x), \phi_2\} = x^3 + x + 1$$

порождает двоичный (7,4,3) код БЧХ. На самом деле, это двоичный циклический код Хемминга! Заметим, что вес Хемминга этого порождающего многочлена равен 3 и, следовательно (в данном случае, но не всегда), конструктивное и истинное расстояние кода совпадают.

Пример 32. Рассмотрим поле **GF**(2^4) , порождаемое примитивным многочленом $p(x) = x^4 + x + 1$, и параметры $t_d = 3$, $b = 1$. Тогда многочлен

$$g(x) = \text{НОК}\{\phi_1(x), \phi_3(x)\} = (x^4 + x + 1)(x^4 + x^3 + x^2 + x + 1) =$$

$$= x^8 + x^7 + x^6 + x^4 + 1$$

порождает двоичный (15,7,5) код БЧХ, исправляющий две ошибки.

Пример 33. В поле **GF**(2^4) , порождаемом примитивным многочленом $p(x) = x^4 + x + 1$, и параметры $t_d = 3$, $b = 1$, многочлен

$$g(x) = \text{НОК}\{\phi_1(x), \phi_3(x)\phi_5(x)\} = (x^4 + x + 1)(x^4 + x^3 + x^2 + x + 1)(x^2 + x + 1)$$
$$= x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1$$

порождает двоичный (15,5,7) код БЧХ, исправляющий три ошибки.

Рассмотрим нижнюю границу минимального расстояния кодов БЧХ известную как *граница БЧХ*. Это полезно не только тем, что позволяет оценить корректирующие способности кода в общем случае, но и тем, что выделяет особые свойства ко-

дов БЧХ. Напомним, что элементы α^b, α^{b+1}, ..., $\alpha^{b+\delta}$, где $\delta = 2t_d - 1$, являются корнями порождающего многочлена $g(x)$ и что все кодовые слова v кода БЧХ, ассоциированные с полиномами $v(x)$, кратны порождающему многочлену кода. Следовательно

$$v(x) \in C \Leftrightarrow v(\alpha^i) = 0, b \le i \le b + 2t_d - 1 \qquad (3.11)$$

Таким образом, на основании (3.11), все кодовые слова удовлетворяют следующей системе $2t_d$ уравнений (в матричной форме)

$$(v_0, v_1, \ldots, v_{n-1}) \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \alpha^b & \alpha^{b+1} & \cdots & \alpha^{b+2t_d-1} \\ \alpha^{2b} & \alpha^{2(b+1)} & \cdots & \alpha^{2(b+2t_d-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha^{(n-1)b} & \alpha^{(n-1)(b+1)} & \cdots & \alpha^{(n-1)(b+2t_d-1)} \end{pmatrix} = \mathbf{0} \qquad (3.12)$$

Соответственно, проверочная матрица двоичного БЧХ кода имеет вид

$$\mathbf{H} = \begin{pmatrix} 1 & \alpha^b & \alpha^{2b} & \cdots & \alpha^{(n-1)b} \\ 1 & \alpha^{b+1} & \alpha^{2(b+1)} & \cdots & \alpha^{(n-1)(b+1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{b+2t_d-1} & \alpha^{2(b+2t_d-1)} & \cdots & \alpha^{(n-1)(b+2t_d-1)} \end{pmatrix} \qquad (3.13)$$

Эта проверочная матрица обладает следующим свойством: любая ее $2t_d \times 2t_d$ подматрица, т.е. образованная любыми $2t_d$ столбцами, является *матрицей Вандермонда* ([GG, стр. 95], [Blah*], стр.144). Следовательно (см. раздел 2.1), любые $2t_d$ столбцов проверочной матрицы линейно независимы. Отсюда следует, что минимальное кодовое расстояние этого кода удовлетворяет неравенству $d \ge 2t_d + 1$ (см. [PW], стр. 270, [MS], стр. 201, [LC], стр. 149). Этот результат можно интерпретировать следующим образом:

Граница БЧХ. Если порождающий многочлен $g(x)$ циклического (n,k,d) кода имеет ℓ последовательных корней, например, α^b, α^{b+1}, ..., $\alpha^{b+\ell}$, то $d \ge 2\ell + 1$.

3.4. Полиномиальные коды

Класс циклических полиномиальных кодов включает циклические коды Рида-Маллера, коды БЧХ и Рида-Соломона, коды над конечными геометриями [KLP, PW, LC]. Задание полиномиальных кодов связано с наложением условий на их корни (нули) следующим образом.

Пусть α примитивный элемент поля $GF(2^{ms})$. Пусть s положительное целое число и b делитель $2^s - 1$. Тогда α^h является корнем порождающего полинома $g(x)$ полиномиального кода порядка μ, если и только если b делит h и

$$\min_{0 \le \ell < s} W_{2^s}(h2^\ell) = jb, \text{ где } 0 < j < \left[\frac{m}{b}\right] - \mu$$

где для любого целого i,

$$i = \sum_{\ell=0}^{m-1} i_\ell 2^{s\ell}$$

функция $W_{a(s)}$ определена как $(a(s) = 2^s)$ – ичный вес целого числа i, т.е.

$$W_{2^s}(i) = \sum_{i=0}^{m-1} i_\ell$$

Согласно этому определению коды БЧХ и Рида-Соломона являются полиномиальными с параметрами $b = m = 1$. Коды Рида-Маллера являются подкодами полиномиальных кодов с параметром $s = 1$. Коды над конечными геометриями ([LC], Глава 8) возникают как *дуальные к полиномиальным кодам* [PW]. Ниже даются свойства нулей конечно-геометрических кодов [LC].

Евклидово геометрические (ЕГ) коды

Пусть α примитивный элемент поля $\mathbf{GF}(2^{ms})$. Пусть h неотрицательное целое меньше $2^{ms} - 1$. Тогда α^h является корнем порождающего многочлена $g(x)$ ЕГ кода порядка (μ, s), длины $2^{ms} - 1$, если и только если

$$0 < \max_{0 \le \ell < s} W_{2^s}(h^{(\ell)}) \le (m - \mu - 1)(2^s - 1)$$

где $h^{(\ell)} = 2^\ell h \bmod (2^{ms} - 1)$.

Для $s = 1$ ЕГ коды совпадают с циклическими $PM^*_{m,\mu}$ кодами и, следовательно, ЕГ коды можно рассматривать как обобщенные коды РМ (Рида-Маллера).

Проективно геометрические (ПГ) коды

Пусть α примитивный элемент поля $\mathbf{GF}(2^{(m+1)s})$. *Пусть h неотрицательное целое меньше* $2^{(m+1)s} - 1$. *Тогда α^h является корнем порождающего многочлена $g(x)$ ПГ кода порядка (μ, s), длины* $(2^{ms} - 1)/(2^s - 1)$, *если и только если h делится на* $2^s - 1$ и

$$0 < \max_{0 \le l < s} W_{2^s}(h^{(l)}) \le j(2^s - 1)$$

где $h^{(\ell)} = 2^\ell h \bmod(2^{ms} - 1)$ *и* $0 \le j \le m\text{-}n$.

3.5. Декодирование двоичных БЧХ кодов

Главной идеей в декодировании БЧХ кодов является использование элементов конечного поля для нумерации позиций кодового слова (или, эквивалентно, в порядке коэффициентов ассоциированного многочлена). Такая нумерация показана на Рисунке 20 для вектора $\mathbf{r} = (r_0, r_1 \ldots r_{n-1})$, соответствующего многочлену $r(x)$.

Позиции ошибок могут быть найдены из решения системы уравнений в поле $GF(2^m)$. Эти уравнения можно получить, вводя многочлен ошибок $e(x)$ и учитывая нули кода $\mathcal{C}$ для $b \le j \le b + 2t_d - 1$, как показано ниже.

Пусть $r(x) = v(x) + e(x)$ представляет полином, ассоциированный с принятым словом, где многочлен ошибок определен как

$$e(x) = e_{j_1} x^{j_1} + e_{j_2} x^{j_2} + \ldots + e_{j_\nu} \tag{3.14}$$

где $\nu \le t_d$ число ошибок в принятом слове. Множества

$$\{e_{j_1}, e_{j_2}, \ldots, e_{j_\nu}\} \quad \text{и} \quad \{\alpha^{j_1}, \alpha^{j_2}, \ldots, \alpha^{j_\nu}\}$$

называют *значениями ошибок* и *локаторами ошибок*, соответственно, где $e_j \in \{0, 1\}$ для двоичных БЧХ[5] кодов и $\alpha \in \mathbf{GF}(2^m)$.

[5] Позднее будет показано, что для кодов Рида-Соломона $e_j \in \mathbf{GF}(2^m)$

значения	r_0 r_1	r_{n-1}
локаторы позиций	1 α	α^{n-1}

Рис. 20. Нумерация позиций кодового слова элементами поля $GF(2^m)$

Синдромы определены как значения принятого полинома $r(x)$ в нулях кода:

$$\begin{aligned}
S_1 &= r(\alpha^b) = e_{j_1}\alpha^{bj_1} + \ldots + e_{j_\nu}\alpha^{bj_\nu} \\
S_2 &= r(\alpha^{b+1}) = e_{j_1}\alpha^{(b+1)j_1} + \ldots + e_{j_\nu}\alpha^{(b+1)j_\nu} \\
&\vdots \\
S_{2t_d} &= r(\alpha^{b+2t_d-1}) = e_{j_1}\alpha^{(b+2t_d-1)j_1} + \ldots + e_{j_\nu}\alpha^{(b+2t_d-1)j_\nu}
\end{aligned}$$

Это определение эквивалентно $\mathbf{s} = \mathbf{H}\mathbf{r}^\top$, где проверочная матрица $\mathbf{H}$ определена в (3.13).

Введем *многочлен локаторов ошибок*

$$\sigma(x) = \prod_{l=1}^{\nu}(1 + \alpha^{j_l} x) = 1 + \sigma_1 x + \sigma_2 x^2 + \ldots + \sigma_\nu \tag{3.15}$$

корни которого равны обратным величинам *локаторов ошибок*. Тогда справедливо следующее соотношение между коэффициентами многочлена локаторов ошибок и синдромами (см., например, [Pet], [LC] стр. 154, [PW] стр. 284):

$$\begin{pmatrix} S_{\nu+1} \\ S_{\nu+2} \\ \vdots \\ S_{2\nu} \end{pmatrix} = \begin{pmatrix} S_1 & S_2 & \cdots & S_\nu \\ S_2 & S_3 & \cdots & S_{\nu+1} \\ \vdots & \vdots & \ddots & \vdots \\ S_\nu & S_{\nu+1} & \cdots & S_{2\nu-1} \end{pmatrix} \begin{pmatrix} \sigma_\nu \\ \sigma_{\nu-1} \\ \vdots \\ \sigma_1 \end{pmatrix} \tag{3.16}$$

Решение *ключевого уравнения*, представленного в (3.16), требует довольно интенсивных вычислений в процедуре декодирования БЧХ кодов. Известны следующие методы решения ключевого уравнения:

1. Алгоритм Берлекемпа-Мэсси (BMA)

Этот алгоритм был предложен Берлекемпом [Ber1] и Мэсси [Mas2]. По числу операций в конечном поле этот алгоритм обладает высокой эффективностью. BMA обычно использу-

ется для программной реализации или моделирования кодов БЧХ и РС.

2. Евклидов алгоритм (ЕА)

Этот метод решения ключевого уравнения в полиномиальной форме был введен в [SKHN] и позднее исследован в [Man]. Из-за высокой регулярности структуры этого алгоритма его широко используют для аппаратной реализации декодеров БЧХ и РС кодов.

3. Прямое решение

Этот алгоритм, предложенный Питерсоном [Pet], находит коэффициенты многочлена локаторов ошибок прямым решением системы (3.16) как системы линейных уравнений. В литературе часто используется термин *PGZ* (*Питерсон-Горенштейн-Цирлер*) *декодер*. Этот термин обязан своим появлением публикации в [GZ], где был использован алгоритм Питерсона для декодирования недвоичных кодов БЧХ и РС. В действительности, так как сложность обращения матрицы растет как куб корректирующей способности кода, прямой алгоритм может быть использован только для малых значений t_d. Решения (3.16) для значений t_d до 6, включительно, даны в [ML] (раздел 5.3).

3.5.1. Общий метод декодирования для БЧХ кодов

На Рисунке 21 показана блок-схема декодера БЧХ кодов (как двоичных, так и недвоичных). Декодер состоит из логических схем и обрабатывающих блоков, реализующих следующие задачи:

- Вычислить синдромы, вычисляя значения принятого полинома в нулях кода[6]

$$S_i = r(\alpha^i),\ i = b,\ b+1,\ldots,b+2t_d-1 \qquad (3.17)$$

- Найти коэффициенты многочлена *локаторов ошибок* $\sigma(x)$.

[6] Для двоичных БЧХ кодов справедливо равенство $S_{2i} = S_i^2$, которое позволяет сократить объем вычислений.

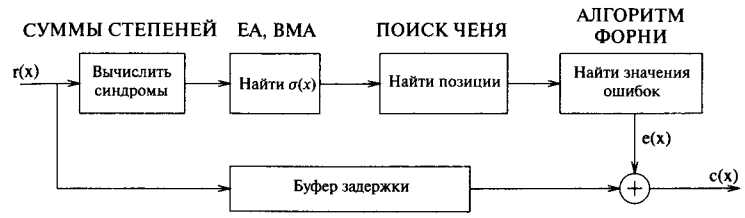


Рис. 21. Архитектура БЧХ декодера.

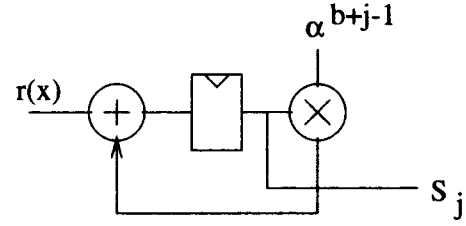


Рис. 22. Пример схемы вычисления синдрома.

- Найти *обратные величины корней* $\sigma(x)$, т.е. позиции ошибок j_1, j_2, ... , j.
- Найти *значения ошибок* (этап ненужный для двоичных кодов)
- *Исправить принятое слово* на вычисленных позициях для вычисленных значений ошибок.

Одним из преимуществ использования арифметики над конечным полем **GF**(2^m) является возможность применения относительно простых логических схем и вычислительных блоков. Например, на Рисунке 22 показана схема вычисления синдромов S_j. Умножение в поле **GF**(2^m) также реализуется относительно простой логической схемой. Некоторые детали аппаратной реализации вычислительных блоков в поле **GF**(2^m) можно найти в [PW], а также в Главах 5 и 10 [WB].

3.5.2. Алгоритм Берлекемпа-Мэсси (ВМА)

Алгоритм Берлекэмпа-Мэсси лучше всего рассматривать как итеративный процесс построения минимального линейного

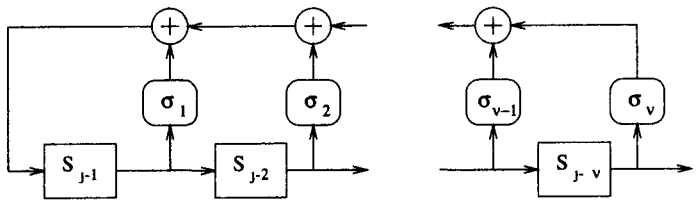


Рис. 23. ЛРОС с отводами $\sigma_1, \sigma_2, ..., \sigma_v$ и выходом $S_1, S_2, ..., S_{2v}$.

регистра (сдвига) с обратной связью (ЛРОС), аналогичного показанному на Рисунке 23, который генерирует известную *последовательность синдромов* $S_1, S_2, ...S_{2t_d}$.

Целью BMA является построение многочлена (обратной связи) $\sigma^{(l+1)}(x)$ наименьшей степени, удовлетворяющего следующему уравнению, выведенному из (3.16):

$$\sum_{j=0}^{\ell_l+1} S_{k-j}\,\sigma_j^{(l+1)} = 0,\ \ell_l < k < l+1 \qquad (3.18)$$

Решение этой задачи эквивалентно условию, что многочлен

$$\sigma^{(l+1)}(x) = 1 + \sigma_1^{(l+1)}x + ... + \sigma_{\ell_{l+1}}^{(l+1)}x^{\ell_{l+1}}$$

является многочленом обратной связи ЛРОС, который генерирует ограниченную последовательность синдромов.

Несовместность (рассогласование, расхождение, различие) на i-ой итерации, определенная как

$$d_l = S_{l+1} + S_l\sigma_1^{(l)} + \cdots + S_{l-\ell_l+1}\sigma_{\ell_l}^{(l)}$$

является мерой соответствия синдромной последовательности и генерируемой ЛРОС и содержит корректирующий множитель для вычисления $\sigma^{(l+1)}$ на следующей итерации. Возможны два случая [Pet][7]:

- Если $d_l = 0$, то уравнение (3.18) удовлетворяется с равенством

$$\sigma^{(l+1)}(x) = \sigma^l(x),\ \ell_{l+1} = \ell_l \qquad (3.19)$$

[7] Существует вариация BMA, введенная Мэсси [Mas], которая используется в некоторых публикациях Эта модификация будет рассмотрена в следующей главе Разумеется, обе вариации BMA дают одинаковый результат!

- Если $d_l \neq 0$, то решение на следующей итерации имеет вид

$$\sigma^{(l+1)}(x) = \sigma^l(x) + d_l d_m^{-1} x^{l-m}\sigma^{(m)}(x)$$
$$\ell_{l+1} = \max\{\ell_l, \ell_m + l - m\}, \qquad (3.20)$$

где является решением на m-ой итерации такое, что $-1 \le m < i$, $d_m \neq 0$ и разность $(m - \ell_m)$ максимальна.

Итеративное вычисление $\sigma^{(l+1)}(x)$ продолжается, пока не удовлетворятся одно или оба условия: либо $i \ge \ell_{l+1} + t_d - 1$, либо $i = 2t_d - 1$.

Начальными условиями алгоритма являются:

$$\sigma^{(-1)}(x) = 1,\ \ell_{-1} = 0,\ d_{-1} = 1$$
$$\sigma^{(0)}(x) = 1,\ \ell_0 = 0,\ d_0 = S_1. \qquad (3.21)$$

Заметим также, что в Алгоритме Берлекэмпа-Мэсси используются программные инструкции (*если – то*). По этой причине BMA не слишком удобен для аппаратной реализации. Тем не менее, по числу операций в конечном поле **GF**(2^m) этот алгоритм весьма эффективен. Эта версия алгоритма реализована в большинстве программ на языке Си для моделирования БЧХ кодов и доступна на ECC веб сайте.

Пример 34. Пусть C двоичный (15, 5, 7) код БЧХ, исправляющий три ошибки из Примера 33. Для проверки вычислений и просто для справки приводится степенное и векторное представление элементов поля **GF**(16), порожденное примитивным многочленом $p(x) = 1 + x + x^4$:

Таблица элементов поля **GF**(2^4), $p(x) = 1 + x + x^4$:

Степень	0	1	α	α^2	α^3	α^4	α^5	α^6
Вектор	0000	0001	0010	0100	1000	0011	0110	1100
Степень	α^7	α^8	α^9	α^{10}	α^{11}	α^{12}	α^{13}	α^{14}
Вектор	1011	0101	1010	01111	1110	1111	1101	1001

Порождающий многочлен кода C равен $g(x) = x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1$. Предположим, что информационный полином равен $u(x) = x + x^2 + x^4$. Тогда соответствующее ему кодовое слово равно $v(x) = x + x^2 + x^3 + x^4 + x^8 + x^{11} + x^{12} + x^{14}$.

Пусть принятое слово равно $r(x) = 1 + x + x^2 + x^3 + x^4 + x^6 + x^8 + x^{11} + x^{14}$, соответствующее вектору **r=v+e**, полученному в результате передачи слова **v** по ДСК. (Вектору **e** соответствует многочлен ошибок $e(x) = 1 + x^6 + x^{12}$. Хотя декодеру этот вектор ошибок неизвестен, он используется здесь для упрощения вычисления синдромов.)

Синдромы равны:

$$S_1 = r(\alpha) = 1 + \alpha^6 + \alpha^{12} = \alpha$$
$$S_2 = S_1^2 = \alpha^2$$
$$S_3 = r(\alpha^3) = 1 + \alpha^3 + \alpha^6 = \alpha^8$$
$$S_4 = S_2^2 = \alpha^4$$
$$S_5 = r(\alpha^5) = 1 + 1 + 1 = 1$$
$$S_6 = S_3^2 = \alpha$$

Алгоритм Берлекемпа-Мэсси (BMA):

- **Итерация 0:** Инициализация.

$\sigma^{(-1)}(x) = 1, \ell_{-1} = 0, d_{-1} = 1, \sigma^{(0)}(x) = 1, \ell_0 = 0, d_0 = S_1 = \alpha.$

- **Итерация 1:**

$i = 0, d_0 = \alpha \neq 0, m = -1 = \arg(\max(-1+0) = -1)$ для $d_{-1} \neq 0.$

$\sigma^{(1)}(x) = \sigma^{(0)}(x) + d_0 d_{-1}^{-1} x^{(0-(-1))} \sigma^{(-1)}(x) = 1 + \alpha x,$

$\ell_1 = \max\{\ell_0, \ell_{-1} + 0 - (-1)\} = 1,$

$\ell_1 + 3 - 1 \leq 0$? Нет:

$d_1 = S_2 + S_1 \sigma_1^{(1)} = \alpha^2 + \alpha(\alpha) = 0.$

- **Итерация 2:**

$i = 1, d1 = 0,$

$\sigma^{(2)}(x) = \sigma^{(1)}(x) = 1 + \alpha x,$

$\ell_2 = \ell_1,$

$\ell_2 + 3 - 1 \leq 1$? Нет:

$d_2 = S_3 + S_2 \sigma_1^{(1)} = \alpha^8 + \alpha^2(\alpha) = \alpha^{13}.$

- **Итерация 3:**

$i = 2, d_2 = \alpha^{13} \neq 0, m = 0 = \arg(\max(0-0) = 0)$ для $d_0 \neq 0.$

$\sigma^{(3)}(x) = \sigma^{(2)}(x) + d_2 d_0^{-1} x^{(2-0)} \sigma^{(0)}(x) =$

$(1 + \alpha x) + \alpha^{13}(\alpha^{-1}) x^2 (1) = 1 + \alpha x + \alpha^{12} x^2,$

$\ell_3 = \max\{\ell_2, \ell_0 + 2 - 0\} = 2,$

$\ell_3 + 3 - 1 \leq 2$? Нет

$d_3 = S_4 + S_3 \sigma_1^{(3)} + S_2 \sigma_2^{(3)} = \alpha^4 + \alpha^8(\alpha) + \alpha^2(\alpha^{12}) = 0.$

- **Итерация 4:**

$i = 3, d_3 = 0,$

$\sigma^{(4)}(x) = \sigma^{(3)}(x) = 1 + \alpha x + \alpha^{12} x^2,$

$\ell_4 = \ell_3,$

$\ell_4 + 3 - 1 \leq 3$? Нет:

$d_4 = S_5 + S_4 \sigma_1^{(4)} + S_3 \sigma_2^{(4)} = 1 + \alpha^4(\alpha) + \alpha^8 \alpha^{12} = 1.$

- **Итерация 5:**

$i = 4, d_4 = 1 \neq 0, m = 2 = \arg(\max(2-1) = 1)$ для $d_2 \neq 0.$

$\sigma^{(5)}(x) = \sigma^{(4)}(x) + d_4 d_2^{-1} x^{(4-2)} \sigma^{(2)}(x) = (1 + \alpha x + \alpha^{12} x^2) +$

$+ (1)(\alpha^{13})^{-1} x^2 (1 + \alpha x) = 1 + \alpha x + \alpha^7 x^2 + \alpha^3 x^3,$

$\ell_5 = \max\{\ell_4, \ell_2 + 4 - 2\} = 3,$

$\ell_5 + 3 - 1 \leq 4$? Нет:

$d_5 = S_6 + S_5 \sigma_1^{(5)} + S_4 \sigma_2^{(5)} + S_3 \sigma_3^{(5)} = \alpha + \alpha + \alpha^4(\alpha^7) + \alpha^8(\alpha^3) = 0.$

- **Итерация 6:**

$i = 5, d_5 = 0,$

$\sigma^{(6)}(x) = \sigma^{(5)}(x) = 1 + \sigma x + \alpha^7 x^2 + \alpha^3 x^3,$

$\ell_6 = \ell_5 = 3,$

$\ell_6 + 3 - 1 \leq 3$? Да: Конец.

Таким образом, $\sigma(x) = 1 + \alpha x + \alpha^7 x^2 + \alpha^3 x^3$.

То, что на нечетных шагах алгоритма $d_i = 0$ в предыдущем примере не случайность. Для *двоичных* кодов БЧХ это закономерность. С тем же результатом можно было выполнять толь-

ко четные шаги алгоритма. Единственное изменение состоит в замене правила остановки на следующее

$$i \geq \ell_{i+2} + t_d - 2$$

В результате снижается сложность декодирования. Читателю предлагается найти $\sigma(x)$ в Примере 34, используя только три итерации.

3.5.3. Декодер PGZ

Впервые этот алгоритм был рассмотрен Питерсоном в [Pet]. Решение ключевого уравнения (3.16) может быть найдено с помощью стандартной техники решения системы линейных уравнений. Это решение дает коэффициенты$\sigma(x)$. Применению этой техники мешает только то, что неизвестно действительное число ошибок в принятом слове. По этой причине приходится проверять гипотезу о том, что действительное количество ошибок в принятом слове равно ν. Предположим, что не все синдромы S_i, $1 \leq i \leq 2t$ равны нулю. Очевидно, что принятое слово совпадает с одним из кодовых, если все синдромы равны нулю. Тогда процедура декодирования на этом завершается!

Декодер начинает с предположения о том, что возникло максимальное число ошибок, $\nu_{max} = t_d$. Он вычисляет определитель Δ_i для $i = \nu_{max} = t_d$

$$\Delta_i = \det \begin{pmatrix} S_1 & S_2 & \cdots & S_i \\ S_2 & S_3 & \cdots & S_{i+1} \\ \vdots & \vdots & \ddots & \vdots \\ S_i & S_{i+1} & \cdots & S_{2i-1} \end{pmatrix} \tag{3.22}$$

и сравнивает его с нулем. Если определитель равен нулю, то действительное число ошибок меньше, чем предполагалось. Значение i уменьшается на единицу и снова проверяется определитель. Процедура повторяется, если это необходимо, пока $i > 1$. Как только окажется, что определитель не равен нулю, вычисляется обратная матрица для матрицы синдромов и вычисляются

значения $\sigma_1, \sigma_2, \ldots, \sigma_\nu$, где $\nu = i$. В случае, когда $\Delta_i = 0$, для $i = 1, 2, \ldots, t_d$, декодирование считается безуспешным и регистрируется обнаружение неисправляемой комбинации ошибок.

Пример 35. В этом примере полином локаторов ошибок для (15, 5, 7) БЧХ кода из Примера 34 находится с помощью PGZ алгоритма. Предположим для начала, что имеется $i = t_d = 3$ ошибки. Определитель Δ_3 вычисляется (с использованием алгебраического дополнения) следующим образом:

$$\Delta_3 = \det \begin{pmatrix} \alpha & \alpha^2 & \alpha^8 \\ \alpha^2 & \alpha^8 & \alpha^4 \\ \alpha^8 & \alpha^4 & 1 \end{pmatrix} = \alpha(\alpha^8 + \alpha^8) + \alpha^2(\alpha^2 + \alpha^{12})$$

$$+ \alpha^8(\alpha^6 + \alpha) = \alpha^{2+7} + \alpha^{8+11} = \alpha^{14}.$$

Так как $\Delta_3 \neq 0$, то подтверждается предположение о том, что в принятом слове три ошибки. Подставляя синдромы, найденные в Примере 34, в ключевое уравнение (3.16), получаем:

$$\begin{pmatrix} \alpha & \alpha^2 & \alpha^8 \\ \alpha^2 & \alpha^8 & \alpha^4 \\ \alpha^8 & \alpha^4 & 1 \end{pmatrix} \begin{pmatrix} \sigma_3 \\ \sigma_2 \\ \sigma_1 \end{pmatrix} = \begin{pmatrix} \alpha^4 \\ 1 \\ \alpha \end{pmatrix} \tag{3.23}$$

Напомним, что $(\Delta_3)^{-1} = \alpha^{-14} = \alpha$. Решение уравнения (3.23) имеет вид:

$$\sigma_3 = \alpha \det \begin{pmatrix} \alpha^4 & \alpha^2 & \alpha^8 \\ 1 & \alpha^8 & \alpha^4 \\ \alpha & \alpha^4 & 1 \end{pmatrix} = \alpha(\alpha^7 + \alpha^{12}) = \alpha^3,$$

$$\sigma_2 = \alpha \det \begin{pmatrix} \alpha & \alpha^4 & \alpha^8 \\ \alpha^2 & 1 & \alpha^4 \\ \alpha^8 & \alpha & 1 \end{pmatrix} = \alpha(\alpha^{11} + \alpha) = \alpha^7,$$

$$\sigma_1 = \alpha \det \begin{pmatrix} \alpha & \alpha^2 & \alpha^4 \\ \alpha^2 & \alpha^8 & 1 \\ \alpha^8 & \alpha^4 & \alpha \end{pmatrix} = \alpha(\alpha^{15} + \alpha^{15} + \alpha^{15}) = \alpha.$$

Таким образом, получили $\sigma(x) = 1 + \alpha x + \alpha^7 x^2 + \alpha^3 x^3$, что совпадает с результатом, полученным по алгоритму БМ в Примере 34.

3.5.4. Евклидов алгоритм (ЕА)

В основе этого алгоритма лежит хорошо известная процедура нахождения наибольшего общего делителя (НОД) двух полиномов (или целых чисел). Ниже рассматривается применение ее для декодирования БЧХ кодов.

Определим *полином значений ошибок* как $\Lambda(x) = \sigma(x)S(x)$, где *синдромный полином* имеет вид

$$S(x) = 1 + S_1 x + \cdots S_{2t_d} x^{2t_d} \tag{3.24}$$

Из уравнения (3.16) следует, что

$$\Lambda(x) = \sigma(x)S(x) \bmod x^{2t_d+1} \tag{3.25}$$

Задача декодирования может быть переформулирована как задача определения многочлена $\Lambda(x)$, удовлетворяющего уравнению (3.25). Это решение может быть найдено применением расширенного алгоритма Евклида к многочленам $r_0(x) = x^d$, $d = 2t_d + 1$ и $r_1(x) = S(x)$. Если на j-*ом* шаге алгоритма получено решение

$$r_j(x) = a_j(x)x^{2t_d+1} + b_j(x)S(x)$$

такое, что $\deg[r_j(x)] \le t_d$, то $\Lambda(x) = r_j(x)$ и $\sigma_i(x) = b_j(x)$. Заметим, что для задачи декодирования полином $a_i(x)$ не представляет интереса, так как решение (3.25) ищется по модулю x^d.

Расширенный алгоритм Евклида вычисления НОД состоит в следующем.

Расширенный алгоритм Евклида: Вычисление НОД($r_0(x)$, $r_1(x)$)

- Вход: $r_0(x)$, $r_1(x)$, $\deg[r_0(x)] \ge \deg[r_1(x)]$,
- Начальные условия: $a_0(x) = 1$, $b_0(x) = 0$, $a_1(x) = 0$, $b_1(x) = 1$.
- На шаге j ($j \ge 2$) применить *длинное деление* к многочленам $r_{j-2}(x)$ и $r_{j-1}(x)$,

$$r_{j-2}(x) = q_j(x)r_{j-1}(x) + r_j(x), \quad 0 \le \deg[r_j(x)] < \deg[r_{j-1}(x)]$$

- Вычислить

$$a_j(x) = a_{j-2}(x) - q_j(x)a_{j-1}(x), \; b_j(x) = b_{j-2}(x) - q_j(x)b_{j-1}(x)$$

- Остановить вычисления на итерации $last = j_{last}$, когда $\deg[r_{last}(x)] = 0$.

Выход: НОД($r_0(x)$, $r_1(x)$) = $r_k(x)$, где k наибольшее ненулевое целое такое, что $r_k(x) \ne 0$ и $k < j_{last}$.

Пример 36. В этом примере вычисляется полином локаторов ошибок $\sigma(x)$ для (15, 5, 7) кода БЧХ из примера 34 по алгоритму Евклида.

- **Начальные условия:**

$$r_0(x) = x^7,$$
$$r_1(x) = S(x) = 1 + \alpha x + \alpha^2 x^2 + \alpha^8 x^3 + \alpha^4 x^4 + x^5 + \alpha x^6,$$
$$b_0(x) = 0, \; b_1(x) = 1.$$

- $j = 2$:

$$x^7 = (1 + \alpha x + \alpha^2 x^2 + \alpha^8 x^3 + \alpha^4 x^4 + x^5 + \alpha x^6)(\alpha^{14} x + \alpha^{13}) +$$
$$+ \alpha^8 x^5 + \alpha^{12} x^4 + \alpha^{11} x^3 + \alpha.$$
$$r_2(x) = \alpha^8 x^5 + \alpha^{12} x^4 + \alpha^{11} x^3 + \alpha,$$
$$q_2(x) = \alpha^{14} x + \alpha^{13},$$
$$b_2(x) = b_0(x) + q_2(x)b_1(x) = \alpha^{14} x + \alpha^{13}.$$

- $j = 3$:

$$S(x) = (\alpha^8 x^5 + \alpha^{12} x^4 + \alpha^{11} x^3 + \alpha^{13})(\alpha^8 x + \alpha^2) +$$
$$+ \alpha^{14} x^4 + \alpha^3 x^3 + \alpha^2 x^2 + \alpha^{11} x.$$
$$r_3(x) = \alpha^{14} x^4 + \alpha^3 x^3 + \alpha^2 x^2 + \alpha^{11} x,$$
$$q_3(x) = \alpha^8 x + \alpha^2,$$
$$b_3(x) = b_1(x) + q_3(x)b_2(x) = \alpha^7 x^2 + \alpha^{11} x.$$

- $j = 4$:

$$\alpha^8 x^5 + \alpha^{12} x^4 + \alpha^{11} x^3 + \alpha^{13} =$$
$$(\alpha^{14} x^4 + \alpha^3 x^3 + \alpha^2 x^2 + \alpha^{11} x)(\alpha^9 x) + \alpha^5 x + \alpha^{13}.$$
$$r_4(x) = \alpha^5 x + \alpha^{13},$$
$$q_4(x) = \alpha^9 x,$$
$$b_4(x) = b_2(x) + q_4(x) b_3(x) = \alpha x^3 + \alpha^5 x^2 + \alpha^{14} x + \alpha^{13}.$$

Так как $\deg[r_4(x)] = 1 \le 3$, алгоритм останавливается.

Очевидно, что многочлен

$$\sigma(x) = b_4(x) = \alpha^{13}(1 + \alpha x + \alpha^7 x^2 + \alpha^3 x^3)$$

имеет те же самые корни, что и многочлены, найденные алгоритмами BMA и PGZ (см. Пример 37 ниже), и отличается только константным множителем[8].

Последний пример показывает, что в общем случае многочлен локаторов ошибок, полученный алгоритмом Евклида, может отличаться на константный множитель от решения, полученного алгоритмами BMA и PGZ. Для декодирования представляют интерес только корни этих полиномов, а не их коэффициенты. Таким образом, все три метода находят требуемый многочлен локаторов ошибок.

3.5.5. Метод Ченя и исправление ошибок

Для поиска корней $\sigma(x)$ на множестве локаторов позиций кодовых символов используется метод проб и ошибок, получивший название *метод Ченя*. Для всех ненулевых элементов $\beta \in \mathbf{GF}(2^m)$, которые генерируются в порядке $1, \alpha, \alpha^2, \ldots$ проверяется условие $\sigma(\beta^{-1}) = 0$. Этот процесс легко реализуется аппаратно. На самом деле, поиск корней (факторизация) многочленов над $\mathbf{GF}(2^m)$ является увлекательной математической задачей, которая еще ждет своего решения.

Для двоичных кодов БЧХ исправление ошибок на позициях $j_1, \ldots, j_\nu$, соответствующих найденным корням, сводится к инвертированию символов, принятых на этих позициях, т.е.

[8] Нормализованный многочлен $\sigma_{norm}(x) = (\sigma_0)^{-1}\sigma(x)$ совпадает с тем, который был найден с помощью процедур BMA и PGZ

$$\hat{v}_{j_\ell} = v_{j_\ell} \oplus 1, \quad 1 \le \ell \le \nu$$

после чего, выдается восстановленное кодовое слово $\hat{v}(x)$.

Пример 37. Продолжая Пример 34, находим, что корни многочлена локаторов ошибок равны: 1, $\alpha^9 = \alpha^{-6}$ и $\alpha^3 = \alpha^{-12}$. Другими словами, получено следующее разложение многочлена на множители

$$\sigma(x) = (1 + x)(1 + \alpha^6 x)(1 + \alpha^{12} x)$$

Соответственно, восстановленный многочлен ошибок равен $e(x) = 1 + x^6 + x^{12}$ и

$$\hat{v}(x) = x + x^2 + x^3 + x^4 + x^8 + x^{11} + x^{12} + x^{14}$$

Три ошибки исправлены.

3.5.6. Исправление стираний и ошибок

Существует много ситуаций, когда решение о принятом символе не может считаться надежным. Рассмотрим, например, передачу двоичных символов по каналу с АБГШ с двоичной фазовой манипуляцией, т.е. с отображением 0→+1 и 1→−1. Если значение принятого сигнала слишком близко к нулю, то возможно, более разумно, с точки зрения минимизации вероятности ошибки декодирования, отказаться от решения о принятом символе. В таких случаях принятый символ «стирается», а такое решение называют *стиранием*[9]. Подобные стирания представляют собой простейшую форму *мягкого решения*, которому будет посвящена Глава 7.

Введение стираний, по сравнению с исправлением только ошибок, обладает тем преимуществом, что *декодеру известны позиции с ненадежными символами*. Пусть d минимальное кодовое расстояние, ν число ошибок и μ число стираний в принятом слове. Тогда минимальное Хеммингово расстояние по нестертым позициям снижается, по меньшей мере, до величины

[9] С теоретико-информационной точки зрения ДСК превращается в канал с двоичным входом и троичным выходом, называемый *двоичным каналом с ошибками и стираниями*.

d-μ. Следовательно, корректирующая способность равна $\lfloor (d-\mu-1)/2 \rfloor$ и справедливо следующее соотношение

$$d \geq 2\nu + \mu \tag{3.26}$$

Последнее неравенство, на уровне интуиции, означает, что исправление ошибок требует вдвое больше усилий (и избыточности), чем исправление стираний, поскольку позиции стираний известны.

Для двоичных линейных кодов, включая коды БЧХ, стирания и ошибки могут быть исправлены следующим ниже методом.

1. Записать нули на стертые позиции и декодировать полученное слово в кодовое слово обозначенное $v_0(x)$.
2. Записать единицы на стертые позиции и декодировать в кодовое слово $v_1(x)$.
3. Выбрать из $v_0(x)$ и $v_1(x)$ в качестве результата декодирования кодовое слово, ближайшее к принятому слову по нестертым позициям. Иначе говоря, в качестве результата декодирования выбирается слово, потребовавшее наименьшего числа исправлений [ML].

Таким образом, любая допустимая по условию (3.26) комбинация стираний и ошибок исправляется за две попытки исправления ошибок.

Пример 38. Рассмотрим циклический (7, 4, 3) код Хемминга с порождающим многочленом $g(x) = 1 + x + x^3$ и конечное поле $\mathbf{GF}(2^3)$ с примитивным элементом α, удовлетворяющим условию $p(\alpha) = 1 + \alpha + \alpha^3$. Предположим, что передано слово $v(x) = x + x^2 + x^4$ и что приемник ввел два стирания. Так как $d = 3 > \mu$, то эти стирания могут быть исправлены. Пусть $r(x) = f + x + x^2 + fx^3 + x^4$ полином, ассоциированный с принятым словом, где через f обозначены стирания.

Первая попытка: декодируем принятое слово при $f = 0$, $r_0(x) = x + x^2 + x^4$. Синдромы для него равны: $S_1 = r_0(\alpha) = 0$ и $S_2 = (S_1)^2 = 0$ Следовательно, $v_0(x) = r_0(x) = x + x^2 + x^4$.

Вторая попытка: теперь декодируем при f=1, $r_1(x) = 1 + x + x^2 + x^3 + x^4$, $S_1 = \alpha$ и $S_2 = \alpha^2$. Уравнение (3.16) имеет в этом случае вид: $\alpha\sigma_1 = \alpha^2$. В результате получаем $\sigma(x) = 1 + \alpha x$, $e(x) = x$ и кодовое слово $v_1(x) = r_1(x) + e(x) = 1 + x^2 + x^3 + x^4$, которое отличается от принятого слова $r(x)$ в одной нестертой позиции. В качестве результата декодирования выбираем слово $v_0(x)$. Таким образом, исправлены два стирания.

Следует напомнить, что для исправления только стираний линейным кодом (при точном отсутствии ошибок) достаточно найти решение системы линейных уравнений $\mathbf{rH}^{\top}=\mathbf{0}$. Решение этой системы единственно, если число стираний меньше кодового расстояния. Кроме того, этим методом можно исправить многие из комбинаций с числом стираний *больше кодового расстояния* (но не более числа проверок), хотя решение может быть не всегда единственным.

3.6. Распределение весов и границы вероятности ошибки

В общем случае, распределение весов двоичного линейного (n,k) кода C может быть найдено перечислением всех 2^k кодовых слов и подсчетом их веса Хемминга. Очевидно, что при больших k это нереализуемая затея. Обозначим A_w число кодовых слов веса Хемминга w. *Тождество Мак-Вильямс* связывает *распределение весов* линейного кода, $A(x) = A_0 + A_1x + A_2x^2 + ... + A_nx^n$, с распределением весов *дуального*[10] ему $(n, n\text{-}k)$ кода $C^{\perp}$, $B(x) = B_0 + B_1x + B_2x^2 + ... + B_nx^n$, следующим соотношением:

$$A(x) = 2^{-n+k}(1+x)^n B\left[\frac{1-x}{1+x}\right] \tag{3.27}$$

которое может быть записано и в обращенной форме

$$B(x) = 2^{-k}(1+x)^n A\left[\frac{1-x}{1+x}\right] \tag{3.28}$$

[10] Напомним, что порождающей матрицей дуального кода $C^{\perp}$ является проверочная матрица кода C

Таким образом, для кодов с высокой скоростью проще сначала подсчитать распределение весов дуального кода, а затем вычислить $A(x)$ по формуле (3.27).

Для некоторых классов кодов может быть использована решетчатая структура кода[11]. В работе [DFK2] представлен метод вычисления распределения весов *расширенного* БЧХ кода длины 128, основанный на треллисном представлении кода. Для кодов меньшей длины распределение весов нетрудно посчитать с помощью тождества Мак-Вильямс. Далее вводится определение *расширенного циклического кода*. Двоичный расширенный циклический код получается из циклического кода добавлением, в начале (или, наоборот, в конце) каждого кодового слова, *общей проверки на четность*. Расширенный код уже не является циклическим. Пусть **H** проверочная матрица циклического кода, тогда проверочная матрица $\mathbf{H}_{ext}$ расширенного кода равна

$$\mathbf{H}_{ext} = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 0 & & & \\ 0 & & \mathbf{H} & \\ 0 & & & \end{pmatrix} \qquad (3.29)$$

В приложении А дано распределение весов всех расширенных двоичных БЧХ кодов длины до 128. Эти данные доступны и на ECC веб сайте. В приложении даны распределения весов, не превышающих $(n+1)/2$, $n=2^m-1$. Для расширенных БЧХ кодов известно, что $A^{(\mathrm{ext})}{}_{n+1-w} = A^{(\mathrm{ext})}{}_{w}$. Эти данные полезны для нахождения распределения весов двоичного циклического БЧХ кода длины до 127. Это делается с помощью следующего результата (который является специальным случаем Теорем 8.14 и 8.15 в [PW]):

Пусть C двоичный циклический (n,k) БЧХ код с распределением весов $A(x)$, полученный исключением общей проверки из двоичного расширенного $(n+1, k)$ с распределением весов $A^{(ext)}(x)$. Тогда для четного веса w справедливо соотношение:

[11] Материал по кодовым решеткам можно найти в Главе 7.

$$(n+1)A_{w-1} = wA_w^{(ext)},$$
$$wA_w = (n+1-w)A_{w-1} \qquad (3.30)$$

Пример 39. Рассмотрим двоичный расширенный (8,4,4) код Хемминга. Этот код имеет проверочную матрицу (см. также Пример 23)

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

Легко проверить, что $A^{(ext)}(x) = 1 + 14x^4 + x^8$. Для распределения весов двоичного циклического (7, 4, 3) кода Хемминга получаем с помощью (3.30)

$$8A_3 = 4A_4^{(ext)} \rightarrow A_3 = 7,$$
$$4A_4 = (8-4)A_3 \rightarrow A_4 = 7,$$
$$8A_7 = 8A_8^{(ext)} \rightarrow A_7 = 1.$$

3.6.1. Оценка вероятности ошибки

Известный спектр весов кода позволяет оценить его вероятности ошибки, как это обсуждалось в Главе 1. Распределение весов и граница объединения (1.34) дают хорошую оценку вероятности ошибки кода при передаче двоичных сигналов по каналу с АБГШ.

В качестве примера граница объединения была посчитана с помощью данных Приложения А для расширенных кодов БЧХ длины от 8 до 64. Результаты представлены на Рисунках 24 – 27. На Рисунке 28 представлена граница объединения (1.40) для канала с общими Релеевскими замираниями и распределения весов из приложения А для БЧХ кода длины 8. Граница посчитана с помощью метода Монте-Карло при замене коэффициента A_w на wA_w/n чтобы учесть двоичные ошибки. Границы для других кодов могут быть получены аналогичным способом.

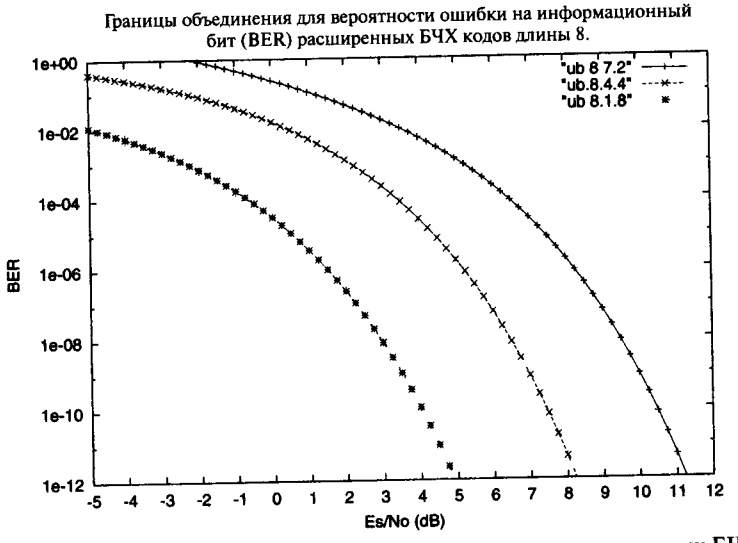


Рис. 24. Оценки BER по границе объединения для расширенных БЧХ кодов длины 8. Двоичный канал с АБГШ.

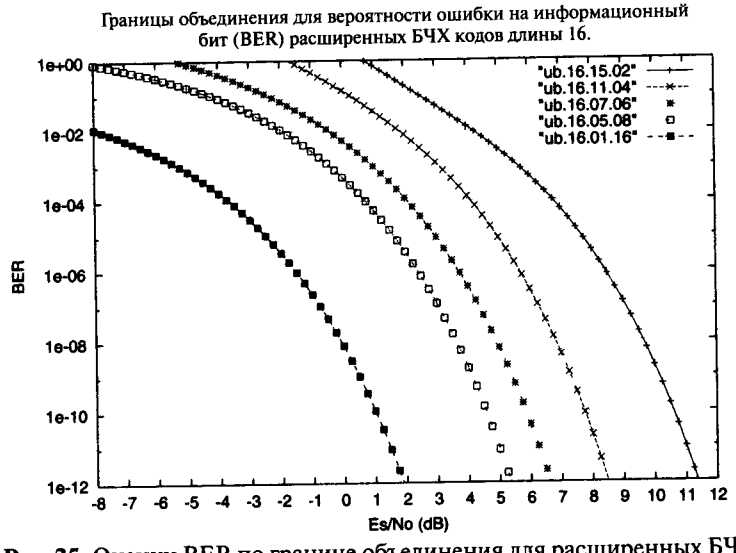


Рис. 25. Оценки BER по границе объединения для расширенных БЧХ кодов длины 16. Двоичный канал с АБГШ.

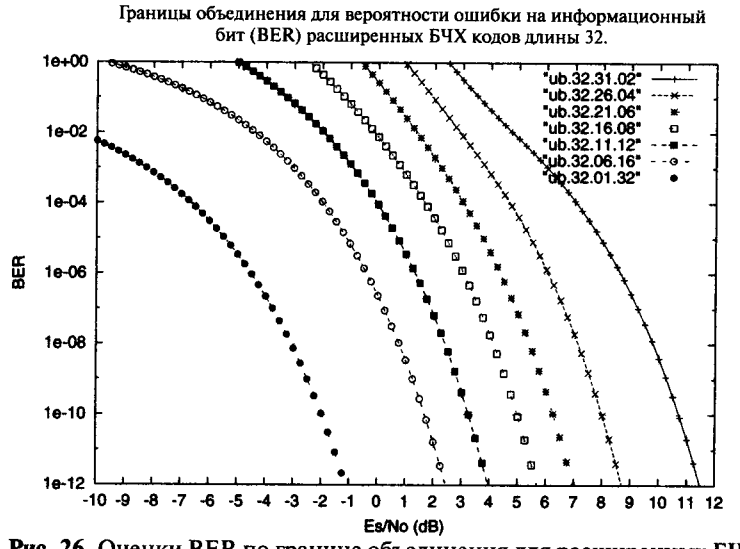


Рис. 26. Оценки BER по границе объединения для расширенных БЧХ кодов длины 32. Двоичный канал с АБГШ.

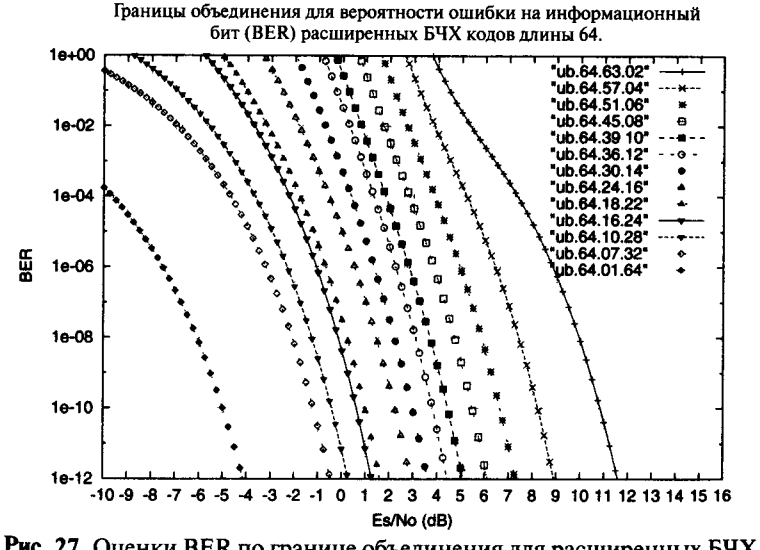


Рис. 27. Оценки BER по границе объединения для расширенных БЧХ кодов длины 64. Двоичный канал с АБГШ.

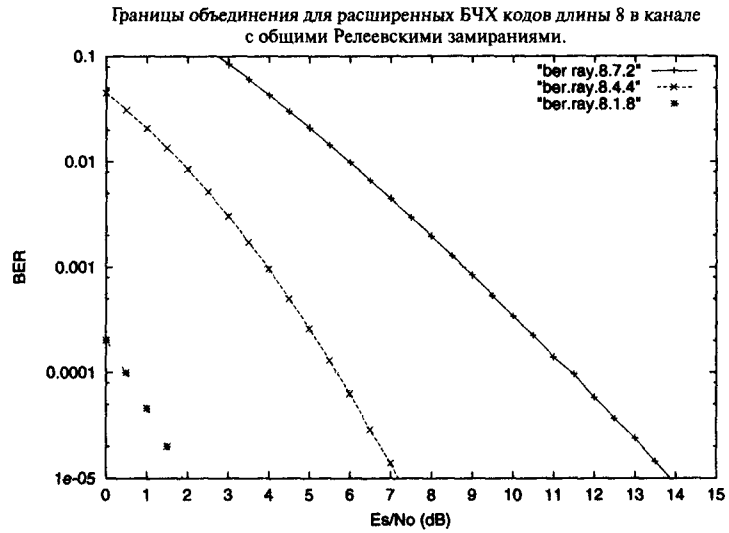


Рис. 28. Оценки BER по границе объединения для расширенных БЧХ кодов длины 8. Двоичный канал с общими Релеевскими замираниями.

Главный вывод этого раздела состоит в следующем. Прежде чем выбирать конкретный алгоритм мягкого декодирования (см. Главу 7) имеет смысл использовать распределение весов кода и границу объединения для предварительной оценки эффективности, достижимой в заданном канале. Для каналов с АБГШ и Релеевскими замираниями граница объединения дает точный ответ для вероятности ошибки на бит менее 10^{-4}.

Глава 4

НЕДВОИЧНЫЕ БЧХ КОДЫ – КОДЫ РИДА-СОЛОМОНА

В этой главе вводятся наиболее известные кодовые конструкции и объясняются алгоритмы их кодирования и декодирования. Коды Рида-Соломона (РС) нашли множество применений в системах цифровой памяти и связи. В качестве примеров упомянем знаменитый (255,223,33) код в системах космической связи НАСА (NASA), укороченные коды РС над $\mathbf{GF}(2^8)$ в системах цифровой записи «компакт-диск» CD-ROM и DVD, а также в наземных цифровых системах HDTV (высокоточное ТВ), расширенные (128,122,7) коды РС над $\mathbf{GF}(2^7)$ для модемов на кабельных линиях, среди многих и многих других.

4.1. Коды РС как полиномиальные коды

Аналогично кодам Рида-Маллера коды РС можно определить как множество кодовых слов, компоненты которых равны значениям некоторых определенных многочленов. В действительности, это определение РС кодов принадлежит Риду и Соломону [RS]. Коды Рида-Маллера, конечно-геометрические коды [LC] и коды РС являются членами большого класса *Полиномиальных кодов* [PW] и тесно связаны с классом *алгебро-геометрических (AG) кодов* [Pre]. Обозначим

$$u(x) = u_0 + u_1 x + \cdots u_{k-1} x^{k-1} \tag{4.1}$$

информационный полином с коэффициентами $u_i \in \mathbf{GF}(2^m)$, $10 \le i < k$. Очевидно, что всего имеется 2^{mk} таких многочленов. Вычисляя значения многочлена (4.1) для ненулевых элементов поля $\mathbf{GF}(2^m)$, получаем кодовое слово $\mathbf{v}$ кода РС с параметрами $(2^m\text{-}1, k, d)$

$$\mathbf{v} = (u(1), u(\alpha), u(\alpha^2), \cdots, u(\alpha^{2^m-2})) \qquad (4.2)$$

4.2. От двоичных кодов БЧХ к РС кодам

Коды РС можно также интерпретировать как *недвоичные* коды БЧХ. Можно сказать, что РС коды являются кодами БЧХ, значения кодовых символов которых взяты из поля **GF**(2^m). В частности, нулями РС кода, исправляющего t_d ошибок, являются $2t_d$ последовательных степеней примитивного элемента поля Галуа. Более того, так как над полем **GF**(2^m) минимальные многочлены имеют вид $\phi_i(x) = (x - \alpha^i)$, $0 \le i < 2^m - 1$, (см. уравнение (3.9)) все делители порождающего код многочлена являются *линейными* (т.е. имеют степень 1) и

$$g(x) = \prod_{j=b}^{b+2t_d-1} (x - \alpha^j) \qquad (4.3)$$

где b целое число, обычно 0 или 1.

Из (4.3) и границы БЧХ следует, что минимальное кодовое расстояние (n,k,d) РС кода над **GF**(2^m) удовлетворяет неравенству $d \ge n - k + 1$. Из границы Синглтона [Sin], утверждающей, что $d \le n - k + 1$, следует, что $d = n - k + 1$. Коды, удовлетворяющие последнему равенству, называют МДР кодами (кодами с *максимальным достижимым расстоянием*) [Sin]. Таким образом, РС код является МДР кодом. Из этого следуют полезные свойства кодов РС. Одно из них состоит в том, что укороченные коды РС являются МДР кодами.

Изоморфизм между **GF**(2^m) и $\{0,1\}^m$ означает, что любому двоичному m – вектору $\mathbf{x}_\beta$ может быть поставлен в соответствие элемент $\beta \in$ **GF**(2^m),

$$m - \text{бит} \Leftrightarrow \beta_j \in \mathbf{GF}(2^m),\ 0 \le j < 2^m - 1$$

Другими словами, m информационных бит можно сгруппировать в блок, образующий символ **GF**(2^m). Если элементы **GF**(2^m) рассматривать как векторы из m бит, то из кода РС получаем двоичный линейный код длины $n = m(2^m - 1)$ и размерности $k = m(2^m - 1 - 2t_d)$. Минимальное расстояние такого кода не меньше $2t_d$. Такое *двоичное отображение* кода РС позволяет исправлять не только до t_d случайных (двоичных) ошибок, но и многократные *случайные пакеты ошибок*. Например, может быть исправлен любой однократный пакет ошибок длиной до $m(t_d - 1) + 1$ бит. Это следует из того, что пакет ошибок длины $m(q - 1) + 1$ или меньше покрывается не более q символами **GF**(2^m). Таким образом, коды РС способны исправлять многие комбинации случайных ошибок и пакетов ошибок. В этом основная причина огромной популярности кодов РС в практических системах.

Пример 40. Пусть m=3 и поле **GF**(2^3) генерируется примитивным элементом α, удовлетворяющим условию $p(\alpha) = \alpha^3 + \alpha + 1 = 0$. Пусть $b = 0$ и $t_d = 2$. Тогда имеется (7,3,5) код РС с порождающим полиномом

$$g(x) = (x+1)(x+\alpha)(x+\alpha^2)(x+\alpha^3) = x^4 + \alpha^2 x^3 + \alpha^5 x^2 + \alpha^5 x + \alpha^6$$

Отображением символов **GF**(2^3) в вектора длины 3 получаем двоичный (21,9,5) код, способный исправлять до двух случайных ошибок и любой пакет до 4-х ошибок.

4.3. Декодирование кодов РС

Алгоритмы декодирования кодов РС весьма близки к алгоритмам для двоичных БЧХ кодов. Существенное различие относится только к вычислению $\nu \le t_d$ *значений ошибок* e_{j_ℓ}, $1 \le \ell \le \nu$. В общем случае это делается с помощью *алгоритма Форни* [For2]. Ниже представлено выражение, справедливое для кодов РС с произвольным множеством $2t_d$ последовательных нулей $\{\alpha^b, \alpha^{b+1}, \ldots, \alpha^{b+\delta}\}$, $\delta = 2t_d - 1$,

$$e_{j_\ell} = \frac{(\alpha^{j_\ell})^{2-b}\,\Lambda(\alpha^{-j_\ell})}{\sigma'(\alpha^{-j_\ell})} \qquad (4.4)$$

где $\sigma'(x)$ есть формальная производная по x многочлена локаторов ошибок $\sigma(x)$. (Аналогичное выражение имеется в [RC],

стр. 276) Многочлен $\Lambda(x)$ в (4.4) это *многочлен значений ошибок*, определяемый как

$$\Lambda(x)=\sigma(x)S(x)\bmod x^{2t_d+1}. \qquad (4.5)$$

Перед изучением первого примера декодирования кода РС мы рассмотрим другой вариант алгоритма БМ, известный как *алгоритм Мэсси* (или МА). Этот алгоритм был предложен Дж. Мэсси в [Mas2], а также рассматривается в [ML, Wic].

Алгоритм Мэсси синтеза ЛРОС

1. Начальные условия: многочлен обратной связи ЛРОС $\sigma(x) = 1$, корректор $\rho(x) = x$, счетчик $i = 1$, длина регистра $\ell = 0$.
2. Взять следующий синдром и вычислить различие:
$$d = S_i + \sum_{j=1}^{\ell} \sigma_j S_{i-j}$$
3. Проверить различие d: если d=0, то перейти к п. 8.
4. Модифицировать многочлен обратной связи:
$$\sigma_{new}(x)=\sigma(x)-d\rho(x)$$
5. Проверить длину регистра: если $\ell \ge i$, то перейти к п. 7.
6. Исправить длину регистра и заменить корректор: положить $\ell = i - \ell$ и $\rho(x) = \sigma(x)/d$.
7. Обновить многочлен обратной связи: $\sigma(x) = \sigma_{new}(x)$.
8. Обновить корректор: $\rho(x) = x\rho(x)$.
9. Обновить счетчик: $i = i + 1$.
10. Условие остановки: если $i < d$, то перейти к п. 2, иначе Стоп.

Пример 41. Пусть C код РС (7,3,5) из Примера 40. Положим, что многочлен

$$r(x)=\alpha x^2+\alpha^5 x^4$$

получен из канала связи. Тогда $S_1 = r(1) = \alpha + \alpha^5 = \alpha^6$, $S_2 = r(\alpha) = \alpha^3 + \alpha^2 = \alpha^5$, $S_3 = r(\alpha^2) = \alpha^5 + \alpha^6 = \alpha$ и $S_4 = r(\alpha^3) = 1 + \alpha^3 = \alpha$. Уравнение (3.16) получаем в виде:

$$\begin{pmatrix}\alpha^6 & \alpha^5\\ \alpha^5 & \alpha\end{pmatrix}\begin{pmatrix}\sigma_2\\ \sigma_1\end{pmatrix}=\begin{pmatrix}\alpha\\ \alpha\end{pmatrix}$$

Ниже рассмотрены три способа решения последнего уравнения.

Прямое решение (PGZ алгоритм)

Предположим, что произошли две ошибки. Тогда $\Delta_2 = \alpha^7 + \alpha^{10} = 1 + \alpha^3 = \alpha \neq 0$. Следовательно, должны были произойти две ошибки и

$$\sigma_2 = \alpha^6 \det\begin{pmatrix}\alpha & \alpha^5\\ \alpha & \alpha\end{pmatrix}= \alpha^6$$

$$\sigma_1 = \alpha^6 \det\begin{pmatrix}\alpha^6 & \alpha\\ \alpha^5 & \alpha\end{pmatrix}= \alpha$$

откуда следует, что

$$\sigma(x)=1+\alpha x+\alpha^6 x^2=(1+\alpha^2 x)(1+\alpha^4 x)$$

Алгоритм Мэсси

$$S_1=\alpha^6, S_2=\alpha^5, S_3=\alpha, S_4=\alpha$$

- $i=0$: $\sigma(x)=1, \ell=0, \rho(x)=x$

- $i=0$: $\sigma(x)=1, \ell=0, \rho(x)=x$

$\sigma_{new}(x)=\sigma(x)+d\rho(x)=1+\alpha^6 x$, $2l=0<i$, $\ell=i-\ell=1$,
$\rho(x)=\sigma(x)/d=\alpha^{-6}=\alpha$,
$\rho(x)=x\rho(x)=\alpha x$, $\sigma(x)=\sigma_{new}(x)$.

- $i=2$: $d=S_2+\sum_{j=1}^{1}\sigma_j S_{2-j}=\alpha^5+\alpha^6\alpha^6=0$, $\rho(x)=x\rho(x)=\alpha x^2$

$$\sigma_{new}(x) = \sigma(x) + d\rho(x) = 1 + \alpha^6 x + \alpha^3 x^2,$$
$$2l = 2 < i,\ \ell = i - \ell = 2,$$
$$\rho(x) = \sigma(x)/d = \alpha^5 + \alpha^4 x, \quad \rho(x) = x\rho(x) = \alpha^5 x + \alpha^4 x^2,$$
$$\sigma(x) = \sigma_{new}(x).$$

- $i = 4$: $d = S_4 + \sum_{j=1}^{2} \sigma_j S_{4-j} = \alpha + \alpha^6 \alpha^5 = \alpha^2,$

$$\sigma_{new}(x) = \sigma(x) + d\rho(x) = 1 + \alpha^6 x + \alpha^3 x^2 + (1)(\alpha^5 x + \alpha^4 x^2) = 1 + \alpha x + \alpha^6 x^2,$$
$$2\ell \geq 4,$$
$$\rho(x) = x\rho(x) = \alpha^5 x^2 + \alpha^4 x^3, \quad \sigma(x) = \sigma_{new}(x).$$

- $i = 5 > d$. **Стоп.**

Евклидов алгоритм

- Начальные условия:

$$r_0(x) = x^5,\ r_1(x) = S(x) = 1 + \alpha^6 x + \alpha^5 x^2 + \alpha x^3 + \alpha x^4,$$
$$b_0(x) = 0,\ b_1(x) = 1.$$

- $i = 2$

$$x^5 = (1 + \alpha^6 x + \alpha^5 x^2 + \alpha x^3 + \alpha x^4)(\alpha^6 x + \alpha^6) + \alpha^5 x + x^2 + \alpha x + \alpha^6.$$
$$r_2(x) = \alpha^5 x^3 + x^2 + \alpha x + \alpha^6,$$
$$q_2(x) = \alpha^6 x + \alpha^6,$$
$$b_2(x) = 0 + (\alpha^6 x + \alpha^6)(1) = \alpha^6 x + \alpha^6.$$

- $i = 2$

$$1 + \alpha^6 x + \alpha^5 x^2 + \alpha x^3 + \alpha x^4 =$$
$$= (\alpha^5 x^3 + x^2 + \alpha x + \alpha^6)(\alpha^3 x + \alpha^2) + \alpha^6 x^2 + \alpha x + \alpha^3.$$
$$r_3(x) = \alpha^6 x^2 + \alpha x + \alpha^3,$$
$$q_3(x) = \alpha^3 x + \alpha^2,$$
$$b_3(x) = 1 + (\alpha^3 x + \alpha^2)(\alpha^6 x + \alpha^6) = \alpha^3 + \alpha^4 x + \alpha^2 x^2.$$

Алгоритм останавливается, так как $\deg[r_3(x)] = 2 = t_d$.
Следовательно, $\sigma(x) = \alpha^3 + \alpha^4 x + \alpha^2 x^2 = \alpha^3(1 + \alpha x + \alpha^6 x^2)$.

Все рассмотренные алгоритмы дали одинаковое, с точностью до константного множителя, решение для многочлена локаторов ошибок:

$$\sigma(x) = 1 + \alpha x + \alpha^6 x^2 = (1 + \alpha^2 x)(1 + \alpha^4 x)$$

Таким образом, ошибки произошли на позициях $j_1 = 2$ и $j_2 = 4$. Координаты ошибок находятся с помощью процедуры Ченя как обратные величины корней многочлена локаторов ошибок $\sigma(x)$. Заметим, что $\sigma'(x) = 0$, так как в поле **GF**(2^m) характеристики 2 справедливо равенство $2a = a + a = 0$.

Для вычисления значений ошибок (любым из рассмотренных алгоритмов) необходим многочлен значений ошибок (4.5),

$$\Lambda(x) = (1 + \alpha x + \alpha^6 x^2)(1 + \alpha^6 x + \alpha^5 x^2 + \alpha x^4 + \alpha x^4) \bmod x^5 =$$
$$= (1 + \alpha^5 x + \alpha^3 x^2) \bmod x^5$$

Важно отметить, что алгоритм Евклида вычисляет $\sigma(x)$ и $\Lambda(x)$ *одновременно*, как $\sigma(x) = b_{last}(x)$ и $\Lambda(x) = r_{last}(x)$ (см. раздел 3.5.4, ЕА). Для проверки этого утверждения заметим, что

$$r_3(x) = \alpha^3 + \alpha x + \alpha^6 x^2 = \alpha^3(1 + \alpha^5 x + \alpha^3 x^2) = \alpha^3 \Lambda(x)$$

Если локаторы ошибок известны, то значения ошибок, найденные из уравнения (4.4), равны

$$e_2 = (\alpha^2)(1 + \alpha^5 \alpha^{-2} + \alpha^3 \alpha^{-4})\alpha^{-1} = \alpha,$$
$$e_4 = (\alpha^4)(1 + \alpha^5 \alpha^{-4} + \alpha^3 \alpha^{-8})\alpha^{-1} = \alpha^5.$$

Таким образом, $e(x) = \alpha x^2 + \alpha^5 x^4$ и декодированное слово равно

$$\hat{c}(x) = r(x) + e(x) = \mathbf{0}$$

Исправлены две ошибки.

Заметим, что оба полинома, найденные алгоритмом Евклида, имеют одинаковый константный множитель β, т.е. алгоритм находит $\beta\sigma(x)$ и $\beta\Lambda(x)$ с некоторым множителем $\beta \in$ **GF**(2^m). Тем не менее, оба полинома имеют *те же самые корни*, что и полиномы, вычисленные алгоритмами БМА или PGZ. В результате и значения ошибок совпадают.

В большинстве программ, моделирующих процедуры кодирования и декодирования РС кодов на ECC веб-сайте, используется следующий эквивалентный способ определения значений ошибок [LC]. Пусть

$$z(x) = 1 + (S_1 + \sigma_1)x + (S_2 + \sigma_1 S_1 + \sigma_2)x^2 + \\ + \ldots + (S_\nu + \sigma_1 S_{\nu-1} + \ldots + \sigma_\nu)x^\nu \qquad (4.6)$$

Тогда значения ошибок равны [Ber]

$$e_{j_\ell} = \frac{(\alpha^{j_\ell})^{1-b} z(\alpha^{j_\ell})}{\prod_{i=1, i\neq\ell}^{\nu}(1 + \alpha^{j_i - j_\ell})} \qquad (4.7)$$

где $1 \le \ell \le \nu$.

Еще одна альтернатива алгоритму Форни, для малых значений t_d, состоит в следующем способе вычисления значений ошибок. Значения ошибок e_{jl}, $1 \le l \le \nu$, связаны с синдромами S_i системой линейных уравнений:

$$S_i = e(\alpha^{b+i-1}) = \sum_{\ell=1}^{\nu} e_{j_\ell} \alpha^{(b+i-1)j_\ell} \qquad (4.8)$$

где $1 \le i \le 2t_d$.

Каждая $\nu \times \nu$ подматрица, образованная (известными) членами $\alpha^{(b+i-1)j_\ell}$ образуют матрицу Вандермонда. Если известно положение всех ν ошибок, то любое подмножество ν уравнений (4.8) может быть использовано для определения значений ошибок. Например, используя первые ν синдромов, получаем следующую систему линейных уравнений,

$$\begin{pmatrix} S_1 \\ S_2 \\ \vdots \\ S_\nu \end{pmatrix} = \begin{pmatrix} (\alpha^{j_1})^b & (\alpha^{j_2})^b & \cdots & (\alpha^{j_\nu})^b \\ (\alpha^{j_1})^{b+1} & (\alpha^{j_2})^{b+1} & \cdots & (\alpha^{j_\nu})^{b+1} \\ \vdots & \vdots & \ddots & \vdots \\ (\alpha^{j_1})^{b+\nu-1} & (\alpha^{j_2})^{b+\nu-1} & \cdots & (\alpha^{j_{\nu 1}})^{b+\nu-1} \end{pmatrix} \begin{pmatrix} e_{j_1} \\ e_{j_2} \\ \vdots \\ e_{j_\nu} \end{pmatrix} \qquad (4.9)$$

решение которой может быть найдено над полем $\mathbf{GF}(2^m)$.

Пример 42. Рассмотрим тот же код РС и принятый многочлен из Примеров 40 и 41. Из (4.9) получаем

$$\begin{pmatrix} \alpha^6 \\ \alpha^5 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ \alpha^2 & \alpha^4 \end{pmatrix} \begin{pmatrix} e_2 \\ e_4 \end{pmatrix}$$

Определитель 2x2 матрицы равен $\Delta = \alpha^4 + \alpha^2 = \alpha$. Отсюда следует, что

$$e_2 = \alpha^{-1} \det \begin{pmatrix} \alpha^6 & 1 \\ \alpha^5 & \alpha^4 \end{pmatrix} = \alpha^6(\alpha^3 + \alpha^5) = \alpha^6\alpha^2 = \alpha$$

и

$$e_4 = \alpha^{-1} \det \begin{pmatrix} 1 & \alpha^6 \\ \alpha^2 & \alpha^5 \end{pmatrix} = \alpha^6(\alpha^5 + \alpha) = \alpha^6\alpha^6 = \alpha^5$$

Полученные значения ошибок совпадают с теми, которые вычислены по алгоритму Форни. Подчеркнем еще раз, что этот способ может быть применен при исправлении только очень небольшого числа ошибок.

4.3.1. Комментарий к алгоритмам декодирования

В отличие от БМА алгоритм Евклида (ЕА) использует все синдромы уже на первом шаге процедуры. Однако, по числу операций в конечном поле алгоритм Берлекэмпа-Мэсси (БМА) обычно эффективнее ЕА. С другой стороны, каждый шаг ЕА имеет идентичную структуру, что приводит к более эффективной аппаратной (микросхемной) реализации. Кроме того, три рассмотренных метода декодирования для (двоичных и недвоичных) кодов БЧХ являются неполными, так как исправляют ограниченное число ошибок – *декодирование с ограниченным расстоянием*. Вследствие этого рассмотренные алгоритмы способны

обнаруживать ситуации, которые превышают их корректирующую способность.

Дополнение переводчика. *Рассмотренные алгоритмы исправляют все комбинации ошибок ограниченного веса (до половины кодового расстояния). Многие из комбинаций ошибок большего веса не могут быть исправлены этими алгоритмами, но обнаруживаются и приводят к отказу от декодирования. В зависимости от конкретного применения кода в этом случае, либо выдаются принятые из канала информационные символы систематического кода, либо все слово заменяется символом (сигналом) стирания. Обнаружение неисправимых комбинаций ошибок происходит, либо в процедуре Ченя, когда хотя бы один из корней многочлена локаторов ошибок не принадлежит множеству локаторов позиций кодового слова, либо на этапе вычисления синдромов, когда количество нулевых синдромов оказывается больше или равным половине кодового расстояния.*

Известны также и другие подходы к декодированию кодов БЧХ, из которых наиболее значительным является использование быстрых алгоритмов дискретного преобразования Фурье над полем $\mathbf{GF}(2^m)$. Этот подход к созданию быстрых алгоритмов декодирования исследуется в [Blah], где читатель найдет необходимые подробности.

Дополнение переводчика. *Рассмотренные выше алгоритмы принято называть декодированием в частотной области, учитывая эквивалентность между вычислением синдрома и преобразованием Фурье над конечным полем. В [Blah, Blah1*] рассматриваются и другой подход к построению декодеров – декодирование во временной области. Этот вариант декодеров РС кодов требует большего количества операций в поле, но считается более удобным для аппаратной (микросхемной) реализации.*

Сравнительно недавно Судан [Sud] предложил (алгебраический) алгоритм, исправляющий ошибки за границей минимального расстояния кода. Этот алгоритм применяется к декодированию РС и, главным образом, АГ кодов. Выходом этого алгоритма является список (ближайших в некоторой метрике) кодовых слов. Алгоритм (*декодирования в список*) основан на процедурах интерполяции и факторизации многочленов от двух переменных над $\mathbf{GF}(2^m)$ и его расширением. Оригинальный алгоритм Судана был позднее улучшен в работе [Gur] и других. Важно, что на основе алгоритма Судана создан алгебраический алгоритм мягкого декодирования кодов РС и близких к ним АГ кодов.

4.3.2. Исправление ошибок и стираний

Для исправления стираний необходимое изменение в процедуре декодирования кодов РС, рассмотренной выше, состоит во введении *многочлена локаторов стираний* $\tau(x)$,

$$\tau(x)=\prod_{\ell=1}^{\mu}(1+y_{J_\ell}x)$$

где $y_{i_\ell}=\alpha^{i_\ell}$, локаторы стираний, $1\le\ell\le\mu$.

По определению, позиции со стираниями известны. Следовательно, требуется определить только *значения стертых позиций*. Это может быть выполнено, как и раньше, по алгоритму Форни. Можно показать, что в процессе вычисления синдромов на стертых позициях могут быть подставлены произвольные значения символов, так как их значения не влияют на результат декодирования.

Процедура декодирования РС с исправлением стираний и ошибок аналогична исправлению только ошибок с учетом следующих ниже модификаций. Прежде всего, формируется модифицированный синдромный многочлен, или *модифицированный синдром Форни*,

$$T(x)=S(x)\tau(x)+1 \bmod x^{2t^{d+1}} \qquad (4.10)$$

Алгоритм БМ применяется для определения многочлена локаторов ошибок с учетом следующих изменений:

1. Рассогласование определяется теперь как

$$d_i = T_{i+\mu+1} + \sum_{j=1}^{\ell_i} \sigma_j^{(i)} S_{i+\mu+1-i} \tag{4.11}$$

где $d_0 = T_{\mu+1}$.

2. Условие остановки алгоритма заменяется следующим:

$$i \geq \ell_{i+1} + t_d - 1 - \mu/2$$

По окончании вычисления $\sigma(x)$ формируется *модифицированный многочлен значений ошибок и стираний*, или многочлен значений искажений, равный по определению

$$\omega(x) = (1 + T(x))\sigma(x) \bmod x^{2t_d+1} \tag{4.12}$$

Кроме того, вычисляется следующий *многочлен локаторов искажений*,

$$\phi(x) = \tau(x)\sigma(x) \tag{4.13}$$

С помощью *модифицированного алгоритма Форни* получаем для значений ошибок:

$$e_{j_\ell} = \frac{(\alpha^{j_\ell})^{2-b}\omega(\alpha^{-j_\ell})}{\phi'(\alpha^{-j_\ell})}, \quad 1 \leq \ell \leq \nu, \tag{4.14}$$

и для значений стираний:

$$f_{i_\ell} = \frac{(y_{i_\ell})^{2-b}\omega(y_{i_\ell}^{-1})}{\phi'(y_{i_\ell}^{-1})}, \quad 1 \leq \ell \leq \mu \tag{4.15}$$

Алгоритм Евклида также может быть применен к модифицированному синдрому $T(x)$ для исправления стираний и ошибок. Для этого в алгоритме исправления только ошибок многочлен $S(x)$ заменяется многочленом $1+T(x)$, а в начальных условиях $r_0(x) = x^d$, $d = 2t_d+1$, и $r_1(x) = 1 + T(x)$. Алгоритм останавливается на j-ом шаге, если $\deg[r_j(x)] \leq \lfloor (d - 1 + \mu)/2 \rfloor$, и выдает как решение $\omega(x) = r_j(x)$ и $\sigma(x) = b_j(x)$.

Пример 43. Пусть C это (15,9,7) код РС над **GF**(2^4) с нулями в точках $\{\alpha, \alpha^2, \ldots, \alpha^6\}$, где примитивный элемент поля

удовлетворяет равенству $p(\alpha) = \alpha^4 + \alpha^3 + 1 = 0$. Для удобства ниже представлена таблица элементов **GF**(2^m) , упорядоченная по степеням примитивного элемента α , $\alpha^4 + \alpha^3 + 1 = 0$.

Таблица элементов **GF**(2^m), $p(x) = x^4+x^3+1$.

Степень	0	1	α	α^2	α^3	α^4	α^5	α^6
Вектор	0000	0001	0010	0100	1000	1001	1011	1111
Степень	α^7	α^8	α^9	α^{10}	α^{11}	α^{12}	α^{13}	α^{14}
Вектор	0111	1110	0101	1010	1101	0011	0110	1100

Порождающий многочлен кода C равен

$$g(x) = \prod_{i=1}^{6}(x + \alpha^i) = x^6 + \alpha^{12}x^5 + x^4 + \alpha^2 x^3 + \alpha^7 x^2 + \alpha^{11}x + \alpha^6$$

Положим, что ассоциированный с кодовым словом полином равен

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

Пусть принятое слово представлено полиномом

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

Предположим, что приемник располагает дополнительной информацией о том, что позиции с локаторами α^0 и α^5 ненадежны и заявлены как стирания.

Заметим, что многочлен искажений[1] имеет вид $e(x) = \alpha^{14} + \alpha^8 x^3 + \alpha^5 x^5 + \alpha x^{12}$.

Таким образом, кроме принятого слова $r(x)$ декодеру известны локаторы $\mu = 2$ двух стираний, α^0 и α^5. Это позволяет вычислить многочлен локаторов стираний

$$\tau(x) = (1 + x)(1 + \alpha^5 x) = 1 + \alpha^{10}x + \alpha^5 x^2 \tag{4.16}$$

[1] Очевидно, что декодеру этот многочлен не известен за исключением только позиций стираний. Этот полином используется здесь только для проверки правильности результата декодирования.

Синдромы для принятого слова равны:

$$S_1 = \mathbf{r}(\alpha) = \alpha^{11},\ S_2 = \mathbf{r}(\alpha^2) = \alpha^5,\ S_3 = \mathbf{r}(\alpha^3) = \alpha^2,$$
$$S_4 = \mathbf{r}(\alpha^4) = \alpha^2,\ S_5 = \mathbf{r}(\alpha^5) = 1,\ S_6 = \mathbf{r}(\alpha^6) = \alpha^{14},$$

Соответственно, синдромный многочлен равен:

$$S(x) = 1 + \alpha^{11}x + \alpha^5 x^2 + \alpha^2 x^3 + \alpha^2 x^4 + x^5 + \alpha^{14}x^6 \qquad (4.17)$$

Модифицированный синдром (4.10) равен

$$\begin{aligned} T(x) &= S(x)\tau(x) + 1 \bmod x^{2t_d+1} \\ &= (1 + \alpha^{11}x + \alpha^5 x^2 + \alpha^2 x^3 + \alpha^2 x^4 + x^5 + \alpha^{14}x^6) \times \\ &\times (1 + \alpha^{10}x + \alpha^5 x^2) + 1 \bmod x^7 \\ &= \alpha^7 x + \alpha^6 x^2 + \alpha^7 x^3 + \alpha^{11}x^4 + \alpha^9 x^5 + x^6. \end{aligned}$$

Алгоритм Берлекэмпа-Мэсси для исправления ошибок и стираний

• **Итерация 0:** Начальные установки.

$$\sigma^{(-1)}(x) = 1,\ \ell_{-1} = 0,\ d_{-1} = 1,\ \sigma^{(0)}(x) = 1,\ \ell_0 = 0,\ d_0 = T_3 = \alpha^7$$

• **Итерация 1:**

$$i = 0,\ d_0 = \alpha^7 \neq 0,\ m = -1 = \arg(\max(-1+0) = -1),\ d_{-1} \neq 0.$$

$$\begin{aligned} &\sigma^{(1)}(x) = \sigma^{(0)}(x) + d_0 d_{-1}^{-1} x^{(0-(-1))}\sigma^{(-1)}(x) = 1 + \alpha^7 x, \\ &\ell_1 = \max\{\ell_0, \ell_{-1} + 0 - (-1)\} = 1, \\ &i \geq \ell_1 + 1\text{? Нет:} \\ &d_1 = T_4 + T_3\sigma_1^{(1)} = \alpha^{11} + \alpha^7\alpha^7 = 1. \end{aligned}$$

• **Итерация 2:**

$$i = 1,\ m = 0 = \arg(\max(0-0) = 0),\ d_0 \neq 0$$

$$\begin{aligned} &\sigma^{(2)}(x) = \sigma^{(1)}(x) + d_1 d_0^{-1}\sigma^{(0)}(x) = 1 + \alpha^4 x, \\ &\ell_2 = \max\{1, 0+1-0\} = 1, \\ &i \geq 2\text{?} \quad \text{Нет:} \\ &d_2 = T_5 + T_4\sigma_1^2 = \alpha^9 + \alpha^{11}\alpha^4 = \alpha^2. \end{aligned}$$

• **Итерация 3:**

$$i = 2,\ m = 0 = \arg(\max(0-0) = 0),\ d_0 \neq 0.$$

$$\begin{aligned} &\sigma^{(3)}(x) = \sigma^{(2)}(x) + d_2 d_0^{-1}\sigma^{(0)}(x) = 1 + \alpha^4 x + \alpha^{10}x^2, \\ &\ell_3 = \max\{1, 0+2-0\} = 2, \\ &i \geq 3\text{?} \ \text{Нет:} \\ &d_3 = T_6 + T_5\sigma_1^{(3)} + T_4\sigma_2^{(3)} = 1 + \alpha^9\alpha^4 + \alpha^{11}\alpha^{10} = \alpha^3. \end{aligned}$$

• **Итерация 4:**

$$i = 3,\ m = 2 = \arg(\max(2-1) = 1),\ d_2 \neq 0.$$

$$\begin{aligned} &\sigma^4(x) = \sigma^{(3)}(x) + d_3 d_2^{-1}\sigma^{(2)}(x) = 1 + \alpha^5 x + x^2, \\ &\ell_4 = \max\{2, 1+3-2\} = 2, \\ &i \geq 2\text{?} \ \text{Да: Алгоритм остановлен.} \end{aligned}$$

Таким образом, получаем многочлен локаторов ошибок

$$\sigma(x) = 1 + \alpha^5 x + x^2 = (1 + \alpha^3 x)(1 + \alpha^{12} x)$$

Напомним, что в последнем равенстве с помощью процедуры Ченя найдены обратные величины локаторов ошибок в поле $\mathbf{GF}(2^m)$. Действительно, проверка дает $\sigma(\alpha^{12}) = 0$ и $\sigma(\alpha^3) = 0$. Следовательно, локаторами ошибок являются $\alpha^{-12} = \alpha^3$ и $\alpha^{-3} = \alpha^{12}$.

Из определения (4.12) находим модифицированный многочлен значений искажений

$$\begin{aligned} \omega(x) &= (1 + T(x))\sigma(x) \bmod x^{2t_d+1} \\ &= (1 + \alpha^7 x + \alpha^6 x^2 + \alpha^7 x^3 + \alpha^{11}x^4 + \alpha^9 x^5 + x^6)(1 + \alpha^5 x + x^2) \bmod x^7 \\ &= 1 + \alpha^{14}x + \alpha^{11}x^2 + \alpha^{11}x^3 + x^4. \end{aligned} \qquad (4.18)$$

Многочлен локаторов искажений равен

$$\phi(x)=\tau(x)\sigma(x)=(1+\alpha^{10}x+\alpha^5x)(1+\alpha^5x+x^2)=1+x+\alpha^5x^2+\alpha^5x^4$$

и, следовательно, $\phi'(x)=1$.

Значения ошибок и стираний могут быть вычислены по формулам (4.14) и (4.15), соответственно,

$$e_3=\alpha^3(1+\alpha^{11}+\alpha^5+\alpha^2+\alpha^3)=\alpha^3\alpha^5=\alpha^8,$$
$$e_{12}=\alpha^{12}(1+\alpha^2+\alpha^2+\alpha^5+\alpha^{12})=\alpha^{12}\alpha^4=\alpha,$$
$$f_0=(1+\alpha^{14}+\alpha^{11}+\alpha^{11}+1)=\alpha^{14},$$
$$f_5=\alpha^5(1+\alpha^9+\alpha+\alpha^{11}+\alpha^{10})=\alpha^5.$$

Отсюда следует, что *многочлен искажений* равен

$$e(x)=\alpha^{14}+\alpha^8x^3+\alpha^5x^5+\alpha x^{12}$$

Результат декодирования $\overline{v}(x)=r(x)+e(x)$ совпадает с переданным словом $v(x)$. Исправлены две ошибки и два стирания.

Прямое вычисление значений искажений

Для кодов РС с малым кодовым расстоянием значения стираний можно получить как решение системы линейных уравнений. Пусть $e(x)$ полином ошибок, ассоциированный с комбинацией ν ошибок и μ стираний,

$$e(x)=\sum_{\ell=1}^{\nu}e_{J_\ell}x^{J_\ell}+\sum_{\ell'=0}^{\mu}f_{J_{\ell'}}x^{J_{\ell'}} \qquad (4.19)$$

Тогда справедлива следующая система линейных уравнений, аналогичная (4.8), связывающая синдромы и значения искажений:

$$S_i=e(\alpha^{b+i})=\sum_{\ell=1}^{\nu}e_{J_\ell}\alpha^{(b+i)J_\ell}+\sum_{\ell'=1}^{\mu}f_{J_{\ell'}}\alpha^{(b+i)j_{\ell'}} \qquad (4.20)$$

где $1\le i\le 2t_d$. Как и раньше, может быть использовано любое подмножество $\nu+\mu\le t_d$ уравнений для определения значений всех стираний и ошибок.

Пример 44. Рассмотрим прямое вычисление значений искажений из предыдущего примера. После выполнения алгоритма Берлекэмпа-Мэсси и процедуры Ченя декодеру известны локаторы искажений и многочлен ошибок в виде

$$e(x)=f_0+e_3x^3+f_5x^5+e_{12}x^{12}$$

Значения искажений могут быть найдены из системы линейных уравнений (4.20), которая в матричной форме имеет следующий вид

$$\begin{pmatrix}1 & \alpha^3 & \alpha^5 & \alpha^{12}\\ 1 & \alpha^6 & \alpha^{10} & \alpha^9\\ 1 & \alpha^9 & 1 & \alpha^6\\ 1 & \alpha^{12} & \alpha^5 & \alpha^5\end{pmatrix}\begin{pmatrix}f_0\\ e_3\\ f_5\\ e_{12}\end{pmatrix}=\begin{pmatrix}\alpha^{11}\\ \alpha^5\\ \alpha^2\\ \alpha^2\end{pmatrix} \qquad (4.21)$$

Легко проверить, что решением (4.21) являются $f_0=\alpha^{14}$, $e_3=\alpha^8$, $f_5=\alpha^5$, $e_{12}=\alpha$. Эти значения совпадают со значениями искажений, вычисленными модифицированным алгоритмом Форни.

4.4. Распределение весов

Как упоминалось выше, коды РС являются МДР кодами. Распределение весов МДР кодов вычисляется точно с помощью следующего выражения [MS, LC]:

Число кодовых слов веса i МДР (n,k,d) кода над **GF**(q) *равно*

$$A_i=\binom{n}{i}(q-1)\sum_{j=0}^{i-d}(-1)^j\binom{i-1}{j}q^{i-j-d} \qquad (4.22)$$

Для оценки помехоустойчивости (n,k,d) РС кода над **GF**(q) можно использовать следующую границу (оценку сверху) вероятности ошибки на бит P_b алгебраического декодера РС[2], которая легко вычисляется и дает достаточно хороший результат,

[2] Следует заметить, что эта граница точна только для *декодеров с ограничением расстояния* таких, как декодеры БМ, ЕА или ПГЦ.

$$P_b \le \frac{2^{m-1}}{2^m-1}\sum_{i=t_d+1}^{n}\frac{i+t_d}{n}\binom{n}{i}P_s^{\,i}(1-P_s)^{n-i} \qquad (4.23)$$

где P_S вероятность *ошибки на символ* на входе декодера РС,

$$P_s = 1-(1-p)^m$$

и p вероятность *ошибки на бит* на входе декодера. Вероятность ошибки на слово (включая неправильное декодирование и отказ от декодирования) ограничена сверху границей (1.31),

$$P_e < 1-\sum_{i=0}^{t_d}\binom{n}{i}P_s^{\,i}(1-P_s)^{n-i} \qquad (4.24)$$

Глава 5

ДВОИЧНЫЕ СВЕРТОЧНЫЕ КОДЫ

Двоичные сверточные коды, впервые введенные Элайесом [Eli2], представляют собой, пожалуй, наиболее популярный вид помехоустойчивого кодирования. Эти коды нашли многочисленные применения. Вот некоторые из них: беспроводная связь (IMT-2000, GSM, IS-95), цифровые наземные и спутниковые системы связи и (радио-ТВ) вещания, системы связи с дальним космосом и т.п. Изначально эти коды были названы в [Eli2] *сверточными кодами с проверками на четность*. Наиболее распространенным, на сегодняшний день, методом их декодирования является *алгоритм Витерби* [Vit1]. Он применяется также для решения таких, эквивалентных по сложности задач, как восстановление последовательностей по максимуму правдоподобия (выравнивание канала) или прием сигналов в каналах с частичным (ограниченным по задержке) откликом (как, например, в канале магнитной записи). Совсем недавно было показано, что сверточные коды в сочетании с каскадной конструкцией и перемешиванием достигают эффективности очень близкой к пределу Шеннона [BGT]. В этой главе рассматриваются процедуры декодирования двоичного сверточного кода и их основные свойства.

5.1. Основные структуры

Сверточные коды это коды, исправляющие ошибки, которые используют непрерывную, или последовательную, обработку информации короткими фрагментами (блоками). Сверточный кодер обладает *памятью* в том смысле, что символы на его выходе зависят не только от (очередного фрагмента) информационных символов на входе, но и предыдущих символов на его

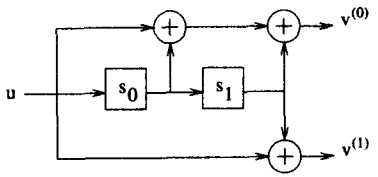


Рис. 29. Кодер сверточного кода скорости 1/2 и памяти 2.

входе и/или выходе. Другими словами, кодер представляет собой *последовательную машину* или автомат с конечным числом состояний. *Состояние* кодера определяется содержимым его памяти. В компьютерных программах, доступных на ECC вебсайте, которые реализуют алгоритм Витерби и другие процедуры декодирования, включающие поиск кратчайшего пути на решетке, используются таблицы переходов (смены состояний), связывающие вход, предыдущее и текущее состояния с текущим выходом автомата.

Сверточный код образуется множеством всех двоичных последовательностей, порождаемых сверточным кодером. Теоретически эти последовательности бесконечны, однако, на практике, состояние сверточного кодера периодически устанавливается в некоторое заранее известное состояние и, следовательно, порождаемый код приобретает характер блокового кода.

Пример 45. Рассмотрим сверточный кодер, показанный на Рисунке 29. Для облегчения анализа и демонстрации декодирования этот же кодер задан Таблицей 2.

Заметим, что *скорость кода* равна 1/2, так как на каждый введенный информационный символ выдаются два кодовых символа.

В общем случае, сверточный кодер скорости k/n использует k регистров сдвига, один регистр на один вводимый информационный символ, и n линейных комбинаций (над двоичным полем, т.е. с использованием «исключительного или») содержимого регистров и информационных символов на входе.

Таблица 2. Вход, выход и состояния кодера.

Начальное состояние $s_0[i]s_1[i]$	Информация $u[i]$	Конечное состояние $s_0[i+1]s_1[i+1]$	Выход $v^{(0)}[i]v^{(1)}[i]$
00	0	00	00
00	1	10	11
01	0	00	11
01	1	10	00
10	0	01	10
10	1	11	01
11	0	01	01
11	1	11	10

Ради простоты изложения и применения на практике, в дальнейшем рассматриваются только двоичные сверточные коды со скоростью $1/n$. Одна из причин такого выбора состоит в том, что эти двоичные коды используются наиболее часто. Техника, известная как *перфорация (выкалывание)* кодовых символов, позволяет построить сверточные кодеры для более высоких скоростей. Перфорация обсуждается в Разделе 5.5.

Общая длина регистров сдвига, используемых кодером, называется *памятью* кода. В этой главе в качестве *меток состояний* приняты целые числа I, ассоциированные с двоичными символами, которые записаны в регистрах, следующим образом:

$$I = \sum_{j=0}^{m-1} s_j(i) 2^{m-1-j}$$

Кодовое ограничение определяется как число информационных символов ($u[i]$, $u[i-1]$,..., $u[i-m]$) на входе кодера, которые влияют на кодовые символы ($v^{(0)}[i]$,..., $v^{(n-1)}[i]$) на его выходе в момент i. Для кодера скорости 1/2 оно равно $K=m+1$. Для кодера на Рисунке 29 $K=3$. Кодер, использующий m элементов памяти, называется в дальнейшем *кодер памяти m*. Сверточный кодер *памяти m* и *скорости* $1/n$ может быть задан также *диаграммой состояний*. На этой диаграмме имеется 2^m состояний. Так как в кодер вводится по одному биту, то в каждое состояния входят и из каждого состояния исходят по два ребра, помеченные метками $u[i]/v^{(0)}[i]$,..., $v^{(n-1)}[i]$.

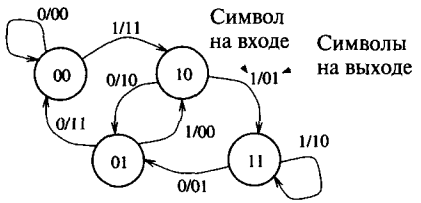


Рис. 30 Диаграмма состояний сверточного кодера памяти-2 и скорости-1/2.

Пример 46. Диаграмма состояний сверточного кодера памяти-2 и скорости 1/2 из Примера 45 показана на Рисунке 30.

Кодер памяти m и скорости $1/n$ двоичного сверточного кода можно рассматривать также как *дискретную линейную инвариантную во времени* систему[1]. Это означает, что импульсный отклик, т.е. выход кодера, полученный для входной последовательности $\mathbf{u} = (1000...00...)$, полностью определяет код.

Соглашение: при записи последовательностей предполагается, что крайний левый символ первым вводится в канал.

Имеется n импульсных откликов сверточного кодера скорости $1/n$, по одному на каждый выход $\mathbf{v}^{(j)}$, $j = 0,1,...,n-1$. По мере того, как импульс проходит через память кодера, он «выявляет» связи между элементами памяти и выходом. Очевидно, что этот кодер представляет собой систему с конечным откликом (FIR, finite-impulse-response).

Обозначим $\mathbf{g}_0,...,\mathbf{g}_{n-1}$ набор импульсных откликов сверточного кодера скорости $1/n$. Этот набор называют *порождающими последовательностями* – или генераторами кода. Генераторы кода задают действительные (физические) связи кодера.

Пример 47. Продолжаем изучение сверточного кода памяти 2 и скорости 1/2 из Примера 45. Кодер на Рисунке 29 порождает генераторы $\mathbf{g}_0=(1,1,1)$ и $\mathbf{g}_1=(1,0,1)$, как показано на Рисунке 31.

Заметим, что через K бит порождающие последовательности переходят в нулевое продолжение (хвост). Это означает, что

[1] Заметим также, что определение и изучение *меняющихся во времени* сверточных кодов дается в [Joh].

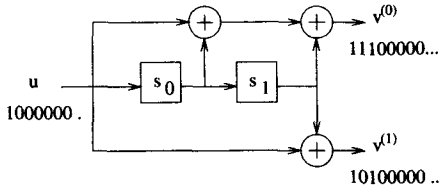


Рис. 31. Порождающие последовательности сверточного кодера памяти 2 и скорости 1/2.

достаточно рассматривать векторы этой длины. Для того чтобы подчеркнуть динамический характер сверточного кодера вводится формальная переменная D (оператор «задержки») и полиномиальная форма записи генераторов кода: $g_0(D),..., g_{n-1}(D)$.

Пример 48. Для кодера на Рисунке 29

$$g_0(D) = 1 + D + D^2 \text{ и } g_1(D) = 1 + D^2.$$

Обычно, говоря о сверточном коде, генераторы записывают в виде $(\mathbf{g}_0,...,\mathbf{g}_1)$ в восьмеричной системе. Для кодера из Примера 48 эта запись имеет вид (7,5). Наиболее широко используемый до настоящего времени сверточный код это код памяти 6 и скорости 1/2 с генераторами (171,133). Это, так называемый, *стандартный* код NASA (Национальный комитет по аэронавтике и исследованию космического пространства, НАСА (США)), который впервые был использован в космической экспедиции Вояджер (Voyager space mission). Он использован также во многих стандартах цифровой связи, например, в стандарте CCSD.

Динамическая структура сверточного кодера позволяет развернуть во времени его диаграмму состояний и построить *решетчатую диаграмму* (trellis diagram). Решетчатая диаграмма (или просто *решетка,* или *треллис*) строится следующим образом: в соответствии с диаграммой состояний кодера (или таблицей кодера) на каждом временном интервале соединяются ребрами состояния на i-ом и $i+1$-ом тактах. Эти ребра помечаются теми же метками, что и на диаграмме состояний.

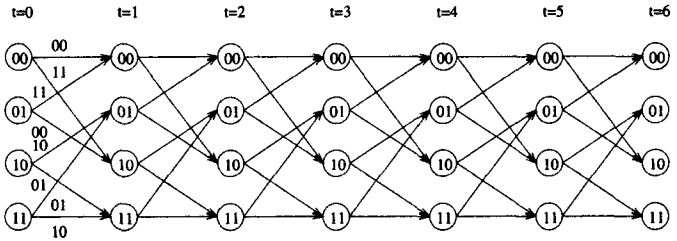


Рис. 32. Шесть секций решетчатой диаграммы сверточного кода скорости 1/2 и памяти 2.

Соглашение: Если не возникает неоднозначности, то входной информационный символ не обязательно должен быть записан на ребре треллиса. Например, для кодера с конечным откликом (FIR) информационный символ может быть извлечен непосредственно из состояния, в которое осуществляется переход[2]:

$$(s_0\, s_1 \ldots s_{m-1}) \to (u\, s_0\, s_1 \ldots s_{m-2}).$$

Следует заметить, что в случае рекурсивного систематического сверточного кодера (IIR, infinite-impulse-response) это не совсем так, а именно, информационный символ определяется парой состояний. Мы вернемся к этому вопросу в разделе 5.1.1.

Пример 49. Для сверточного кода скорости 1/2 и памяти 2 из последних примеров решетка (или треллис) показана на Рисунке 32. Заметим, что переходы из состояния $s(i)$ в состояние $s'(i+1)$, вызванные нулевым входным символом $u(i)=0$, представлены на диаграмме верхними ребрами.

Пример 50. Рассмотрим кодер, представленный на Рисунке 29, и входную последовательность **u** = (110100). Тогда соответствующая выходная (кодовая) последовательность может быть получена непосредственно из таблицы кодирования (Таблица 2) или как путь на диаграмме, показанный на Рисунке 33. Соответствующая кодовая последовательность равна **v** = (11 01 01 00 10 11).

[2] Это простое наблюдение используется при обратном проходе в памяти путей декодера Витерби для того, чтобы восстановить декодированную последовательность.

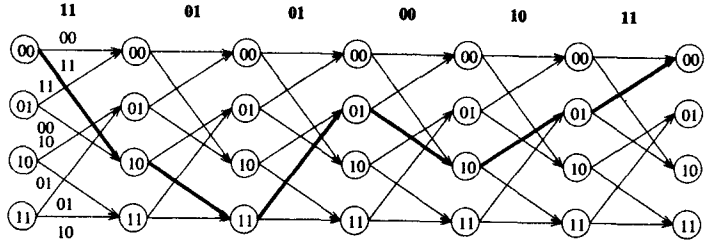


Рис. 33. Путь на треллисесверточного кода скорости 1/2 и памяти 2.

Сверточный кодер является линейной *постоянной во времени* (time-invariant) системой, импульсный отклик которой задан генераторами кода $g_0(D),\ldots, g_{n-1}(D)$, где

$$g_j(D) = g_j[0] + g_j[1]D + g_j[2]D^2 + \cdots + g_j[m]D^m \qquad (5.1)$$

для $0 \le j < n$. С помощью этих генераторов выходную последовательность кода можно записать как

$$v^{(j)}[i] = \sum_{l=0}^{m} u[i-\ell]g_j[\ell] \qquad (5.2)$$

$0 \le j < n$. Выходные последовательности $v^{(j)}(D)$, как и ожидалось, равны дискретной *свертке* входной последовательности $u(D)$ с генераторами кода $g_0(D),\ldots, g_{n-1}(D)$. Этим фактом объясняется название «сверточный» кодер [Eli2].

Покажем теперь, как можно записать уравнение (5.2) в матричной форме

$$\mathbf{v} = \mathbf{u}G, \qquad (5.3)$$

где **G** – *порождающая матрица* сверточного кода. В частности, для сверточного кода памяти m и скорости 1/2 имеем

$$\mathbf{G} = \begin{pmatrix} g_0[0]g_1[0] & \cdots & g_0[m]g_1[m] & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & g_0[0]g_1[0] & \cdots & g_0[m]g_1[m] & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & g_0[0]g_1[0] & \cdots & g_0[m]g_1[m] \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \ddots & \ddots \end{pmatrix} \qquad (5.4)$$

Обобщение этой конструкции матрицы на кодер скорости $1/n$ очевидно.

Пример 51. Для кодера памяти 2 и скорости 1/2 на Рисунке 29 порождающая матрица равна

$$\mathbf{G}=\begin{pmatrix} 11 & 10 & 11 & & & & \\ & 11 & 10 & 11 & & & \\ & & 11 & 10 & 11 & & \\ & & & 11 & 10 & 11 & \\ & & & & \ddots & & \ddots \end{pmatrix}$$

где пустые ячейки матрицы заполнены нулями. Это, так называемая, ленточная матрица с шириной ленты равной $(m+1)n$, для кода памяти m и скорости $1/n$. Пусть $\mathbf{u}$=(110100). Тогда из (5.3) следует, что $\mathbf{v} = \mathbf{uG}$ = (11,01,01,00,10,11). Этот результат совпадает с выходной последовательностью Примера 50.

Пусть $v(D) = v^{(0)}(D) + Dv^{(1)}(D) + \ldots + D^{n-1}v^{(n-1)}(D)$. Тогда соотношение между последовательностями на входе и выходе кодера может быть записано как

$$v(D)=u(D)G(D) \tag{5.5}$$

где генераторы сверточного кода скорости $1/n$ записаны в виде *полиномиальной порождающей матрицы*,

$$G(D)=(g_0(D)\, g_1(D)\cdots g_{n-1}(D)) \tag{5.6}$$

5.1.1. Рекурсивные систематические сверточные коды

Полиномиальная порождающая матрица (5.6) может быть преобразована к матрице вида

$$G'(D)=\begin{pmatrix} I & \frac{g_1(D)}{g_0(D)} & \cdots & \frac{g_{n-1}(D)}{g_0(D)} \end{pmatrix} \tag{5.7}$$

которая порождает *рекурсивный систематический сверточный* код (RSC код). В сокращенной записи этой матрицы генераторы кода имеют вид $(1, g_1/g_0,\ldots, g_{n-1}/g_0)$. Выполняя деление и умножение правой части (5.5) на $g_0(D)$, получаем

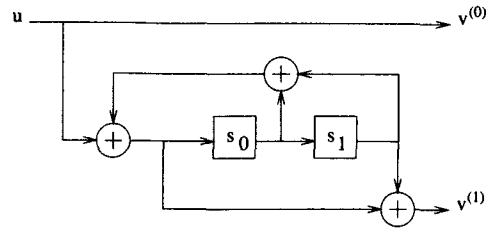


Рис. 34. Кодер памяти 2 и скорости 1/2 рекурсивного систематического сверточного кода.

$$v(D)=u'(D)G'(D) \tag{5.8}$$

где $u'(D) = g_0(D)u(D)$. Отсюда следует, что несистематический кодер и систематический рекурсивный кодер имеют одно и тоже множество кодовых последовательностей (слов), но с другим соответствием между информационными и кодовыми словами.

Систематический кодер также является примером дискретной, постоянной во времени, линейной системы. Так как порождающая матрица содержит рациональные функции, то систематический кодер является линейной системой с *бесконечным импульсным откликом* (IIR), в отличие от несистематического кодера, который является линейной системой с *конечным импульсным откликом* (FIR). Предпочтительной формой изображения структуры кодера является *каноническая логическая форма* [For4]. Кодер RSC кода состоит из регистров сдвига, схемы[3] деления на $g_0(D)$ и $n-1$ схем умножения на $g_1(D),\ldots, g_{n-1}(D)$.

Пример 52. На Рисунке 34 показан RSC кодер скорости 1/2 и памяти 2 с генераторами (1, 5/7). Решетка этого кода такая же, как и у несистематического кода с генераторами (5,7), од-

[3] Общая структура схем деления и умножения двух многочленов имеется, например, в [PW], Рисунок 7.8, или в [LC] Раздел 4.7.

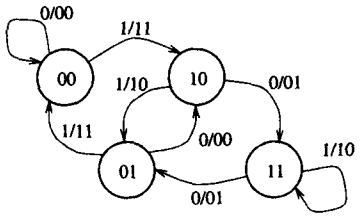


Рис. 35. Диаграмма состояний памяти 2 и скорости 1/2 рекурсивного систематического сверточного кода.

нако метки ребер (входные/выходные символы) различны. Диаграмма состояний этого кодера показана на Рисунке 35. Сравните ее с диаграммой на рисунке 30. Различное отображение входов/выходов означает, что нулевая последовательность на входе не обязательно возвратит кодер в начальное нулевое состояние $s_0 s_1 = 00$. Это ясно из Рисунка 35. Наконец, импульсный отклик кодера равен (11,01,01,00,01,01,00,01,01,00,...).

5.1.2. Свободное расстояние

Свободным расстоянием d_f сверточного кода называют минимальное расстояние между двумя кодовыми последовательностями. Длина последовательностей должна быть достаточно большой, значительно больше кодового ограничения данного кода. Свободное расстояние сверточного кода может быть получено из нумератора весов кода, как будет показано в Разделе 5.3. Существуют и другие расстояния, относящиеся к сверточным кодам. Однако они не будут рассматриваться в этой книге. Дополнительные подробности о структуре и свойствах сверточных кодов скорости k/n имеются, например, в [LC] и [Joh].

5.2. Связь с блоковыми кодами

Из приведенного выше описания сверточных кодеров видно, что существует связь между сверточными и блоковыми кодами. Как уже отмечалось ранее, информационные последовательности обычно разбивают на блоки *конечной длины* (например, несколько тысяч бит). Как правило, в конце каждого информационного блока добавляется фиксированная последовательность длины m. Эта последовательность представляет собой *уникальное слово*, которое служит для синхронизации приемника и заставляет сверточный кодер возвращаться к известному (начальному) состоянию.

В дальнейшем, ради простоты изложения, будем считать, что C несистематический (FIR) сверточный код со свободным расстоянием d_f, образованный сверточным кодером памяти m и скорости $1/n$. Аналогичные характеристики относятся и к RSC кодам.

5.2.1. Терминированная конструкция (нулевой хвост)

Добавление «хвоста» из m нулей к каждому информационному блоку длины $(K-m)$ приводит к завершению всех путей на треллисе, соответствующих 2^{K-m} кодовым словам, в нулевом состоянии. В результате получаем линейный блоковый $(nK,(K-m),d_{ZT})$ код C_{ZT}. Если K достаточно велико, то скорость кода C_{ZT} приближается к скорости кода C. Если $K>m$ и K достаточно велико, то минимальное расстояние кода C_{ZT} удовлетворяет равенству $d_{ZT} = d_f$.

Пример 53. Пусть C сверточный код со свободным расстоянием[4] $d_f = 5$, порождаемый кодером памяти 2 и скорости 1/2 на Рисунке 29. Предположим, что кодируется информационный вектор $K = 3$ бит, дополненный $m = 2$ нулями. Тогда терминированная конструкция приводит к двоичному блоковому линейному (10, 3, 5) коду C_{ZT} с порождающей матрицей

$$\mathbf{G} = \begin{pmatrix} 11 & 10 & 11 & 00 & 00 \\ 00 & 11 & 10 & 11 & 00 \\ 00 & 00 & 11 & 10 & 11 \end{pmatrix}$$

[4] Доказательство этого значения d_f будет дано в разделе 5.3.

Распределение весов этого кода равно $A(x) = 1 + 3x^5 + 3x^6 + x^7$ и $d_{ZT} = 5$. Скорость кода C_{ZT} равна 3/10, т.е. меньше скорости $k/n = 1/2$ исходного кода C.

5.2.2. Усеченная конструкция (direct truncation)

В этом случае кодовые слова линейного блокового (nK, K, d_{DT}) кода ассоциируются с кодовыми последовательностями сверточного кодера скорости k/n, которые соответствуют на трellисе всем путям, начинающимся в нулевом состоянии и заканчивающимся, после ввода K информационных бит, в произвольном состоянии. Минимальное расстояние полученного блокового кода удовлетворяет неравенству $d_{DT} < d_f$.

Пример 54. Рассмотрим кодер скорости 1/2 памяти 2 на Рисунке 29. Кодируются информационные векторы длины $K = 3$. Конструкция с прямым усечением кодовых последовательностей приводит к двоичному линейному (6,3, d_{DT}) коду с порождающей матрицей

$$\mathbf{G} = \begin{pmatrix} 11 & 10 & 11 \\ 00 & 11 & 10 \\ 00 & 00 & 11 \end{pmatrix}$$

Распределение весов этого кода равно $A(x) = 1 + x^2 + 3x^3 + 2x^4 + x^5$ и $d_{DT} = 2 < d_f$.

5.2.3. Кольцевая (циклическая или циклически замкнутая) (tail-biting) конструкция

Кодовые слова кольцевой конструкции блокового кода C_{TB} представляют собой множество всех путей на решетке, которые начинаются в состоянии, заданной последними m символами информационного вектора длины K бит. Таким образом, кодовые слова C_{TB} начинаются и заканчиваются в *одном и том же состоянии* кодовой решетки. Скорость полученного блокового (nK, K, d_{TB}) кода совпадает со скоростью $1/n$ исходного

сверточного кода. Однако даже при достаточно большом $K > m$ минимальное расстояние удовлетворяет условию $d_{TB} \le d_f$.

Пример 55. Рассмотрим кодер скорости 1/2 памяти 2 на Рисунке 29. Предположим, что кодируется блок $K = 5$ информационных символов. Тогда кольцевая конструкция приводит к двоичному линейному блоковому (10, 5, d_{TB}) коду с порождающей матрицей

$$\mathbf{G} = \begin{pmatrix} 11 & 10 & 11 & 00 & 00 \\ 00 & 11 & 10 & 11 & 00 \\ 00 & 00 & 11 & 10 & 11 \\ 11 & 00 & 00 & 11 & 10 \\ 10 & 11 & 00 & 00 & 11 \end{pmatrix}$$

Распределение весов[5] равно $A(x) = 1 + 5x^3 + 5x^4 + 6x^5 + 10x^6 + 5x^7$ и $d_{TB} = 3 < d_f$.

5.2.4. Распределение весов

В этом разделе рассматривается метод определения спектра весов линейного блокового кода, построенного из сверточного кода скорости $1/n$ с помощью рассмотренных выше конструкций [WV]. Обозначим $\Omega(x)$ *переходную матрицу состояний* кодера размера 2^m x 2^m и вида:

$$\Omega_{ij}(x) = \delta_{ij} x^{h_{ij}} \qquad (5.9)$$

где $\delta_{ij} = 1$, если и только если имеется переход из состояния i в состояние j. В противном случае $\delta_{ij} = 0$. Величина h_{ij} равна Хемингову весу соответствующего выходного вектора (длины n).

Пример 56. Для сверточного кодера, диаграмма состояний которого дана на Рисунке 30, переходная матрица состояний $\Omega(x)$ равна

[5] Это распределение весов получено с помощью программы на ECC - Интернет сайте, которая вычисляет спектр двоичного линейного кода по его порождающей матрице.

$$\Omega(x)=\begin{pmatrix} 1 & 0 & x^2 & 0 \\ x^2 & 0 & 1 & 0 \\ 0 & x & 0 & x \\ 0 & x & 0 & x \end{pmatrix}$$

Распределение весов двоичного линейного блокового (n, k) кода, построенного любым из рассмотренных выше способов, может быть получено просто возведением данной переходной матрицы в ℓ-ую степень, обозначенную $\Omega^{\ell}(x)$, и комбинированием различных членов.

Каждый элемент $\Omega^{\ell}_{ij}(x)$ переходной матрицы дает распределение весов путей на решетке, которые начинаются в состоянии i и заканчиваются в состоянии j через ℓ шагов (тактов ввода). Для конструкции ZT (нулевой хвост) значение $= k + m$, тогда как для конструкций DT и TB $l = k$. Распределение весов для каждой из рассмотренных выше конструкций было получено следующим способом.

- **Терминированная конструкция (ZT):**

$$A(x)=\Omega_{00}^{k+m}(x)$$

- **Усеченная конструкция (DT):**

$$A(x)=\sum_{j=0}^{2^m}\Omega_{0j}^{k}(x)$$

- **Кольцевая конструкция (TB):**

$$A(x)=\sum_{j=0}^{2^m}\Omega_{jj}^{k}(x)$$

Пример 57. Рассмотрим снова сверточный кодер памяти 2 и скорости 1/2. Для него находим

$$\Omega^{3}(x)=\begin{pmatrix} 1+x^5 & x^3+x^4 & x^2+x & x^3+x^4 \\ x^2+x^3 & x^5+x^2 & x^4+x & x^5+x^2 \\ x^3+x^4 & x^2+x^3 & x^5+x^2 & x^2+x^3 \\ x^3+x^4 & x^2+x^3 & x^5+x^2 & x^2+x^3 \end{pmatrix}$$

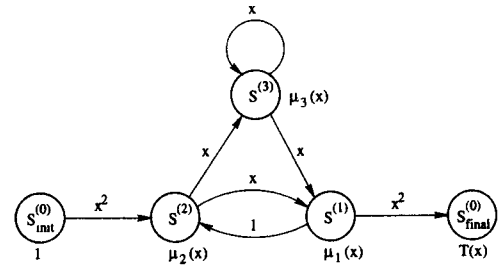


Рис. 36. Модифицированная диаграмма состояний сверточного кода памяти 2 и скорости 1/2.

Распределение весов усеченной конструкции из Примера 54 получается суммированием членов первой строки полученной выше матрицы.

Описание других методов вычисления спектра весов линейного блокового кода, построенного из сверточного, имеется в [FLC], [CK], [DFK1].

5.3. Нумераторы весов и границы вероятности ошибки

Оценка эффективности сверточных кодов в каналах без памяти может быть получена с помощью границы объединения, также как это было сделано в Главе 1 для блоковых кодов. Витерби [Vit2] был первым, кто рассмотрел эффективность сверточных кодов. Обширный материал имеется и в [ViOm]. Для сверточных кодов *нумератор весов* (WES) определяется как

$$T(x)=T_1x+T_2x^2+\cdots+T_wx^w+\cdots, \qquad (5.10)$$

где коэффициент T_w в $T(x)$ равен числу кодовых последовательностей веса w. Заметим, что, в принципе, на входе сверточного кода появляются бесконечные последовательности. В результате этого, нумератор весов (WES) имеет бесконечное число членов. Тем не менее, можно получить замкнутую форму выражения для нумератора весов сверточного кода, как это будет показано ниже.

Обозначение: Состояние $(s_0 s_1 \ldots s_{m-1})$ сверточного кодера будет обозначено как S^I, где

$$I = \sum_{i=0}^{m-1} s_i 2^{m-1-i}$$

Нумератор весов будет вычислен с помощью *модифицированной диаграммы состояний*. На этой диаграмме нулевое (начальное) состояние расщепляется на начальное состояние $S^{(0)}_{init}$ и финальное состояние $S^{(0)}_{final}$. Ребра на этой диаграмме помечены множителями x^w, где w вес Хемминга выходной последовательности.

Пример 58. Пусть C сверточный код памяти 2 и скорости 1/2 с генераторами (7,5), который используется всюду в этой главе. Модифицированная диаграмма состояний кода C показана на Рисунке 36.

Известны два метода вычисления $T(x)$. Первый из них рассматривает модифицированную диаграмму состояний как сигнальный граф. Для вычисления $T(x)$ применяется правило Мэйсона. Этот метод рассматривается в большинстве книг по теории помехоустойчивого кодирования. Читатель может найти все детали в [LC].

Другой метод, предложенный Витерби [Vit2], состоит в приписывании состояниям $S^{(j)}$, $j = 1, 2, \ldots, 2^m - 1$, формальных переменных $\mu_j(x)$. Каждая переменная $\mu_j(x)$ представляет собой линейную комбинацию взвешенных переменных, приписанных состояниям, которые связаны с состоянием $S^{(j)}$,

$$\mu_j(x) = \sum_{\substack{k=1 \\ k \neq l}}^{2^m-1} \mu_k \delta_{kj} x^{h_{kj}} \qquad (5.11)$$

для всех $k \in \{1, 2, \ldots, 2^m - 1\}$, $k \neq j$, где δ_{ij} и h_{ij} определены в (5.9). Начальному состоянию $S^{(0)}_{init}$ приписана переменная со значением 1, а нумератор весов $T(x)$ приписан как переменная финальному состоянию $S^{(0)}_{final}$.

Теперь можно построить систему линейных уравнений, которая может быть решена относительно $\mu_1(x)$. Учитывая обозначение состояний, получаем

$$T(x) = \mu_1(x) x^{h_{10}} \qquad (5.12)$$

Пример 59. Рассмотрим модифицированную диаграмму состояний на Рисунке 36. Ей соответствует следующая система уравнений

$$\begin{aligned} \mu_1(x) &= \mu_2(x)x + \mu_3(x)x \\ \mu_2(x) &= x^2 + \mu_1(x) \\ \mu_3(x) &= \mu_2(x)x + \mu_3(x)x \end{aligned} \qquad (5.13)$$

и нумератор $T(x) = \mu_1(x)x^2$. Уравнение (5.13) в матричной форме имеет вид

$$\begin{pmatrix} 1 & -x & -x \\ -1 & 1 & 0 \\ 0 & -x & 1-x \end{pmatrix} \begin{pmatrix} \mu_1(x) \\ \mu_2(x) \\ \mu_3(x) \end{pmatrix} = \begin{pmatrix} 0 \\ x^2 \\ 0 \end{pmatrix} \qquad (5.14)$$

Решение этой системы относительно $\mu_1(x)$ равно

$$\mu_1(x) = \frac{x^3}{1-2x}$$

Окончательно получаем

$$T(x) = \frac{x}{1-2x} = x^5 + 2x^6 + 4x^7 + 8x^8 + \cdots \qquad (5.15)$$

Отсюда следует, что сверточный код памяти 2 (т.е. с числом состояний 4) и скорости 1/2 с генераторами (7,5) имеет свободное расстояние $d_f = 5$.

Более подробное исследование структуры сверточного кода возможно, если на модифицированной диаграмме состояний ввести метки вида $x^w y^\ell z^m$, где w вес Хемминга кодовой последовательности, ℓ вес Хемминга входной информационного вектора (длины равной k для сверточного кода скорости k/n), m число ненулевых ребер (см. [LC, стр. 302]). Соответствующий этой модификации результат $T(x,y,z)$ называется *полным нумератором весов* (CWES). Полный нумератор весов может быть использован для получения различных оценок вероятно-

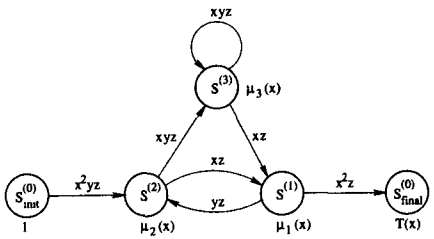


Рис. 37. Модифицированная диаграмма состояний сверточного кода памяти 2 и скорости 1/2.

сти ошибки сверточных кодов [Vit2, LC, Joh]. Нумератор весов аналогичный полному нумератору был использован для оценки эффективности турбо кодов в [BM].

Пример 60. Продолжим изучение сверточного кода скорости 1/2 с числом состояний 4, модифицированная диаграмма состояний которого показана на Рисунке 37. По этой диаграмме получаем следующую систему уравнений

$$\begin{pmatrix} 1 & -xz & -xz \\ -yz & 1 & 0 \\ 0 & -xyz & 1-xyz \end{pmatrix}\begin{pmatrix} \mu_1(x) \\ \mu_2(x) \\ \mu_3(x) \end{pmatrix}=\begin{pmatrix} 0 \\ x^2yz \\ 0 \end{pmatrix}$$

с решением вида $T(x, y, z) = x^2z\mu_1(x)$. Решением системы является полный нумератор равный

$$T(x,y,z)=\frac{x^5y^3z}{1-xy(1+y)z}=x^5y^3z+x^6y^4(1+y)z^2+x^7y^5(1+y)^2z^3+\cdots$$

С помощью полного нумератора весов можно получить границы вероятности ошибки на бит для сверточного кода скорости k/n. В ДСК и в случае передачи двоичных сигналов по каналу с АБГШ [ViOm] с декодированием по максимуму правдоподобия (МПД)[6] справедливы следующие границы, соответственно,

$$P_b < \frac{1}{k}\left.\frac{\partial T(x,y,z)}{\partial y}\right|_{x=\sqrt{2p(1-p)},\, y=1,\, z=1} \qquad (5.16)$$

[6] Примером МПД является декодер Витерби, который рассматривается в следующем разделе.

$$P_b < \frac{1}{k}\left.\frac{\partial T(x,y,z)}{\partial y}\right|_{x=\exp(-E_b/N_0),\, y=1,\, z=1} \qquad (5.17)$$

Для оценки вероятности ошибки на бит (BER) сверточного кода можно применить границу объединения. Читатель должен понимать, что границы (5.16) и (5.17) довольно слабы. К счастью, существуют более точные (т.е. близкие к действительным значениям BER) границы для относительно хороших условий, т.е. достаточно малых значений вероятности ошибки в ДСК и достаточно высоких отношений E_s/N_0 в канале с АБГШ, которые приводятся в [Joh]:

$$P_b < \frac{1}{k}\left(\frac{1+x}{2}\frac{\partial T(x,y,z)}{\partial y}+\frac{1-x}{2}\frac{\partial T(-x,y,z)}{\partial y}\right)\Bigg|_{x=\sqrt{2p(1-p)},\, y=1,\, z=1} \qquad (5.18)$$

$$P_b < \frac{1}{k}Q\left(\sqrt{2Rd_f\frac{E_b}{N_0}}\right)e^{2Rd_fE_b/N_0}\left.\frac{\partial T(x,y,z)}{\partial y}\right|_{x=\exp(-E_b/N_0),\, y=1,\, z=1} \qquad (5.19)$$

Пример 61. Граница объединения (5.18) для вероятности ошибки на бит для сверточного кода скорости 1/2 и памяти 2 из Примера 60 в ДСК с вероятностью ошибки p и МПД построена на Рисунке 38.

5.4. Декодирование: Алгоритм Витерби в Хемминговой метрике

Решетка сверточного кода имеет регулярную структуру. Повторяемость секций решетки может быть эффективно использована при декодировании. Декодирование по максимуму правдоподобия (в обычном смысле) блоковых кодов, полученных из сверточных кодов, при большой длине информационных последовательностей слишком сложно и непрактично для реализации.

Эффективным решением проблемы декодирования является алгоритм динамического программирования, известный

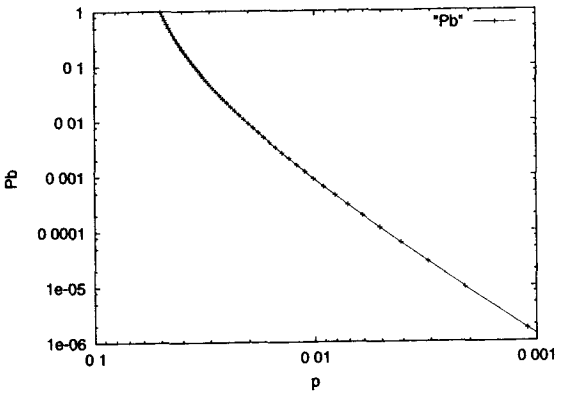


Рис. 38. Граница объединения (5.18) вероятности ошибки на бит сверточного кода памяти 2, скорости 1/2 со свободным расстоянием $d_f = 5$. Передача по ДСК с вероятностью ошибки p и МПД.

как *алгоритм Витерби* или *декодер Витерби* (ВД). Декодер Витерби реализует *метод максимального правдоподобия* в несколько ином смысле, но очень близком к классическому. Это зависит от конкретного применения декодера. В наиболее распространенном варианте применения ВД находит кодовую последовательность, ближайшую к принятой, обрабатывая ее бит за битом. Вместо подсчета веса каждой из возможных кодовых последовательностей декодер Витерби *прослеживает состояния решетки* (*и проходящие через них пути*).

5.4.1. Декодирование по максимуму правдоподобия и метрики

Правдоподобие принятой последовательности **r** при условии, что по каналу с шумом без памяти была передана последовательность **v**, определяется условной вероятностью

$$P(\mathbf{r} \mid \mathbf{v}) = \prod_{i=0}^{n-1} P(r_i \mid v_i) \qquad (5.20)$$

Легко показать, что в ДСК с параметром p

$$P(\mathbf{r} \mid \mathbf{v}) = \prod_{i=0}^{n-1} (1-p) \left(\frac{p}{1-p} \right)^{d_H(r_i, v_i)} \qquad (5.21)$$

где $d_H(r_i, v_i)$ есть расстояние Хемминга между битами r_i и v_i, $d_H(r_i, v_i) = 1$, если $r_i \neq v_i$ и $d_H(r_i, v_i) = 0$, если $r_i = v_i$. Для канала с АБГШ правдоподобие измеряется вероятностью

$$P(\mathbf{r} \mid \mathbf{v}) = \prod \frac{1}{\sqrt{\pi N_0}} \exp\left(-\frac{1}{N_0} (r_i - m(v_i))^2 \right) \qquad (5.22)$$

где функция $m(.)$ представляет двоичный модулированный сигнал. Здесь рассматривается взаимно однозначное отображение двоичных символов {0,1} в вещественные числа $\{-\sqrt{E}, +\sqrt{E}\}$.

Декодирование по максимуму правдоподобия (МПД) означает выбор такой кодовой последовательности **v′**, которая максимизирует (5.20). Логарифмируя (по любому основанию) выражение (5.20), легко показать следующее. Для ДСК такое декодирование эквивалентно выбору кодовой последовательности, которая минимизирует *Хемммингово расстояние*

$$d_H(\mathbf{r}, \mathbf{v}) = \sum_{i=0}^{n-1} d_H(r_i, v_i) \qquad (5.23)$$

Аналогично, для канала с АБГШ декодер минимизирует квадрат *Евклидова расстояния*

$$d_E(\mathbf{r}, \mathbf{v}) = \sum (\mathbf{r} - \mathbf{m}(\mathbf{v}))^2 \qquad (5.24)$$

В этой главе рассматривается ДСК с вероятностью ошибки на бит p. Гауссов канал рассматривается в Главе 7.

5.4.2. Алгоритм Витерби

Обозначим $S^{(k)}_t$ состояние (узел) на решетке в момент i. Каждому состоянию решетки припишем *метрику* (вес) $M(S^{(k)}_t)$ и некоторый *путь* на решетке $\mathbf{y}^{(k)}$ (заканчивающийся в данном состоянии). Ключевым свойством (для понимания) алгоритма Витерби является следующее.

В момент i наилучшие (т.е. ближайшие к принятой последовательности) пути $\mathbf{y}^{(k)}$, *заканчивающиеся в состояниях* $S^{(k)}{}_i$, *имеют общее начало в некоторый момент* $i-\ell$.

В работе [Vit] Витерби отметил, что для двоичного сверточного кода скорости 1/2 величина ℓ должна удовлетворять условию $\ell > 5m$. *Строго говоря, эта величина зависит от отношения сигнал-шум или вероятности ошибки в ДСК.* Обычно декодер Витерби работает в окне размера L принятых из канала n-ребер. Эта величина называется *глубиной декодирования*. Очевидно, что условие $L>\ell$ является необходимым (величина окна влияет на вероятность ошибки декодера Витерби).

В дальнейшем, рассматривается применение алгоритма Витерби к двоичному сверточному коду памяти m, скорости $1/n$. Работа алгоритма показана на простом примере. Нам понадобятся следующие дополнительные обозначения. Пусть $\mathbf{v}[i] = (v_0[i]\ v_1[i] \ldots\ v_{n-1}[i])$ обозначает метку (т.е. кодовые символы) ребра кодовой решетки, а $\mathbf{r}[i] = (r_0[i]\ r_0[i] \ldots r_0[i])$ обозначает i-ое ребро на выходе канала.

Основные этапы декодирования

Начальные условия

Положим $i = 0$. Установим нулевые начальные значения метрик и путей

$$M(S_0^{(k)}) = 0, \quad \mathbf{y}_0^{(k)} = (\ ) (\text{пусто})$$

Конкретный способ инициализации путей, как будет показано ниже, не имеет значения. Для того, чтобы упростить описание алгоритма, предполагается , что пути представляются *списком*, начальным значением которого является *пустой список*.

1. Вычисление метрик ребер решетки

В i-ом сечении решетки (на i-ом шаге процедуры) вычислить (частичные) *метрики ребер*

$$BM_i^{(b)} = d_H(\mathbf{r}[i], \mathbf{v}[i]), \quad b = \sum_{\ell=0}^{n-1} v_\ell[i] 2^{n-1-\ell} \qquad (5.25)$$

которые определяются кодовыми символами $\mathbf{v}[i]$ на каждом ребре решетки и принятыми из канала символами $\mathbf{r}[i]$.

2. Прибавить, сравнить и выбрать (ПСВ)

Для каждого состояния $S^{(k)}{}_i$, $k = 0, 1, \ldots, 2^m-1$, и соответствующей пары ребер, входящих в данное состояние из двух предшествующих $S^{(ka)}{}_{i-1}$, $a \in \{1, 2\}$, вычислить и сравнить суммы $M(S^{(ka)}{}_{i-1}) + BM^{(ba)}{}_i$, где $b_a = v_0[a]2^{n-1} + v_1[a]2^{n-2} + \ldots + v_{n-1}[a]2^0$. Выбрать *выжившее ребро*, которое дает наименьшую метрику пути, и запомнить,

$$M(S_i^{(k)}) = \min\{M(S_{i-1}^{(k_1)}) + BM_i^{(b_1)}, (S_{i-1}^{(k_2)}) + BM_i^{(b_2)}\} \qquad (5.26)$$

3. Обновление памяти путей

Для каждого состояния $S^{(k)}{}_i$, $k = 0, 1, \ldots, 2^m-1$, запомнить *выжившие пути* $\mathbf{y}^{(k)}$, дополнив каждый из них выжившим ребром $\mathbf{v}_{kj}$, $j \in \{1, 2\}$,

$$\mathbf{y}_i^{(k)} = (\mathbf{y}_{i-1}^{(k_j)}, \mathbf{v}_{k_j}) \qquad (5.27)$$

4. Декодирование символов

Если $i > L$, то выдать как оценку кодовой последовательности ребро $\mathbf{y}^{(\gamma)}{}_{i-L}$, где γ индекс состояния $S^{(\gamma)}$ с наименьшей метрикой. Установить $i = i + 1$ и вернуться в п. 1 процедуры.

Следует заметить, что это не единственный способ реализации алгоритма Витерби. Рассмотренную выше процедуру можно считать *классической*. Другие варианты реализации могут обладать различными преимуществами в зависимости от конкретной структуры сверточного кодера (см., например, [FL2]). Кроме того, на последнем шаге процедуры можно выдавать непосредственно информационные символы. Этот вариант чаще используется в программной реализации декодера Витерби, доступной на ECC интернет-сайте. Для реализации на интегральных микросхемах обычно применяется метод *обратного прохода памяти путей* с восстановлением информационных символов из переходов между состояниями кодовой решетки. Этот способ будет рассмотрен ниже в этой главе.

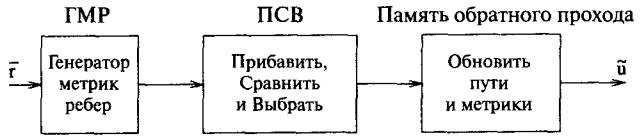


Рис. 39. Структурная схема декодера Витерби.

Пример 62. Рассмотрим еще раз сверточный код памяти 2, скорости 1/2 с генераторами (7,5). Напомним, что этот код имеет $d_f = 5$. В этом примере показано исправление одиночной ошибки. Предположим, что кодовая последовательность $\mathbf{v} = (11, 01, 01, 00, 10, 11)$ передана по двоичному симметричному каналу (ДСК), на выходе которого получена последовательность $\mathbf{r}=(10, 01, 01, 00, 10, 11)$, содержащая одну ошибку на второй позиции. Действия декодера Витерби показаны на Рисунках 40 – 45. Значения метрик (выживших) путей по всем состояниям решетки и на первых 6 шагах процедуры представлены в следующей ниже таблице:

Состояние/Шаг	$i=0$	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$
$S_i^{(0)}$	0	1	1	1	1	2	1
$S_i^{(1)}$	0	0	0	1	2	1	3
$S_i^{(2)}$	0	1	1	1	1	2	2
$S_i^{(3)}$	0	0	1	1	2	2	2

После обработки шести секций решетки ($i = 6$) состоянием с наименьшей метрикой является $S^{(0)}{}_6$ с ассоциированным (выжившим) путем на решетке $\mathbf{y}^{(0)}{}_6 = \mathbf{v}$. Таким образом, исправлена одна ошибка.

5.4.3. Проблемы реализации

В этом разделе обсуждаются некоторые проблемы, относящиеся к реализации декодера Витерби. Рассмотренные ниже приемы могут быть использованы для реализации декодеров Витерби в любых каналах с аддитивной метрикой таких как, например, ДСК, каналы с АБГШ и каналы с общими Релеевскими замираниями.

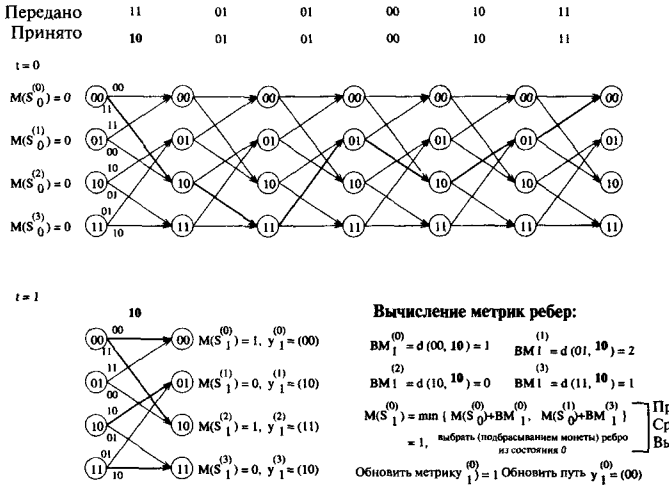


Рис. 40. Диаграмма вычислений декодера Витерби, Пример 62 , шаги $i = 0$ и $i = 1$.

Инициализация метрик путей

Декодер Витерби может работать в одном и том же режиме с самого начала ($i = 0$). Выжившим путям можно задать произвольные начальные значения без всякого влияния на эффективность декодера. В результате первые L декодированных ребер будут иметь случайные значения, не содержащие какой-либо информации. По этой причине величина L определяет задержку декодирования и известна как *глубина декодирования*. Более того, при условии, что L достаточно велико ($L>>\ell$, где $\ell > 5m$ для двоичного кода скорости 1/2), декодированные символы могут быть взяты из пути с наименьшей метрикой или из нулевого состояния пути ($\mathbf{y}^{(0)}$). Последний вариант легче реализовать и он не приводит к снижению эффективности декодирования. Программы на ECC интернет-сайте, реализующие декодирование по максимуму правдоподобия с применением алгоритма Витерби, работают именно таким образом.

Дополнение переводчика. *Эта свобода в выборе решения является следствием наблюдения (свойства декодера Витерби), сформулированного в начале раздела 5.4.2, а именно – на доста-*

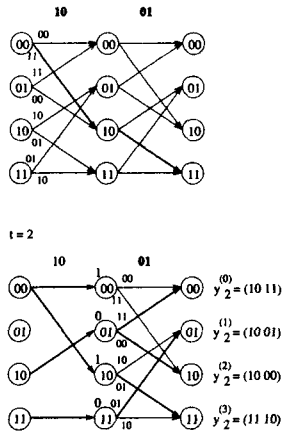


Рис. 41. Диаграмма вычислений декодера Витерби, Пример 62 , шаги $i = 2$.

точно большой глубине все выжившие пути имеют одинаковое начало.

Заметим еще, что в Примере 62 метки ребер (кодовые символы) запоминаются в выживших путях. Это сделано здесь только для того, чтобы облегчить понимание алгоритма. В практической реализации хранятся обычно соответствующие *информационные символы*. Это будет обсуждаться ниже в связи с задачей управления памятью путей. Строго говоря, нет необходимости запоминать какие-либо символы, так как они однозначно восстанавливаются из известного в каждом узле направления перехода (т.е. смены состояния решетки).

Синхронизация

Символы ребер решетки должны быть строго согласованы с последовательностью принятых символов. Любое несоответствие может быть обнаружено с помощью непрерывного наблюдения за значением некоторой случайной характеристикой декодера. Обычно используются следующие две характеристики: (1) *рост метрик путей* и (2) *оценка вероятности ошибки в канале* (BER). Статистика этих случайных величин

(*характеристик синхронизации*) позволяет обнаруживать ненормальное поведение декодера.

Предположим, что принятая последовательность смещена относительно меток, т.е. принятая последовательность $\mathbf{r}[i]$ не согласована (не синхронизирована) с метками ребер $\mathbf{v}[i]$.

Пример 63. На Рисунке 46 показан пример несинхронного приема последовательности $\mathbf{r}$ относительно (переданной) кодовой последовательности $\mathbf{v}$ скорости 1/2.

Другими словами, в этом случае не все биты принятой подпоследовательности $\mathbf{r}[i]$ принадлежат одному и тому же состоянию кодовой решетки в декодере. В этом случае возможны два события: (1) путевые метрики близки по величине и быстро возрастают, (2) оценка BER канала приближается к 1/2. На Рисунке 47 показана блок-схема декодера Витерби и измеритель (монитор) BER.

Здесь должен быть добавлен этап синхронизации, внешний по отношению к собственно декодеру, задачей которого является поиск правильного положения ребер последовательности $\mathbf{v}$ в декодере, при котором наблюдаемые статистики возвратятся к нормальному состоянию. Это может быть сделано (достигнуто) за счет выбрасывания некоторых принятых символов (не более $n-1$ раз), пока характеристики синхронизации не обнаружат нормальное поведение декодера. Этот процесс показан на Рисунке 46 для сверточного кода скорости 1/2 из Примера 63.

Нормализация метрик

Когда декодер Витерби работает непрерывно, метрики путей растут пропорционально длине принятой (обработанной) последовательности. Для того чтобы избежать переполнения или насыщения метрик (в зависимости от их числового представления), необходимо их нормализовать. Известны два способа. Оба используют следующие два свойства алгоритма Витерби:

1. выбор наилучшего пути (т.е. решения по максимуму правдоподобия) зависит только от *разностей метрик*,
2. разности метрик путей на решетке ограничены.

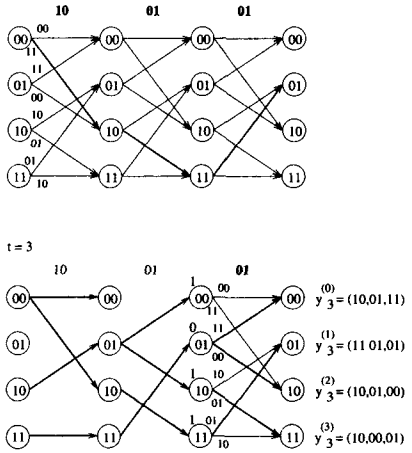


Рис. 42. Операции ВД на шаге 3 в Примере 62.

• **Пороговый способ**

Для нормализации метрик на каждом шаге декодирования значение наименьшей метрики пути

$$M_{\min} = \min_{0 \le k \le 2^m - 1} \{M(S_i^{(k)})\}$$

сравнивается с порогом T. Если $M_{\min} > T$, то величина T вычитается из всех накопленных метрик. Очевидно, что это действие не повлияет на результат выбора, так как разности метрик не изменились. Этот способ легко применяется в программной реализации декодера.

• **Модулярный способ**

Все метрики на каждом шаге алгоритма вычисляются по модулю N. Таким образом, значения метрик в каждом узле решетки находятся в диапазоне $[0, N-1]$, где $N = 2\Delta_{\max}$ и $\Delta_{\max}$ – максимальная разность метрик выживших путей. Очевидно, что $\Delta_{\max}$ зависит от значений отсчетов сигналов, принимаемых из канала. Учитывая оба приведенных выше свойства алгоритма Витерби, можно показать, что при вычислении метрик путей с помощью модулярной арифметики будет выбран тот же самый *путь максимального правдоподобия*. В частности, возможно применение арифметики *дополнения двойки*[7] таким образом, что возникающие переполнения не влияют на процесс выбора. Подробности можно найти в [НЕК, ОСС]. Этот способ преимущественно используется в аппаратной (микросхемной) реализации декодера Витерби.

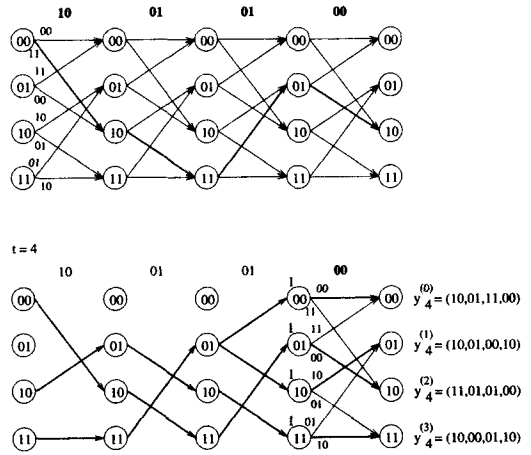


Рис. 43. Операции ВД, Пример 62, шаг 4.

Управление памятью путей

В декодере Витерби должны храниться и обновляться выжившие пути и их метрики. При непрерывном режиме работы декодера пока происходит обновление выживших путей на i-ом такте, выдается оценка наиболее вероятных информационных символов выживших путей на $(i\text{-}L)$-ом такте, где L глубина декодера. Важно отметить, что с высокой вероятностью (порядка (1 – *вероятность ошибки декодера*)) все выжившие пути сливаются в один, т.е. имеют общее начало при достаточно большой глубине декодера. Это означает, что наиболее вероятными являются символы, принадлежащие общей части выживших путей. Известны различные способы хранения путей

[7] Арифметика, использующая аддитивную группу $\{-2^{m-1}, 1-2^{m-1}, \ldots, 2^{m-1}-1\}$ m-разрядных целых чисел [НЕК].

и извлечения декодированных символов. Наиболее распространенными являются следующие два: (1) обмен между регистрами памяти, (2) обратный проход памяти путей.

• *Обмен между регистрами памяти*

Этот способ наиболее прост для программной реализации декодера. Все выжившие пути обновляются на каждом шаге алгоритма Витерби. Путь длины L, выбранный в некоторой вершине решетки и дополненный новым ребром, переписывается вместе с метрикой в регистр, сопоставленный этой вершине, из регистра, относящегося к предшествующей вершине. Таким образом, информационные (декодированные) символы могут быть считаны прямо из начальной части выжившего пути (причем любого регистра, например, всегда соответствующего нулевому состоянию решетки). Если же этот способ применяется в микросхемной реализации, то скорость декодирования может оказаться слишком низкой из-за слишком больших затрат времени на чтение и перезапись регистров. Для того чтобы упростить управление регистрами памяти, можно применить *циклический указатель* (пойнтер) с длиной цикла L. При использовании циклического указателя $(i-L)$-ая позиция в памяти путей (считывание декодированных символов) на i-ом шаге декодирования соответствует $((i+1) \bmod L)$ –ой позиции циклического регистра.

• *Обратный проход*

Этот способ предпочтителен при микросхемной реализации декодера. Выжившие пути набираются (составляются) из *решений*, каждое из которых представляет *переход на кодовой решетке* (или выжившее ребро). Выжившие пути реконструируются по *последовательности состояний* в обратном порядке.

Память *решений* для обратного прохода может быть организована как прямоугольный массив, k-ая строка которого соответствует состояниям $S^{(k)}{}_i$, $k = 0, 1, \ldots, 2^m-1$. На i-ом шаге декодирования в *память решений* (или память обратного прохода TB_RAM) каждого состояния $S^{(j)}{}_i$, $0 \le j \le 2^m-1$, решетки записывается крайний правый бит предыдущего состояния $S^{(b_j)}{}_{i-1}$,

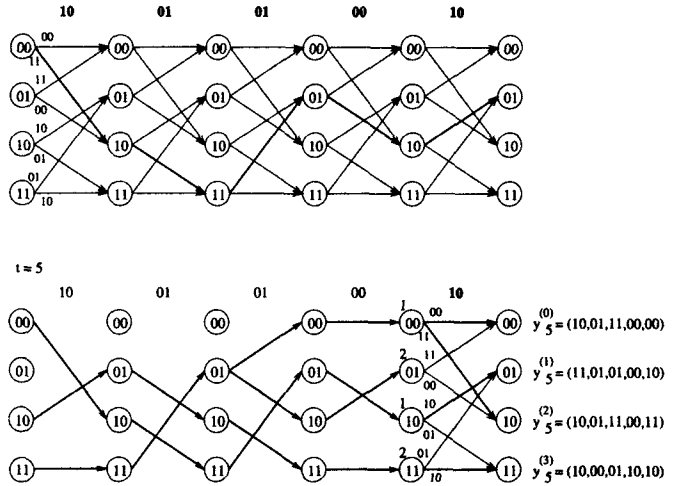


Рис. 44. Операции ВД, Пример 62, шаг 5.

$j \in \{1, 2\}$, ассоциированного с выбранным ребром (связывающим эти состояния, см. описание блока ACS, раздел 5.4.2).

Метод обратного прохода обеспечивает обмен между памятью и скоростью декодирования, так как записывается только один бит на состояние на каждом шаге декодирования, вместо L бит на состояние в случае обмена регистров памяти. Информационные биты восстанавливаются с помощью просмотра в обратном порядке последовательности переходов (на состояниях решетки) и использования копии кодера (или просмотра таблицы переходов кодера). Для реализации непрерывного режима работы декодера требуется дополнительная память (дополнительные столбцы прямоугольной памяти), так как в одно и то же время считываются информационные символы (декодированные) и записываются новые решения.

Пример 64. Продолжим Пример 62 применением процедуры обратного прохода. После приема из канала пары бит на шаге $i = 6$ содержимое памяти обратного прохода показано в Таблице 3.

Память обратного прохода считывается, начиная с последнего (записанного) бита (т.е. в режиме LIFO) , $i = 6$ (в общем

случае указатель устанавливается в состояние L). Адрес строки определяется состоянием с лучшей метрикой. В Примере это состояние S^0. Считывается *бит перехода* b_6 (в общем случае b_L). Адрес строки (ROW) на шаге i=5 (i=L−1) для считывания следующего бита перехода b_5 образуется сдвигом адреса k, соответствующего моменту i=6 (в общем случае i=L), и присоединением к нему предыдущему биту решения b_6 (в общем случае b_L). Это может быть выполнено с помощью следующей программной строки на языке C:

```
ROW[j]=((ROW[j+1]<<1)&&MASK)^TB_RAM[ROW[j+1]][j+1];
```

(где MASK=2^m − 1).

Продолжая таким же образом, получаем следующую последовательность состояний:

$$S_6^{(0)} \to S_6^{(1)} \to S_6^{(2)} \to S_6^{(1)} \to S_6^{(3)} \to S_6^{(2)} \to S_6^{(0)}$$

Из этой последовательности восстанавливается соответствующая ей последовательность информационных символов, с помощью обратного считывания последовательности переходов, которая принимает следующий вид: **u** = (110100).

Для высокоскоростной реализации декодера Витерби память обратного прохода должна быть разделена на несколько блоков. В этом случае, пока в один блок происходит запись решений (о выборе ребер) текущего шага декодирования i, из другого блока считываются декодированные символы, соответствующие моменту i-L, $L>\ell>5m$, а еще в одном (или более чем в одном) блоке выполняются операции обратного прохода. Подобное разделение памяти увеличивает скорость вычислений, но одновременно увеличивает и задержку декодирования декодера [Col, BABSZ].

Дополнение переводчика

Следует заметить, что память обратного прохода и сам обратный проход легко реализуются в виде конвейерной структуры с естественным разделением по секциям решетки. Обозначим $R=k_0/n_0$ скорость сверточного кода. Рассмотрим работу конвейерной структуры, в ПСВ блок которой вводятся Vn_0 бит и на выходе схемы обратного прохода накапливаются Vk_0 бит решений. Уравнение равновесия этой структуры может быть записано как: $(L+V)k_0\tau \le Vn_0\tau_{псв}$, или $LR\tau \le V(1-R)\tau_{псв}$, или $LR \le V(1-R)$, где τ величина соответствующего тактового интервала.

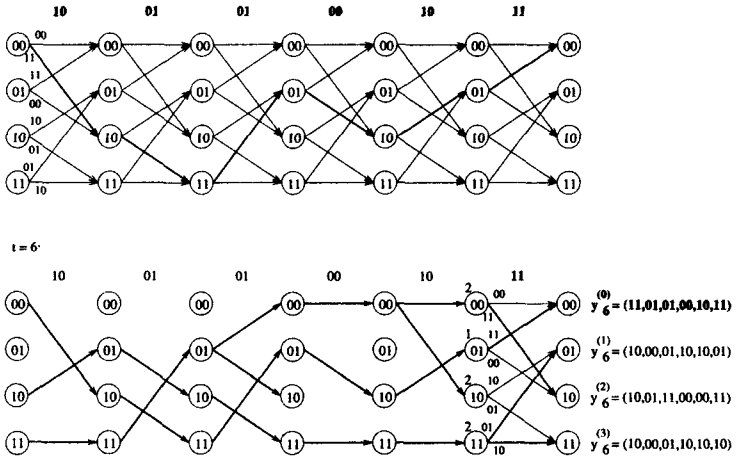


Рис. 45. Операции ВД, Пример 62, шаг 6.

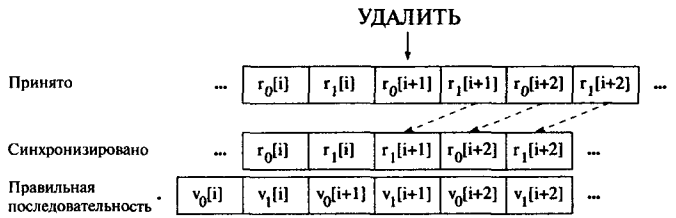


Рис. 46. Пример рассинхронизации ребер декодера Витерби.

Таблица 3. Обратный проход пути, законченного в состоянии S^0.

k	i = 1	i = 2	i = 3	i = 4	i = 4	i = 6
0	0	1	1	0	0	1
1	0	1	1	0	0	1
2	0	1	1	1	0	0
3	1	0	0	1	1	1

Рис. 47. Оценка вероятности ошибки в двоичном симметричном канале.

ПСВ

Для двоичных сверточных кодов памяти m и скорости $1/n$ базовым элементом треллиса является *бабочка*, показанная на Рисунке 48. Операция *прибавить-сравнить-выбрать* (ПСВ) реализуется с помощью этой структуры на 2^{m-1} парах состояний. Таким образом, операции ПСВ могут быть выполнены либо последовательно (в программном цикле), либо параллельно с помощью 2^{m-1} вычислительных блоков (по одному на бабочку). Если в данном коде для каждого кодового слова имеется противоположное (antipodal код), то генераторы кода имеют единицу в первой и последней позиции. В этом случае метки ребер, инцидентных состоянию $S^{(2j)}$, совпадают с метками ребер, инцидентных состоянию $S^{(2j+1)}$, принадлежащему той же самой бабочке. Более того, метки ребер, инцидентных состоянию $S^{(2j)}$, равны дополнению меток другого ребра[8]. Эти факты использованы в [FL2] для реализации техники вычислений, основанной на разности реберных метрик. В результате, для кодов скорости $1/n$ получается декодер Витерби, имеющий архитектуру *сравнить-прибавить-выбрать* (СПВ) и меньшую сложность реализации устройства, чем обычный ПСВ (*прибавить-сравнить-выбрать*) подход.

5.5. Перфорированные сверточные коды

Перфорированные сверточные коды были введены в [Cla]. Перфорация кода состоит в систематическом удалении из процесса передачи в канал некоторых битов (символов) с выхода низкоскоростного кодера[9]. Так как структура решетки низкоскоростного кодера не изменяется, то количество информационных символов не изменяется. В результате, выходная последовательность принадлежит *перфорированному сверточному* (РС) *коду* более высокой скорости. Последующее обсуждение относится к кодам, получаемым из двоичных сверточных кодов памяти m и скорости $1/n$.

Матрица перфорации P задает правило удаления выходных символов. Матрица P есть k x n_p двоичная матрица, элементы которой p_{ij} указывают, что соответствующий выходной двоичный символ будет передан ($p_{ij} = 1$) или нет ($p_{ij} = 0$). Обычно, правило перфорации является периодическим. РС кодер скорости k/n_p, основанный на кодере скорости $1/n$, имеет матрицу перфорации P, которая содержит ℓ нулей, где $n_p = kn - \ell$, $0 <= \ell < kn$.

Пример 65. Сверточный код памяти 2 скорости 2/3 может быть построен с помощью перфорации выходных бит кодера на Рисунке 29 согласно матрице перфорации

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

Соответствующий кодер показан на Рисунке 50. Кодовая последовательность

$$\mathbf{v} = (\cdots, v_i^0 v_i^1, v_{i+1}^0 v_{i+1}^1, v_{i+2}^0 v_{i+2}^1, v_{i+3}^0 v_{i+3}^1, \cdots)$$

кодера скорости 1/2 преобразуется в кодовую последовательность

$$\mathbf{v}_p = (\cdots v_i^0 v_i^1, v_{i+1}^0, v_{i+2}^0 v_{i+2}^1, v_{i+3}^0, \cdots)$$

где удален второй бит каждого второго ребра.

Одной из возможных целей перфорации является возможность использования одного и того же декодера для всего множе-

---

[8] Дополнение бита a равно $(1+a) \bmod 2$

[9] Иногда исходный код называют *материнским кодом*.

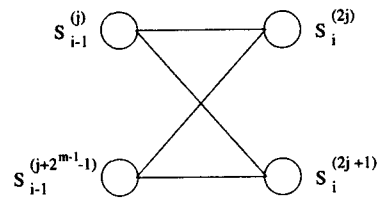


Рис. 48. Бабочка сверточного кода памяти m, скорости $1/n$.

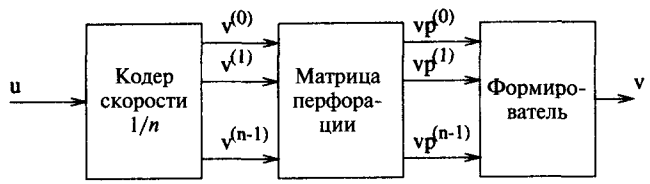


Рис. 49. Кодер сверточного кода, основанный на коде скорости $1/n$.

ства перфорированных кодов. Один способ декодирования перфорированного кода с помощью декодера Витерби низкоскоростного кода состоит в восстановлении «удаленных» символов на своих позициях (на которых они не были переданы) в качестве стираний. Этот процесс называют *деперфорацией*. Восстановленные как стирания «удаленные» символы метятся специальным флагом. Это может быть сделано, например, с помощью дополнительного бита, устанавливая его в 1 на позициях стираний.

Таблица 4. Матрица перфорации стандартного кода скорости 1/2

Скорость	Матрица	Кодовая последовательность	d_f
1/2	$\begin{pmatrix}1\\1\end{pmatrix}$	$v_i^{(0)}, v_i^{(1)}$	10
2/3	$\begin{pmatrix}1&0\\1&1\end{pmatrix}$	$v_i^{(0)}, v_i^{(1)}, v_{i+1}^{(1)}$	6
3/4	$\begin{pmatrix}1&0&1\\1&1&0\end{pmatrix}$	$v_i^{(0)}, v_i^{(1)}, v_{i+1}^{(1)}, v_{i+2}^{(0)}$	5
5/6	$\begin{pmatrix}1&0&1&0&1\\1&1&0&1&0\end{pmatrix}$	$v_i^{(0)}, v_i^{(1)}, v_{i+1}^{(1)}, v_{i+2}^{(0)}, v_{i+3}^{(1)}, v_{i+4}^{(0)}$	4
7/8	$\begin{pmatrix}1&0&0&0&1&0&1\\1&1&1&1&0&1&0\end{pmatrix}$	$v_i^{(0)}, v_i^{(1)}, v_{i+1}^{(1)}, v_{i+2}^{(1)}, v_{i+3}^{(1)}, v_{i+4}^{(0)}, v_{i+5}^{(1)}, v_{i+6}^{(0)}$	3

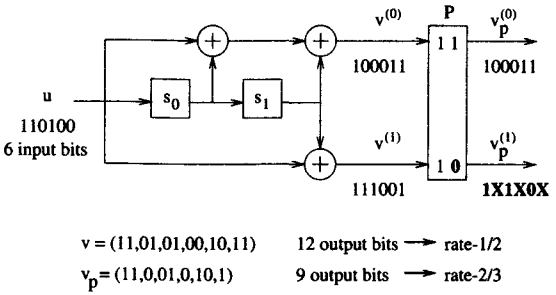


Рис. 50. Кодер перфорированного сверточного кода скорости 2/3 с памятью 2.

Если позиция бита помечена флагом, то символ на этой позиции не учитывается при вычислении метрики ребра[10].

При программной реализации применяется другой метод, состоящий в проверке элементов $p_{m,j}$ матрицы P в порядке передачи символов, $m = 0, 1, \ldots, n-1, j = i, i+1, \ldots, i+k-1$. Если $p_{m,j} = 0$, то соответствующий ему (не переданный) символ не учитывается при обновлении метрики ребра. Декодер перемещает указатель позиции на следующий символ ребра решетки и повторяет проверку $p_{m,j}$. Если же $p_{m,j} = 1$, то принятый символ используется для подсчета метрики. Каждый раз, когда передвигается указатель позиции символа на ребре, выполняется проверка того, что обработаны все символы ребра. Если это так, то выполняются операции ПСВ. Этот метод использован в некоторых программных реализациях декодера Витерби для перфорированных кодов на ECC интернет-сайте.

В Таблице 4 показаны матрицы перфорации, применяемые к стандартному (де-факто) сверточному коду памяти 6 и скорости 1/2 с генераторами $(\mathbf{g}_0, \mathbf{g}_1) = (171, 133)$. В этой таблице нижний индекс элемента кодовой последовательности соответствует номеру такта, а верхний индекс соответствует генератору кода $\mathbf{g}_i$, $i = 0, 1$. Другие матрицы перфорации можно найти в [YKH].

[10] Некоторые авторы пишут, что «метрика ребра равна нулю», в качестве эквивалентного утверждения.

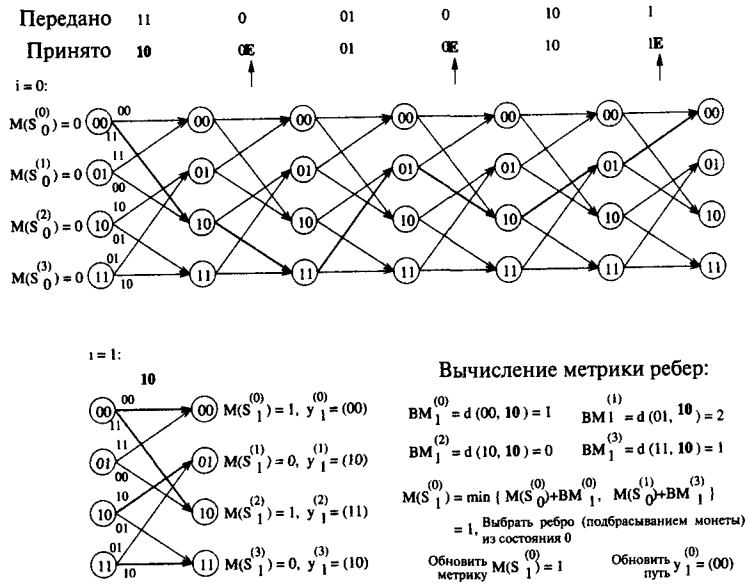


Рис. 51. Работа декодера Витерби для перфорированного кода (РСС) скорости 2/3, памяти 2, $i = 1$.

Пример 66. (Продолжение Примера 65). Пусть вектор **u** тот же самый, что и в Примере 62. Предположим, что при передаче сообщения по ДСК произошла ошибка во второй позиции. Пусть, по условию перфорации, кодовые символы $v^{(1)}[i]$, $i = 0,2,4,...$, не передавались. Декодер, которому известен этот факт, восстанавливает вместо них символы, помеченные флагом E. При вычислении реберных метрик эти символы игнорируются. Обработка декодером Витерби удаленных символов показана на Рисунках 51 и 52 для $i = 1$ и $i = 2$, соответственно. Читателю предлагается закончить этот пример в качестве упражнения.

5.5.1. Соображения по реализации перфорированных сверточных кодов

В системе, использующей перфорированные сверточные коды, процесс синхронизации становится более продолжительным, так как выходные символы распределены по нескольким ребрам (секциям) решетки. Таким образом, декодеру потребуется проверка гипотез о скорости кода и примененной матрице (маске) перфорации. Другими проблемами реализации перфорированных сверточных кодов являются:

1. Необходимо *увеличение глубины декодирования* L. Увеличение должно быть тем больше, чем больше символов удалено. Это связано с тем, что высокоскоростные коды (например, скорости 7/8) имеют малое кодовое расстояние Хемминга (в этом примере $d_{\min} = 3$) и выжившие пути сливаются через большее количество шагов.
2. Необходим *дополнительный уровень синхронизации* принятой последовательности по маске перфорации.

Зависимость длины маски перфорации от скорости перфорированного кода может быть устранена за счет использования одинакового *периода перфорации*. Эта возможность реализована в *перфорированных кодах, совместимых по скорости* (rate-compatible PC или RCPC).

5.5.2. RCPC коды

Совместимые по скорости перфорированные коды были предложены Хагенауэром в [Hag]. Эти коды строятся из низкоскоростных кодов с помощью периодической перфорации. Обозначим M период перфорации. Тогда, как и раньше, маска перфорации представляется двоичной $n \times M$ матрицей. Однако, в общем случае, $M > kn - \ell$.

Для построения семейства RCPC кодов необходимо выполнение следующего условия. Обозначим P^H и P^L матрицы (маски) перфорации высокоскоростного кода C^H и низкоскоростного кода C^L , соответственно, причем оба кода получены из одного и того же (материнского) кода меньшей скорости. Тогда коды C^H и C^L *совместимы по скорости*, если

$$p_{ij}^{(H)} = 1 \rightarrow p_{ij}^{(L)} = 1$$

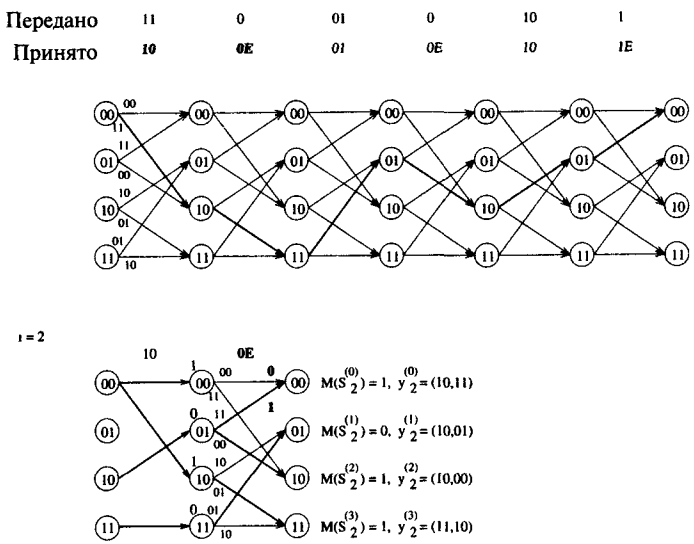


Рис. 52. Работа декодера Витерби для РСС скорости 2/3 и памяти 2, шаг $i = 2$.

Эквивалентным условием является следующее:

$$p_y^{(L)} = 0 \rightarrow p_y^{(H)} = 0$$

Пример 67. Пусть C сверточный код скорости 1/2. Пусть $M = 4$. Тогда матрицы $P(1) - P(4)$, представленные ниже, порождают RCPC коды скорости 4/5, 4/6, 4/7, 4/8 (=1/2), соответственно.

$$P(1)=\begin{pmatrix}1&1&1&0\\1&0&0&1\end{pmatrix},\quad P(2)=\begin{pmatrix}1&1&1&0\\1&\mathbf{1}&0&1\end{pmatrix},$$

$$P(3)=\begin{pmatrix}1&1&1&\mathbf{1}\\1&1&0&1\end{pmatrix},\quad P(4)=\begin{pmatrix}1&1&1&1\\1&1&\mathbf{1}&1\end{pmatrix}.$$

RCPC коды [Hag] нашли применение в системах с автоматическим переспросом (ARQ), благодаря их способности к наращиванию избыточности, а также возможности построения кодов с переменной скоростью или кодов *с неравной защитой*.

Глава 6

МОДИФИКАЦИЯ И КОМБИНИРОВАНИЕ КОДОВ

В этой главе рассматриваются некоторые способы модификации существующих кодов или их комбинирования, с помощью которых достигается известная гибкость в конструировании кодов, исправляющих ошибки. Многие из наилучших на сегодня кодов получены не как представители известного семейства кодов, а с помощью модификации и комбинирования [Bro].

6.1. Модификация кодов

Обозначим C линейный блоковый (n, k, d) код над $\mathbf{GF}(q)$ с порождающей матрицей $\mathbf{G}$ и проверочной матрицей $\mathbf{H}$.

6.1.1. Укорочение кодов

Укороченные коды были описаны в Разделе 3.1.5. Пусть s целое число, $0 \le s < k$. В общем случае, линейный укороченный $(n\text{-}s, k\text{-}s, d_s)$ код C_s имеет расстояние $d_s \ge d$. Порождающая матрица кода C_s может быть получена из матрицы $\mathbf{G}$ исходного кода следующим образом. Предположим, что матрица $\mathbf{G}$ задана в систематической форме, т.е.

$$\mathbf{G} = (\mathbf{I}_k \mid \mathbf{P}) \qquad (6.1)$$

Тогда $(k\text{-}s)\times(n\text{-}s)$ порождающая матрица $\mathbf{G}_s$ укороченного кода C_s может быть получена удалением s столбцов единичной матрицы $\mathbf{I}_k$ и s строк, соответствующих ненулевым элементам выбранных (удаляемых) столбцов. Эта операция показана на примере.

Пример 68. Рассмотрим (7,4,3) код Хемминга из Примера 13 Главы 2. Этот код задан порождающей матрицей

$$\mathbf{G} = \begin{pmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} & 1 & 0 & 1 & 1 & 0 \\ \mathbf{0} & \mathbf{0} & 0 & 1 & 0 & 1 & 1 \end{pmatrix} \tag{6.2}$$

(Предлагается сравнить с **H** в Примере 13.)

Для построения укороченного (5, 2, 3) кода можно удалить любые два из четырех левых столбцов матрицы **G**. Положим, что удаляются первые два столбца и, следовательно, первая и вторая строки матрицы **G**. Эти столбцы и строки отмечены жирным шрифтом в (6.2). Оставшиеся элементы образуют матрицу

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

Очень полезно изучить стандартную таблицу укороченного (5,2,3) кода из Примера 68, чтобы понять усиление корректирующих свойств укороченного кода по отношению к исходному.

Пример 69. Стандартная таблица укороченного (5, 2, 3) кода из Примера 68 представлена в Таблице 5.

Таблица 5. Стандартная таблица укороченного (5,2,3) кода.

s	u = 00	u = 10	u = 01	u = 11
000	00000	10110	01011	11101
110	10000	00110	11011	01101
011	01000	11110	00011	10101
100	01100	10010	01111	10101
010	00010	10100	01001	11111
001	00001	10111	01010	11100
101	11000	01110	10011	00101
111	01100	11010	00111	10001

Из стандартной таблицы следует, что имеется два сочетания ошибок веса Хемминга два, а именно: 11000 и 01100, которые могут быть исправлены, хотя минимальное расстояние кода осталось равным 3.

Заметим, что описанная выше операция укорочения кода уменьшает его длину и размерность, тогда как число проверочных символов остается прежним. Следовательно, должно исправляться большее число комбинаций ошибок. Это легко доказать с помощью границы Хемминга (1.24) для линейного блокового (n, k, d) кода, исправляющего t ошибок, которая приводится ниже для удобства

$$2^{n-k} \geq \sum_{l=0}^{t} \binom{n}{l} \tag{6.3}$$

По отношению к исходному (n, k, d) коду его укороченная версия $(n\text{-}s, k\text{-}s, d_s)$ имеет ту же избыточность. Следовательно, левая часть неравенства (6.3) имеет то же самое значение. Другими словами, *число смежных классов кода не изменилось*. Но, с другой стороны, правая часть при $s > 0$ стала меньше. Другими словами, число комбинаций ошибок веса t или меньше уменьшилось. Если исходный код не удовлетворял границе Хемминга (6.3) с равенством (а, как известно, среди двоичных кодов только коды Хемминга, коды Голея, коды с одной проверкой и коды повторения удовлетворяют равенству), то разность

$$2^{n-k} - \sum_{i=0}^{t} \binom{n}{i}$$

представляет те дополнительные комбинации ошибок веса большего t, которые могут быть исправлены исходным кодом. Число дополнительных комбинаций ошибок, которые исправляются укороченным кодом, равно

$$\Delta_t = 2^{n-k} - \sum_{l=0}^{t} \binom{n-s}{l} - \left(2^{n-k} - \sum_{l=0}^{t} \binom{n}{l}\right) = \sum_{l=0}^{t} \left[\binom{n}{l} - \binom{n-s}{l}\right] \tag{6.4}$$

т.е. равно разности объемов Хемминговых сфер, имеющих одинаковый радиус t, но различную размерность: n и $n\text{-}s$.

6.1.2. Расширение

В общем случае, расширение кода C означает добавление ε проверочных символов. Расширенный $(n+\varepsilon,\ k,\ d_{\text{ext}})$ код C_{ext} имеет минимальное расстояние $d_{\text{ext}} \geq d$. Расширенная проверочная $(n\text{-}k+\varepsilon) \times (n+\varepsilon)$ матрица получается из проверочной матрицы $\mathbf{H}$ кода C добавлением ε строк и столбцов,

$$\mathbf{H}_{\text{ext}} = \begin{pmatrix} h_{1,1} & \cdots & h_{1,\varepsilon} & \cdots & h_{1,n+\varepsilon} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ h_{\varepsilon,1} & \cdots & h_{\varepsilon,\varepsilon} & \cdots & h_{\varepsilon,n+\varepsilon} \\ \vdots & \ddots & \vdots & & H \\ h_{n-k+\varepsilon,1} & \cdots & h_{n-k+\varepsilon,} & & \end{pmatrix} \tag{6.5}$$

Наиболее известный способ расширения кода состоит в добавлении общей проверки на четность. В этом случае расширенная матрица имеет следующий вид,

$$\mathbf{H}_{\text{ext}} = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 0 & & & \\ 0 & & H & \\ 0 & & & \end{pmatrix} \tag{6.6}$$

В результате получаем $(n+1,\ k,\ d_{\text{ext}})$ код C_{ext}. Если минимальное расстояние исходного кода нечетно, то $d_{\text{ext}} = d+1$.

Пример 70. Пусть C есть (7,4,3) код Хемминга. Тогда расширенный (8,4,4) код C_{ext} имеет проверочную матрицу

$$\mathbf{H}_{\text{ext}} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{pmatrix} \tag{6.7}$$

Перестановкой столбцов эта матрица преобразуется в порождающую матрицу кода Рида-Маллера $\text{RM}_{1,3}$ из Примера 16, который является *самодуальным кодом*[1].

[1] См. раздел 1.3.1, Пример 3.

6.1.3. Перфорация (выкалывание)

Техника перфорации кодов рассматривалась в разделе 5.5 в связи со сверточными кодами. В общем случае, перфорация линейных блоковых кодов состоит в удалении проверочных символов, что приводит к линейному блоковому $(n\text{-}p,\ k,\ d_p)$ коду C_p с минимальным расстоянием $d_p < d$. Скорость кода возрастает, так как размерность кода не изменяется, а избыточность (число проверок) уменьшается.

Проверочная матрица $\mathbf{H}_p$ перфорированного кода C_p получается удалением определенных столбцов проверочной матрицы H исходного кода. Эта техника эквивалентна укорочению дуального кода. Если

$$\mathbf{H} = (\mathbf{p}_0\, \mathbf{p}_1 \cdots \mathbf{p}_{k-1} \,|\, \mathbf{I}_{n-k})$$

есть проверочная матрица кода C, то удалением любых p столбцов среди $n-k$ правых столбцов матрицы $\mathbf{H}$ и p строк, соответствующих ненулевым элементам выбранных для удаления столбцов, получаем $(n-k-p) \times (n-p)$ матрицу

$$\mathbf{H}_p = (\mathbf{p}'_0\, \mathbf{p}'_1 \cdots \mathbf{p}'_{k-1} \,|\, \mathbf{I}_{J_0} \mathbf{I}_{J_1} \cdots \mathbf{I}_{J_k-p-1})$$

где через $\mathbf{I}_j$ обозначен вектор-столбец веса 1, а через $\mathbf{p}'^{\ell}_j$ обозначен столбец, оставшийся после определенного выше удаления p строк.

Пример 71. Рассмотрим (5,2,3) код C_s из Примера 68. Его проверочная матрица равна

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

Удаление третьего столбца и верхней строки дает проверочную матрицу $\mathbf{H}_p$ перфорированного (4,2,2) кода C_p,

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

который совпадает с LUEP (НРЗ) кодом из Примеров 3 и 6 с точностью до перестановки первой и второй позиций кодового слова.

6.1.4. Пополнение и выбрасывание.

Известны еще два способа модификации существующих линейных кодов, которые могут привести к нелинейным кодам. Эти способы, известные как *пополнение* и *выбрасывание*, очень полезны в анализе некоторых классов кодов, таких как коды Рида-Маллера и БЧХ.

Пополнение кода означает добавление кодовых слов к исходному коду. Очевидным способом выполнения этой модификации при сохранении линейности кода является добавление (линейно независимых) строк в порождающую матрицу. Это эквивалентно построению *суперкода*, являющегося объединением *смежных классов* исходного кода C. Построенный таким образом код C_{aug} имеет параметры $(n, k+\delta, d_{aug})$, где δ число добавленных строк. Минимальное расстояние C_{aug} есть $d_{aug} < d$. Пусть $\mathbf{G}=(\mathbf{g}_0{}^T,..., \mathbf{g}_{k-1}{}^T,)^T$ порождающая матрица кода C, где $\mathbf{g}_i{}^T$ транспонированный вектор длины n , $0 \le i < n$. Тогда порождающая матрица пополненного кода равна

$$\mathbf{G}_{aug} = \begin{pmatrix} \mathbf{p}_0 \\ \vdots \\ \mathbf{p}_{\delta-1} \\ \mathbf{G} \end{pmatrix} \qquad (6.8)$$

Техника пополнения тесно связана с *разложением кода на смежные классы* [For6]. Эта техника будет рассматриваться ниже в связи с обсуждением прямой суммы кодов, конструкций $[\mathbf{u}|\mathbf{u}+\mathbf{v}]$ и близких к ним идей. Общеизвестным способом пополнения линейного блокового (n, k, d) кода C является добавление кодового слова из всех единиц (если таковое уже не принадлежит коду). И наоборот, если слово из всех единиц принадлежит коду C, то

$$C = C_0 \bigcup \{\mathbf{1} + C_0\}$$

где $(n, k-1, d_0)$ подкод $C_0 \subset C$ и $d_0 = d-1$.

Таблица 6. Основные методы модификации кодов.

Метод	Действие	Параметры кода
Укорочение	Удаление информационных символов	$(n-l, k-l, d_s \ge d)$
Удлинение	Добавление информационных и проверочных символов	$(n+l, k+l, d_s \le d)$
Расширение	Добавление проверочных символов	$(n+l, k, d_{ext} \ge d)$
Выкалывание	Удаление проверочных символов	$(n-l, k, d_p \le d)$
Дополнение	Добавление кодовых слов	$(n, k+l, d_{aug} \le d)$
Выбрасывание	Удаление кодовых слов	$(n, k-l, d_{ext} \ge d)$

Выбрасывание состоит в удалении некоторых кодовых слов из кода (из множества кодовых слов). В общем случае, это приводит к нелинейному коду. Способ, сохраняющий линейность, состоит в удалении строк порождающей матрицы $\mathbf{G}$ исходного кода. В результате получается код C_{exp} с параметрами $(n, k-l, d_{exp})$, $d_{exp} \ge d$. Следует заметить, что для систематического кода операции выбрасывания и укорочения совпадают.

Пример 72. Рассмотрим (7,4,3) код Хемминга с порождающей матрицей

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} \qquad (6.9)$$

Удаление первой строки приводит к (7,3,3) коду, первый символ которого всегда нулевой (и, следовательно, должен быть удален тоже). Таким образом, получаем код, эквивалентный (6,3,3) коду.

Важно отметить, что получение наибольшего кодового расстояния после операции выбрасывания зависит от выбора порождающей матрицы, или *базиса исходного кода*. Эта зависимость рассмотрена в следующем примере.

Пример 73. Как показано в Примере 39 код Хеммнга (7,4,3) содержит семь кодовых слов веса 3 и столько же слов веса 4.

Выбирая четыре линейно независимых кодовых слова веса 4 и одно веса 3, получаем следующую порождающую матрицу,

$$\mathbf{G}'=\begin{pmatrix}1&0&0&0&1&0&1\\1&1&0&0&1&1&0\\0&1&1&1&1&0&0\\0&1&1&0&0&1&1\end{pmatrix} \qquad (6.10)$$

Матрица (6.10) порождает (7.4.3) код Хемминга из предыдущего Примера. Однако при удалении верхней строки получается порождающая матрица (7,4,3) кода максимальной длины (или *симплексного кода*).

Операция *удлинения* кода выполняется добавлением l столбцов и l строк к порождающей матрице, т.е. добавляются l проверочных и l информационных символов. В частности, обычный способ удлинения кода состоит в добавлении одного информационного символа и общей проверки на четность [MS].

Рассмотренные способы модификации кодов сведены в Таблицу 6.

6.2. Комбинирование кодов

В этом разделе представлены некоторые способы комбинирования кодов. С помощью этой техники можно получить очень сильные результаты, что подтверждается, например, появлением в 1993 г. *турбо кодов* [BGT]. В дальнейшем, без дополнительных указаний, C_i означает линейный блоковый код с параметрами (n_i, k_i, d_i), $i = 1,2$.

6.2.1. Последовательное соединение (time-sharing) кодов

Рассмотрим два кода C_1 и C_2. Тогда *последовательное соединение* кодов C_1 и C_2 эквивалентно поочередной передаче кодовых слов $\mathbf{c}_1 \in C_1$ и $\mathbf{c}_2 \in C_2$,

$$|C_1|C_2| = \{(\mathbf{c}_1, \mathbf{c}_2): \mathbf{c}_i \in C_i,\ i = 1,2\} \qquad (6.11)$$

Результатом последовательного соединения (*чередования*) m линейных блоковых (n_i, k_i, d_i) кодов, $i = 1,2,...,m$, является (n, k, d) код с параметрами

$$n = \sum_{i=1}^{m} n_i, \quad k = \sum_{i=1}^{m} k_i, \quad d = \min_{1 \le i \le m}\{d_i\} \qquad (6.12)$$

Обозначим $\mathbf{G}_i$ порождающую матрицу *компонентного кода* C_i, $i = 1, 2, \ldots, m$. Тогда порождающая матрица последовательного соединения кодов имеет вид:

$$\mathbf{G}_{TS} = \begin{pmatrix}\mathbf{G}_1 & & & \\ & \mathbf{G}_2 & & \\ & & \ddots & \\ & & & \mathbf{G}_m\end{pmatrix} \qquad (6.13)$$

где незаполненные ячейки являются нулевыми.

Последовательное соединение кодов (time-sharing) иногда называют «прямой суммой» кодов [MS] или «каскадным соединением» (*concatenation*) [Rob]. Однако, в этой книге термин «каскадное соединение» кодов имеет другой смысл, который обсуждается в разделе 6.2.4.

Пример 74. Пусть C_1 код-повторение (4,1,4), а C_2 код Хемминга (7,4,3). Тогда *последовательное соединение* этих кодов дает линейный блоковый (11,5,3) код с порождающей матрицей

$$\mathbf{G}_{TS} = \begin{pmatrix}\mathbf{G}_1 & \\ & \mathbf{G}_2\end{pmatrix} = \begin{pmatrix}1&1&1&1&0&0&0&0&0&0&0\\0&0&0&0&1&0&0&0&1&0&1\\0&0&0&0&0&1&0&0&1&1&1\\0&0&0&0&0&0&1&0&1&1&0\\0&0&0&0&0&0&0&1&0&1&1\end{pmatrix}$$

Техника чередования кодов широко используется в системах связи, требующих разный уровень защиты от ошибок, или *неравную защиту от ошибок*, как, например, в RCPC кодах

Раздела 5.5.2, см. [Hag]. Заметим еще, что m – кратное последовательное соединение одного и того же кода эквивалентно m – кратной повторной передаче кодового слова. Более подробно это будет рассмотрено при обсуждении кодов-произведений в Разделе 6.2.3.

6.2.2. Прямые суммы кодов

Пусть C_i означает линейный блоковый код с параметрами (n_i, k_i, d_i), $i = 1, 2, \ldots, m$. Прямая сумма кодов C_{DS} определена как

$$C_{\mathrm{DS}} = \{\mathbf{v} \mid \mathbf{v} = \mathbf{v}_1 + \mathbf{v}_2 + \cdots + \mathbf{v}_m, \mathbf{v}_i \in C_i, i = 1,2,\ldots,m\}$$

Эта техника позволяет увеличить размерность кода. Однако, при этом обычно убывает расстояние. Обозначим $\mathbf{G}_i$ порождающую матрицу компонентного кода C_i, для $i = 1, 2, \ldots, m$. Тогда порождающая матрица кода, построенного как прямая сумма компонентных кодов, $C_{\mathrm{DS}} = C_1 + C_2 + \ldots + C_m$, равна

$$\mathbf{G}_{\mathrm{DS}} = \begin{pmatrix} \mathbf{G}_1 \\ \mathbf{G}_2 \\ \vdots \\ \mathbf{G}_m \end{pmatrix} \tag{6.14}$$

Код C_{DS} является линейным блоковым (n, k, d) кодом размерности $k \le k_1 + k_2 + \ldots + k_m$ с кодовым расстоянием $d \le \min_i\{d_i\}$.

Пример 75. Пусть C_1 код-повторение (4,1,1) и C_2 линейный блоковый (4,2,2) код с порождающей матрицей

$$\mathbf{G}_2 = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

(Этот код эквивалентен двукратному повторению двух-битного сообщения) Тогда код $C_{\mathrm{DS}} = C_1 + C_2$ является (4,3,2) кодом с одной проверкой и порождающей матрицей

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

Техника прямой суммы кодов может быть использована не только для комбинирования кодов малой размерности, но и для *разложения* некоторого кода в *объединение подкодов* $C_i \subset C$ таких, что C может быть представлен прямой суммой компонентных подкодов. Дополнения см. в Разделе 6.2.4.

Очевидно, что любой линейный блоковый (n, k, d) код C с порождающей матрицей $\mathbf{G}$ может разложен в композицию k линейных блоковых $(n, 1, d_i)$ подкодов C_i, $1 \le i \le k$. Каждый из этих подкодов имеет порождающую матрицу, состоящую из одной строки $\mathbf{g}_i$ матрицы $\mathbf{G}$. Однако известна техника, которая дает разложение кода на подкоды большей размерности с известным минимальным расстоянием. Эта техника рассматривается в следующем разделе.

Конструкции $|u|u + v|$ и соответствующая техника

Комбинирование последовательного соединения кодов с прямой суммой приводит к интересным конструкциям кодов. Первой из них является конструкция |u|u+v| [Plo]. Эта конструкция основана на двух кодах C_1 и C_2 длины $n = n_1 = n_2$ и состоит в формировании кода[2] $C = |C_1|C_1 + C_2|$ с порождающей матрицей

$$\mathbf{G} = \begin{pmatrix} \mathbf{G}_1 & \mathbf{G}_1 \\ \mathbf{0} & \mathbf{G}_2 \end{pmatrix} \tag{6.15}$$

Код C имеет параметры $(2n, k_1 + k_2, d)$. Минимальное кодовое расстояние кода C равно $d = \min\{2d_1, d_2\}$. Этот факт может быть доказан для любых двоичных векторов $\mathbf{x}$ и $\mathbf{y}$ с помощью неравенства треугольника в следующей форме: $\mathrm{wt}(\mathbf{x} + \mathbf{y}) \le \le \mathrm{wt}(\mathbf{x}) - \mathrm{wt}(\mathbf{y})$.

[2] Если длины кодов не одинаковы, например, $n_1 > n_2$, тогда в конце слова кода C_2 добавляется $n_2 - n_1$ нулей.

Конструкция |u|u+v| позволяет дать удобное описание кодов Рида-Маллера.

Коды Рида-Маллера

$$RM(r+1,m+1)=|RM(r+1,m)\,|\,RM(r+1,m)+RM(r,m)|$$

Эта интерпретация кодов РМ была использована для разработки эффективных методов мягкого декодирования, основанных на «рекурсивной» структуре этого класса кодов (см. [For6, SB] и ссылки в этих работах). В Разделе 6.2.4 будет показано, что эта конструкция является частным случаем более мощной техники комбинирования кодов. Конструкция **|u|u+v|** известна также как *конструкция удвоения* [For6].

Пример 76. Пусть C_1 код RM(1,2), т.е. (4,3,2) код с одной проверкой (SPC), и C_2 код RM(0,2), т.е. код повторение (4,1,4). Тогда C=|C_1|C_1+C_2| является кодом Рида-Маллера RM(1,3) или расширенным (8,4,4) кодом Хемминга с порождающей матрицей

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

Конструкция X

Эта конструкция является обобщением конструкции **|u|u+v|** [Slo]. Обозначим C_i линейный блоковый (n_i, k_i, d_i) код для i=1, 2, 3. Предположим, что C_3 является подкодом C_2, т.е. $n_3 = n_2$, $k_3 \le k_2$ и $d_3 \ge d_2$. Предположим, что размерность кода C_1 равна $k_1 = k_2 - k_3$. Пусть $(\mathbf{G}_2{}^{\mathrm{T}}\ \mathbf{G}_3{}^{\mathrm{T}})^{\mathrm{T}}$ и $\mathbf{G}_3$ порождающие матрицы кода $C_2 \supset C_3$ и подкода C_3, соответственно. Заметим, что $\mathbf{G}_2$ есть множество *представителей смежных классов* кода C_3 в C_2 [For6]. Тогда код C_X с порождающей матрицей

$$\mathbf{G}_X=\begin{pmatrix}\mathbf{G}_1&\mathbf{G}_2\\\mathbf{0}&\mathbf{G}_3\end{pmatrix} \qquad (6.17)$$

является линейным блоковым кодом с параметрами $(n_1 + n_2, k_1 + k_2, d_X)$, где $d_X = \min\{d_3, d_1 + d_2\}$.

Пример 77. Пусть C_1 код (3,2,2) с одной проверкой (SPC) и C_2 (4,3,2) SPC код, подкодом которого является (4,1,4) код-повторение C_3. Тогда

$$\mathbf{G}_1=\begin{pmatrix}1&0&1\\0&1&1\end{pmatrix},\quad \begin{pmatrix}\mathbf{G}_2\\\mathbf{G}_3\end{pmatrix}=\begin{pmatrix}1&0&0&1\\0&1&0&1\\1&1&1&1\end{pmatrix}$$

и $\mathbf{G}_3$ = (1 1 1 1). Конструкция **X** дает код C_X = |C_1|C_2 + C_3| с порождающей матрицей

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

который является кодом максимальной длины (7,3,4) (или симплекс кодом). Этот код эквивалентен коду, полученному из (7,4,3) кода Хемминга удалением одного информационного символа, как показано в Примере 73.

Конструкция X3

Развивая идею использования представителей смежных классов подкодов некоторого кода, в этой конструкции комбинируются три кода, один из которых имеет два уровня *разложения на смежные* классы по подкодам [Slo]. Пусть C_3 линейный блоковый (n_1, k_3, d_3) код, размерность которого удовлетворяет равенству $k_3 = k_2 + a_{23} = k_1 + a_{12} + a_{23}$. Код C_3 является объединением $2^{a_{23}}$ непересекающихся смежных классов линейного блокового (n_1, k_2, d_2) кода C_2 размерности $k_2 = k_1 + a_{12}$. В свою очередь, код C_2 является объединением $2^{a_{12}}$ непересекающихся смежных классов линейного блокового (n_1, k_1, d_1) кода C_1. Тогда каждое кодовое слово кода C_3 может быть записано как сумма $\mathbf{x}_i + \mathbf{y}_i + \mathbf{v}$, где $\mathbf{v} \in C_1$, $\mathbf{x}_i$ является представителем смежного класса кода C_2 в C_3, а $\mathbf{y}_i$ есть представитель смежного класса кода C_1 в C_2. Пусть C_4 и C_5 линейные блоковые коды с параметрами (n_4, a_{23}, d_4) и (n_5, a_{12}, d_5), соответственно. Линейный блоковый код C_{X3} с параметрами $(n_1 + n_4 + n_5, k_3, d_{X3})$ задается следующим образом

$$C_{X3} = \{|\mathbf{x}_i + \mathbf{y}_i + \mathbf{v}|\mathbf{w}|\mathbf{z}| : \mathbf{x}_i + \mathbf{y}_i + \mathbf{v} \in C_3, \mathbf{w} \in C_4, \mathbf{z} \in C_5\}$$

и имеет минимальное расстояние $d_{X3} = \min\{d_1, d_2 + d_4, d_3 + d_5\}$. Порождающая матрица кода C_{X3} имеет вид

$$\mathbf{G}_{X3} = \begin{pmatrix} \mathbf{G}_1 & \mathbf{0} & \mathbf{0} \\ \mathbf{G}_2 & \mathbf{G}_4 & \mathbf{0} \\ \mathbf{G}_3 & \mathbf{0} & \mathbf{G}_5 \end{pmatrix}$$

где $(\mathbf{G}_1)$, $(\mathbf{G}_1{}^{\mathrm{T}}\,\mathbf{G}_2{}^{\mathrm{T}})^{\mathrm{T}}$ и $(\mathbf{G}_1{}^{\mathrm{T}}\,\mathbf{G}_2{}^{\mathrm{T}}\,\mathbf{G}_3{}^{\mathrm{T}})^{\mathrm{T}}$ порождающие матрицы кодов C_1, C_2 и C_3, соответственно.

Пример 78. Пусть C_1, C_2 и C_3 являются расширенными БЧХ кодами с параметрами (64, 30, 14), (64, 36, 12) и (64, 39, 10), соответственно. Пусть также C_4 есть (7, 6, 2) код с одной проверкой и C_5 симплексный (7, 3, 4) код. Конструкция **X3** дает (78, 39, 14) код. Этот код имеет на четыре информационных символа больше, чем укороченный (78, 35, 14) код, полученный укорочением (128, 85, 14) БЧХ кода.

Обобщение конструкций **X** и **X3** и семейства хороших кодов представлены в [Slo, MS, Kas1, Sug, FL3]. Применение этих конструкций для построения кодов с неравной защитой рассматривается в работах [Van, MH].

6.2.3. Произведения кодов

В этом разделе представлен важный метод построения кодов, известный как произведение. Простейшим способом комбинирования кодов является *последовательное* (повторное) кодирование. В этом случае выход первого кодера является входом второго кодера и т.д. Реализация этой идеи для двух кодеров показана на Рисунке 53. По существу, это прямой способ образования *произведения кодов*. Несмотря на простоту, *прямое произведение* дает очень хорошие коды. Например, сверточные коды с очень низкой скоростью могут быть построены как произведение двоичного сверточного кода с блоковым кодом – повторением.

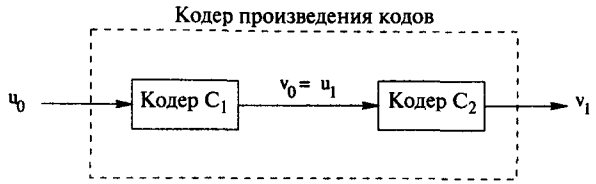


Рис. 53. Блок схема кодера кода-произведения.

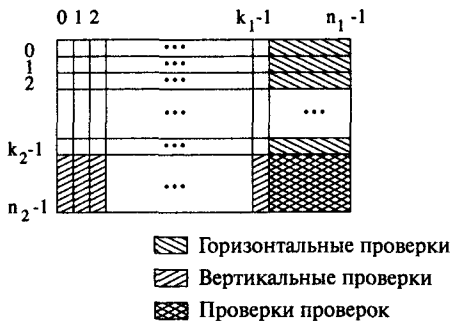


Рис. 54. Кодовое слово двумерного кода-произведения.

Пример 79. Рассмотрим схему, в которой стандартный сверточный кодер памяти 6, скорости 1/2, с генераторами (171, 133) и минимальным расстоянием $d_f = 10$, подключен последовательно к кодерам последовательного соединения кодов – повторений (2,1,2) или (3,1,3), а именно, |(2,1,2)|(2,1,2)| или |(3,1,3)|(3,1,3)|. Другими словами, каждый бит кодового слова повторяется два или три раза.

Эти схемы порождают коды памяти 6, скорости 1/4 или 1/6 с генераторами (171, 171, 133, 133) или (171, 171, 171, 133, 133, 133) и расстояниями $d_f = 20$ или $d_f = 30$, соответственно. Эти коды оптимальны [Dho] в том смысле, что они обладают наибольшим свободным расстоянием при заданном числе состояний кодера. По-видимому, это первый пример описания этих кодов в терминах их генераторов.

Тем не менее, кроме случая двух кодеров, второй из которых генерирует последовательное соединение кодов, всегда

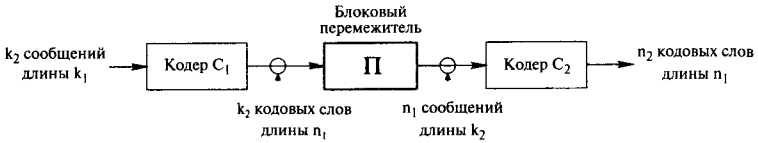


Рис. 55. Кодер двумерного кода-произведения с блоковым перемежителем (интерливером).

возникает вопрос о последовательном [3]соединении между двумя кодерами: Как следует соединить выход первого кодера с входом второго кодера? Назовем C_1 *внешним кодом* и C_2 – *внутренним кодом*. Код C_1, или C_2, или оба могут быть сверточными или блоковыми кодами. Если G_1 и G_2 порождающие матрицы компонентных кодов, то порождающая матрица произведения кодов есть их Кронекеровское произведение, т.е. $G = G_1 \otimes G_2$.

В 1954 году Элайес (Elias) [Eli1] ввел произведение или итерацию кодов. Главная идея состоит в следующем. Предположим, что оба кода C_1 и C_2 являются систематическими. Кодовые слова внешнего кода C_1 (порядок внешнего и внутреннего кодирования устанавливается по отношению к каналу: канал и внутренний код образуют составной канал для внешнего кода) записываются по строчкам прямоугольного массива с n_1 столбцами: по одному символу в элемент массива. После заполнения k_2 строк оставшиеся $n_2 - k_2$ строк заполняются по столбцам проверочными символами внутреннего кода. Прямоугольный массив размера $n_1 \times n_2$, заполненный кодовыми словами компонентных кодов по строчкам и столбцам, является кодовым словом *произведения кодов* $C_P = C_1 \otimes C_2$. На Рисунке 54 показана структура кодового слова двумерного кода-произведения.

Кодовое слово из массива передается по столбцам. Возвращаясь к первоначальному описанию произведения кодов, Рисунок 53, Элайеса, двумерный код – произведение может быть интерпретирован как последовательное соединение кодеров через перемежитель (interleaver). Схематически это изображено на Рисунке 55.

[3] Эта схема известна как *каскадное кодирование* (или *последовательное каскадирование*) Однако в этом разделе термин *каскадирование* используется в несколько ином значении

	j=0	j=1	j=2	j=3	j=4
i=0	0	2	4	6	8
i=1	1	3	5	7	9

Рис. 56. Два-на-пять блоковый перемежитель.

По определению Рамсей (Ramsey) [Ram] перемежитель является устройством, *которое изменяет порядок передачи последовательности символов некоторым взаимно однозначным детерминированным способом*. Для произведения линейных блоковых кодов это устройство известно, естественно, как *блоковый перемежитель*. Интерливер (перемежитель) определяет соответствие $m_b(i,j)$ между элементами $a_{i,j}$ $k_2 \times n_1$ массива, образованного k_2 кодовыми словами C_1 по строчкам, и элементами u_μ, $\mu = m_b(i,j)$ *информационного вектора* $\mathrm{u} = (u_0\, u_1 \ldots u_{N-1})$, $N = n_1 n_2$.

Взаимно однозначное отображение на множество целых чисел, индуцированное $m_1 \times m_2$ блоковым перемежителем, можно рассматривать также как *перестановку* П: $i \to \pi(i)$, действующую на множестве целых чисел по модулю $m_1 m_2$. Разворачивая массив в одномерный вектор u с чередованием элементов от первой до m_1 строки, получаем вектор

$$\mathbf{u} = (u_0\, u_1 \cdots u_{m_1 m_2 - 1})$$

который считывается на выходе перемежителя через перестановку П как

$$\mathbf{u}_\pi = (u_{\pi(0)}\, u_{\pi(1)} \cdots u_{\pi(m_1 m_2 - 1)}) \qquad (6.18)$$

где

$$\pi(i) = m_2 (i \bmod m_1) + \left\lfloor \frac{i}{m_1} \right\rfloor \qquad (6.19)$$

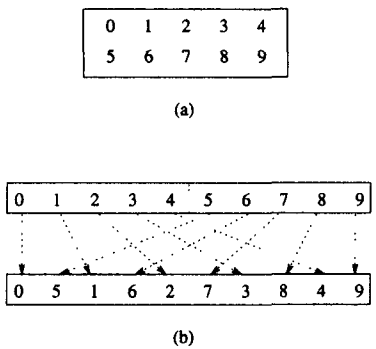


Рис. 57. (а) Кодовые слова C1 как строки матрицы; (б) эквивалентный вектор u и его перестановка up.

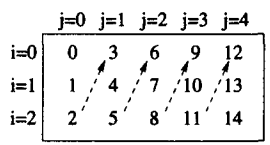


Рис. 58. Отображение mb(i,j), реализуемое 3-на-5 блоковым перемежителем.

Пример 80. Пусть C_1 и C_2 линейные блоковые коды с одной проверкой (5, 4, 2) и (3, 2, 2), соответственно. Блоковый перемежитель, показанный на Рисунке 56, приводит к коду – произведению (15, 8, 4). Соответствующая перестановка имеет вид

$$\pi(i) = 5(i \bmod 2) + \left\lfloor \frac{i}{2} \right\rfloor$$

а вектор $\mathbf{u} = (u_0, u_1, u_2, u_3, \ldots, u_9)$ преобразуется в вектор

$$\mathbf{u}_\pi = ((u_0\ u_5)(u_1\ u_6)\cdots(u_4\ u_9)) = (\mathbf{u}_0\ \mathbf{u}_1 \cdots \mathbf{u}_4)$$

Это отображение показано на Рисунке 57. Подвекторы $\mathbf{u}_i = (u_i\ u_{i+5})$, $0 \le i < 5$, составляют информационные векторы, которые кодируются кодом C_2. Обычно кодовые слова представляются двумерными массивами

$$\mathbf{v} = (a_{0,0}\ a_{1,0}\ a_{2,0}\ a_{0,1}\ a_{1,1} \cdots a_{2,4})$$

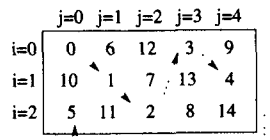


Рис. 59. Циклическое отображение m_c для $n_1 = 5$, $n_2 = 3$.

где строки $(a_{\ell,0}\, a_{\ell,1} \ldots a_{\ell,4}) \in C_1$ для $\ell = 0,1,2$, а столбцы $(a_{0,\ell}\, a_{1,\ell}\, a_{2,\ell}) \in C_2$ для $\ell = 0, 1, \ldots, 4$. Соответствующий порядок показан на Рисунке 58. В одномерной записи получаем такой же вектор

$$\mathbf{v} = ((\mathbf{u}_0, v_0)(\mathbf{u}_1, v_1)\cdots(\mathbf{u}_4, v_4))$$

где $(\mathbf{u}_i, v_i) \in C_2$.

Пример 81. Пусть C_1 и C_2 двоичные (3, 2, 2) коды с одной проверкой. Тогда C_P есть (9, 4, 4) код. Хотя этот код имеет на один проверочный бит больше, чем расширенный код Хемминга (или код Рида-Маллера RM(1,3)), он способен очень легко исправлять ошибки, используя только проверки по строкам и столбцам. Предположим, что передавалось нулевое кодовое слово по ДСК и что принятое слово равно

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

Напомним, что синдром двоичного $(n, n-1, 2)$ кода с одной проверкой равен просто сумме n символов. Вторая строка и первый столбец имеют ненулевые синдромы, что указывает на присутствие нечетного числа ошибок. Более того, так как другие столбцы и строки имеют нулевые синдромы, то правильный вывод состоит в том, что первый символ второй строки содержит единственную ошибку (или второй символ первого столбца). Декодирование заканчивается инвертированием символа на обнаруженной искаженной позиции.

Код в Примере 81 принадлежит семейству кодов, известных под названием *матричные коды* (*array codes* см. также

$v_{0,0}$	$v_{0,1}$	...	v_{0,n_1-1}
$v_{1,0}$	$v_{1,1}$	...	v_{1,n_1-1}
...	...		...
$v_{n_2-1,0}$	$v_{n_2-1,1}$	...	v_{n_2-1,n_1-1}

Рис. 60. Кодовое слово блокового кода с перемежением степени $I = n_2$.

[Kas2, BL]). Так как эти коды являются кодами-произведениями, они способны исправлять пакеты ошибок в дополнение к одиночным ошибкам. Матричные коды имеют удобную решетчатую структуру [HM] и связаны с классом *обобщенных каскадных кодов* (OK или GC), которые будут рассматриваться в Разделе 6.2.4.

Пусть C_i линейный блоковый (n_i, k_i, d_i) код, $i = 1, 2$. Тогда произведение $C_P = C_1 \otimes C_2$ является линейным блоковым (n_1n_2, k_1k_2, d_P) кодом с кодовым расстоянием $d_P = d_1d_2$. Код C_P может исправлять, дополнительно, все (одиночные) *пакеты ошибок* длиной до $b = \max\{n_1t_2, n_2t_1\}$, где $t_i = \lfloor (d_i - 1)/2 \rfloor$, для $i = 1, 2$. Параметр b называется *корректирующей способностью к пакетам ошибок*.

Пример 82. Пусть C_1 и C_2 коды Хемминга (7,4,3). Тогда C_P есть (49, 16, 9) код, способный к исправлению четырех случайных ошибок или одного пакета ошибок длины 7 и меньше.

Если компонентные коды циклические, то и их произведение является циклическим кодом [BW]. Точнее, пусть C_i циклический (n_i, k_i, d_i) код с порождающим многочленом $g_i(x)$, $i = 1, 2$. Тогда код $C_P = C_1 \otimes C_2$ является циклическим, если выполняются следующие условия:

1. Длины кодов взаимно просты, т.е. существуют целые a и b такие, что $an_1 + bn_2 = 1$.
2. Использовано *циклическое отображение* $m_c(i, j)$ элементов прямоугольного массива $a_{i,j}$, показанного на Рисунке 54, в элементы v_μ, $\mu = m_c(i,j)$, кодового вектора $v(x) = v_0 + v_1x + \ldots + v_{N-1}x^{N-1}$, $N = n_1n_2$, такое, что

$$m_c(i,j) = [(j-i)bn_1 + i] \bmod n_1n_2 \qquad (6.20)$$

где $m_c(i, j) = 0, 1, \ldots, n_1n_2\text{-}1$.

Если оба условия выполнены, то производящий многочлен циклического кода-произведения равен

$$g(x) = GCD\left(g_1(x^{bn_2})g_2(x^{an_1}), x^{n_1n_2} + 1\right) \qquad (6.21)$$

Пример 83. Пример циклического отображения для $n_1 = 5$ и $n_2 = 3$, показан на Рисунке 59. В этом случае $(-1)5 + (2)3 = 1$, т.е. $a = -1$ и $b = 2$. Следовательно, отображение имеет вид

$$m_c(i,j) = (6j - 5i) \bmod 15$$

Для проверки, если $i = 1, j = 2$, то $m_c(1,2) = (12 - 5) \bmod 15 = 7$, если $i = 2$ и $j = 1$, то $m_c(2,1) = (6 - 10) \bmod 15 = -4 \bmod 15 = 11$.

Отображение $m_c(i, j)$ определяет *порядок передачи элементов массива* [BW]. Этот порядок не совпадает с обычным порядком считывания символов из блокового перемежителя для обычного кода – произведения. Отображение (6.20) называют *циклическим перемежением*. Другие типы перемежителей обсуждаются в Разделе 6.2.4.

С появлением турбо-кодов в 1993 г. [BGT] активизировалось исследование новых структур перемежителей, реализующих псевдослучайную перестановку на словах кода C_1 перед выполнением кодирования кодом C_2. В следующем разделе рассматриваются коды с перемежением. В Главе 8 обсуждаются типы структур перемежителей полезных для итеративного декодирования кодов-произведений.

Блоковые коды-перемежения

Это частный случай кода-произведения, когда второй кодер является тривиальным $(n_2, n_2, 1)$ кодом. В этом случае, слова кода C_1 записываются по строкам $n_2 \times n_1$ прямоугольного массива и считываются для передачи по столбцам как для обычного кода-произведения. Величина $I = n_2$ известна как *степень перемежения* [LC] или *глубина перемежения* (*interleaving degree or depth*).

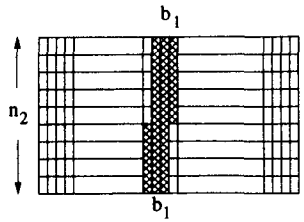


Рис. 61. Исправляемый пакет ошибок в слове кода-перемежения.

Построенный блоковый код-произведение, обозначаемый здесь и в дальнейшем как $C_1^{(n_2)}$, может быть декодирован тем же алгоритмом, что и код C_1, после того как принятое слово будет записано по столбцам и считано по строкам массива. На Рисунке 60 показано размещение символов кода-перемежения, где

$$(v_{i,0}\ v_{i,1} \cdots v_{i,n_1-1}) \in C_1, \quad 0 \le i < n_2$$

Если код C_1 исправляет $t_1 = \lfloor (d_1-1)/2 \rfloor$ ошибок, то код $C_1^{(n_2)}$ может исправить любой одиночный пакет ошибок длины $b = t_1 n_2$ и меньше. Этот факт продемонстрирован на Рисунке 61. Напомним, что порядок передачи символов кодового слова устанавливается по столбцам. Если возникает пакет ошибок, то он искажает не более t_1 позиций в строке, которые могут быть исправлены кодом C_1. Максимальная длина такого пакета ошибок равна $n_2 t_1$. Более того, если код C_1 может исправлять (или обнаруживать) одиночный пакет ошибок длины b_1 и меньше, то код-перемежение может исправить (или обнаружить) любой одиночный пакет ошибок длины $b_1 n_2$ и меньше.

Если C_1 циклический код, то из (6.21) следует, что $C_1^{(n_2)}$ тоже циклический код с порождающим многочленом $g_1(x^{n_2})$ [PW, LC]. Если применяются укороченные коды, то справедлив следующий результат ([PW, стр. 358]):

Перемежение степени l укороченного циклического (n, k) кода дает укороченный (ln, lk) код с исправляющей способностью к пакетам ошибок в l раз больше, чем у исходного кода.

Заметим еще, что корректирующая способность кода-произведения к случайным ошибкам, $t_P = \lfloor (d_1 d_2 - 1)/2 \rfloor$, может быть реализована только с помощью специально разработанного метода декодирования.

Большинство методов декодирования кода-произведения используют *двух этапную* процедуру. На первом этапе исправляются ошибки в строках массива с помощью алгебраического декодера для кода C_1. После этого, декодированным символам приписываются *надежности* в зависимости от исправленного числа ошибок. Чем больше исправлено ошибок (в строке), тем менее надежной считается оценка результата декодирования кодом C_1.

На втором этапе используется алгебраическое исправление ошибок и стираний кодом C_2 в столбцах массива. На этом этапе выполняется много попыток исправления с возрастающим числом стираний, соответствующих *наименее надежным позициям* (т.е. позициям, которым приписаны наименьшие надежности), до тех пор, пока выполняется достаточное условие на исправляемое число ошибок (и стираний). Этот подход был предложен в [For1, For3, RR, Wel]. Обычно второй этап реализуется в виде *декодирования по минимуму обобщенного расстояния*, который обсуждается в Разделе 7.6. Дополнительный материал по декодированию кода-произведения может быть найден в Главе 8.

6.2.4. Каскадные коды

В 1966 г. Форни [For1] предложил замечательный метод комбинирования двух кодов, названный *каскадным кодированием*. Заметим, что аналогичный подход разрабатывался в работах Блоха, Зяблова [BZ2*, BZ3*]. Каскадная схема показана на Рисунке 62. Каскадные коды[4], основанные на внешних кодах Ри-

[4] В английской литературе используется термин *concatenated codes*, а иногда и *cascaded codes*. В русской литературе теперь используется только термин *каскадные коды*, хотя в 70-х использовался также термин *коды с локализацией ошибок*.

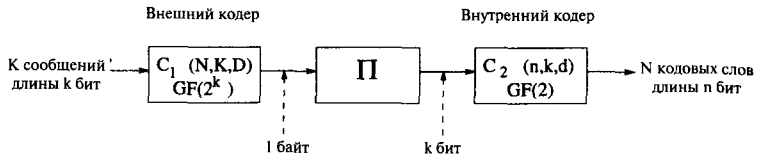


Рис. 62. Кодер каскадного кода.

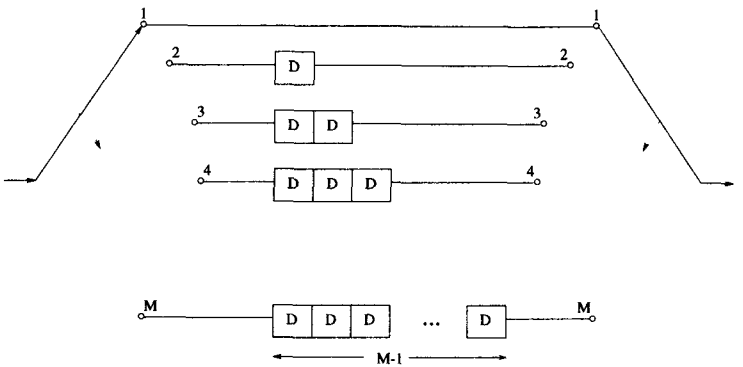


Рис. 63. Сверточный интерливер.

да-Соломона и внутренних сверточных кодах, были до настоящего времени наиболее распространенными[5] кодовыми конструкциями для цифровой связи. В общем случае, внешний код, обозначаемый C_1, является недвоичным блоковым (N, K, D) кодом над $\mathbf{GF}(2^k)$. Кодовые слова кода C_1 записываются в память перемежителя (интерливера). На выходе перемежителя считываются байты и пропускаются через кодер внутреннего кода C_2. Внутрений код может быть блоковым или сверточным. Когда рассматриваются блоковые коды, то C_2 есть двоичный блоковый (n, k, d) код. Структура соответствующего кодера показана на Рисунке 62. Обозначим $C = C_1 \bullet C_2$ каскадный код с внешним кодом C_1 и внутренним кодом C_2. Тогда C является двоичным линейным блоковым (Nn, Kk, Dd) кодом.

[5] По крайней мере до появления турбо кодов и практической реализации кодов с низкой плотностью проверок.

Назначение интерливера между внешним и внутренним кодами двоякое. Во-первых, он требуется для преобразования байтов размера k в векторы, размерность которых соответствует размерности (информационным символам) внутреннего кода. Внутренний код может быть двоичным или недвоичным линейным блоковым (n, k', d) кодом или сверточным кодом скорости k'/n с несовпадающими, как правило, значениями k и k'. Во-вторых, как обсуждалось ранее, перемежение позволяет разбить пакеты ошибок. Это полезно, например, в каскадных схемах со сверточным внутренним кодом, так как декодер Витерби сам создает пакеты ошибок (в случае ошибочного декодирования) [CY, Mor]. На практике используются несколько типов интерливеров. Вероятно, наибольшее распространение получил *сверточный интерливер* Форни [For5], который является частным случаем *интерливера Рэмси* [Ram]. Базовая структура сверточного перемежителя показана на Рисунке 63. Деинтерливер (или деперемежитель) имеет такую же структуру, за исключением того, что начальным положением переключателя является M и его вращение выполняется в обратном направлении.

Важным преимуществом каскадных кодов (и кодов-произведений) является то, что декодирование может быть основано на декодировании отдельных компонентных кодов. Это существенно снижает сложность декодирования по сравнению с декодированием полного кода.

Пример 84. Пусть C_1 (7, 5, 3) код РС[6] с нулями $\{1, \alpha\}$, α примитивный элемент поля $\mathbf{GF}(2^3)$ и $\alpha^3 + \alpha + 1 = 0$. Пусть (7, 4, 3) код максимальной длины из Примера 77. Тогда $C = C_1 \bullet C_2$ есть двоичный блоковый (49, 15, 12) код. Этот код имеет на 6 информационных символов меньше, чем укороченный (49, 21, 12) код, построенный из расширенного БЧХ кода (64, 21, 12). Однако этот код легче декодировать. Пусть $v(x) = (x^4 + \alpha^4)g(x) = \alpha + x + \alpha^4 x^2 + \alpha x^4 + \alpha^3 x^5 + x^6$ кодовое слово (7, 5, 3) кода РС, где $g(x) = x^3 + \alpha^3 x + \alpha$.

[6] Коды Рида-Соломона рассматривались в Главе 4.

α^5	1	α^4	0	α	α^3	1
1	0	1	0	0	0	0
1	0	1	0	1	1	0
1	1	0	0	0	1	1
0	1	1	0	0	1	1
0	1	1	0	1	0	1
0	0	0	0	1	1	0
1	1	0	0	1	0	1

Рис. 64. Кодовое слово каскадного кода $C_1 \bullet C_2$, где C_1 код РС (7, 5, 3) над GF(2^3) и C_2 двоичный циклический (7, 4, 3) код.

Используя таблицу из Примера 27, можно представить элементы GF(2^3) векторами размера 3 бита. В результате получаем матрицу размера 3-на-7, столбцы которой являются двоичным представлением коэффициентов кодового многочлена (кодового слова). Затем, кодируя столбцы с помощью порождающего многочлена кода C_2 , получаем 4 дополнительных строки матрицы, представляющей кодовое слово. Здесь была использована следующая систематическая форма порождающей матрицы кода C_2, полученная перестановкой третьего и шестого столбцов матрицы G в Примере 77,

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

На Рисунке 64 показано кодовое слово (матрица), соответствующее вектору $\mathbf{v} \in C_1$.

6.2.5. Обобщенные каскадные коды

В 1974 г. Э.Л. Блох, В.В. Зяблов [BZ] и В.А. Зиновьев [Zin] ввели мощный класс обобщенных каскадных кодов (ОКК или GC – generalized concatenated codes). Это семейство кодов способно исправлять как случайные ошибки, так и случайные пакеты

ошибок. Из названия видно, что ОКК являются обобщением базовой конструкции каскадных кодов, предложенной Форни. Обобщение состоит в том, что вводится *иерархия вложенных подкодов* (разбиение на подкоды) внутреннего кода C_I и нескольких внешних кодов, по одному на каждый *уровень вложения* (разбиения). Конструкция ОКК является комбинацией идеи прямой суммы кодов, или разложения по смежным классам, и каскадного конструирования кодов. Перед определением этого класса кодов необходимо ввести некоторые обозначения.

Линейный блоковый (n, k, d) код C называется *разложимым* на линейные блоковые (n, k_i, d_i) подкоды C_i , $1 \le i \le M$, если выполняются следующие условия:

(Сумма) $C = C_1 + C_2 + \ldots + C_M$;

(D) Для любого $\mathbf{v}_i \in C_i$, $1 \le i \le M$, $\mathbf{v}_1 + \mathbf{v}_2 + \ldots + \mathbf{v}_M = \mathbf{0}$, если и только если $\mathbf{v}_1 = \mathbf{v}_2 = \ldots = \mathbf{v}_M = \mathbf{0}$.

Обозначим C_{Ii} , $1 \le i \le M$, линейный блоковый (n_I, k_{Ii}) код над **GF**(q) такой, что

(D_I) Для любого $\mathbf{u}_i \in C_i$, $1 \le i \le M$, $\mathbf{u}_1 + \mathbf{u}_2 + \ldots + \mathbf{u}_M = \mathbf{0}$, если и только если $\mathbf{u}_i = \mathbf{0}$ для всех $1 \le i \le M$.

Обозначим δ_i минимальное расстояние Хемминга прямой суммы кодов $C_{Ii} + C_{Ii+1} + \ldots + C_{IM}$. Пусть C_{Oi} есть линейный блоковый (n_O, k_{Oi}, d_{Oi}) код над $\mathbf{GF}(q^\kappa)$, $\kappa = k_{Ii}$. Обозначим $C_i^\bullet = C_{Oi} \bullet C_{Ii}$. Обобщенный каскадный код C определяется как прямая сумма

$$C = C_1^\bullet + C_2^\bullet + \cdots + C_M^\bullet \qquad (6.22)$$

Тогда условие (D) на код C следует из условия (D_I). Тогда минимальное расстояние Хемминга d кода C ограничено снизу [BZ, Zin, TYFKL] как

$$d \ge \min_{1 \le i \le M} \delta_i d_{Oi} \qquad (6.23)$$

Как и для простого (одноуровневого) каскадного кода главное преимущество ОКК состоит в том, что с помощью многошагового декодирования возможна реализация правой части границы (6.23) [BZ3*, TYFKL, MFKL].

Неравная защита от ошибок

Другим преимуществом этого класса кодов является относительно легко реализуемая возможность выбрать расстояния внутренних и внешних кодов таким образом, чтобы получить блоковый или сверточный код с *неравной защитой от ошибок*. Если указанные выше условия на прямую сумму кодов удовлетворяются и, кроме того, удовлетворяются следующие условия на произведения минимальных расстояний,

$$\delta_1 d_{O1} \geq \delta_2 d_{O2} \geq \cdots \geq \delta_M d_{OM} \qquad (6.24)$$

то кодовым словам размерности $k_{Oi}k_{Ii}$ над **GF**(q) соответствует корректирующая способность $\lfloor(\delta_i d_{O\,i} - 1)/2\rfloor$, которая убывает с номером уровня i, $1 \leq i \leq M$. В результате, сообщение, закодированное на верхнем уровне (малые значения i) разбиения, имеет усиленную корректирующую способность по сравнению с нижними уровнями. Конструкции этого типа рассматривались в [DYS, MH, BZ3*].

Конструкция

Предположим, что линейный блоковый (n_I, k_1, d_1) код C_1 над **GF**(q) *раскладывается* на последовательность M вложенных подкодов C_i с параметрами (n_I, k_i, d_i), $i = 2, 3, \ldots, M+1$, такую, что

$$C_1 \supset C_2 \supset \cdots \supset C_{M+1}$$

где для удобства принято, что $C_{M+1} = \{\mathbf{0}\}$ и $d_{M+1} = \infty$.

Обозначим $C_{Ii} = [C_i / C_{i+1}]$ линейный блоковый (n_I, k_{Ii}, δ_i) подкод кода C_i, который представляет собой *множество представителей смежных классов* кода C_{i+1} в коде C_i, имеет размерность $k_{Ii} = k_i - k_{i+1}$ и минимальное расстояние Хемминга $\delta_i \geq d_i$. Тогда код C_1 имеет следующее *разложение на смежные классы* [For6]

$$C_1 = C_{I1} + C_{I2} + \cdots + C_{IM} \qquad (6.25)$$

Обозначим C_{Oi} линейный блоковый код с параметрами (n_O, k_{Oi}, d_{Oi}) над расширением степени $k_{I\,i}$ базового поля

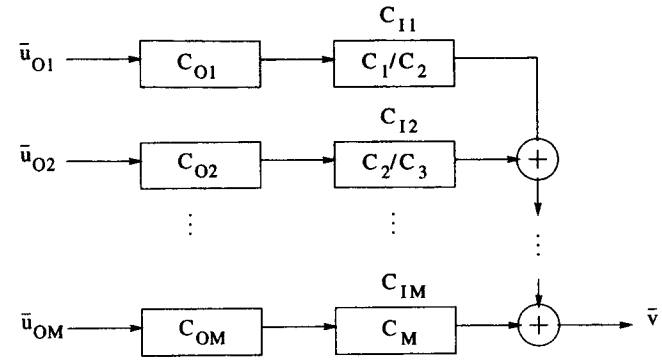


Рис. 65. Структура кодера обобщенного каскадного кода, имеющего M уровней разложения на вложенные подкоды.

GF(q), где $k_{I\,i} = \dim(C_i / C_{i+1}) = k_i - k_{i+1}$, $i = 1, 2, \ldots, M$. Тогда прямая сумма каскадных кодов дает линейный блоковый $(n_O n_I, k, d)$ код, размерность и минимальное расстояние которого равны, соответственно, [BZ]

$$k = \sum_{i=1}^{M} k_{Ii} k_{Oi} \quad \text{и} \quad d \geq \min_{1 \leq i \leq M} \{\delta_i d_{Oi}\} \qquad (6.26)$$

Заметим, что равенство в (6.26) достигается, когда все вложенные подкоды $C_{I\,i}$ содержат нулевое кодовое слово.

Рассмотрим выбор компонентных кодов. Так как размерности представителей смежных классов известны точно, то внешние коды могут быть выбраны из кодов Рида-Соломона или укороченных РС кодов.

Двоичные коды Рида-Маллера являются хорошими кандидатами в качестве внутренних кодов, так как они обладают следующим *свойством вложенности подкодов*,

$$\mathrm{RM}(r, m) \subset \mathrm{RM}(r+1, m),$$

для $0 \leq r < m$, $m \geq 2$.

Пример 85. Рассмотрим подкоды порядка r кода Рида-Маллера длины 8, получаемые из RM(r, 3). Имеет место следующая вложенность подкодов

$$RM(3, 3) \supset RM(2, 3) \supset RM(1, 3) \supset RM(0, 3) \supset \{\mathbf{0}\}.$$

Из (6.25) следует, что

$$RM(3, 3) = [RM(3, 3) / RM(2, 3)] + [RM(2, 3) / RM(1, 3)] + [RM(1, 3) / RM(0, 3)] + RM(0, 3).$$

Эту декомпозицию легко представить в матричной форме преобразованием порождающей матрицы кода RM(3, 3)

$$\mathbf{G} = \begin{pmatrix} \mathbf{G}_1 \\ \mathbf{G}_2 \\ \mathbf{G}_3 \\ \mathbf{G}_4 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

где матрица $\mathbf{G}_i$ определена как порождающая матрица множества представителей смежных классов кода RM(3–i, 3) в коде RM(3–i+1, 3) или [RM(3–i+1, m) / RM(3–i, m)], для i = 1, 2, 3, а матрица $\mathbf{G}_4$ есть порождающая матрица RM(0, 3).

В соответствии с разложением кода RM(3, 3) на вложенные подкоды возможно построение ОКК с четырьмя уровнями. Заметим, что внешние коды для первого и четвертого уровней должны быть двоичными, а для второго и третьего над $\mathbf{GF}(2^3)$.

Очевидно, что некоторые из подкодов RM(3, 3) могут быть использованы в качестве базового внутреннего кода для построения ОКК с меньшим числом уровней. Если код RM(2, 3) выбран базовым внутренним кодом ОКК, то число уровней равно трем. Порождающая матрица RM(2, 3) получается из матрицы кода RM(3, 3) удалением $\mathbf{G}_1$ (верхняя строка).

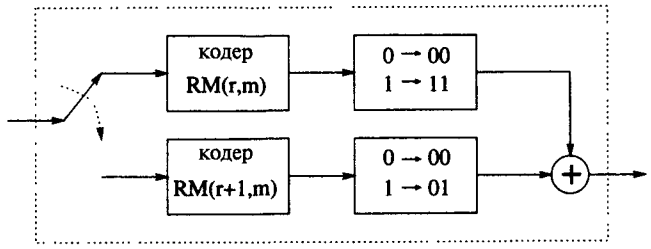


Рис. 66. Кодер двоичного RM(r+1, m+1) кода в виде двухуровневого ОКК.

Например, RM(2, 3) может быть выбран в качестве внутреннего кода. Тогда внешними кодами выбираются: код Рида-Соломона RS(8, 1, 8) в качестве C_{O1} над $\mathbf{GF}(2^3)$, код RS(8, 5 , 4) в качестве C_{O2} над $\mathbf{GF}(2^3)$ и двоичный линейный код с одной проверкой (8, 7, 2) в качестве C_{O3}. В результате получаем двоичный (64, 25, 16) ОКК, который имеет на один информационный символ больше, чем двоичный расширенный БЧХ (64, 24, 16) код.

Теперь уже известно, каким образом коды Рида-Маллера могут быть представлены в виде ОКК. Напомним, что код РМ порядка (r + 1), длины 2^{m+1} может быть задан в виде конструкции **X** как RM(r + 1, m + 1) = |RM(r + 1, m)|RM(r + 1, m) + RM(r, m)| (Раздел 6.2.2, уравнение (6.16)). Обозначим $\mathbf{G}(r, m)$ порождающую матрицу кода RM(r, m). Тогда получаем

$$\mathbf{G}(r+1,m+1) = \begin{pmatrix} \mathbf{G}(r+1,m) & \mathbf{G}(r+1,m) \\ 0 & \mathbf{G}(r,m) \end{pmatrix} = \mathbf{G}(r+1,m)\begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix} + \mathbf{G}(r,m)\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \quad (6.27)$$

Представление (6.27) соблюдается для кодов Рида-Маллера любого порядка r, при $r \geq 1$ и $m > 1$. Следовательно, возможно рекурсивное построение кода. Из (6.27) следует, что RM(r + 1, m + 1) как ОКК задается внутренним кодом (2, 2, 1) с разложе-

нием в смежные классы кода-повторения (2,1,2) и внешними кодами RM(r, m) и RM(r+1, m), соответственно. Кодер для этого случая показан на Рисунке 66. Рекурсивная структура кодов Рида-Маллера является весьма ценным свойством для построения алгоритмов мягкого декодирования[7], которые могут быть основаны на кодах-повторениях и кодах с одной проверкой. Именно это и было сделано в работе [SB].

[7] Мягкое декодирование рассматривается в Главе 7.

Глава 7

ДЕКОДИРОВАНИЕ С МЯГКИМ РЕШЕНИЕМ

В этой главе рассматривается декодирование с (дополнительной) *«мягкой» информацией*, получаемой из канала. Рассмотрим простой случай передачи двоичных сигналов по каналу с АБГШ (аддитивным белым гауссовым шумом). Чтобы обосновать целесообразность применения декодирования с мягким решением заметим, что шумовая компонента в задаче восстановления данных или приема сигналов является непрерывной, т.е. не дискретной, по своей природе. Это означает, что принятые символы представляются (квантованными) *действительными числами* (соответствующими напряжению, току и т.п.), а не двоичными символами или символами из конечного поля $\mathbf{GF}(2^m)$.

Когда выбираются *жесткие* решения относительно принятых символов, при посимвольной обработке, то могут происходить ошибки. Это иллюстрируется на Рисунке 67.

В принципе существуют два метода декодирования помехоустойчивых кодов, основанных на принятой последовательности действительных чисел:

1. Декодирование с жестким решением (hard decision decoding) (HDD):
при формировании жестких решений относительно принятых из канала величин происходят ошибки. Цель HDD состоит

Рис. 67. Пример возникновения ошибок при декодировании с жестким решением.

в исправлении двоичных ошибок, возникших в процессе выбора жестких решений. Первая часть этой книги была посвящена описанию различных методов HDD для линейных блоковых кодов, циклических и сверточных кодов.

2. Декодирование с мягким решением (soft-decision decoding) (*SDD*):
принятые из канала величины вводятся непосредственно в декодер для формирования оценок кодовой последовательности. Особым случаем SDD является декодирование по максимуму правдоподобия (maximum-likelihood decoding) (MLD или МПД), при котором в качестве решения декодера выбирается ближайшая (в некоторой метрике) кодовая последовательность. Здесь важно помнить, что для SDD необходимо знать статистику шума в канале связи.

В общем случае SDD более трудоемко, чем HDD. Отметим две основных причины этого. Одна из них состоит в том, что SDD требует выполнения операций с действительными числами. В практических применениях эти числа *квантуются* с конечной точностью (т.е. представляются конечным числом бит). Для некоторых систем передачи двоичных сигналов известно, что квантование на 8 уровней (или трех-битное представление чисел) обеспечивает эффективность системы, близкую к использованию вычислений с бесконечной точностью (без квантования) [Mas3, OCC].

Другая причина увеличения сложности SDD связана с необходимостью вычисления *апостериорных* статистик для кодовых символов. Тем не менее, увеличение трудоемкости окупается потенциальным повышением эффективности системы кодирования. Как показано в Главе 1 для двоичных сигналов в Гауссовом канале в случае SDD та же самая эффективность достигается при отношении сигнал-шум на *2 – 3 dB меньше*, чем при HDD. Это означает, что в случае SDD излучаемая передатчиком мощность может быть снижена на 50 – 63% по сравнению со случаем HDD. Эта экономия мощности преобразуется в меньший размер передающей антенны или в меньший размер приемной антенны при той же мощности передатчика.

7.1. Передача двоичных сигналов по каналам с АБГШ

Для каналов с АБГШ (или гауссовых каналов) *апостериорная* вероятность принятого значения r_i при условии, что был передан символ v_i, равна

$$p(r_i \mid v_i) = p_{n_i}(r_i - m(v_i)) = \frac{1}{\sqrt{\pi N_0}} e^{-(r_i - m(v_i))^2 / N_0}. \qquad (7.1)$$

Как показано в Главах 1 и 5 *квадрат Евклидова расстояния*, $D^2(r_i, m(v_i)) = (r_i - m(v_i))^2$, является метрикой для декодирования по максимуму правдоподобия (MLD). Обозначим E энергию переданного сигнала. Для отображения символов в сигналы (или модуляции) используем двоичную фазовую модуляцию (BPSK[1]),

$$m(v_i) = \begin{cases} \sqrt{E}, \text{если } v_i = 0 \\ -\sqrt{E}, \text{если } v_i = 1 \end{cases} \qquad (7.2)$$

или иначе

$$m(v_i) = (-1)^{v_i} \sqrt{E}$$

В двоичном канале с фазовой модуляцией и Гауссовым шумом вычисление метрики декодирования может быть упрощено, если заметить, что при условии $v_i = 1$ имеем

$$p(r_i \mid 1) = p_{n_i}(r_i + 1) = \frac{1}{\sqrt{\pi N_0}} \exp(-(r_i + 1)^2 / N_0).$$

Легко видеть, что после удаления константных членов натуральный логарифм отношения условных вероятностей $p(r \mid v)/p(r \mid 1-v)$ (индекс i опущен), называемый *метрикой логарифма правдоподобия*, пропорционален $-r$, если $v = 1$, и про-

[1] Binary phase-shift keying.

порционален r, если $v = 0$. Таким образом, весьма полезным является следующее замечание:

В двоичном канале с Гауссовым шумом вычисление метрики сводится к изменению знака принятой из канала величины.

7.2. Алгоритм Витерби с Евклидовой метрикой

Алгоритм Витерби (VD) может быть применен для декодирования данных, которые закодированы двоичным сверточным кодом и переданы по каналу с двоичной фазовой модуляцией и АБГШ. По сравнению с алгоритмом Витерби с жестким решением, который изучался в разделе 5.4, необходимы следующие два изменения.

1. Блок генерации метрики ребер:

Как было показано в предыдущем разделе, метрика для канала с аддитивным Гауссовым шумом пропорциональна корреляции между принятой из канала последовательностью отсчетов и проверяемой кодовой последовательностью. Таким образом, вместо Евклидовых растояний используются *корреляционные* метрики.

2. Блок Прибавить-сравнить-выбрать (ACS):

Алгоритм Витерби *максимизирует* корреляционную метрику вместо минимизации расстояния.

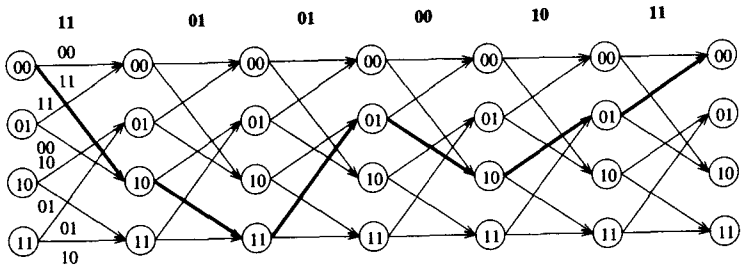


Рис. 68. Путь на треллисе, соответствующий последовательности $\mathbf{u} = (110100)$.

Пример 86. Рассмотрим сверточный кодер памяти 2 и скорости 1/2 с генераторами (7, 5). Предположим, что информационная последовательность равна $\mathbf{u} = (110100)$. Соответствующий путь на кодовой решетке показан на Рисунке 68.

Переданная (кодовая) последовательность (нормализованная относительно E) равна

$$t(\mathbf{v}) = (-1,-1,1,-1,1,-1,1,1,-1,1,-1,-1).$$

Пусть принятая последовательность после передачи по каналу с АБГШ и квантования выхода канала на 8 уровней (3 бита) равна:

$$\mathbf{r} = (-4,-1,-1,-3,+2,-3,+3,+3,-3,+3,-3,+1).$$

Заметим, что жесткое решение (соответствующее знаку) имеет вид:

$$\mathbf{r}_H = (11,\mathbf{1}1,01,00,10,1\mathbf{0}),$$

т.е. процесс детектирования сигналов с жестким решением ввел две ошибки (выделены жирным шрифтом). Вычисления декодера Витерби иллюстрируются на Рисунках с 69 по 74. Изменение значений метрик в процессе декодирования по шагам показано в следующей таблице:

Состояние/шаг	$i=0$	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$
$S_i^{(0)}$	0	+5	+7	+6	+12	+14	+26
$S_i^{(1)}$	0	+3	+5	+12	+14	+24	+18
$S_i^{(2)}$	0	+5	+9	+8	+18	+14	+22
$S_i^{(3)}$	0	+3	+7	+14	+14	+20	+24

Декодированная информационная последовательность равна $\mathbf{u} = (1\ 1\ 0\ 1\ 0\ 0)$. Таким образом, исправлены две ошибки.

Все соображения по реализации декодера Витерби, которые обсуждались в разделах 5.4.3. и 5.5.1, применимы и в случае мягкого декодирования. В частности, должна быть аккуратно выполнена нормализация метрики.

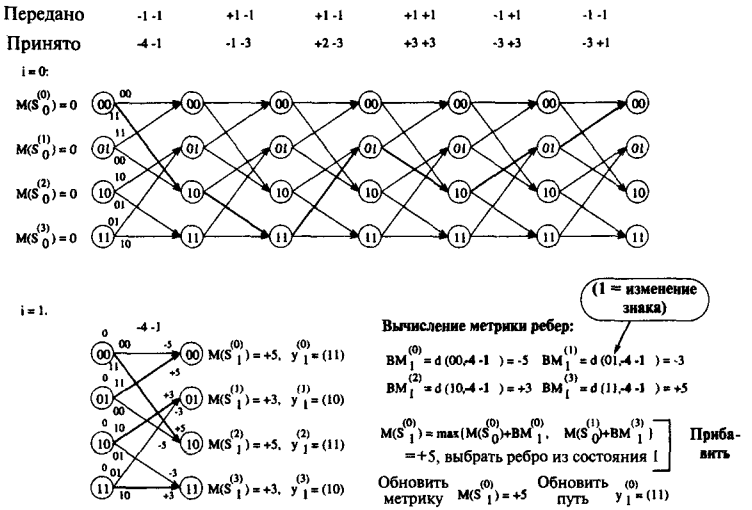


Рис. 69. Работа декодера Витерби с мягким решением на шаге $i = 1$.

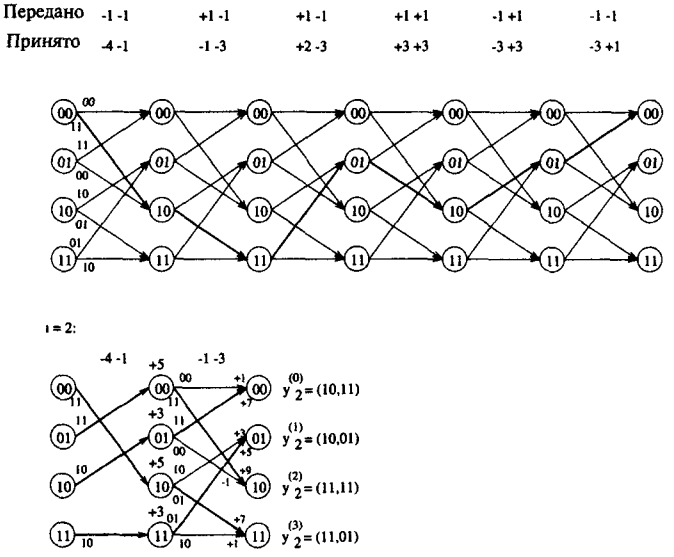


Рис. 70. Работа декодера Витерби с мягким решением на шаге $i = 2$.

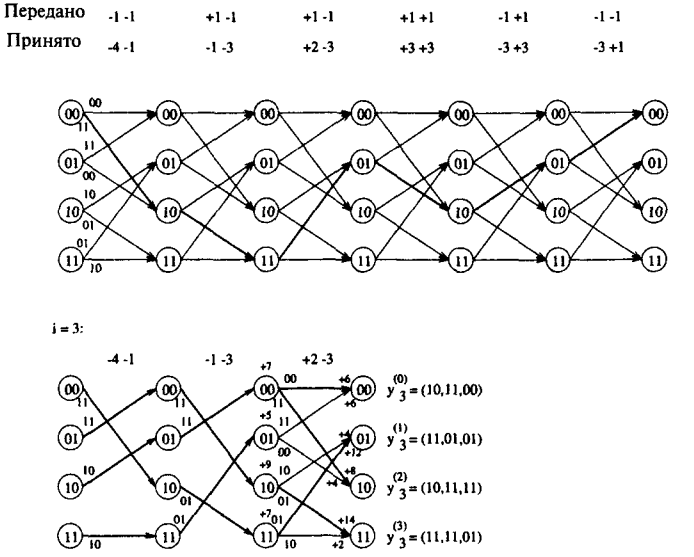


Рис. 71. Работа декодера Витерби с мягким решением на шаге $i = 3$.

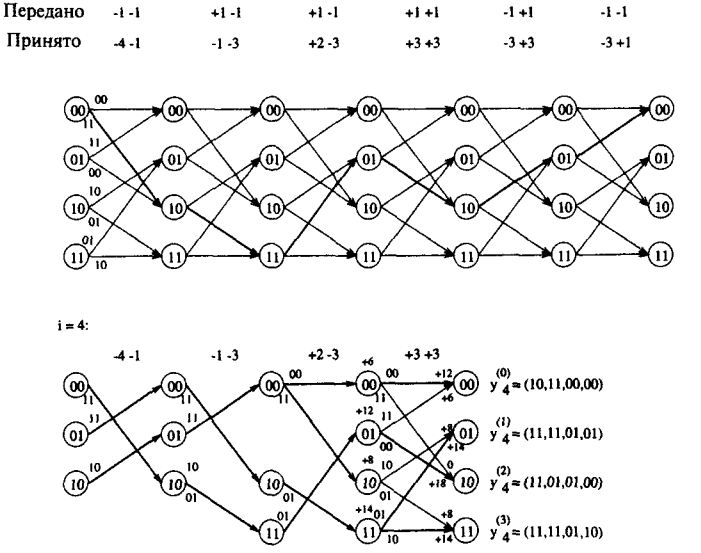


Рис. 72. Работа декодера Витерби с мягким решением на шаге $i = 4$.

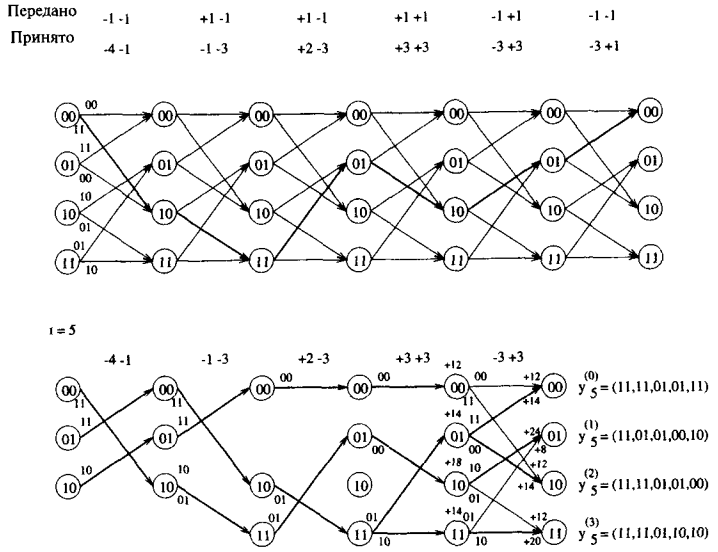


Рис. 73. Работа декодера Витерби с мягким решением на шаге $i = 5$.

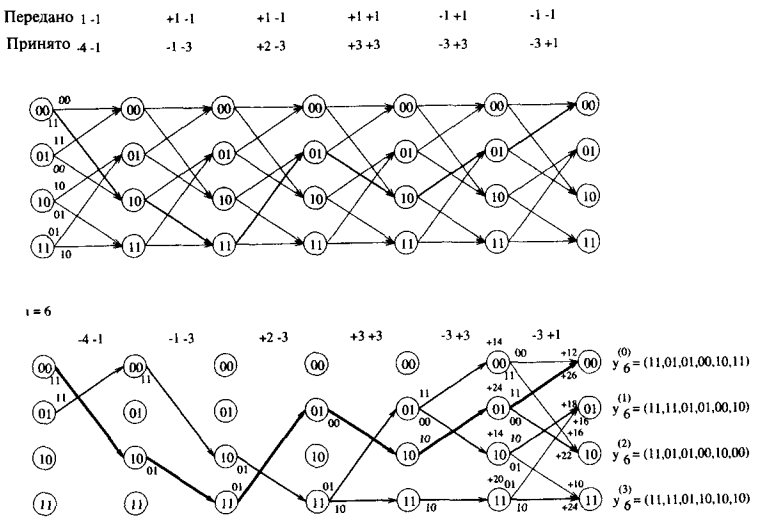


Рис. 74. Работа декодера Витерби с мягким решением на шаге $i = 6$.

7.3. Декодирование двоичных линейных блоковых кодов с помощью решетки

Алгоритм Витерби может быть применен и к блоковым кодам. *Синдромная решетка* для двоичного линейного блокового (N, K) кода C может быть построена из проверочной матрицы следующим образом [Wol]. Обозначим $(v_1, v_2, \ldots, v_N)$ кодовое слово кода C. В момент i, $1 \le i \le N$, состояния решетки определяется *частичными синдромами*:

$$\mathbf{s}_i = \sum_{j=1}^{i} v_j \mathbf{h}_j \qquad (7.3)$$

где сумма вычисляется в $\mathbf{GF}(2)$, $\mathbf{h}_j$ есть j-ый столбец проверочной матрицы $\mathbf{H}$ и $\mathbf{s}_0^{\top} = (0\,0\,\ldots\,0)$. Максимальное число состояний синдромной решетки равно $\min\{2^K, 2^{N-K}\}$. Синдромная решетка обладает свойством минимальности числа состояний. Решетка, удовлетворяющая условию минимальности числа состояний, называется *минимальной решеткой*.

Пример 87. Рассмотрим двоичный циклический (7, 4, 3) код Хемминга с порождающим многочленом $g(x) = x^3 + x + 1$. Проверочный многочлен равен $h(x) = 1 + x + x^2 + x^4$ и проверочная матрица кода C имеет вид

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

Сопоставим состоянию $\mathbf{s}_j^{\top} = (s_0\, s_1\, s_2)$ целое число I_j следующим образом:

$$I_j = s_0 + 2s_1 + 2^2 s_2.$$

Решетка этого кода имеет 8 состояний (так как $2^{N-K} = 2^3$) в сечениях для $i = 3$ и $i = 4$, как показано на Рисунке 75.

Интересно, что на синдромной решетке нет необходимости метить ребра кодированными битами. Переход между состояниями с одинаковой меткой соответствует кодовым битам

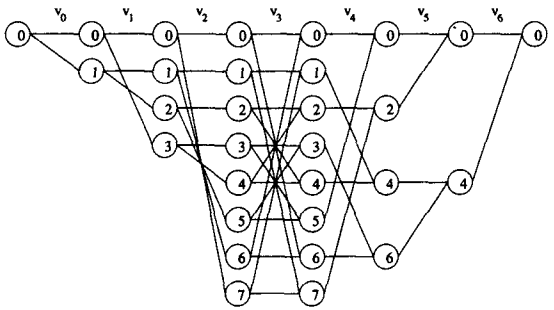


Рис. 75. Решетка кода Хемминга (7, 4, 3).

0, как это видно из (7.3). Если кодовый бит равен 0, то сумма не изменяется. Отметим следующие наблюдения о структурах решеток блоковых кодов. В общем случае, решетка блокового циклического кода имеет нерегулярную структуру. В результате заметно усложняется реализация алгоритма Витерби по сравнению со сверточными кодами.

Для некоторых классов кодов, таких как расширенные[2] коды БЧХ и коды Рида-Маллера, решетка может быть разделена на секции. В результате получается более регулярная и симметричная решетчатая структура с большим числом параллельных подрешеток, которые могут быть использованы для построения очень быстрых декодеров Витерби для блоковых кодов [LKFF, HM]. Фундаментальная теория минимальных решеток имеется в [Sch*].

7.4. Алгоритм Чейза

Алгоритм Чейза [Cha], который использует *множество или список наиболее вероятных последовательностей ошибок*, обеспечивает почти оптимальное декодирование. Это множество ошибок выбирается на основе оценок *надежности* принятых символов. Каждая комбинация ошибок прибавляется к слову, принятому с жесткими (посимвольными) решениями, и результат декодируется с использованием *декодера с жестким решением*. Для каждого декодированного кодового слова подсчитывается его *метрика* (расстояние) относительно принятой с *мягким решением* последовательности символов. В качестве наиболее вероятного решения выбирается декодированное кодовое слово с *наилучшей метрикой*.

[2] Расширение кодов рассматривалось в Главе 6.

Пусть C двоичный линейный блоковый (N, K, d) код, исправляющий любую комбинацию случайных ошибок веса $t = \lfloor (d-1)/2 \rfloor$ или меньше. Пусть $\mathbf{r} = (r_1, r_2, \ldots, r_N)$ полученное на выходе канала слово, где $r_i = (-1)^{c(i)} + w_i$ и w_i есть Гауссова случайная величина с нулевым средним и дисперсией $N_0/2$ для всех $i = 1, 2, \ldots, N$. Знаковые разряды на выходе канала представляют жесткое решение относительно символов принятого слова,

$$\mathbf{z} = (z_{0,0}\, z_{0,1} \cdots z_{0,N-1}), \quad z_{0,J} = \operatorname{sgn}(r_j), \quad 0 \le j < N, \qquad (7.4)$$

где

$$\operatorname{sgn}(x) = \begin{cases} 0, \text{ если } x > 0; \\ 1, \text{ в противном случае.} \end{cases}$$

Надежностями символов на выходе канала при приеме двоичных сигналов с АБГШ являются амплитуды $|r_i|$. Полученные надежности символов упорядочиваются с помощью какого-либо алгоритма сортировки (например, *быстрой сортировки*). Выходом алгоритма является список индексов $I_j, j = 1, 2, \ldots, N$, согласно которому

$$|r_{I_1}| \le |r_{I_2}| \le \ldots \le |r_{I_N}|.$$

На первом этапе декодирования принятое слово с жесткими посимвольными решениями вводится в декодер с жестким решением. Обозначим $\mathbf{v}_0$ декодированное кодовое слово. Тогда метрикой $\mathbf{v}_0$ относительно принятого слова $\mathbf{r}$ (с мягкими решениями) является величина

$$v_0 = \sum_{j=1}^{N} (-1)^{v_{0j}} \times r_j, \qquad (7.5)$$

которая вычисляется и запоминается как максимум.

Чейз ввел три типа алгоритмов соответственно множествам исправляемых комбинаций ошибок.

• **Тип – I**

Проверяются все комбинации ошибок на расстоянии не более $(d - 1)$ от принятого слова.

• **Тип –II**

Проверяются комбинации ошибок веса $\lfloor (d - 1)/2 \rfloor$ и меньше, которые размещаются на любых позициях за исключением $\lfloor d/2 \rfloor$ позиций с наименьшими надежностями. Эффективность этого алгоритма лишь немного уступает алгоритму Типа – I однако сложность его на много меньше за счет меньшего количества проверяемых комбинаций ошибок.

• **Тип – III**

Проверяются те комбинации ошибок, для которых i ошибок размещаются на i наименее надежных позициях, i нечетно, $1 \le i \le d - 1$. Следует заметить, что этот алгоритм тесно связан с алгоритмом декодирования по минимуму обобщенного расстояния (GMD или MOP) (см. также [ML], стр. 168). Декодирование по MOP кодов Рида-Соломона является темой раздела 7.6.

Из-за невысокой сложности и хорошей эффективности алгоритм Чейза типа-II стал наиболее популярным из перечисленных выше алгоритмов. Структурная схема алгоритма Чейза показана на Рисунке 76.

Алгоритм Чейза, Тип -II

- Для $i = 1, 2, \ldots, 2^t - 1$ прибавить комбинацию ошибок $\mathbf{e}_i$ к принятому с жестким решением слову: $\mathbf{z}_i = \mathbf{e}_i \oplus \mathbf{z}_0$.
- Комбинации ошибок генерируются на t наименее надежных позициях. Таковыми являются позиции с индексами $\{I_1, I_2, \ldots, I_t\}$, надежности (амплитуды) которых минимальны.
- Каждый вектор $\mathbf{z}_i$ вводится в декодер с жестким решением, который формирует кодовое слово $\mathbf{v}_i$, $i = 1,2, \ldots, 2^t - 1$.
- Вычислить метрику (7.5) и, если она максимальна, запомнить кодовое слово $\mathbf{v}_i$ как наиболее вероятное.

Рис. 76. Структурная схема алгоритма Чейза, Тип -II.

Если это необходимо, алгоритм может формировать список кодовых слов (в качестве возможных решений). Это свойство алгоритма Чейза с успехом может быть использовано для формирования мягкого выхода при итеративном декодировании блоковых турбо кодов (см. раздел 8.3).

7.5. Декодирование по упорядоченным статистикам

Алгоритм декодирования по упорядоченным статистикам (ordered statistics decoding – OSD) [FL1] аналогичен алгоритму Чейза в том смысле, что он тоже создает список комбинаций ошибок и использует декодирование с жестким решением. Однако, в отличие от алгоритмов Чейза, которые работают на *наименее надежных позициях*, OSD алгоритм создает список кодовых слов на множестве *наиболее надежных позиций*.

Предположим, что для передачи двоичных сигналов по каналу с АБГШ используется линейный блоковый (N, K) код C с порождающей матрицей $\mathbf{G}$ и минимальным расстоянием $d_H \ge 2t + 1$. Пусть $\mathbf{c} = (c_1, c_2, \ldots, c_N)$ кодовое слово из кода C и пусть $\mathbf{r} = (r_1, r_2, \ldots, r_N)$ принятая последовательность.

Как и в алгоритме Чейза декодирование начинается с упорядочивания компонент принятой последовательности по убыванию (не возрастанию) надежностей. Обозначим

$$\mathbf{y} = (y_1, y_2, \cdots, y_N)$$

упорядоченную последовательность, в которой $|y_1| \geq |y_2| \geq \ldots \geq y_N|$. Это переупорядочивание назовем *перестановочным отображением* (или просто *перестановкой*) λ_1 такой, что $\mathbf{y} = \lambda_1(\mathbf{r})$.

Следующий шаг алгоритма состоит в перестановке столбцов порождающей матрицы:

$$\mathbf{G}' = \lambda_1[\mathbf{G}] = (\mathbf{g}_1' \, \mathbf{g}_2' \cdots \mathbf{g}_N'),$$

где $\mathbf{g}_j'$ есть j-ый столбец матрицы $\mathbf{G}'$.

Продолжение алгоритма состоит в построении *наиболее надежного базиса* некоторого эквивалентного кода. Начиная с первого столбца матрицы $\mathbf{G}'$, находятся первые K линейно независимых столбцов, которым соответствуют наибольшие надежности. Далее эти K столбцов используются как первые K столбцов новой матрицы $\mathbf{G}''$ в порядке, соответствующем их надежности. Остальные $(N\text{-}K)$ столбцов тоже упорядочиваются в порядке убывания их надежности. Этот процесс назовем вторым перестановочным отображением λ_2 таким, что

$$\mathbf{G}'' = \lambda_2[\mathbf{G}'] = \lambda_2[\lambda_1[\mathbf{G}]]$$

Применяя отображение λ_2 к последовательности $\mathbf{y}$, получаем новую переупорядоченную последовательность $\mathbf{z}$:

$$\mathbf{z} = \lambda_2[\mathbf{y}] = (z_1, z_2, \cdots, z_k, z_{k+1}, \cdots, z_N),$$

где $|z_1| \geq |z_2| \geq \ldots \geq |z_K|$ и $|z_{K+1}| \geq |z_{K+2}| \geq \ldots \geq |z_N|$. С помощью элементарных операций над строками матрицы $\mathbf{G}''$ получаем в систематической форме матрицу $\mathbf{G}_1$:

$$\mathbf{G}_1 = (\mathbf{I}_k \mathbf{P}) = \begin{pmatrix} 1 & 0 & \cdots & 0 & p_{1,1} & \cdots & p_{1,N-K} \\ 0 & 1 & \cdots & 0 & p_{2,1} & \cdots & p_{2,N-K} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & p_{k1} & \cdots & p_{k,N-K} \end{pmatrix}.$$

Легко видеть, что код, генерируемый матрицей $\mathbf{G}_1$, эквивалентен коду C в том смысле, что генерируются те же кодовые слова с перестановкой позиций кодовых символов.

Следующий шаг состоит в реализации *жесткого декодирования*, учитывающего k наиболее надежных позиций переупорядоченной принятой последовательности $\mathbf{z}$. Обозначим $\mathbf{u}_a = (u_1, u_2, \ldots, u_K)$ результат декодирования. Соответствующее кодовое слово из кода C_1 может быть найдено как

$$\mathbf{v} = \mathbf{u}_a \mathbf{G}_1 = (v_1 \, v_2 \cdots v_K \, v_{K+1} \cdots v_N).$$

Оценка $\mathbf{v}_{\mathrm{HD}}$ кодового слова кода C может быть получена из $\mathbf{v}$ с помощью обратного отображения как

$$\mathbf{v}_{HD} = \lambda_1^{-1}[\lambda_2^{-1}[\mathbf{v}]]. \qquad (7.6)$$

На основе полученного жесткого решения $\mathbf{v}$ алгоритм выполняет повторные проходы (итерации) до тех пор, пока не будет достигнута *практическая оптимальность* или достаточная эффективность.

Итеративный процесс l-го порядка определяется следующим образом.

Для $1 \leq i \leq l$ выполняются все возможные замены i из k наиболее надежных символов кодового слова $\mathbf{v}$.

Для каждого пробного изменения находится соответствующее кодовое слово $\mathbf{v}'$ из C_1 и его отображение в двоичную последовательность $\mathbf{x}'$ действительных чисел с помощью отображения $\{0,1\} \rightarrow \{+1, -1\}$.

Для каждого кодового слова $\mathbf{v}'$ вычисляется квадрат Евклидова расстояния (1.33) между сгенерированной последовательностью $\mathbf{x}'$ и переупорядоченной принятой последовательностью $\mathbf{z}$. После того как закончится генерация

$$1 + k + \binom{k}{2} + \ldots + \binom{k}{l}$$

пробных кодовых слов и сравнение их расстояний, алгоритм выбирает кодовое слово $\mathbf{v}''$, ближайшее к $\mathbf{z}$. Наиболее вероят-

ное кодовое слово $\mathbf{v}''_{\mathrm{ML}}$ вычисляется из $\mathbf{v}''$ с помощью обратного отображения (7.6).

Итерации могут быть организованы так, чтобы минимизировать количество вычислений. В частности, как упоминалось в Разделе 7.1, для двоичных сигналов в Гауссовом канале достаточно вычислять *корреляцию* между генерируемыми кодовыми словами и переупорядоченной принятой последовательностью. Таким образом, не требуется и вычисление действительной последовательности $\mathbf{x}'$. Все, что требуется в этом случае, это сложение принятых значений с измененными знаками, соответствующими изменениям в последовательности $\mathbf{v}$ при генерации $\mathbf{v}''$.

7.6. Декодирование по минимуму обобщенного расстояния

В 1966 г. Форни [For3] ввел декодирование по минимуму обобщенного расстояния (МОР). Основная идея состояла в том, чтобы расширить понятие стирания, разделяя принятые значения по *классам (уровням) надежности*. Стратегия декодирования аналогична алгоритму Чейза, Тип-III, с использованием *комбинаций стираний*. В процессе декодирования по МОР вводятся дополнительные стирания на $d-1$ наименее надежных позициях и проверяется *условие достаточности* для выбора оптимального решения. Итеративный процесс продолжается до тех пор, пока не выполнится это условие или не будет введено максимальное число стираний.

Пусть C линейный блоковый (N, K, d) код. Предположим, что имеется *декодер*, *исправляющий стирания и ошибки*, способный исправлять любую комбинацию e ошибок и s стираний в пределах корректирующей способности кода, т.е. $2e + s \le d - 1$. Такие декодеры рассматривались в Разделе 3.5.6 для БЧХ кодов и в Разделе 4.3.2 для РС кодов.

Пусть $\mathbf{r} = (r_1, r_2, \ldots, r_N)$ принятое слово на выходе канала. На i-ой позиции для переданного *символа* c_i имеем оценку $r_i = (-1)^{c_i} + w_i$, где w_i Гауссова случайная величина с нулевым средним и дисперсией $N_0/2$, $i = 1, 2, \ldots, N$. Предполагается, что слово $\mathbf{r}$ нормализовано таким образом, что все его компоненты ограничены интервалом $[-1, +1]$ (с помощью двустороннего ограничителя). Как обычно, знаковый бит представляет жесткое (посимвольное) решение относительно принятого слова,

$$\mathbf{z} = (z_1 z_2 \cdots z_N),\ z_j = \operatorname{sgn}(r_j),\ 1 \le j \le N.$$

Надежности принятых из канала величин упорядочиваются, как в алгоритмах Чейза и OSD, и формируется список индексов $I_j, j = 1, 2, \ldots, N$, такой, что

$$|r_{I_1}| \le |r_{I_2}| \le \cdots \le |r_{I_N}|.$$

На первом этапе декодирования принятое с жесткими решениями слово вводится в алгебраический декодер (если МОР декодирование применяется к кодам БЧХ или РС) с исправлением *только ошибок*. Пусть $\mathbf{v}$ есть результат такого декодирования. Корреляционная метрика слова $\mathbf{v}$ относительно принятого слова $\mathbf{r}$ равна

$$\nu = \sum_{j=1}^{N} (-1)^{v_j} \times r_j. \tag{7.7}$$

Если выполняется следующее *условие достаточности*

$$\nu > n - d, \tag{7.8}$$

то слово $\mathbf{v}$ выбирается как наиболее вероятное и на этом декодирование заканчивается.

В противном случае выполняется следующий этап декодирования. На втором этапе вводятся два стирания, $s = 2$, на позициях с индексами I_1 и I_2 и выполняется исправление *ошибок-и-стираний*. Далее вычисляется корреляционная метрика (7.7) между $\mathbf{r}$ и результатом декодирования $\mathbf{v}$ и проверяется условие (7.8).

Если это необходимо, то выполняется очередной этап процедуры. На каждом новом этапе добавляются по два стирания, $s = s + 2$, на наименее надежных позициях, пока их количество не превысит максимально возможного ($s_{\max} = d - 1$). Если

в результате всех этапов декодирования (включая $s = 0$) не будет найдено ни одного кодового слова, то либо фиксируется *отказ* от декодирования (неисправимая ошибка), либо выдается жесткое решение **z**. Заметим, что в случае РС кода всегда можно восстановить кодовое слово по k самым надежным позициям, т.е. с исправлением $d - 1$ стираний.

7.6.1 Оптимизированные условия достаточности

Условие (7.8), использованное в декодировании по МОР, может быть улучшено и применено к другим алгоритмам, формирующим список решений, таким как алгоритмы Чейза и OSD. Эти алгоритмы являются примерами *списочного декодирования*. Критерий (7.8) является слишком жестким, вследствие чего часто отбрасываются возможные кодовые слова, среди которых могут быть и наиболее вероятные (т.е. такие, которые выбрал бы декодер по максимуму правдоподобия). Известны несколько вариантов улучшенных (оптимизированных) условий достаточности. Два из них представлены ниже без доказательств. Для объяснения этих условий необходимо ввести некоторые определения.

Пусть **x** есть двоичное кодовое слово с фазовой модуляцией (7.2), $\mathbf{x} = m(\mathbf{v})$, где $\mathbf{v} \in C$ и $x_i = (-1)^{v_i}$, $1 \le i \le N$. Введем следующие множества: $S_e = \{i: \operatorname{sgn}(x_i) \ne \operatorname{sgn}(r_i)\}$ – множество позиций с ошибками, $U = \{I_j, j = 1, 2, ..., d\}$ – множество наименее надежных позиций и множество $T = \{i: \operatorname{sgn}(x_i) = \operatorname{sgn}(r_i),\ i \in U\}$. Тогда *расширенное расстояние* или *корреляционный дефект* между кодовым словом **v** и принятым словом **r** определяется [TP] следующим образом

$$d_e(\mathbf{v}, \mathbf{r}) = \sum_{i \in S_e} |r_i| \tag{7.9}$$

Улучшенный критерий выбора (оптимального) кодового слова основан на верхней границе корреляционного дефекта (7.9) и на увеличении мощности множеств тестируемых позиций. Два улучшения условий, предложенных Форни:

• **Taipale-Pursley** [TP] Существует оптимальное кодовое слово $\mathbf{x}_{\text{opt}}$ такое, что

$$d_e(x_{opt}, \mathbf{r}) = \sum_{i \in T} |r_i|. \tag{7.10}$$

• **Касами и др.** [KKTFL] Существует оптимальное кодовое слово $\mathbf{x}_{\text{opt}}$ такое, что

$$d_e(x_{opt}, \mathbf{r}) = \sum_{i \in T_k} |r_i| \tag{7.11}$$

где $T_k = \{i: \operatorname{sgn}(x_i) \ne \operatorname{sgn}(r_i)\}$, $i \in U$.

Полезные материалы по методам декодирования по МОР, их расширению и комбинациям с алгоритмами Чейза имеются в [KNIH], [Kam], [TKK], [FL4] и [TLFL], а также [BZ3*].

7.7. Списочное декодирование

Списочное декодирование (т.е. выдача списка возможных решений декодера) было введено Элайесом и Возенкрафтом (Elias, Wozencraft) [Eli3]. Совсем недавно списочное декодирование полиномиальных кодов стало объектом пристального внимания, вызванного главным образом публикациями Судана и его коллег [Sud, Gur] о декодировании кодов РС за пределами их конструктивной корректирующей способности. Предложенный ими метод, который называют *алгоритмом Судана*, использует интерполяцию и факторизацию многочленов двух переменных над расширением базового поля. Алгоритм Судана можно рассматривать как развитие алгоритма Берлекэмпа-Велча [Ber2]. Этот алгоритм был применен для декодирования с мягким решением кодов РС [KV].

7.8. Алгоритмы декодирования с мягким выходом (soft-output)

Предыдущие разделы этой главы были посвящены алгоритмам декодирования, формирующим на выходе наиболее вероятную

кодовую последовательность или кодовое слово (или список кодовых слов). Однако, с появлением «революционной» работы по турбо кодам в 1993 г. [BGT] возникла потребность в алгоритмах, которые вычисляют не только наиболее вероятное кодовое слово (или список кодовых слов), но и оценивают *надежность* их символов для дальнейшей обработки. В области помехоустойчивого кодирования алгоритмы с мягким выходом были введены намного раньше, в 1962 г. в работе Галагера (Gallager) [Gal] по кодам с низкой плотностью проверок на четность[3] (LDPC – low-density parity-check code) и несколько позже в работе Бала (Bahl) с соавторами [BCJR]. В обоих случаях алгоритмы выполняют проходы *вперед – назад* для вычисления надежностей кодовых символов. В следующем разделе рассматриваются базовые алгоритмы декодирования с мягким выходом. Компьютерные программы, моделирующие эти алгоритмы декодирования, могут быть найдены на ECC вебсайте.

В следующих ниже разделах, для более простого изложения, предполагается, что линейный блоковый код построен с помощью терминирования двоичного сверточного кода памяти m и скорости $1/n$ и что сигналы передаются по каналу с АБГШ. Предполагается также, что сверточный кодер стартует с нулевого начального состояния $S_0^{(0)}$ и через N шагов (трellисных секций) останавливается в нулевом состоянии $S_N^{(0)}$.

7.8.1. Алгоритм Витерби с мягким выходом

В 1989 г. была предложена модификация алгоритма Витерби с выдачей побитовых оценок надежности [HH]. *Алгоритм Витерби с мягким выходом* (*soft-output viterbi algorithm - SOVA*) вычисляет надежности (или мягкий выход) информационных символов как *логарифм отношения правдоподобия* (*log-likelihood ratio – LLR*),

$$\Lambda(u_i) = \log\left(\frac{\Pr(u_i = 1 \mid \mathbf{r})}{\Pr(u_i = 0 \mid \mathbf{r})}\right), \qquad (7.12)$$

[3] LDPC коды рассматриваются в Главе 8.

где $\mathbf{r}$ принятая последовательность.

Работа декодера SOVA может быть разделена на две части. В первой части декодирование выполняется также как и в обычном алгоритме Витерби, который выбирает наиболее вероятную кодовую последовательность $\mathbf{v}$, соответствующую пути (на решетке) с максимальной (корреляционной) метрикой на n-ом шаге. Подробности в Разделе 5.4. В дополнение к обычному алгоритму здесь требуется сохранение метрик путей на каждом шаге и для каждого состояния декодера. Эти метрики потребуются в последней части алгоритма для вычисления *мягких выходов*. Во второй части SOVA алгоритм Витерби выполняется в обратном направлении, т.е. вычисляются метрики и пути, начинающиеся из N-ой секции решетки и заканчивающиеся в i-ой секции, $i \geq 0$. Заметим, что во втором проходе не требуется сохранение выживших путей, а необходимо только сохранение метрик для каждого состояния решетки. Таким образом, после второго прохода для каждого узла решетки имеются две метрики: одна для левой части пути, а вторая для правой части пути, проходящего через него. Затем для каждой секции решетки вычисляются мягкие выходы.

Обозначим $M_{\max}$ (корреляционную) метрику наиболее правдоподобной последовательности $\mathbf{v}$, найденной алгоритмом Витерби. Условная вероятность соответствующей информационной последовательности $\mathbf{u}$ при заданной принятой последовательности, иначе *апостериорная вероятность* (APP) , зависит от $M_{\max}$ так как справедлива следующая оценка

$$\Pr\{\mathbf{u} \mid \mathbf{r}\} = \Pr\{\mathbf{v} \mid \mathbf{r}\} \approx e^{M_{\max}} \qquad (7.13)$$

Без потери общности можно считать, что оценка APP относительно информационного бита u_i имеет вид

$$\Pr\{u_i = 1 \mid \mathbf{r}\} \approx \exp(M_i(1)),$$

где $M_i(1) \triangleq M_{max}$. Обозначим $M_i(0)$ максимальную метрику пути ассоциированного с дополнением информационного символа u_i. Легко показать, что

$$\Lambda(u_i) \approx M_i(1) - M_i(0). \qquad (7.14)$$

Таким образом, для i-ой секции решетки мягкий выход может быть получен из разности между максимальной метрикой путей на решетке, для которых $u_i = 1$, и максимальной метрикой путей с $u_i = 0$.

На стадии формирования мягкого выхода алгоритма SOVA для i-ой секции определяется наиболее правдоподобное значение информационного символа $u_i = a$, $a \in \{0,1\}$, и соответствующая ему максимальная метрика (которая была найдена на прямом проходе алгоритма Витерби) устанавливается равной $M_i(u_i)$. Метрика наилучшего пути для дополнения $M_i(u_i \oplus 1)$ может быть найдена следующим образом [Vuc]

$$M_i(v_i \oplus 1) = \min_{k_1,k_2}\{M_f(S_{i-1}^{k_1}) + BM_i^{(b_1)}(u_i \oplus 1) + M_b(S_i^{k_2})\}, \qquad (7.15)$$

где $k_1, k_2 \in \{0, 1, 2, \ldots, 2^m - 1\}$,

- $M_f(S_{i-1}{}^{(k1)})$ метрика выжившего пути на прямом проходе алгоритма для $(i-1)$-ой секции и состояния $S^{(k1)}$,
- $BM_i^{(b1)}(u_i \oplus 1)$ метрика ребра для инвертированного информационного символа, ассоциированного с переходом из состояния $S^{(k1)}$ в $S^{(k2)}$,
- $M_b(S_i^{(k2)})$ метрика выжившего пути на обратном проходе момента i и состояния $S^{(k2)}$.

Окончательно, мягкий выход вычисляется как разность

$$\Lambda(u_i) = M_i(1) - M_i(0). \qquad (7.16)$$

Пример 88. Пусть C терминированный (12, 4) код, построенный из сверточного кода памяти 2, скорости 1/2 с генераторами (7, 5). Базовая структура решетки этого кода такая же, как в Примере 86. Предположим, что информационная последовательность (включая хвост) имеет вид $\mathbf{u} = (110100)$ и что принятая двоичная последовательность на выходе канала с АБГШ имеет вид

$$\mathbf{r} = (-4, -1, -1, -3, +2, -3, +3, +3, -3, +3, -3, +1).$$

На Рисунке 77 показана решетка этого кода. Для всех $i = 0, 1, \ldots, 6$ над каждым состоянием решетки написана метка в виде

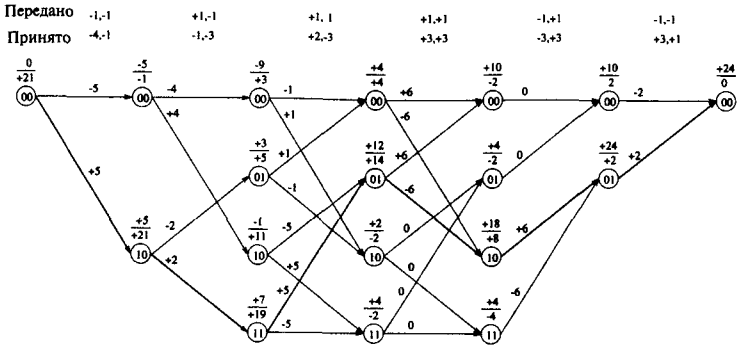


Рис. 77. Решетчатая диафрагма, использованная для SOVA декодирования в Примере 88.

$$\frac{M_f(S_i^{(m)})}{M_b(S_i^{(m)})}$$

Ребра этой решетки помечены реберной метрикой BM_i. Мягкие выходы $\Lambda(u_i)$ для $i = 0, 1, \ldots, 6$ равны, соответственно, −34, −16, +22, -16, +24, +24.

Замечания по реализации

В декодере, реализующем алгоритм SOVA, алгоритм Витерби используется дважды. На прямом проходе он работает как обычный декодер Витерби с тем только отличием, что должны сохраняться метрики путей для каждого состояния решетки. На обратном проходе алгоритма Витерби сохраняются только метрики для всех состояний декодера без сохранения выживших путей. Заметим, что выполнение обратного прохода и вычисление мягких выходов можно совместить в одном процессе. Особое внимание необходимо уделить нормализации метрик для каждого состояния декодера в обоих направлениях прохода. Другие соображения по реализации те же, что и для алгоритма Витерби, обсуждались ранее в Разделах 5.4.3 и 5.5.1.

Декодер SOVA может быть использован в режиме *скользящего окна* (*sliding window decoder*), как обычный декодер Витер-

би. За счет увеличения объема вычислений декодер может работать непрерывно, т.е. не в блоковом режиме, без периодического принудительного возврата декодера в нулевое состояние. Здесь используется та же самая идея, что и в декодере Витерби с памятью обратного прохода, которая обсуждалась в разделе 5.4.3. В этом варианте прямой и обратный проходы декодера Витерби, прямой и обратный проходы алгоритма SOVA и вычисления мягких выходов выполняются с использованием различных блоков памяти (см. [Vit3]).

7.8.2. Алгоритм декодирования по максимуму апостериорной вероятности (MAP)

Алгоритм BCJR [BCJR] является оптимальным алгоритмом посимвольного MAP декодирования линейных блоковых кодов, который минимизирует вероятность ошибки на символ. Цель MAP декодирования состоит в вычислении апостериорных вероятностей информационных символов при заданной принятой последовательности $\mathbf{r}$ по правилу (7.12). Ниже рассматривается алгоритм MAP декодирования в версии близкой к оригиналу [BCJR].

Переходы между состояниями (или ребра) на решетке имеют вероятности

$$\Pr\{S_i^{(m)} \mid S_{i-1}^{(m')}\}, \tag{7.17}$$

а для (кодовых) символов $\mathbf{v}_i$

$$q_i(x_i \mid m', m) \triangleq \Pr\{x_i = x \mid S_{i-1}^{(m')}, S_i^{(m)}\}, \tag{7.18}$$

где

$$x = \pm 1,\ x_i = m(v_i) = (-1)^{v_i},\ 0 < i \le N.$$

Последовательность $\mathbf{x}$ передается по каналу с АБГШ и принимается как последовательность $\mathbf{r}$ с переходными вероятностями

$$\Pr\{\mathbf{r} \mid \mathbf{x}\} = \prod_{i=1}^{N} p(\mathbf{r}_i \mid \mathbf{x}_i) = \prod_{i=1}^{N}\prod_{j=0}^{n-1} p(r_{i,j} \mid x_{i,j}) \tag{7.19}$$

где вероятность $p(r_{i,j}|x_{i,j})$ определена в (7.1).

Обозначим $B_i^{(j)}$ множество ребер, связывающих состояния $S_{i-1}^{(m')}$ с $S_i^{(m)}$ таким образом, что соответствующий информационный бит $u_i = j, j \in \{0, 1\}$. Тогда

$$\Pr(u_i = j \mid \mathbf{r}) = \sum_{(m',m)\in B_i^{(j)}} \Pr(S_{i-1}^{(m')}, S_i^{(m)}, \mathbf{r}) \triangleq \sum_{(m',m)\in B_i^{(j)}} \sigma_i(m', m). \tag{7.20}$$

Величина $\sigma_i(m', m)$ в (7.20) равна

$$\sigma_i(m', m) = \alpha_{i-1}(m')\gamma_i^{(j)}(m', m)\beta_i(m), \tag{7.21}$$

где совместная вероятность $\alpha_i(m) \triangleq \Pr\{S_i^{(m)}, \mathbf{r}\}$ определена рекурсивно как сумма

$$\alpha_i(m) = \sum_{m'} \alpha_{i-1}(m') \sum_{j=0}^{1} \gamma_i^{(j)}(m', m). \tag{7.22}$$

Обычно ее называют *метрикой прямого прохода*.

Условная вероятность $\gamma_i^{(j)}(m', m) \triangleq \Pr\{S_i^{(m)}, \mathbf{r} \mid S_{i-1}^{(m')}\}$ задается следующей суммой

$$\gamma_i^{(j)}(m', m) = \sum_{x} p_i(m \mid m') \Pr\{x_i = x \mid S_{i-1}^{(m')}, S_i^{(m)}\} \Pr\{r_i \mid x\} \tag{7.23}$$

Вероятность $p_i(m \mid m') = \Pr\{S_i^{(m)} \mid S_{i-1}^{(m')}\}$. В канале с АБГШ (7.23) можно записать в следующем виде

$$\gamma_i^{(j)}(m', m) = \Pr\{u_i = j\}\delta_{ij}(m, m') \exp\left(-\frac{1}{N_0}\sum_{q=0}^{n-1}(r_{i,q} - x_{i,q})^2\right), \tag{7.24}$$

где $\delta_{ij}(m', m) = 1$, если $\{m', m\} \in B_i^{(j)}$, и $\delta_{ij}(m', m) = 0$ в противном случае. Вероятность $\gamma_i^{(j)}(m', m)$ называют *метрикой ребра*.

Условная вероятность $\beta_i(m) \triangleq \Pr\{\mathbf{r}_f \mid S_i^{(m)}\}$ определяется как

$$\beta_i(m) = \sum_{m'} \beta_{i+1}(m') \sum_{j=0}^{1} \gamma_i^{(j)}(m', m). \tag{7.25}$$

Эту величину называют *метрикой обратного прохода*.

Комбинируя (7.25), (7.24), (7.22), (7.21) и (7.12), получаем мягкий выход (LLR – логарифм отношения правдоподобия) для информационного символа u_i в виде

$$\Lambda(u_i)=\log\left(\frac{\Pr\{u_i=1\mid \mathbf{r}\}}{\Pr\{u_i=0\mid \mathbf{r}\}}\right)=\log\left(\frac{\sum_m\sum_{m'}a_{i-1}(m')\gamma_i^{(1)}(m',m)\beta_i(m)}{\sum_m\sum_{m'}a_{i-1}(m')\gamma_i^{(0)}(m',m)\beta_i(m)}\right). \tag{7.26}$$

Жесткое решение равно $(\hat{u}_i)=\operatorname{sgn}(\Lambda(u_i))$, а надежность решения u_i есть $|\Lambda(u_i)|$. Выведенные выше уравнения можно интерпретировать следующим образом. В этом алгоритме декодирования может быть применен двухпроходный алгоритм Витерби, подобно тому, как это делается в декодере SOVA (см. предыдущий раздел). На прямом проходе для заданной вероятности смены состояния в момент i вычисляется совместная вероятность принятой последовательности до момента i и состояния в момент i. На обратном проходе вычисляется вероятность принятой последовательности от момента $i+1$ до момента N при заданном состоянии в момент i. Тогда мягкий выход зависит от совместной вероятности смены состояний и принятого символа в момент i.

MAP алгоритм.

- **Начальные установки**

  Для $m = 0, 1,\ldots, 2^m - 1$, $\alpha_0(0) = 1$, $\alpha_0(m) = 0$, $m \neq 0$.

  Для $m' = 0, 1, \ldots, 2^m - 1$, $\beta_N(0) = 1$, $\beta_N(m') = 0$, $m' \neq 0$.

- **Прямой проход**

  Для $i = 1, 2, \ldots, N$,

1. Для $j = 0, 1$ вычислить и сохранить метрики ребер $\gamma_i^{(j)}(m', m)$ в соответствии с (7.24).
2. Для $m = 0, 1, \ldots, 2^m - 1$, вычислить и сохранить метрики прямого прохода $\alpha_i(m)$ (7.22).

- **Обратный проход**

  Для $i = N-1, N-2, \ldots, 0$,

1. Вычислить обратные метрики $\beta_i(m)$ соответственно (7.25), используя метрики ребер, вычисленные на прямом проходе.

2. Вычислить логарифм отношения правдоподобия $\Lambda(u_i)$ соответственно (7.26).

Реализация

Реализация MAP декодера похожа на реализацию декодера SOVA. Оба декодера используют прямой и обратный проходы. Все предложения, упоминавшиеся в предыдущем разделе, применимы и для MAP декодера. Дополнительно заметим, что метрики ребер зависят от плотности мощности шума N_0, оценка которой должна быть известна для сохранения оптимальности декодера. Для того чтобы избежать вычислительной нестабильности, вычисление вероятностей $\alpha_i(m)$ и $\beta_i(m)$ следует масштабировать на каждом шаге декодера так, чтобы соблюдалось условие нормировки вероятностей (т.е. суммы этих вероятностей должны давать 1).

7.8.3. Log-MAP алгоритм

Для того, чтобы снизить вычислительную сложность MAP алгоритма могут быть использованы логарифмы метрик. Алгоритм, использующий такой переход, называют *log*-MAP алгоритмом.

Из (7.22) и (7.25) получаем [RVH]

$$\log\alpha_i(m)=\log\left(\sum_{m'}\sum_{j=0}^{1}\exp(\log\alpha_{i-1}(m')+\log\gamma_i^{(j)}(m',m))\right),$$
$$\log\beta_i(m)=\log\left(\sum_{m'}\sum_{j=0}^{1}\exp(\log\beta_{i+1}(m')+\log\gamma_i^{(j)}(m',m))\right). \tag{7.27}$$

Беря логарифм $\gamma_i^{(j)}(m',m)$ в (7.24), получаем

$$\log\gamma_i^{(j)}(m',m)=\delta j_{i,j}(m,m')\left\{\log\Pr\{u_i=j\}-\frac{1}{N_0}\sum_{q=0}^{n-1}(r_{i,q}-x_{i,q})^2\right\} \tag{7.28}$$

Вводя обозначения: $\alpha''_i(m) = \log\alpha_i(m)$, $\beta''_i(m) = \log\beta_i(m)$, $\gamma''^{(j)}{}_i(m', m) = \log\gamma^{(j)}{}_i(m', m)$, можно переписать (7.26) следующим образом

$$\Lambda(u_i)=\log\left(\frac{\sum_m\sum_{m'}\exp(\alpha_i''(m')+\gamma_i''^{(0)}(m',m)+\beta_i(m))}{\sum_m\sum_{m'}\exp(\alpha_i''(m')+\gamma_i''^{(1)}(m',m)+\beta_i(m))}\right). \quad (7.29)$$

В результате получен алгоритм, работающий в логарифмической форме.

Для того, чтобы избежать суммирования экспоненциальных членов, можно применить выражение, известное как Якобианов логарифм [RVH],

$$\log(e^a+e^b)=\max(a,b)+\log(1+e^{-|a-b|}). \quad (7.30)$$

Функция $\log(1 + \exp(-|a-b|))$ может быть записана в небольшую таблицу. Известно, что всего несколько значений (например, 8 в [RVH]) достаточно, чтобы достичь практически такой же точности, как и в исходном алгоритме MAP. Таким образом, вместо нескольких обращений к относительно медленной (или дорогой в аппаратном исполнении) функции $\exp(x)$ использование таблицы дает практически такой же результат.

7.8.4. Max-Log-MAP алгоритм

Модификация MAP алгоритма, известная как *Max-Log-MAP* алгоритм, обладает меньшей вычислительной сложностью, но теряет свойство оптимальности. Как было показано выше, эта модификация состоит в том, что логарифмируется MAP метрика и используется аппроксимация [RVH]

$$\log(e^a+e^b)\approx\max(a,b), \quad (7.31)$$

которая совпадает с первым членом правой части (7.30). В результате логарифм отношения правдоподобия для информационного символа u_i принимает вид

$$\Lambda(u_i)\approx\max_{m',m}\{\alpha_{i-1}''(m')+\gamma_i''^{(0)}(m',m)+\beta_i''(m)\}-\max_{m',m}\{\alpha_{i-1}''(m')+\gamma_i''^{(1)}(m',m)+\beta_i''(m)\}. \quad (7.32)$$

Вычисления на прямом и обратном проходах могут быть представлены следующим образом

$$\alpha_i''(m)=\max_{m'}\max_{j\in\{0,1\}}\{\alpha_{i-1}''(m')+\gamma_i''^{(j)}(m',m)\},$$
$$\beta_i''(m)=\max_{m'}\max_{j\in\{0,1\}}\{\beta_{i+1}''(m')+\gamma_i''^{(j)}(m',m)\}. \quad (7.32)$$

В случае применения двоичных кодов, построенных из сверточных кодов скорости $1/n$ (с помощью перфорации), сложность декодирования (измеряемая числом операций умножения и сложения) по алгоритму SOVA примерно вдвое меньше, чем по max-log-MAP алгоритму. Сложность алгоритма log-MAP примерно вдвое больше чем для max-log-MAP алгоритма. В работе [FBLH] было показано, что по достижимой вероятности ошибки max-log-MAP алгоритм эквивалентен модифицированному алгоритму SOVA. Для алгоритмов log-MAP и MAP вероятность ошибки одинакова и минимальна.

7.8.5. OSD алгоритм с мягким выходом

Алгоритм, основанный на упорядоченных оценках надежности символов, также может быть модифицирован для получения оценок надежности декодированных символов [FL5]. Эта модификация известна как *алгоритм OSD с мягким выходом* или SO-OSD. Алгоритм SO-OSD реализуется в два этапа с итерациями i-го порядка. Первый этап совпадает с обычным OSD алгоритмом, определяющим наиболее правдоподобное слово $\mathbf{v}_{\mathrm{ML}}$ итеративным процессом до i-го порядка, включительно. Для описания следующего этапа необходимо ввести следующие определения.

Для j-ой из K наиболее надежных позиций, $1\le j\le K$, найдем кодовое слово $\mathbf{v}_{\mathrm{ML}}(j)$, содержащее инвертированный символ на j-ой позиции кодового слова $\mathbf{v}_{\mathrm{ML}}$,

$$\mathbf{v}_{ML}(j) = \mathbf{v}_{ML} + \mathbf{e}(j),$$

где $\mathbf{e}(j)$ есть множество векторов длины K веса 1. Вектор $\mathbf{e}(j)$ является *представителем смежного класса* в разложении кода C_1 (эквивалентного исходному коду C после переупорядочивания позиций, см. OSD алгоритм, Раздел 7.5) на два множества кодовых слов, имеющих на j-ой позиции символ 0 или 1. Эти множества после удаления j-ой позиции образуют два подкода кода C_1, которые обозначаются $C(0)$ и $C(1)$. Пусть $C(j) = C(0)$.

Алгоритм SO-OSD состоит из следующих этапов:

1. Найти наиболее правдоподобное кодовое слово $\mathbf{v}_{\mathrm{ML}}$ итеративным процессом i-го порядка.
2. Для каждого подкода $C(j)$, $1 \le j \le K$,

(а) Установить начальные значения мягких выходов наименее надежных позиций для векторов $\mathbf{v}_{\mathrm{ML}}$ и $\mathbf{v}_{\mathrm{ML}}(j)$. Для этого вычислить

$$L_j = r_j + \sum_{\substack{\ell \neq j \\ v_\ell(0) \neq v_\ell(1)}} m(v_\ell(1)) r_\ell, \text{ где } m(v_i) = (-1)^{v_i}. \tag{7.34}$$

(б) Определить наиболее правдоподобное кодовое слово $\mathbf{v}(j) \in C(j)$ итеративным процессом i-го порядка.

(в) Вычислить L_j по формуле (7.34) для векторов $\mathbf{v}_{\mathrm{ML}}$ и $\mathbf{v}(j)$.

(г) Заменить значения мягкого выхода для наименее надежных позиций суммы $\mathbf{v}_{\mathrm{ML}} + \mathbf{v}(j)$ на величину L_j.

3. Для $K+1 \le j \le N$ выбрать наименьшее значение мягкого выхода, ассоциированного с каждой из наименее надежных позиций.

Вероятность ошибки SO-OSD алгоритма совпадает с вероятностью ошибки декодирования по max-log-MAP алгоритму для многих двоичных линейных блоковых кодов длины до 128 и высокоскоростных кодов [FL5]. Заметим еще, что уменьшение масштаба внешних (extrinsic) надежностей (7.34) улучшает эффективность на несколько десятых долей dB.

Глава 8

ИТЕРАТИВНО ДЕКОДИРУЕМЫЕ КОДЫ

Итеративное декодирование можно определить как метод, многократно использующий алгоритм декодирования с мягким выходом для того, чтобы уменьшить вероятность ошибки схемы помехоустойчивого кодирования и приблизиться к декодированию по максимуму правдоподобия при меньшей сложности вычислений. Если правильно выбран (построен) базовый код, исправляющий ошибки, то увеличение числа итераций приводит к повышению помехоустойчивости.

Технология итеративного декодирования появилась в 1954 году в работе Элайса (Elias) [Eli1], посвященной *итеративным кодам*. Позднее, в 60-х годах, важный вклад в эту технологию сделали Галлагер [Gal] и Мэсси [Mas1]. Итеративное декодирование стали называть *вероятностным декодированием* (*probabilistic decoding*). Главная идея была той же самой, что и сегодня – максимизировать апостериорную вероятность символа, который мог быть передан, при условии, что известна искаженная версия кодовой последовательности.

В этой главе представлены кодовые конструкции, для которых возможно итеративное декодирование. В литературе эти кодовые схемы были названы *турбо-подобными* (*turbo-like*) коды. Опираясь на применения итеративных алгоритмов декодирования и учитывая направленность этой главы, кодовые конструкции можно относительно свободно разделить на два класса:

Коды произведения

Примерами кодов этого класса являются *параллельные каскадные коды* или *турбо коды* и *последовательные каскадные коды*. Членами этого класса являются также блоковые коды-произведения, рассмотренные в Разделе 6.2.3.

Коды с низкой плотностью проверок (low-density parity-check (LPDC) codes)

Это линейные коды, обладающие тем свойством, что их проверочные матрицы имеют низкое отношение числа ненулевых элементов к общему числу элементов.

В обоих классах компонентные коды могут быть сверточными или блоковыми, с систематическим или несистематическим кодированием, а также любая их комбинация. Изучение итеративной техники декодирования, начнем с обсуждения фундаментальной структуры турбо кодов.

Турбо коды

Последние восемь с лишним лет отмечалось огромное количество исследований, посвященных анализу итеративных алгоритмов декодирования и конструированию *итеративно декодируемых* кодов или «турбо-подобных кодов», приближающихся к *пределу Шеннона*[1]. Турбо коды были введены в 1993 [BGT]. Результаты, опубликованные тогда, вызвали шок во всем научном сообществе.

Например, конструкция скорости 1/2, состоящая из двух рекурсивных систематических сверточных (RSC) кодов скорости 2/3 (перфорированный код скорости 1/2) с 16 состояниями и перемежителя размера 256×256, обеспечила вероятность ошибки (BER) порядка 10^{-5} при передаче двоичных сигналов в Гауссовом канале и отношении сигнал-шум на бит, $E_b/N_0 =$ 0,7 dB. Эта точка оказалась ближе к пределу Шеннона (0,2 dB), чем когда-либо прежде. См. Рисунок 78.

Основные элементы схемы турбо кодирования, предложенные в [BGT], состоят в следующем.

Схема кодирования

Схема *произведения кодов* (названная в оригинальной работе [BGT] *параллельным каскадным кодом*) с компонентными *рекурсивными систематическими* сверточными кодами.

[1] Предел Шеннона относится к пропускной способности канала с дискретным входом и непрерывным выходом

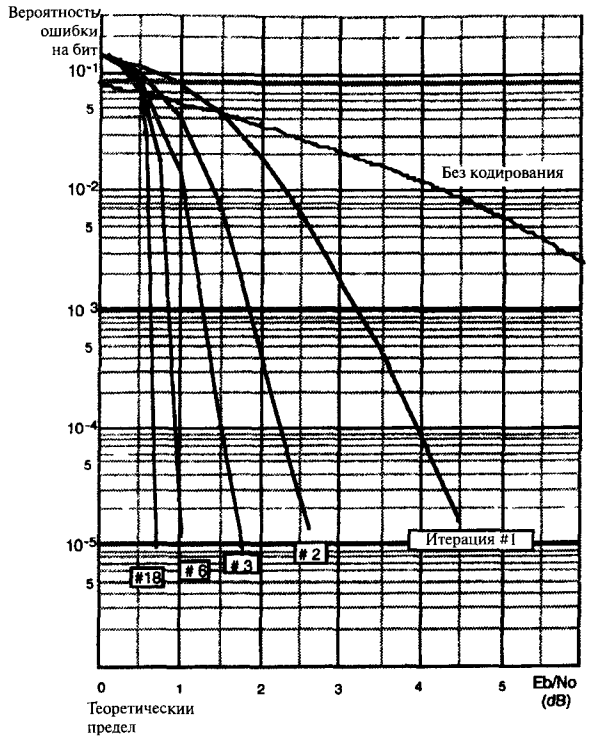


Рис. 78. Вероятность ошибки турбо кода скорости 1/2 с компонентными RSC кодами скорости 2/3, памяти 4, $g_0 = 21$ и $g_1 = 37$, и блоковым перемежителем размера 256×256 = 65536. Скопировано из [BG2] с разрешения IEEE (©1996 IEEE).

Оценка надежности

Для компонентных кодов применяются декодеры (SISO – с мягким входом и мягким выходом) по максимуму апостериорной вероятности (MAP), чтобы генерировать *логарифмические отношения правдоподобия*.

Итеративное декодирование

Надежности некоторых символов (в виде *внешней (extrinsic) информации*) передаются через обратную связь от внешнего декодера (по столбцам) к внутреннему (по строкам) и наоборот.

Случайное перемешивание

Между двумя кодерами используется *случайный перемежитель* большой длины. Его назначение состоит, главным образом, в том, чтобы на каждой итерации обеспечить независимость оценок информационных символов, получаемых компонентными MAP декодерами.

Эти четыре элемента подвергались интенсивному исследованию последние годы. В работах [Vuc, ISTC1, ISTC2, JSAC1, JSAC2, HeW] даны хорошие примеры исследований.

8.1. Итеративное декодирование

Цель этой части состоит в том, чтобы ввести основные идеи, лежащие в основе итеративного декодирования. Рассмотрим апостериорное логарифмическое отношение правдоподобия (LLR)[2] для информационного символа. Выражение (7.12) повторяется здесь для удобства,

$$\Lambda(u_i) = \log\left(\frac{\Pr\{u_i = 1 \mid \mathbf{r}\}}{\Pr\{u_i = 0 \mid \mathbf{r}\}}\right). \tag{8.1}$$

Обозначим $C_i(0)$ и $C_i(1)$ множества модулированных последовательностей $\mathbf{x} = m(\mathbf{v}) \in m(C)$ таких, что на i-ой позиции последовательности $\mathbf{v} \in C$ находится символ 0 или 1, соответственно. Напомним, что модуляция есть взаимно однозначное отображение двоичных символов, $v \in \{0, 1\}$, на $x \in \{+1, -1\}$, как в выражении (7.2). Тогда LLR (8.1) для символа может быть записано следующим образом

$$\begin{aligned}\Lambda(u_i) &= \log\left(\frac{\sum_{\mathbf{x}\in C_i(1)}\prod_{l=1}^{N} p(r_l, x_l)}{\sum_{\mathbf{x}\in C_i(0)}\prod_{l=1}^{N} p(r_l, x_l)}\right) \\ &= \log\left(\frac{p(r_i \mid x_i = -1)\Pr(x_i = -1)\sum_{\mathbf{x}\in C_i(1)}\prod_{l\neq i}^{N} p(r_l, x_l)}{p(r_i \mid x_i = +1)\Pr(x_i = +1)\sum_{\mathbf{x}\in C_i(0)}\prod_{l\neq i}^{N} p(r_l, x_l)}\right).\end{aligned} \tag{8.2}$$

[2] Некоторые авторы вместо LLR используют термин «L – значение».

Отсюда следует, что LLR для информационного символа может быть представлено суммой

$$\Lambda(u_i) = \Lambda_{ci} + \Lambda_a(u_i) + \Lambda_{i,e}(C), \tag{8.3}$$

где Λ_{ci} называют LLR *канала*, величину $\Lambda_a(u_i)$ называют *априорным LLR* информационного символа и $\Lambda_{i,e}(C)$ называют *внешним* (extrinsic) LLR, характеризующим ограничения, накладываемые *другими информационными символами кода*. Значения этих величин определяются следующими соотношениями:

$$\begin{aligned}\Lambda_{ci} &= \log\left(\frac{p(r_i \mid x_i = -1)}{p(r_i \mid x_i = +1)}\right) \\ &= \begin{cases} (-1)^{r_i}\log\left(\dfrac{1-p}{p}\right), & \text{для ДСК с параметром } p; \\ \dfrac{4}{N_0} r_i, & \text{для канала с АБГШ } (\sigma_n^2 = N_0/2); \\ \dfrac{4}{N_0} a_i r_i, & \text{для общих Релеевских замираний.} \end{cases}\end{aligned} \tag{8.4}$$

$$\Lambda_a(u_i) = \log\left(\frac{\Pr(x_i = -1)}{\Pr(x_i = +1)}\right) = \log\left(\frac{\Pr(u_i = 1)}{\Pr(u_i = 0)}\right), \tag{8.5}$$

$$\Lambda_{i,e}(C) = \log\left(\frac{\sum_{\mathbf{x}\in C_i(1)}\prod_{l\neq i}^{N} p(r_l, x_l)}{\sum_{\mathbf{x}\in C_i(0)}\prod_{l\neq i}^{N} p(r_l, x_l)}\right). \tag{8.6}$$

Предположим, что двоичные сигналы передаются по каналу с АБГШ и что в качестве компонентных кодеров используются двоичные рекурсивные систематические кодеры скорости $1/n$. В итеративной процедуре декодирования *внешняя* (*extrinsic*) информация $\Lambda_{i,e}(C)$ может быть введена в декодер (через обратную связь) и использована в качестве априорной

вероятности на следующей итерации. В процедуре декодирования, использующей прямой и обратный проходы по решетке (см. Раздел 7.8.2), внешнее LLR может быть записано в следующем виде

$$\Lambda_{i,e}(C)=\log\left(\frac{\sum_{(m,m')\in B_i(1)}\alpha_{i-1}(m')\,\xi_i(m',m)\,\beta_i(m)}{\sum_{(m,m')\in B_i(0)}\alpha_{i-1}(m')\,\xi_i(m',m)\,\beta_i(m)}\right) \tag{8.7}$$

где вероятности $\alpha_i(m)$ и $\beta_i(m)$ определены в (7.22) и (7.25), соответственно.

Для вычисления внешнего LLR необходимо ввести *модифицированную метрику ребра*

$$\xi_i(m',m) = \delta_{ij}(m,m')\exp\left(\frac{E}{N_0}\sum_{q=1}^{n-1} r_{i,q}x_{i,q}\right) \tag{8.8}$$

где, как обычно,

$$\delta_{ij}(m,m')=\begin{cases}1, & \text{если } (m,m')\in B_i^{(j)};\\ 0, & \text{в остальных случаях.}\end{cases}$$

Внешнее (логарифмическое) отношение правдоподобия (LLR) для i-ой информационной позиции не содержит какой-либо переменной, непосредственно связанной с u_i, $i = 1, 2,\ldots, N$. Так как кодирование по определению систематическое, то все метки ребер решетки имеют вид $(u_i, v_{i,1}, \ldots, v_{i,n-1})$ и, следовательно, суммирование в выражении для модифицированной метрики начинается с $q = 1$[3].

В общем случае двумерного кодирования первый декодер (т.е. по строчкам) вычисляет $\Lambda_{i,j}^{(1)}(C)$ и передает его второму декодеру (по столбцам) как априорную информацию $\Lambda_a^{(2)}(u_i)$, необходимую для вычисления LLR информационного символа u_i. Другими словами, вешняя информация дает (мягкий) выход, зависящий от тех (мягких) входов (надежностей), которые не связаны непосредственно с информационным символом u_i. Итеративное декодирование составных кодов является темой следующего раздела.

[3] Сравните это выражение с метрикой ребер $\gamma_i^{(j)}(m', m)$ в (7.24).

8.2. Составные коды

В этом разделе представлены схемы кодирования составных кодов с перемежителями. Эти схемы позволяют использовать простые декодеры компонентных кодов. В итеративной процедуре посимвольные MAP декодеры компонентных кодов могут обмениваться внешними оценками LLR, существенно повышая надежность оценки информационного символов с увеличением числа итераций.

8.2.1. Параллельная схема: турбо коды

На Рисунке 79 показана структурная схема кодера *параллельного составного кода*, более известного как *турбо код*. Используются два кодера, каждый из которых вычисляет только проверочные символы. Например, если используются RSC коды скорости 1/2, то кодеры соответствуют дробной части $g_1(D)/g_0(D)$ полиномиальной порождающей матрицы $\mathbf{G}(D)$. Входами кодеров являются последовательности $\mathbf{u}$ и $\mathbf{P}\,\mathbf{u}$, где $\mathbf{P}$ *матрица перестановок*[4], ассоциированная с перемежителем. Выход кодера, соответствующий входному символу u_i, имеет вид $(u_i, v_{P1,i}, v_{P2,i})$, где $v_{P1,i}$ и $v_{P2,i}$ представляют проверочные символы для $i = 1, 2, \ldots, N$ и N длина блока.

В оригинальной работе в конструкции турбо кода были использованы RSC коды. Для изучения этой схемы предположим, что в качестве компонентных кодов использованы два двоичных систематических линейных блоковых (n_i, k_i) кода, $i = 1, 2$. Скорость составного кода равна (см. Рисунок 80)

$$R=\frac{K}{N}=\frac{k_1k_2}{n_1n_2-(n_1-k_1)(n_2-k_2)}=\frac{R_1R_2}{R_1+R_2-R_1R_2} \tag{8.8}$$

где $R_i = k_i/n_i$, $i = 1, 2$. Обозначим $\mathbf{G}_i = (\mathbf{I}_i \,|\, \mathbf{P}_i)$ порождающую матрицу кода C_i. Тогда порождающая матрица параллельного составного кода C_{PC} может быть приведена к следующему виду

[4] Матрица перестановок есть двоичная матрица, в каждой строке и каждом столбце которой имеется ровно одна единица. Если $\pi_{i,j} = 1$, то символ из i-ой позиции на входе перемещается на j-ую позицию на выходе.

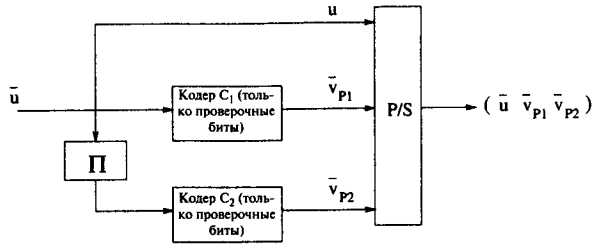


Рис. 79. Структура кодера параллельного составного (турбо) кода.

Рис. 80 Параллельный составной код, интерпретируемый как *перфорированный код-произведение.*

$$\mathbf{G}_{PC} = \left(\mathbf{I}_{k_1 k_2} \left(\begin{matrix} \mathbf{P}_1 & & \\ & \mathbf{P}_1 & \\ & \mathbf{0} & \\ & & \mathbf{P}_1 \end{matrix} \right) \Pi \left(\begin{matrix} \mathbf{P}_2 & & \\ & \mathbf{P}_2 & \\ & \mathbf{0} & \\ & & \mathbf{P}_2 \end{matrix} \right) \right) = \left(\mathbf{I}_{k_1 k_2} | \mathbf{P}_1' | \mathbf{P}_2' \right) \tag{8.10}$$

где П матрица перестановок, ассоциированная с перемежителем, и $\mathbf{P}_i$ проверочная часть порождающей матрицы кода C_i, $i = 1, 2$. Подматрица $\mathbf{P_1}$ появляется в блочно-диагональной подматрице $\mathbf{P}_1'$ матрицы $\mathbf{G_{PC}}$ ровно k_2 раз, а подматрица $\mathbf{P}_2$ появляется k_1 раз в подматрице $\mathbf{P}_2'$. Остальные элементы подматриц $\mathbf{P}_1'$ и $\mathbf{P}_2'$ равны нулю. Отсюда следует, что кодовое слово кода C_{PC} имеет вид $(\mathbf{u} \mid \mathbf{u}\mathbf{P}_1' \mid \mathbf{u}\mathbf{P}_2')$.

Пример 89. Пусть C_1 код повторение (2, 1, 2) и C_2 код с одной проверкой (3, 2, 2). Тогда $k_1 k_2 = 2$. В этом случае существует единственная матрица перестановок $\mathbf{P}$. Проверочные подматрицы и матрица перестановок равны

$$\mathbf{P}_1 = (1), \quad \mathbf{P}_2 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad \Pi = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{8.8}$$

соответственно. Из (8.10) следует, что порождающая матрица (5, 2) параллельного составного кода C_{PC} имеет вид

$$\begin{aligned} \mathbf{G}_{PC} &= \left(\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} (1) & 0 \\ 0 & (1) \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \right) \\ &= \left(\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} (1) & 0 \\ 0 & (1) \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \right) \\ &= \begin{pmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \end{pmatrix} \end{aligned} \tag{8.2}$$

Заметим, что минимальное расстояние этого кода $d_{\mathrm{PC}} = 3$, т.е. меньше чем для обычного каскадного кода, составленного из кодов C_1 и C_2.

Один из способов интерпретации параллельного составного (турбо) кода с точки зрения линейных блоковых кодов состоит в том, что он рассматривается как *перфорированный код-произведение*, у которого удалены проверочные символы, соответствующие проверкам проверок. Эта интерпретация показана на Рисунке 80. Единственное существенное различие между турбо кодом и блоковым кодом-произведением состоит в том, что его перемежитель не является просто прямоугольным, т.е. с записью по строчкам и считыванием по столбцам, а вносит достаточный беспорядок в информационную последовательность для того, чтобы работала итеративная процедура. Развивая эту интерпретацию, можно рассматривать турбо код как *обобщенный каскадный код* (ОКК)[5], заменяя перфорацию эквивалентным умножением на двоичную матрицу с последующим суммированием кодовых слов. Обозначим $\mathbf{M}_1$ матрицу размера $1 \times N$ с $n_1 k_2$ последовательными единицами, за которыми следуют $N - n_1 k_2$ нулей. Обозначим $\mathbf{M}_2$ другую

[5] См. Раздел 6.3.5.

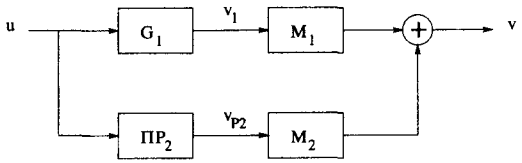


Рис. 81. Турбо кодер как кодер обобщенного каскадного кода.

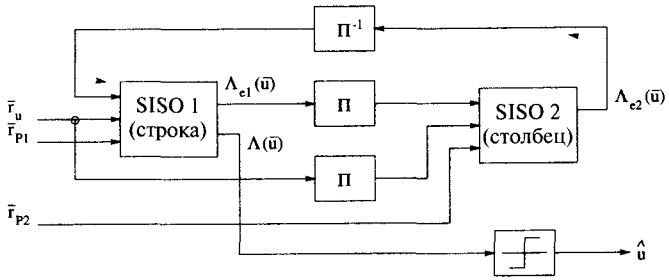


Рис. 82. Структурная схема итеративного декодера для параллельного составного (каскадного) кода.

матрицу размера $1\times N$ с $N-(n_2-k_2)k_1$ нулями, за которыми следуют $(n_2-k_2)k_1$ единиц. Тогда турбо кодер может быть представлен в виде кодера ОКК, как показано на Рисунке 81.

Итеративное декодирование параллельного составного кода

Базовая структура итеративного декодера для параллельной схемы составного кодирования с двумя компонентными кодами показана на Рисунке 82. Каждая итерация состоит из двух фаз, по одной на компонентный декодер.

Первая фаза

На первой итерации, в первой фазе SISO декодер (декодер с мягким входом и мягким выходом) первого компонентного кода вычисляет апостериорные LLR в предположении, что все символы одинаково вероятны, т.е. $\Lambda_a(\mathbf{u})=0$. Этот декодер вычисляет внешнюю информацию для каждого информационного символа, $\Lambda_{e1}(\mathbf{u})$, используя ту часть принятой последовательности, которая соответствует проверочным символам, $\mathbf{r}_{P1}$, и отдает результат второму SISO декодеру.

Вторая фаза

Во второй фазе первой итерации процедуры декодирования перемешанные (перемеженные) элементы внешней информации от первого декодера используются как априорные LLR, т.е. $\Lambda_a(\mathbf{u})=\Pi\Lambda_{e1}(\mathbf{u})$. Затем вычисляется внешняя информация $\Lambda_{e2}(\mathbf{u})$ на основе той части принятой последовательности, которая соответствует проверочным символам второго компонентного кода, $\mathbf{r}_{P2}$, завершая, таким образом, первую итерацию декодирования. В этот момент может быть сделано решение об информационных символах, основанное на значениях апостериорных LLR $\Lambda(\mathbf{u})$.

На последующих итерациях первый декодер использует деперемеженные элементы внешней информации от второго декодера, $\Pi^{-1}\Lambda_{e2}(\mathbf{u})$, как априорные LLR для вычисления мягкого выхода (т.е. апостериорного LLR) $\Lambda(\mathbf{u})$. Эта процедура может повторяться до тех пор, пока не выполнится условие остановки [HOP, SLF], либо реализуется заданное максимальное число итераций. Заметим, что формирование решений об информационных символах после первого декодера позволяет убрать один деперемежитель.

Как и в Разделе 7.8.4 итеративное MAP декодирование может быть упрощено за счет аппроксимации функции "log" функцией "max" как в (7.3.1). Кроме того, имеется итеративная версия алгоритма SOVA, представленная в Разделе 7.8.1, в которой по существу вычисляется апостериорное LLR $\Lambda(\mathbf{u})$ и после вычитания LLR канала, Λ_c, и внешнего LLR, $\Lambda_{ej}(\mathbf{u})$, от другого декодера формируется внешнее LLR [FV].

Пример 90. Рассмотрим (5, 2, 3) турбо код из Примера 89. Кодовое слово записано в 3×2 массив с удаленным элементом правого нижнего угла. Предположим, что записанное в массив кодовое слово v

$u_1=0$	$v_{11}=0$
$u_2=1$	$v_{12}=1$
$v_{21}=1$	X

(где символом X обозначена удаленная или игнорируемая позиция) было передано по каналу с АБГШ с помощью двоичной модуляции и было принято в следующем виде (значения LLR на выходе канала, Λ_c)

−0,5	+2,5
−3,5	+1,0
+0,5	X

Для дальнейших вычислений мы используем тот факт, что для LLR суммы двух двоичных переменных x_1 и x_2 справедливо следующее выражение [HOP]

$$\Lambda(x_1 + x_2) = \log\left[\frac{\exp(\Lambda(x_1) + \Lambda(x_2))}{(1+\exp(\Lambda(x_1))) + (1+\exp(\Lambda(x_2)))}\right]. \quad (8.11)$$

Первая итерация

В первой фазе декодирования декодер кода-повторения (строчный) вычисляет следующие величины:

$$\Lambda_{e1}(u_1) = \Lambda(r_{11}) = +2{,}5;$$
$$\Lambda_{e1}(u_2) = \Lambda(r_{12}) = +1{,}0.$$

Во второй фазе декодирования эти элементы внешней информации строчного декодера используются как априорные LLR для декодера кода с одной проверкой (декодер столбцов). Обновленные значения LLR равны

−0,5	+2,5
−3,5	+1,0
+0,5	X

Во второй фазе первой итерации декодирования декодер столбцов вычисляет следующие внешние LLR:

$$\Lambda_{e2}(u_1) = \Lambda(u_2 + r_{21}) = \log\left[\frac{1+e^{-2}}{e^{-2,5}+e^{+0,5}}\right] = -0{,}42;$$

$$\Lambda_{e2}(u_2) = \Lambda(u_1 + r_{21}) = \log\left[\frac{1+e^{+2,5}}{e^{+2}+e^{+0,5}}\right] = +0{,}38.$$

В конце первой итерации декодер выдает следующие значения апостериорных LLR:

$$\Lambda(u_1) = \Lambda_{c1} + \Lambda_{e1}(u_1) + \Lambda_{e2}(u_1) = +1{,}58;$$
$$\Lambda(u_2) = \Lambda_{c2} + \Lambda_{e1}(u_2) + \Lambda_{e2}(u_2) = -2{,}13.$$

Итеративный декодер мог бы принять решение в этой точке процедуры, основанное на знаках апостериорных LLR, и выдать (правильные) оценки $u_1 = 0$ и $u_2 = 1$. Увеличение числа итераций в данном примере улучшает только мягкие выходы, ассоциированные с информационными битами.

Вторая итерация

Строчный декодер:

$$\Lambda_{e1}(u_1) = \Lambda(r_{11}) + \Lambda_{e2}(u_1) = +2{,}5 - 0{,}42 = +1{,}08;$$
$$\Lambda_{e1}(u_2) = \Lambda(r_{12}) + \Lambda_{e2}(u_2) = +1{,}0 + 0{,}38 = +1{,}38.$$

+0,58	+2,5
−2,12	+1,0
+0,5	X

Декодер столбцов:

$$\Lambda_{e2}(u_1) = \Lambda(u_2 + r_{21}) = \log\left[\frac{1+e^{-1,62}}{e^{-2,12}+e^{+0,5}}\right] = -0{,}17;$$

$$\Lambda_{e2}(u_2) = \Lambda(u_1 + r_{21}) = \log\left[\frac{1+e^{-1,08}}{e^{+0,58}+e^{+0,5}}\right] = +0{,}16.$$

Значения апостериорных LLR:

$$\Lambda(u_1) = \Lambda_{c1} + \Lambda_{e1}(u_1) + \Lambda_{e2}(u_1) = +0{,}41;$$
$$\Lambda(u_2) = \Lambda_{c2} + \Lambda_{e1}(u_2) + \Lambda_{e2}(u_2) = -1{,}96.$$

После второй итерации решение, выбранное по знакам величин LLR, совпадает с прежним (правильным) жестким решением. Более того, заметим, что мягкие выходы изменились из-за влияния внешних LLR, вычисленных на первой итерации. В частности, надежность (абсолютное значение LLR) первого бита уменьшилась с $|\Lambda(u_1)| = +1{,}58$ до $|\Lambda(u_1)| = +0{,}41$.

Эффективность

На Рисунке 78 видно, что увеличением числа итераций вероятность ошибки уменьшается. Обычно для схем турбо кодирования фактом является монотонное снижение скорости роста выигрыша от кодирования с увеличением числа итераций. Например, из Рисунка 78 видно, что увеличение числа итераций от 2 до 6 уменьшает требуемое SNR на 1,7дБ, тогда как дополнительные итерации с 6 до 18 увеличивают выигрыш от кодирования только на 0,3дБ.

С момента открытия турбо кодов достигнут некоторый прогресс в понимании поведения вероятности ошибки этих кодов. В процессе написания этой книги, по-видимому, появилось единое мнение среди исследователей о причинах, по которым турбо коды имеют столь замечательные характеристики:

- Турбо коды имеют распределение весов, которое на больших интервалах совпадает с распределением для случайных кодов.
- Рекурсивное сверточное кодирование с правильно выбранным перемежением преобразует большую часть сообщений малого веса в кодовые слова большого веса.
- Систематические кодеры позволяют эффективно совмещать итеративную технику с MAP декодированием. Оценки (начальные) информационных символов доступны прямо из канала связи.

Кроме того, совсем недавно (конец 2001) появилось множество интересных полу-аналитических методов для изучения сходимости итеративных алгоритмов декодирования, основанных на эволюции плотности (вероятности) [RU], на Гауссовой аппроксимации [EH] и взаимной информации [tBr2] или на передаточных характеристиках SNR [DDP] (после каждой итерации).

Искусство перемешивания

Устройство перемешивания – перемежитель (или интерливер) является самой ответственной частью турбо кода для достижения высокой эффективности при итеративном декодировании. В конструкции турбо кода перемежитель служит для достижения трех целей: (1) построение очень длинных кодов с распределением весов, близким к распределению для случайных кодов, (2) обеспечение, на сколько это возможно, декорреляции входных LLR декодера SISO (с мягкими входом и выходом) в итеративной процедуре, (3) правильное терминирование кодовой решетки (в заданном состоянии) после передачи коротких или средних по длине сообщений с тем, чтобы избежать краевых эффектов, которые могли бы увеличить долю путей малого веса на кодовой решетке компонентных кодов. Чтобы подчеркнуть особую роль интерливера, рассмотрим блоки данных (фреймы) сравнительно небольшой длины, например, до одной тысячи символов. Известно огромное количество публикаций, посвященных конструированию перемежителей для турбо кодов. В этом разделе мы приведем краткое описание основных типов перемежителей со ссылками на литературу.

В 1970 были введены несколько типов оптимальных интерливеров [Ram]. В частности, (n_1, n_2) перемежитель был определен как устройство, которое «переупорядочивает последовательность таким образом, чтобы никакие последовательности смежных n_2 символов не содержали бы любой набор символов, разнесенных менее чем на n_1 символов в исходном порядке».

Обозначим $\ldots a_{l_1}, a_{l_2}, a_{l_3}, \ldots$ выходную последовательность (n_1, n_2) перемежителя, где $\ell_1, \ell_2, \ldots$ индексы (позиции) этих символов во входной последовательности. Тогда предыдущее определение сводится к следующим условиям:

$$\left|\ell_i - \ell_j\right| \geq n_1,$$

всегда, когда

$$|i-j| < n_1$$

Можно показать, что деинтерливер для (n_1, n_2) интеливера является, в свою очередь, (n_1, n_2) интерливером. Задержка интерливера и деинтерливера в совокупности равна задержке, пре-

дусмотренной для выполнения этих операций. Другим важным параметром устройства перемежения является необходимый объем памяти. Рэмси ввел четыре типа (n_1, n_2) перемежителей, которые оптимальны по задержке и используемой памяти. Эти устройства известны как *перемежители* (*интерливеры*) *Рэмси*. В тоже время Форни [For5] предложил устройство такой же конструкции, как и (n_1, n_2) интерливер Рэмси, известное под названием *сверточный интерливер*. Сверточные перемежители уже обсуждались в Разделе 6.2.3. Они были применены для построения хороших схем турбо кодирования [HaW].

Сейчас известно несколько новых подходов к анализу и построению интерливеров. Один из них основан на *случайном перемежении* и его *свойстве рассеивания*. Эта идея была впервые предложена в работе [DP] с простым алгоритмом построения *S-случайных перемежителей*, который состоит в следующем: генерировать случайные числа в интервале [1, N] и использовать ограничения на *расстояние перемежения*. Это ограничение, очевидно эквивалентно определению (S_2, S_1+1) интерливера Рэмси (это было замечено в работе [Vuc], стр. 211-213). Далее были введены дополнительные ограничения (основанные на эмпирической корреляции между последовательными значениями внешних LLR) для того, чтобы управлять выбором перестановок позиций символов на выходе перемежителя [HEM, SSSN]. Другой подход к построению интерливеров состоит в том, что рассматривается структура турбо кодера в целом. Для этого вычисляются его минимальное расстояние и коэффициент ошибок (число кодовых последовательностей на минимальном расстоянии) [GPB, BH]. Этот путь дает точную оценку *зоны насыщения* (error floor) функции вероятности ошибки для средних и высоких значений SNR. Другие важные предложения по построению коротких интерливеров для турбо кодов имеются в работах [TC, BP].

8.2.2. Последовательная схема

Составные коды последовательного типа были введены в работе [BDMP1]. Структурная схема последовательного составного

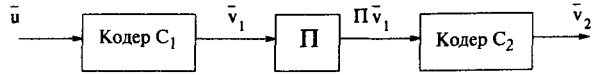


Рис. 83. Структурная схема последовательного составного кодера

кодера для двух линейных кодов показана на Рисунке 83. Опираясь на результаты Раздела 6.2.3 и на детальное сравнение схем на Рисунках 83 и 55, последовательная схема из двух линейных кодов легко интерпретируется как *каскадный код* (произведение кодов). Заметим, что в отличие от турбо кода в последовательной конструкции не удаляются какие-либо проверочные символы.

Кодер последовательного составного кода имеет ту же структуру, что и каскадный код. Сохраняя обозначения оригинала [BDMP1], имеем внешний $(n_1=p,\ k_1=k,\ d_1)$ код C_1 скорости $R_1=k/p$ и внутренний $(n_2=n,\ k_2=n_1,\ d_2)$ код C_2 скорости $R_2=p/n$. Эти коды связаны таким же образом, как и в произведении блоковых кодов с включением блокового интерливера длины $L=mp$. Как и прежде, m кодовых слов длины p записываются в перемежитель, из которого считываются в другом порядке, заданном матрицей перестановок П. Последовательность L бит на выходе интерливера посылается блоками длины p в кодер внутреннего кода C_2. Скорость составного $(N,\ K,\ d_1d_2)$ кода C_{SC} равна $R_{SC}=k/n$, где $N=nm$ и $K=km$.

Порождающая матрица кода C_{SC} может быть записана как произведение порождающей матрицы кода C_1, матрицы перестановок П размера $k_2\times n_1$ и порождающей матрицы кода C_2 следующим образом

$$\mathbf{G}_{SC}=\begin{pmatrix}\mathbf{G}_1 & & \\ & \mathbf{G}_1 & \\ & \mathbf{0} & \\ & & \mathbf{G}_1'\end{pmatrix}\Pi\begin{pmatrix}\mathbf{G}_2 & & \\ & \mathbf{G}_2 & \\ & \mathbf{0} & \\ & & \mathbf{G}_2\end{pmatrix}=\left(\mathbf{G}_1'\,|\,\Pi\,|\,\mathbf{G}_2'\right) \tag{8.12}$$

Здесь $\mathbf{G}_i'$ есть порождающая матрица кода C_i, $i = 1, 2$. Число появлений матрицы $\mathbf{G}_1$ в блочно диагональной матрице $\mathbf{G}_1'$ равно k_2, а число появлений $\mathbf{G}_2$ в $\mathbf{G}_2'$ равно n_1. Все прочие элементы матриц $\mathbf{G}_i'$ равны нулю.

Пример 91. Пусть C_1 и C_2 коды из примера 89, т.е. двоичный (2, 1, 2) код-повторение и (3, 2, 2) код с одной проверкой, соответственно. Тогда последовательный составной код C_{SC} или произведение кодов C_1 и C_2 является двоичным линейным блоковым (6, 2, 4) кодом. Заметим, что минимальное расстояние кода C_{SC} больше, чем у кода C_{PC} в Примере 89. Порождающие матрицы имеют вид:

$$\mathbf{G}_1 = (1\ \ 1),\quad \mathbf{G}_2 = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

В предположении, что применяется обычное произведение кодов, матрица перестановок, ассоциированная с перестановками по строкам и по столбцам, равна

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

Таким образом, порождающая матрица кода C_{SC} имеет вид

$$\mathbf{G}_{SC} = \begin{pmatrix} (1\ 1) & \mathbf{0}_{12} \\ \mathbf{0}_{12} & (1\ 1) \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} & \mathbf{0}_{23} \\ \mathbf{0}_{23} & \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} & \mathbf{0}_{23} \\ \mathbf{0}_{23} & \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \end{pmatrix}$$

$$= \begin{pmatrix} (1\ 0\ 1) & (1\ 0\ 1) \\ (0\ 1\ 1) & (0\ 1\ 1) \end{pmatrix}$$

$$= \left(\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \right),$$

где $\mathbf{0}_{ij}$ нулевая матрица размера $i \times j$.

Этот результат легко проверить, заметив, что последнее равенство содержит порождающую матрицу (3, 2, 2) кода *два раза*, соответственно с (2, 1, 2) кодом-повторением. Интересно отметить также, что последний код является минимальным представителем семейства кодов *с накоплением и повторением* [DJM].

Заметим, что главное различие между последовательной схемой и произведением кодов, обсуждавшимся в Разделе 6.2.3, состоит в том, что там интерливер выполняет либо перестановки строк и перестановки столбцов, либо циклические перестановки (если компонентные коды циклические). В данном случае, как и в случае турбо кодов, эффективность последовательной схемы тем выше, чем ближе к «случайным» перестановки, реализуемые интерливером

В отличие от турбо кодов последовательные составные коды не обладают свойством насыщения функции «усиление интерливера» (т.е. не имеют зоны насыщения). Используя вероятностные рассуждения для интерливеров длины N, можно показать, что функция вероятности ошибки декодирования для произведения кодов содержит множитель $\exp(\ln(N)\lfloor -(d_{Of}+1)/2 \rfloor)$, где d_{Of} есть минимальное расстояние внешнего кода, тогда как для параллельной схемы имеется только множитель N^{-1}[BDMP1].

Вследствие этого произведение кодов превосходит турбо коды в области относительно высоких значений SNR, где проявляется эффект насыщения (error floor). Однако при низких отношениях сигнал-шум имеет значение лучшее распределение весов кодовых слов у турбо кодов [PSC], что приводит к лучшим характеристикам, чем у произведения кодов.

Следующие правила были выработаны для выбора компонентных сверточных кодов в последовательных схемах составного кодирования[6].

[6] Следует заметить, что эти критерии были выработаны на основе *границы объединения* для вероятности ошибки на бит

- Внутренний код должен быть рекурсивным систематическим сверточным (RSC) кодом.
- Внешний код должен иметь большое, насколько это возможно, минимальное расстояние.
- Внешний код может быть нерекурсивным и несистематическим сверточным кодом.

Последний критерий необходим для того, чтобы минимизировать число кодовых слов минимального веса (эта характеристика известна как *экспонента ошибки* (*error exponent*)) а также вес входных последовательностей, порождающих кодовые слова минимального веса.

Итеративное декодирование последовательных схем составного кодирования

Возвращаясь к Рисунку 84, отметим, что не систематичность внешнего сверточного кода означает невозможность получения внешней информации из SISO декодера [BDMP1]. Следовательно, в этом случае алгоритм итеративного декодирования отличается от того, который использовался для турбо кодов. Если для турбо кодов обновлялись LLR только информационных символов, то здесь неизбежно обновляются LLR как информационных, так и кодовых символов. Работа мягкого декодера внутреннего кода не изменяется. Однако, для внешнего мягкого декодера априорные LLR всегда устанавливаются нулевыми. Значения LLR кодовых символов вычисляются и пересылаются мягкому декодеру внутреннего кода после операции интерливинга как априорные LLR для следующей итерации. Как и для турбо кодов в этом случае существует max-log-MAP версия итеративного алгоритма. Кроме того возможна модификация алгоритма SOVA в качестве аппроксимации MAP декодирования в итеративной процедуре декодирования кодов-произведений [FV].

8.2.3. Произведение блоковых кодов

Хотя турбо коды и последовательные схемы кодирования, повидимому, доминируют среди кодовых конструкций, допуска-

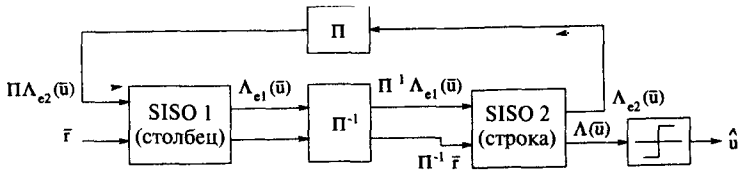


Рис. 84. Блок-схема интерактивного декодера последовательного составного кода.

ющих итеративное декодирование, блоковые коды произведения тоже можно декодировать итеративно, как это следует из предыдущего обсуждения. В 1993, на той же самой конференции, где Берроу с коллегами сообщили об открытии турбо кодов, была представлена работа об итеративном декодировании произведений кодов и каскадных кодов [LYHH]. В частности, был рассмотрен трехмерный (4096, 1331, 64) код, построенный из расширенных (16, 11, 4) кодов Хемминга, с итеративным MAP декодированием и продемонстрирована его впечатляющая эффективность.

Одним годом позже в работе [PGPJ] (см. также [Pyn]) было введено близкое к оптимальному турбо подобное декодирование кодов произведений. Здесь рассматривалось произведение относительно высокоскоростных расширенных кодов БЧХ с исправлением одной и двух ошибок. В предложенной итеративной схеме декодирования компонентные коды используют алгоритм Чейза второго типа[7] (алгоритм Чейза тип-II). После того, как найден список кодовых слов, для них вычисляются значения LLR. Эта итеративная процедура и ее улучшения рассматриваются в следующем разделе.

Итеративное декодирование с использованием алгоритма Чейза

В работах [PGPJ, Pyn] для генерации списка кандидатов (кодовых слов, близких к принятому слову) применяется алгоритм Чейза тип-II. Внешние LLR вычисляются на основе двух ближайших кандидатов из списка. Если найдено только одно кодовое слово (список объема 1), то аппроксимированные

[7] Алгоритмы Чейза рассматривались в Разделе 7.4.

значения LLR выдаются на выход декодера. Пусть C двоичный линейный блоковый (N, K, d) код, способный исправить любую комбинацию $t = \lfloor (d-1)/2 \rfloor$ или меньше случайных ошибок. Пусть $\mathbf{r} = (r_1, r_2, \ldots r_N)$ принятое слово на выходе канала, $r_i = (-1)^{v_i} + w_i$, где $\mathbf{v} \in C$ и w_i Гауссова случайная величина с нулевым средним и дисперсией $N_0/2$. Структурная схема алгоритма Чейза тип-II обработки принятого слова $\mathbf{r}$ показана на Рисунке 76.

В результате выполнения алгоритма Чейза тип-II возможны следующие три события:

1. Найдены два или более кодовых слова, $\{\hat{\mathbf{v}}_1, \ldots, \hat{\mathbf{v}}_l\}$, $l \geq 2$;
2. Найдено одно кодовое слово $\hat{\mathbf{v}}_1$;
3. Нет ни одного кодового слова.

В последнем случае декодер может поднять флаг о неисправимой ошибке и выдать принятую последовательность в том виде, в каком она есть. Альтернативным вариантом является увеличение числа проверяемых комбинаций ошибок до тех пор, пока не найдется хотя бы одно слово, как предложено в работе [Pyn].

Обозначим $\chi_j(\ell)$ множество модулированных кодовых слов кода C, у которых j-ая компонента $x_j = \ell$, $\ell \in \{-1, +1\}$, для $1 \leq j \leq N$. Пусть $\mathbf{x}_j(\ell)$ и $\mathbf{y}_j(\ell)$ из множества $\chi_j(\ell)$ представляют ближайшее (по Евклидову расстоянию) к принятому слову $\mathbf{r}$ и следующее за ним модулированные кодовые слова, соответственно.

С помощью log-max аппроксимации, $\log(e^a + e^b) \approx \max(a, b)$, значение LLR (8.2) для j-го символа может быть записано в следующем виде [Pyn, FL5]

$$\begin{aligned} \Lambda(u_1) &= \Lambda_{c1} + \Lambda_{e1}(u_1) + \Lambda_{e2}(u_1) = +0{,}41; \\ \Lambda(u_2) &= \Lambda_{c2} + \Lambda_{e1}(u_2) + \Lambda_{e2}(u_2) = -1{,}96. \end{aligned} \tag{8.13}$$

Отсюда после нормализации и замены обозначений, $x_m = x_m(+1)$ и $y_m = y_m(-1)$, находим, что *мягкий выход* равен

$$\Lambda'(u_j) = r_j + \sum_{\substack{x_m \neq y_m \\ m=1, m \neq j}}^{N} r_m x_m = x_j \sum_{\substack{x_m \neq y_m \\ m=1}}^{N} r_m x_m \tag{8.14}$$

Сумма

$$w_j = x_j \sum_{\substack{x_m \neq y_m \\ m=1, m \neq j}}^{N} r_m x_m \tag{8.15}$$

интерпретируется как *поправочный член* мягкого входа r_j, который зависит от двух модулированных кодовых слов, ближайших к принятому слову $\mathbf{r}$, и играет ту же роль, что и внешнее LLR. Для каждой позиции j, $1 \leq j \leq N$, величина w_j выдается следующему декодеру как значение внешнего LLR с некоторым масштабирующим множителем α_c, а именно, величина

$$r_j' = r_j + \alpha_c w_j \tag{8.16}$$

вычисляется как мягкий вход для следующего декодера. Множитель α_c вводится для того, чтобы компенсировать различие дисперсий Гауссовых случайных величин r_i и r_j'. Структурная схема процедуры вычисления мягких входов и внешних LLR показана на Рисунке 85.

Если для некоторой j-ой позиции, $1 \leq j \leq N$, алгоритм Чейза не находит ни одной пары последовательностей $\boldsymbol{x}_j(+1)$ и $\boldsymbol{y}_j(-1)$, то предлагается использовать [Pyn] следующее значение LLR для символа u_j

$$\Lambda'(u_j) = \beta_c x_j \tag{8.17}$$

где β_c является *корректирующим множителем* для компенсации ошибки аппроксимации внешней информации. С помощью моделирования этот множитель оценивается как

$$\beta_c = \left| \log\left(\frac{\Pr\{v_j \text{ верно}\}}{\Pr\{v_j \text{ неверно}\}} \right) \right| \tag{8.18}$$

т.е. как значение LLR для экспериментальной оценки вероятности ошибки на символ. В работах [PP, MT] показано, каким образом можно *адаптивно* вычислять корректирующие множители α и β на основе статистик обрабатываемых кодовых слов. Следует заметить, что алгоритм с мягким выходом, предложенный в [FL5] и рассмотренный в Разделе 7.5, также может

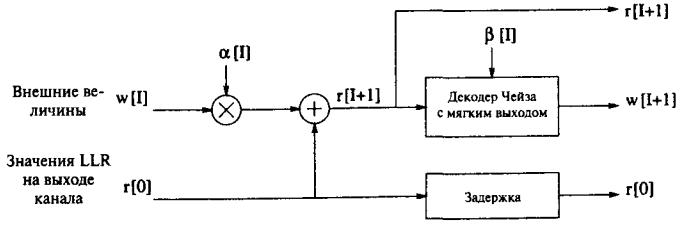


Рис. 85. Блок схема декодера Чейза с мягким выходом.

быть применен. В этом случае тоже необходимы адаптивные веса для того, чтобы масштабировать значения внешних LLR.

Суммируя сказанное выше, получаем следующее формализованное описание алгоритма итеративного декодирования с мягким выходом, основанного на построении списка кодовых слов с помощью алгоритма Чейза тип-II.

Шаг 0: Начальные установки

Установить счетчик итераций $I = 0$. Установить $\mathbf{r}[0] = \mathbf{r}$ (принятая из канала последовательность).

Шаг 1: Мягкие входы

Для $j = 1, 2, \ldots, N$,

$$r_j[I+1] = r_j[0] + \alpha_c[I] w_j[I].$$

Шаг 2: Алгоритм Чейза

Выполнить алгоритм Чейза тип-II для последовательности $\mathbf{r}[I+1]$. Определить число найденных слов n_c (длину списка). Найти, если это возможно, два ближайших к принятой последовательности модулированных кодовых слова **x** и **y**.

Шаг 3: Внешняя информация

Для $j = 1, 2, \ldots, N$,

• Если $n_c \geq 2$,

$$w_j[I+1] = x_j \sum_{\substack{x_m \neq y_m \\ m=1, m \neq j}}^{N} r[I+1]_m x_m$$

• Иначе

$$w_j[I+1] = \beta_c[I] x_j$$

Шаг 4: Мягкий выход

Увеличить номер итерации, $I = I + 1$. Если $I < I_{max}$ (максимальное число итераций) или не выполнилось условие остановки, то вернуться на Шаг 1.

Иначе вычислить мягкий выход.

Для $j = 1, 2, \ldots, N$,

$$\Lambda(u_j) = \alpha_c[I] w_j[I] + r_j[0] \qquad (8.19)$$

Стоп.

В случае двоичной фазовой модуляции величины α_c и β_c были вычислены для четырех итераций (всего восемь величин для первого и второго декодера) [Pyn].

$$\alpha_c = (0{,}0\ \ 0{,}2\ \ 0{,}3\ \ 0{,}5\ \ 0{,}7\ \ 0{,}9\ \ 1{,}0\ \ 1{,}0),$$
$$\beta_c = (0{,}2\ \ 0{,}4\ \ 0{,}6\ \ 0{,}8\ \ 1{,}0\ \ 1{,}0\ \ 1{,}0\ \ 1{,}0).$$

8.3. Коды с низкой плотностью проверок на четность

В 1962 Галлагер в работе [Gal] ввел класс линейных кодов, известный теперь как коды с низкой плотностью проверок (LPDC), и представил два итеративных вероятностных алгоритма декодирования. Позднее Таннер [Tan] обобщил вероятностный алгоритм Галлагера на случай, когда проверки определяются подкодами, а не просто проверочными уравнениями для проверок на четность. Перед этим было показано, что LPDC коды имеют минимальное расстояние, растущее линейно с длиной кода, и что все ошибки вплоть до минимального расстояния могут быть исправлены при почти линейной сложности алгоритма декодирования [ZP].

В работах [MN, Mac] показано, что LDPC коды могут быть также близки к пределу Шеннона, как и турбо коды. Позднее в [RSU] было показано, что нерегулярные LDPC коды могут превосходить турбо коды при примерно одинаковых длинах

и скоростях, когда длина блока достаточно велика. В момент написания книги лучший известный двоичный код скорости 1/2 с длиной блока 10 000 000 есть LDPC код, достигший рекордные 0,0045дБ от предела Шеннона для случая передачи двоичных сигналов по каналу с АБГШ [CFRU].

Регулярный LDPC код является линейным (N, K) кодом, проверочная матрица которого $\mathbf{H}$ имеет Хеммингов вес столбца и строки равный J и K, соответственно, причем обе величины много меньше длины кода N. В результате LDPC код имеет очень сильно разреженную матрицу. Если вес Хемминга столбцов и строк $\mathbf{H}$ выбирается в соответствии с некоторым неравномерным распределением вероятностей, то получаются *нерегулярные* LDPC коды [RSU]. Маккей предложил несколько методов построения LDPC матриц с помощью компьютерного поиска [Mac].

8.3.1. Графы Таннера

Для любого линейного (N, K) кода существует двудольный граф, инцидентный матрице $\mathbf{H}$. Этот граф известен как *граф Таннера* [Tan, SS], названный так по имени его открывателя. С помощью введения вершин-состояний графы Таннера были обобщены до факторграфов [For7, KFL]. Вершины графа Таннера ассоциируются с переменными двух типов, каждой из которых приписываются значения LLR.

Граф Таннера линейного (N, K) кода C имеет N *кодовых вершин*, т.е. вершин, соответствующих переменным x_ℓ – кодовым символам, и, по меньшей мере, N-K *проверочных вершин*, z_m, соответствующих проверочным уравнениям. Для регулярного графа степень (т.е. число входящих и исходящих ребер) каждой кодовой вершины равна J, а степень проверочных вершин равна K.

Пример 92. Чтобы проиллюстрировать Граф Таннера на каком-нибудь коде, рассмотрим (7, 4, 3) код Хемминга. Его проверочная матрица имеет следующий вид[8]

[8] См. Пример 13.

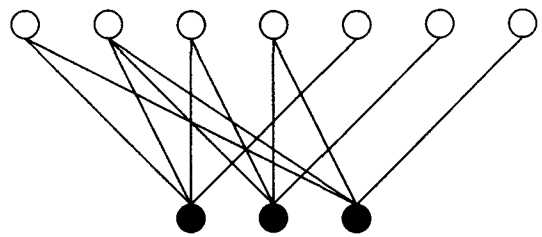


Рис. 86. Граф Таннера (7, 4, 3) кода Хемминга.

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

Соответствующий ей граф Таннера показан на Рисунке 86. Соединение кодовых вершин графа с проверочными определяется строками проверочной матрицы.

Первая строка дает проверочное уравнение $v_1 + v_2 + v_3 + v_5 = 0$. Как отмечалось выше, переменные x_ℓ и z_m приписаны каждому кодовому символу и каждому проверочному уравнению, соответственно. Следовательно, имеем следующие проверочные уравнения

$$z_1 = x_1 + x_2 + x_3 + x_5,$$
$$z_2 = x_2 + x_3 + x_4 + x_6,$$
$$z_3 = x_1 + x_2 + x_4 + x_7.$$

В соответствии с первым уравнением кодовые вершины x_1, x_2, x_3 и x_5 соединены с проверочной вершиной z_1. Аналогично, столбцы проверочной матрицы указывают, в каких проверочных уравнениях участвует данный кодовый символ. Так, левый столбец $\mathbf{H}$, $(1\ 0\ 1)^{\mathrm{T}}$ указывает, что символ (вершина) x_1 связан с проверочными вершинами z_1 и z_3.

Пример 93. В работе Галлагера [Gal] приводится следующая проверочная матрица LDPC кода (20, 5) с параметрами $J = 3$ и $K = 4$

$$\mathbf{H} = \begin{pmatrix}
1111 & 0000 & 0000 & 0000 & 0000 \\
0000 & 1111 & 0000 & 0000 & 0000 \\
0000 & 0000 & 1111 & 0000 & 0000 \\
0000 & 0000 & 0000 & 1111 & 0000 \\
0000 & 0000 & 0000 & 0000 & 1111 \\
1000 & 1000 & 1000 & 1000 & 0000 \\
0100 & 0100 & 0100 & 0000 & 1000 \\
0010 & 0010 & 0000 & 0100 & 0100 \\
0001 & 0000 & 0010 & 0010 & 0010 \\
0000 & 0001 & 0001 & 0001 & 0001 \\
1000 & 0100 & 0001 & 0000 & 0100 \\
0100 & 0010 & 0010 & 0001 & 0000 \\
0010 & 0001 & 0000 & 1000 & 0010 \\
0001 & 0000 & 1000 & 0100 & 1000 \\
0001 & 0000 & 1000 & 0100 & 1000
\end{pmatrix}$$

Граф Таннера для этой матрицы показан на Рисунке 87. Отметим, что каждая кодовая вершина соединена точно с тремя проверочными вершинами. Другими словами, *степень* кодовых вершин равна $J = 3$. Аналогично находим, что степень проверочных вершин равна $K = 4$.

Графы Таннера могут быть использованы для оценки кодовых слов LDPC кода C с помощью итеративного вероятностного алгоритма декодирования с жестким или мягким решениями. Ниже представлены оба итеративных алгоритма, предложенные Галлагером.

8.3.2. Итеративное декодирование с жестким решением: алгоритм с перевертыванием бита

В своей работе 1962 года Галлагер привел следующий алгоритм [Gal][9].

[9] Этот алгоритм известен также как А-алгоритм Галлагера

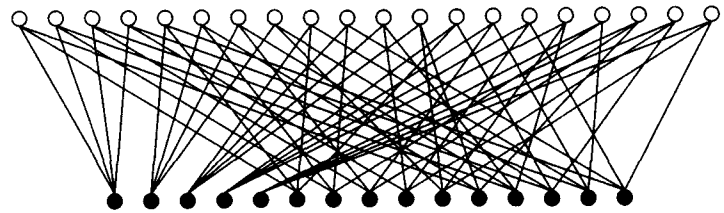


Рис. 87. Граф Таннера для кода Галлагера (20, 5).

Декодер вычисляет все проверки и затем изменяет бит на любой из позиций, содержащихся в большем некоторого заданного порога числе неудовлетворенных проверочных уравнений. Проверки перевычисляются для новых значений символов и процесс продолжается, пока не удовлетворятся все проверки.

Входом этого алгоритма является вектор $\mathbf{r}_h = (\mathrm{sgn}(r_1), \mathrm{sgn}(r_2), \ldots, \mathrm{sgn}(r_N))$, где

$$r_i = (-1)^{v_i} + w_i, \mathbf{v} \in C,$$

w_i случайная Гауссова величина с нулевым средним и дисперсией $N_0/2$. Далее, $\mathrm{sgn}(x) = 1$, если $x < 0$, и $\mathrm{sgn}(x) = 0$ в противном случае.

Обозначим T порог, при превышении которого символ инвертируется. Таким образом, если число неудовлетворенных проверок среди тех, в которые входит символ v_i, превысит этот порог, то $v_i = v_i \oplus 1$.

На Рисунках 88 и 89 показаны результаты моделирования для (20, 5) кода Галлагера C_G и (7, 4, 3) кода Хемминга C_H, соответственно, в двоичном канале с АБГШ. Максимальное число итераций было установлено 4 для C_G и 2 для C_H. В обоих случаях значения порога устанавливались равными $T = 1, 2, 3$. Для кода Хемминга, Рисунок 89, показаны также результаты моделирования при исправлении одной ошибки (HD), для мягкого декодирования по максимуму правдоподобия (SD) и граница объединения (1.35).

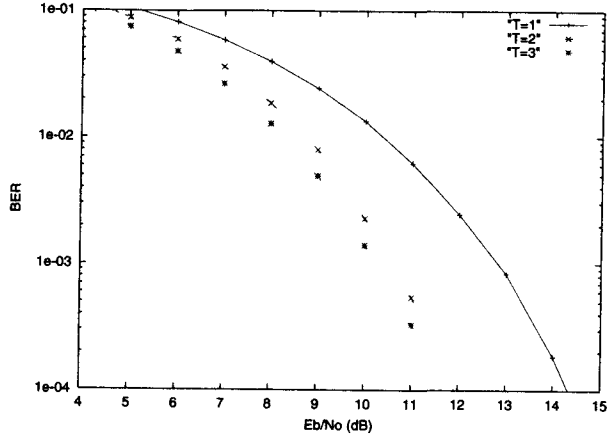


Рис. 88. Характеристики кода Галлагера (20, 5) для алгоритма с перевертыванием бита. Четыре итерации с различными порогами.

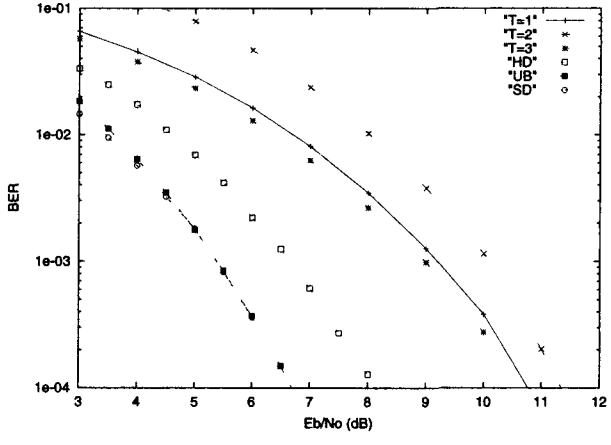


Рис. 89. Характеристики кода Хемминга (7, 4, 3) для алгоритма с перевертыванием бита. Две итерации с различными порогами.

На Рисунках 90 – 92 сравнивается эффективность алгоритмов Берлекэмпа-Мэсси (BM) Галлагера с перевертыванием бита (BF) для кодов БЧХ (31, 26, 3), (31, 21, 5) и (31, 16, 7). Очевидно, что эффективность алгоритма BF несколько ниже, чем

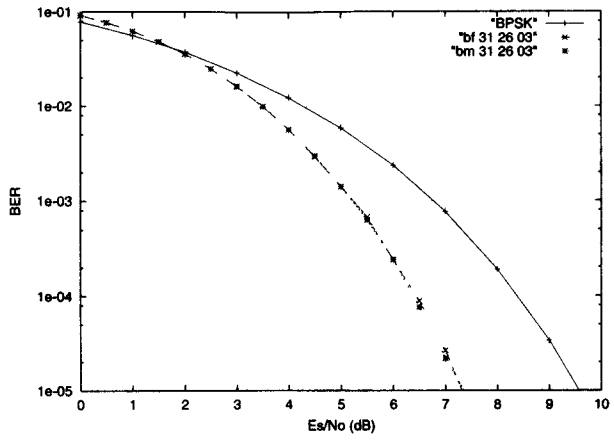


Рис. 90. Эффективность алгоритмов Берлекэмпа-Мэсси и BF-алгоритма Галлагера для двоичного (31, 26, 3) БЧХ кода.

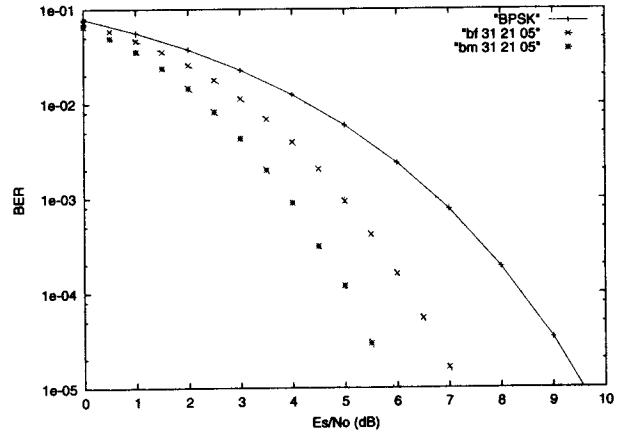


Рис. 91. Эффективность алгоритмов Берлекэмпа-Мэсси и BF-алгоритма Галлагера для двоичного (31, 21, 5) БЧХ кода.

у алгоритма BM, несмотря на большую корректирующую способность LDPC кода. С другой стороны, BF алгоритм Галлагера использует только простые элементы: сложение по модулю 2 и сравнение, вместо арифметического процессора для конеч-

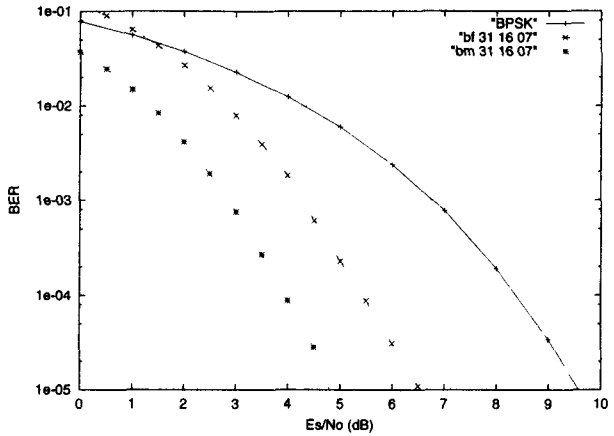


Рис. 92. Эффективность алгоритмов Берлекэмпа-Мэсси и BF-алгоритма Галлагера для двоичного (31, 16, 7) БЧХ кода.

ного поля GF(2^m) в алгоритме BM. Следовательно, можно предположить, что для некоторых высокоскоростных кодов, таких как, например, БЧХ коды с исправлением одной или двух ошибок, алгоритм Галлагера с перевертыванием бита может конкурировать с алгоритмом BM.

Дополнительное свойство алгоритма BF, как и итеративной вероятностной процедуры декодирования, состоит в том, что сложность декодирования зависит только от степени вершин графа Таннера. Другими словами, для фиксированных параметров J и K сложность декодирования растет линейно с длиной кода.

8.3.3. Итеративное вероятностное декодирование: распространение доверия

В этом разделе представлен алгоритм декодирования с *итеративным распространением доверия* (IBP, *iterative belief-propagation*). Этот алгоритм известен также как алгоритм Перла [Prl] и как алгоритм *сумма-произведений* [Wib, KFL, For7]. Уиберг

в своей диссертации показал, что двухпроходные алгоритмы, такие как турбо декодирование и алгоритм Галлагера (рассматривается ниже), являются частными случаями алгоритма с суммированием произведений. Более того, как было показано в [MMC], итеративные алгоритмы декодирования турбо кодов и произведений кодов являются частными случаями IBP декодирования. Приводимое ниже объяснение очень близко к работе [Prl, Mac] для двоичных LDPC кодов. Введем следующие обозначения. Пусть h_{ij} элемент i-ой строки и j-го столбца матрицы **H**. Обозначим

$$\zeta(m)=\{\ell: h_{m,\ell}=1\} \tag{8.20}$$

множество *кодовых позиций*, участвующих в m-ом проверочном уравнении, и

$$\mu(\ell)=\{m: h_{m,\ell}=1\} \tag{8.21}$$

множество *проверочных позиций*, в которых участвует кодовая позиция ℓ.

Алгоритм итеративно вычисляет два типа условных вероятностей:

- $q^x{}_{m\ell}$ – вероятность того, что ℓ-ый бит вектора **v** имеет значение x по информации, полученной от всех проверочных вершин (графа Таннера) кроме вершины m;
- $r^x{}_{m\ell}$ – вероятность[10] того, что уравнение, соответствующее проверочной вершине m, удовлетворяется, если значение ℓ-го бита равно x, а остальные биты независимы с вероятностями $q_{m\ell'}$, $\ell' \in \zeta(m) \setminus \ell$.

Как отмечалось в работах [Prl, Mac] алгоритм декодирования, соответствующий IBP, вычисляет точные апостериорные вероятности после некоторого количества итераций, *если граф Таннера для данного кода не содержит циклов*. Ниже предполагается передача двоичных сигналов по каналу с АБГШ. Как и раньше, модулированные символы $m(v_i)$ передаются по Гаус-

[10] Замечание. это плохое обозначение, так как r_i использовалось для i-ой компоненты принятого вектора. Тем не менее, оно позволяет сохранить обозначения, принятые в большей части литературы по данной теме.

сову каналу и принимаются в виде $r_i = m(v_i) + w_i$, где w_i Гауссова случайная величина с нулевым средним и дисперсией $N_0/2$, $1 \le i \le N$.

Начальные установки

Для $\ell \in \{1, 2, \ldots, N\}$ установить априорные вероятности кодовых вершин (графа Таннера) равными

$$p_\ell^1 = \frac{1}{1+\exp\left(r_\ell \frac{4}{N_0}\right)} \tag{8.22}$$

$p_\ell^0 = 1 - p_\ell^1$. Для каждой пары (l, m) такой, что $h_{m\ell} = 1$,

$$q_{m,\ell}^0 = p_\ell^0, \quad q_{m,\ell}^1 = p_\ell^1 \tag{8.23}$$

Обработка сообщения

Шаг 1. Снизу вверх (по горизонтали):

Для каждой пары ℓ, m вычислить

$$\delta r_{m,\ell} = \prod_{\ell' \in \xi(m)\setminus\ell} (q_{m,\ell'}^0 - q_{m,\ell'}^1) \tag{8.24}$$

и

$$r_{m,\ell}^0 = (1+\delta r_{m,\ell})/2, \quad r_{m,\ell}^1 = (1-\delta r_{m,\ell})/2 \tag{8.25}$$

Шаг 2. Сверху вниз (по вертикали):

Для каждой пары ℓ, m вычислить

$$q_{m,\ell}^0 = p_\ell^0 \prod_{m' \in \mu(\ell)\setminus m} r_{m',\ell}^0, \quad q_{m,\ell}^1 = p_\ell^1 \prod_{m' \in \mu(\ell)\setminus m} r_{m',\ell}^1 \tag{8.26}$$

и нормировать с множителем $\alpha = 1/(q^0{}_{m,\ell} + q^1{}_{m,\ell})$,

$$q_{m,\ell}^0 = \alpha q_{m,\ell}^0, \quad q_{m,\ell}^1 = \alpha q_{m,\ell}^1 \tag{8.27}$$

Для каждого ℓ вычислить апостериорные вероятности

$$q_\ell^0 = p_\ell^0 \prod_{m \in \mu(\ell)} r_{m,\ell}^0, \quad q_\ell^1 = p_\ell^1 \prod_{m \in \mu(\ell)} r_{m,\ell}^1 \tag{8.28}$$

и нормировать их с множителем $\alpha = 1/(q_\ell^0 + q_\ell^1)$,

$$q_\ell^0 = \alpha q_\ell^0, \quad q_\ell^1 = \alpha q_\ell^1 \tag{8.29}$$

Декодирование и формирование мягких выходов

Для i = 1, 2, ..., N вычислить

$$v_i = \operatorname{sgn}(q_i^0) \tag{8.30}$$

Если $\hat{\mathbf{v}}\mathbf{H} = \mathbf{0}$, то оценкой кодового слова и мягкими выходами являются

$$\Lambda(v_i) = \log(q_i^1) - \log(q_i^0), \quad 1 \le i \le N \tag{8.31}$$

и процесс завершается.

Иначе, процедура возвращается к Шагу 2. Если число итераций превышает заранее установленный порог, то фиксируется отказ от декодирования (ошибка). На выход выдаются принятые значения символов. Процесс завершается.

На Рисунке 93 показана эффективность IBP декодирования для двоичного сверточного циклического (273, 191, 17) PG

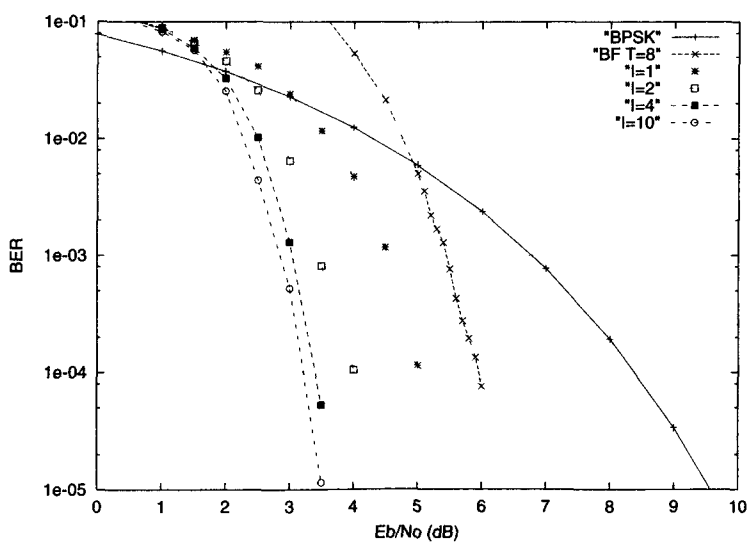


Рис. 93. Эффективность итеративного декодирования двоичного (273, 191) кода.

кода. Этот код является недлинным представителем семейства двоичных конечно-геометрических кодов с низкой плотностью проверок, введенного недавно в работе [KLF]. Показана также зависимость для BF алгоритма с порогом 8.

Замечания

Как предлагалось еще в работе [Gal], IBP алгоритм декодирования можно очевидным образом модифицировать так, чтобы использовать логарифм отношений правдоподобия вместо вероятностей. Это позволяет убрать масштабирование на Шаге 2 процедуры обработки сообщения. В этом случае, на Шаге 1 обработки сообщения потребуется функция

$$F(x) = \frac{e^x + 1}{e^x - 1} = \frac{1}{\tanh(x/2)} \tag{8.32}$$

или ее обратная функция. Программа, реализующая логарифмическую версию IBP декодирования, доступна на ECC вебсайте.

Заметим, что $F(x)$ может быть реализована как таблица. Численные результаты показывают, что квантование $F(x)$ на 8 уровней практически не приводит к потерям по сравнению с вычислениями с плавающей запятой. В работе [FMH] представлено несколько версий IBP алгоритма со сниженной сложностью и показан предпочтительный обмен между сложностью и эффективностью декодирования. Другие интересные применения итеративных алгоритмов декодирования включают быстрые корреляционные атаки в криптоанализе [MG, Glc].

Примечательно то, что LDPC коды способны обнаруживать ошибки декодирования, тогда как турбо коды и коды-произведения, основанные на сверточных кодах, не способны обнаружить большое количество ошибок [Mac]. Это очевидным образом следует из сказанного выше. LDPC коды имеют довольно низкую сложность реализации по сравнению итеративным декодированием, использующим MAP декодирование для компонентных кодов. Это положение справедливо, если

речь идет о сложности, измеряемой числом вещественных сложений и умножений на блок за итерацию.

Тем не менее, необходимо отметить, что IBP декодирование достигает максимума правдоподобия только тогда, *когда граф Таннера не содержит циклов*. Так как для большинства используемых на практике кодов граф Таннера содержит короткие циклы (длины 4 и 6), то сходимость IBP алгоритма декодирования либо не гарантируется, либо медленная [RSU, KFL, For7]. Таким образом, в общем случае IBP алгоритм декодирования для LDPC кодов может потребовать намного больше итераций, чем итеративный алгоритм для произведений кодов с MAP декодированием компонентных кодов.

Для реализации IBP декодирования применяются следующие две архитектуры. Быстрая параллельная архитектура, которая может быть реализована на *N процессорах типа X* для кодовых вершин графа и *M* процессорах типа *Z* для проверочных вершин, соединения между которыми устанавливаются многоадресным вычислительным блоком (ACU). Эта архитектура показана на Рисунке 94. На этом рисунке через λ и π обозначены LLR значения, ассоциированные с условными вероятностями, которые используются IBP алгоритмом. Другая архитектура использует только один *X*-процессор и один *Z*-процессор в режиме разделения времени между вершинами графа, метри-

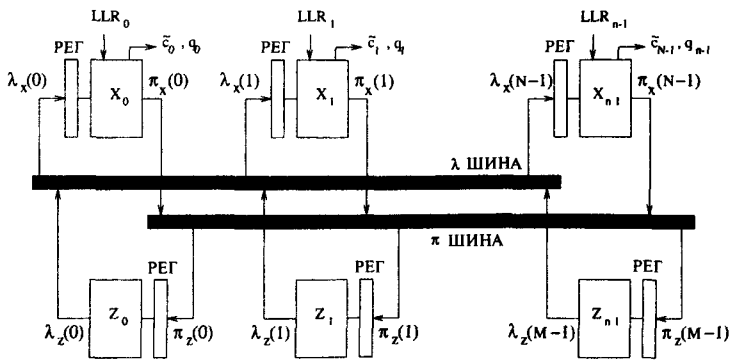


Рис. 94. Параллельная архитектура IBP декодера.

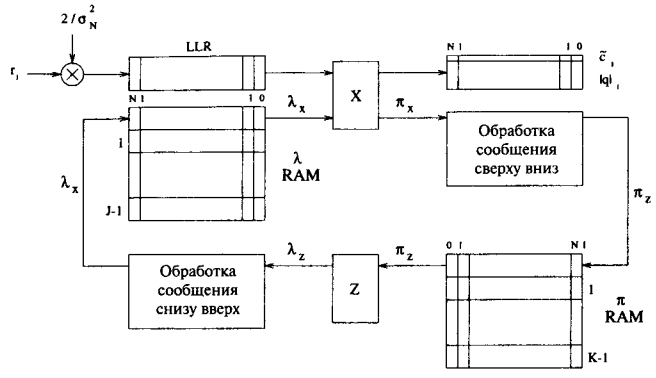


Рис. 95. Последовательная архитектура IBP декодера.

ки которых хранятся в двух блоках памяти (λ-память и π-память) как показано на Рисунке 95. Эти архитектуры представляют собой два крайних решения задачи поиска наилучшего обмена между сложностью декодера и задержкой.

Число вычислительных операций для алгоритма IBP декодирования может быть уменьшено за счет предварительных *жестких решений*, формируемых из амплитуд логарифмов отношений правдоподобия (надежностей) для символов, полученных из канала. Эта идея была предложена в работах [FK] и [Prl] как способ работы с короткими циклами в Байесовых сетях (графах Таннера). Множество позиций с очень высокой надежностью может быть выявлено с помощью сравнения LLR значений на выходе канала с некоторым порогом T. Если надежность символа из канала превышает T, то соответствующая ему кодовая вершина фиксируется как жесткое решение. В результате, соответствующий X-процессор не используется в процедуре декодирования. Вместо вычислений всегда выдается либо максимальная вероятность (LLR), либо поднимается флаг, указывающий на высокую надежность этой позиции. Вследствие этого, Z-процессор, которому пересылается жесткое решение, выполняет меньший объем вычислений, так как из-за высокой надежности данного входа он не участвует в вычислении вероятности или LLR.

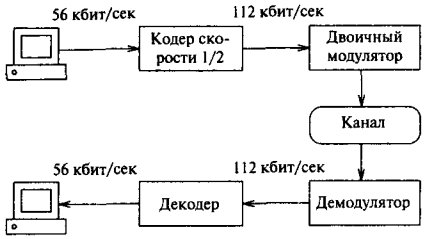
Глава 9

КОМБИНИРОВАНИЕ КОДОВ И ЦИФРОВОЙ МОДУЛЯЦИИ

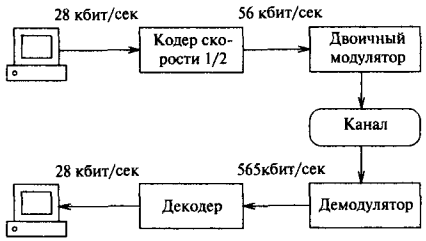
При обсуждении мягкого декодирования все внимание фокусировалось на передаче двоичных сигналов, т.е. $\{-1, +1\}$. Назовем *полосой Найквиста* для передачи (или записи) сигналов скорость R_s , с которой символы передаются (или записываются на некоторый носитель). В случае передачи двоичных символов двум возможным значениям 0 или 1 приписываются два значения сигнала, +1 или −1, соответственно. Следовательно, для передачи этих символов со скоростью R_s (бит/сек) требуется полоса Найквиста R_s(Гц). *Спектральная эффективность* μ двоичной передачи равна 1(бит/сек/Гц) или 1(бит/символ). Кодовой модуляцией называют совместное конструирование помехоустойчивого кодирования и дискретной модуляции (манипуляции) для увеличения спектральной (полосовой) эффективности цифровых систем связи.

9.1. Мотивация

Предположим, что для увеличения надежности двоичной системы передачи (или хранения) информации требуется применение кодов, исправляющих ошибки. Обозначим через $R_c = k/n$ скорость кода. Тогда спектральная эффективность системы равна $\mu = R_c$(бит/сек/Гц). Таким образом, в двоичной системе передачи с кодированием достижимая спектральная эффективность $\mu \leq 1$(бит/сек/Гц). Это означает, что для сохранения заданной скорости передачи двоичных информационных символов требуется увеличение скорости модуляции, т.е. *увеличение полосы пропускания*. Эквивалентно, для того, чтобы сохра-



(а) Сохранение заданной скорости передачи информации



(б) Сохранение заданной скорости передачи символов

Рис. 96 Эффект кодирования для двоичной системы передачи информации.

нить заданную полосу пропускания системы или заданную скорость передачи сигналов, необходимо снизить скорость передачи информации (при использовании кодирования).

Пример 94. Предположим, что требуемая скорость передачи сигналов по каналу составляет 56(кбит/сек). Тогда для передачи двоичных сигналов без кодирования требуется полоса Найквиста 56(кГц). Предположим, что используется сверточный код скорости 1/2. Тогда спектральная эффективность кодированной системы при передаче двоичных сигналов равна $\mu = 0{,}5$(бит/сек/Гц). Влияние параметра μ на скорость передачи данных и скорость модуляции иллюстрируется на Рисунке 96.

Увеличение полосы (эффективной ширины спектра) сигнала, как показано на Рисунке 96(а), не является практичным решением задачи, так как, обычно, *полоса пропускания канала*

является дорогой характеристикой (или ограниченной как, например, в телефонном канале). Снижение скорости передачи данных (Рисунок 96(б)) является возможным решением, однако оно устанавливает предел на количество предлагаемых услуг или приложений. Кроме того, *увеличенная задержка передачи* информации может оказаться недопустимой (как, например, для речи или видео).

Следующий вопрос является фундаментальным: как увеличить скорость передачи информации без расширения полосы пропускания (или скорости передачи сигналов)? Ответ на этот вопрос, предложенный Унгербёком [Ung1] и Имаи с Хирокавой [IH] состоит в использовании расширенного множества сигналов (например, 2^{ν}-ичной фазовой (2^{ν}-ФМ) или амплитудно-фазовой цифровой модуляции (2^{ν}-АФМ или 2^{ν}-КАМ)) совместно с помехоустойчивым кодированием для увеличения Евклидова расстояния между кодированными последовательностями.

9.1.1. Примеры сигнальных множеств.

Некоторые сигнальные множества из тех, которые используются в цифровых системах связи, показаны на Рисунке 97. Координаты I и Q представляют *ортогональные* (несущие) сигналы, которые используются для передачи информации[1]. В цифровых системах связи обычно

$$I = \cos(2\pi f_c t), \quad Q = \sin(2\pi f_c t) \tag{9.1}$$

где несущие I и Q считаются *синфазной* и *квадратурной* (ортогональной) компонентами сигналов, соответственно. Если точка на IQ-плоскости имеет координаты (x, y), то передаваемый сигнал имеет вид

$$s(t) = R\cos(2\pi f_c t + \phi), \quad (k-1)T \le t \le kT \tag{9.2}$$

[1] Ради упрощения обозначений принято, что $I = I(t)$ и $Q = Q(t)$.

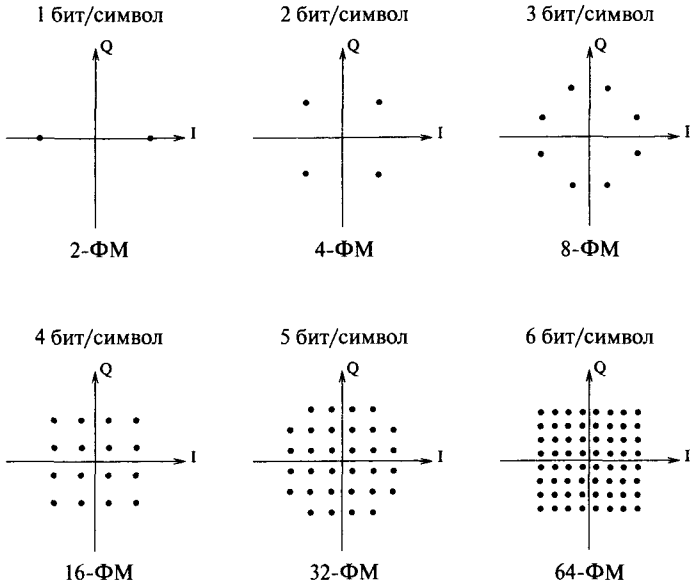


Рис. 97. Примеры M-ФМ и M-КАМ сигнальных созвездий.

где $R = (x^2+y^2)^{1/2}$, $\phi = \tan^{-1}(y/x)$ и T длительность символа. В других системах (например, в системах записи информации) могут использоваться ортогональные импульсы (или последовательности), показанные в качестве примера на Рисунке 98. Каждому сигналу на двумерной IQ-плоскости соответствует *сигнальная точка*. Множество точек (сигналов) на плоскости называют *созвездием* (или *модуляционным алфавитом*).

Добавление переводчика. *Вообще говоря, в качестве несущих могут использоваться любые ортогональные функции времени с ограниченным (или быстро затухающим) спектром (см. например, [Gal*]). Кроме того, существует много других конфигураций созвездий, обладающих меньшей средней мощностью сигнала при сохранении минимального расстояния и числа точек. Наибольшую известность получили конфигурации типа «прямоугольник» и «крест». Подробности можно найти в [Pro*] и [Sch*].*

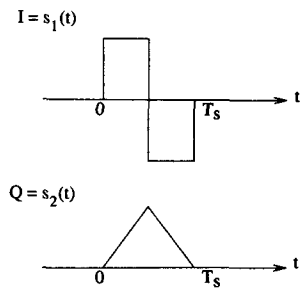
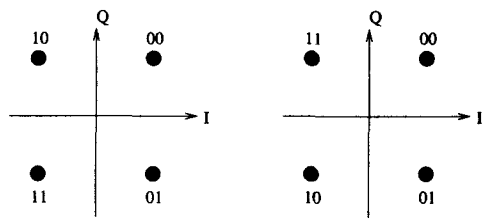


Рис. 98. Пример ортогональных импульсов.



(а) Отображение Грея (б) Естественное отображение

Рис. 99. Два способа нумерации точек 4-ФМ.

С точки зрения цифровой обработки сигналов *модуляция* является *отображением*. Другими словами, это процесс сопоставления ν-мерного вектора $\mathbf{b}$ сигнальной точке $(x(\mathbf{b}), y(\mathbf{b}))$ данного созвездия (модуляционного алфавита). В предыдущих главах рассматривался только случай двоичной фазовой манипуляции (BPSK), $\nu = 1$. В случае $\nu > 1$ имеется много вариантов отображения двоичных векторов на множество сигнальных точек. Иначе, имеется много способов *нумерации сигнальных точек*. На Рисунке 99 показаны два различных (неэквивалентных) способа нумерации сигнальных точек 4-ФМ (QPSK), $\nu = 2$.

Переход от двоичной к 2^{ν}-ичной модуляции обладает тем преимуществом, что увеличивает число бит, передаваемых одним символом, в ν раз, т.е. увеличивает *спектральную эффективность* системы. С другой стороны, при этом возрастает средняя энергия сигналов (M-КАМ) или уменьшается рассто-

яние между сигнальными точками (M-ФМ). На практике мощность передатчика всегда ограничена некоторым максимальным значением. Это означает, что *сигнальные точки оказываются ближе* друг к другу. Напомним, что вероятность ошибки в канале с АБГШ для точек с Евклидовым расстоянием между ними D_e равна [Нау, Pro]

$$\Pr(\varepsilon)=Q\left(\sqrt{\frac{D_e^2}{2N_0}}\right) \qquad (9.3)$$

где функция $Q(x)$ определена в (1.2). В результате уменьшения расстояния ожидается *увеличение вероятности ошибки* на приемном конце. Таким образом, назначение помехоустойчивого кодирования состоит в снижении вероятности ошибки и улучшении качества системы.

9.1.2. Кодовая модуляция

В 1974 Мэсси ввел ключевую концепцию, рассматривающую кодирование и дискретную модуляцию как единый процесс обработки сигналов [Mas3], см. Рисунок 100. Таким образом, кодовая модуляция это согласованное сочетание помехоустойчивого кодирования и модуляционных схем.

При комбинировании кодирования и модуляции возникают два фундаментальных вопроса:

1. Как построить нумерацию сигнальных точек?
2. Как сопоставить кодированным *двоичным последовательностям* кодированные *последовательности символов (сигналов)*?

В 1970-х были предложены два основных подхода к конструированию схем кодовой модуляции:

1. *Решетчатая Кодовая Модуляция (TCM) [Ung1]*

   Применить естественное отображение двоичной последовательности в сигналы с помощью декомпозиции множества точек на подмножества. Используя некоторый конечный автомат, сопоставить последовательности символов путям на решетке. Применить *декодирование Витерби* в приемнике.

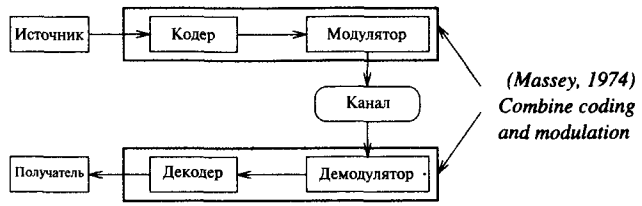


Рис. 100. Идея совместного кодирования и модуляции.

2. *Многоуровневая Кодовая Модуляция (MCM)[IH]*

   Применить отображение кодовых слов на двоичные последовательности с помощью двоичной декомпозиции сигнального созвездия. Для 2^{ν}-ичной модуляции использовать ν кодов, исправляющих ошибки, т.е. один код на каждый битовый уровень сигнального номера. Применить *многоуровневое декодирование* в приемнике.

В обоих случаях (TCM и MCM) основная идея состоит в *расширении созвездия* для того, чтобы получить *избыточность*, необходимую для помехоустойчивого кода, и использовании кодирования для увеличения *минимального Евклидова расстояния* между последовательностями модулированных сигналов.

9.1.3. Расстояние

Для того чтобы продемонстрировать возможность увеличения минимального расстояния между последовательностями сигналов из расширенного множества сигнальных точек с помощью комбинирования кода, исправляющего ошибки, и цифровой модуляции, рассмотрим следующий пример.

Пример 95. На Рисунке 101 показана схема кодовой модуляции на сигналах 4-ФМ (QPSK) с блоковым кодом. Кодовые слова (8, 4, 4) кода Хемминга разделены на четыре пары символов, каждая из которых отображена на сигнальные точки 4-ФМ с помощью отображения Грея. Замечательное свойство нумерации точек 4-ФМ кодом Грея состоит в том, что квадрат расстояния Евклида между точками на плоскости равен удвоенному расстоянию Хемминга между их метками (номерами),

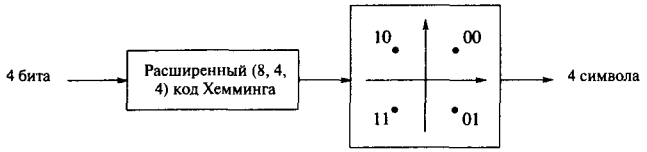


Рис. 101. Конструкция блоковой кодовой модуляции на сигналах 4-ФМ с расширенным (8, 4, 4) кодом Хемминга.

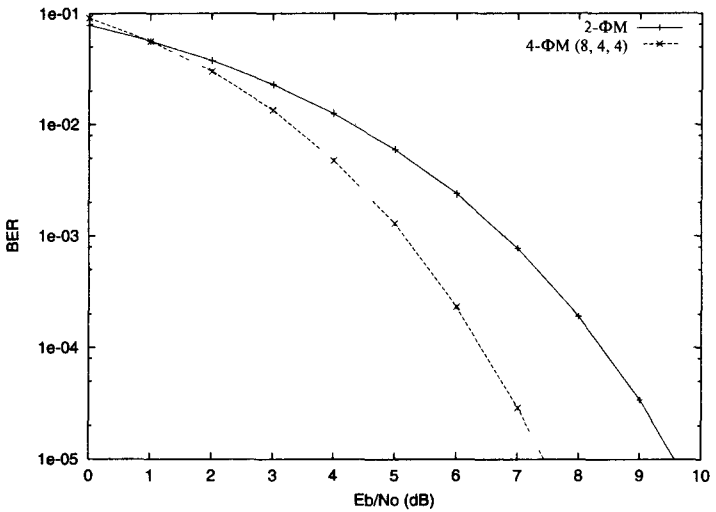


Рис. 102. Результаты моделирования конструкции на сигналах 4-ФМ с расширенным (8, 4, 4) кодом Хемминга в Гауссовом канале.

если принять за единицу радиус окружности, на которой лежат эти четыре точки. В результате получаем конструкцию с блоковой кодовой модуляцией, спектральной эффективностью μ = 1(бит/символ) и *квадратом минимального Евклидова расстояния*, МКЕР, равным $D^2_{min} = 8$. В системе без кодирования такая же эффективность достигается при двоичной модуляции (2-ФМ или BPSK), которая имеет расстояние $D^2_{unc} = 4$ (при той же средней мощности сигнала). Таким образом, *асимптотический выигрыш от кодирования* для этой конструкции кодовой модуляции равен

$$G = 10\log_{10}\left(\frac{D^2_{\min}}{D^2_{unc}}\right) = 3dB \qquad (9.4)$$

В Гауссовом канале кодовая конструкция на 4-ФМ (QPSK) сигналах требует, для достижения той же самой вероятности ошибки на бит $P(\varepsilon)$, вдвое меньшую мощность сигнала по сравнению с 2-ФМ (BPSK) сигналами без кодирования. На Рисунке 102 показаны результаты моделирования вероятности ошибки на информационный символ (BER) для этих конструкций.

Таким образом, хотя использование расширенного созвездия 2^ν точек модуляции связано с уменьшением расстояния между сигналами, правильно выбранная схема кодирования позволяет построить последовательности сигналов с минимальным расстоянием большим, чем в системе без кодирования при той же спектральной эффективности.

9.2. Решетчатая кодовая модуляция (TCM)

Главная идея TCM, предложенная Унгербёком в 1976, состоит в том, чтобы реализовать *отображение через разбиения (декомпозицию) множества (сигнальных точек)*. Для этого выбирается базовая структура решетки, ассоциированная с переходами на состояниях *конечного автомата*, и подмножества сигналов отображаются на ребра решетки. В системах, требующих высокой спектральной эффективности, допускается присваивание информационных (не кодированных) битов параллельным ребрам решетки.

9.2.1. Разбиение множества точек и отображение на решетку

Метки, приписываемые сигнальным точкам, определяются с помощью разбиения (декомпозиции) сигнального созвездия. На множестве 2^ν модуляционных точек S применяется схема

вложенных разбиений (древовидная декомпозиция) по ν уровням. На i-ом уровне разбиения, $1 \le i \le \nu$, подмножество сигналов разбивается на два подмножества: $S_1(0)$, $S_1(1)$, если $i = 1$, и $S_i(b_1 \ldots b_{i-1}\ 0)$ и $S_i(b_1 \ldots b_{i-1} 1)$, $i > 1$, так, чтобы *расстояние* δ_i^2 между точками в каждом подмножестве было максимальным. *Битовый разряд* метки $b_i \in \{0, 1\}$ ассоциируется с выбором подмножества, $S_i(b_1 \ldots b_{i-1} b_i)$, на i-ом уровне разбиения. Этот процесс разбиения завершается полной *нумерацией* всех сигнальных точек. Каждая сигнальная точка получает свою (уникальную) метку (номер) из ν бит $b_1 b_2 \ldots b_\nu$, обозначаемую в дальнейшем $s(b_1 b_2 \ldots b_\nu)$. Описанная процедура реализует *стандартное разбиение (по Унгербёку)* созвездия точек 2^ν-ичной модуляции. При таком разбиении внутренние расстояния в подмножествах образуют неубывающую последовательность $\delta_1^2 \le \delta_2^2 \le \ldots \le \delta_\nu^2$. Результат соответствует *естественной нумерации* точек M-ФМ модуляции, т.е. двоичному представлению целых чисел, величина которых возрастает с переходом по часовой стрелке (или по счетчику). На Рисунке 103 показана естественная нумерация точек 8-ФМ, дающая в результате $\delta_1^2 = 0{,}586$, $\delta_2^2 = 2$, $\delta_3^2 = 4$.

Унгербёк рассматривал кодер «как конечный автомат с заданным числом состояний и заданными переходами на множестве состояний». Он предложил несколько практических правил отображения подмножеств сигналов и точек на ребра решетки. Эти правила сводятся к следующим.

Правило 1: все подмножества должны появляться на решетке с одинаковой вероятностью.

Правило 2: входящие и исходящие переходы одного и того же состояния, должны быть приписаны подмножествам, находящимся на наибольшем Евклидовом расстоянии.

Правило 3: параллельные переходы присваиваются сигнальным точкам, разделенным наибольшим Евклидовым расстоянием (высшие уровни разбиения).

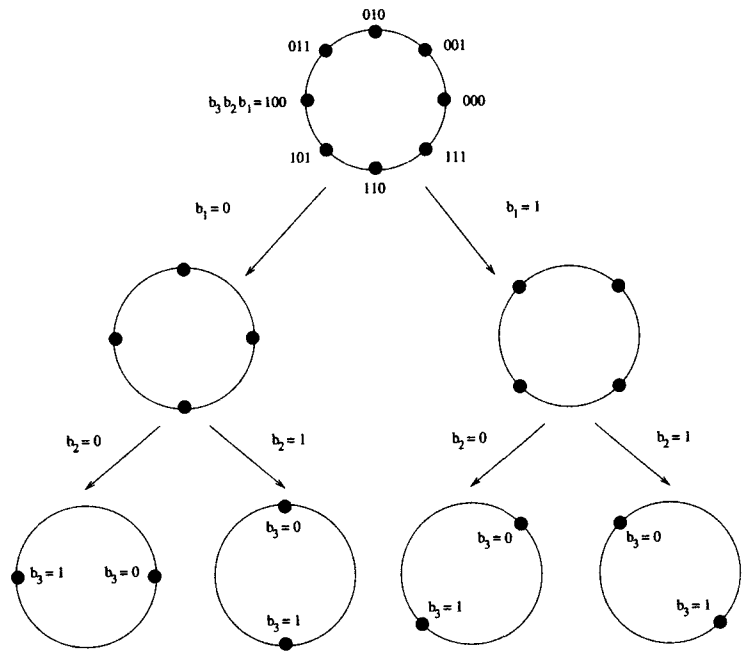


Рис. 103. Естественное отображение для 8-ФМ созвездия.

Общая структура TCM кодера показана на Рисунке 104. В общем случае решетчатой кодовой модуляции скорости $(\nu - 1)/\nu$ структура решетки определяется сверточным кодом скорости $(k+1)/k$. Информационные символы, которые не кодируются, соответствуют *параллельным ребрам* на решетке.

Пример 96. В этом примере рассматривается система кодовой модуляции скорости 2/3 на решетке с четырьмя состояниями. Созвездие для 8-ФМ показано на Рисунке 103. Спектральная эффективность равна $\mu = 2$(бит/символ). Структурная схема кодера показана на Рисунке 105. Двоичный сверточный код скорости 1/2 и памяти 2 тот же, что использовался в Главе 5. На Рисунке 106 видно, что структура решетки совпадает с решеткой сверточного кода с тем только исключением, что каждое ребро на исходной решетке заменяется двумя параллельными ребрами, ассоциированными с не кодируемым битом u_1.

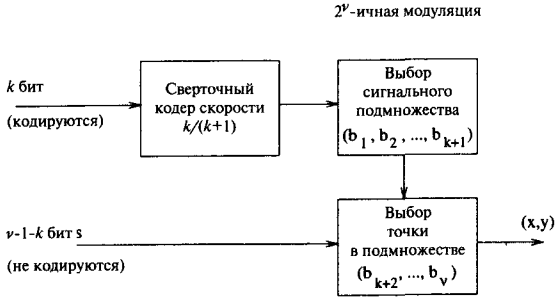


Рис. 104. Структура TCM кодера скорости $(\nu - 1)/\nu$.

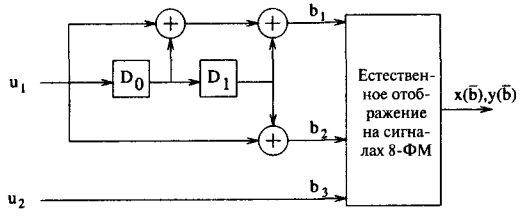


Рис. 105. Кодер скорости 2/3 с четырьмя состояниями для решетчатой кодовой модуляции на сигналах 8-ФМ.

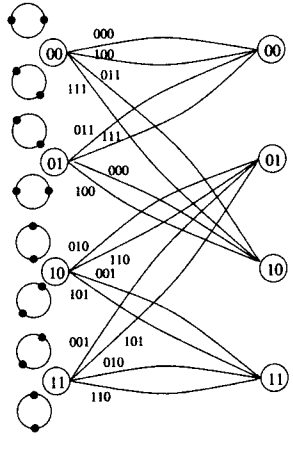


Рис. 106. Структура решетки кодовой модуляции скорости 2/3 на сигналах 8-ФМ, соответствующая кодеру на Рисунке 105.

9.2.2. Декодирование по максимуму правдоподобия

Для выбора наиболее правдоподобных TCM последовательностей можно использовать алгоритм Витерби[2] при условии, что генератор (вычислитель) реберных меток учитывает параллельные ребра. Кроме того, должен быть изменен блок выбора лучшего ребра и выживших не кодированных символов. Память выживших путей (или память обратного прохода) должна включать $(\nu - k - 1)$ не кодированных двоичных символов в отличие от одного только бита в случае сверточного кода скорости $1/n$. Важно заметить, что в случае 2^{ν}-ичной ФМ (2^{ν}-PSK) или 2^{ν}-ичной КАМ (2^{ν}-QAM) *корреляционные метрики* для *двумерных символов* имеют вид $x_p x_r + y_p y_r$, где (x_p, y_p) представляет эталон сигнальной точки в созвездии, а (x_r, y_r) является принятой точкой. Все соображения о реализации, обсуждавшиеся в Разделах 5.4 и 7.2, относятся и к декодеру TCM последовательностей.

9.2.3. Расстояние и вероятность ошибки

Помехоустойчивость TCM последовательностей можно анализировать так же, как для сверточных кодов. Это означает, что из диаграммы состояний TCM кодера может быть получен нумератор спектра весов по методу из Раздела 5.3. Единственная разница состоит в том, что теперь степени будут не целыми числами (соответствующими расстоянию Хемминга), а вещественными (соответственно расстоянию Евклида). Необходимо аккуратно учитывать факт наличия параллельных переходов на диаграмме состояний. Последнее означает, что модифицированная диаграмма состояний содержит два члена, подробности в [BDMS].

Пример 97. На Рисунке 107 показана модифицированная диаграмма состояний для решетчатой кодовой модуляции с четырьмя состояниями на сигналах 8-ФМ из Примера 96. Ребра

[2] Алгоритм Витерби рассматривался в Разделах 5.4 и 7.2.

решетки помечены целыми числами, соответствующими восьми значениям фазы сигнала. Для вычисления нумератора весов $T(x)$ применяется процедура из Раздела 5.3. Анализируя не-

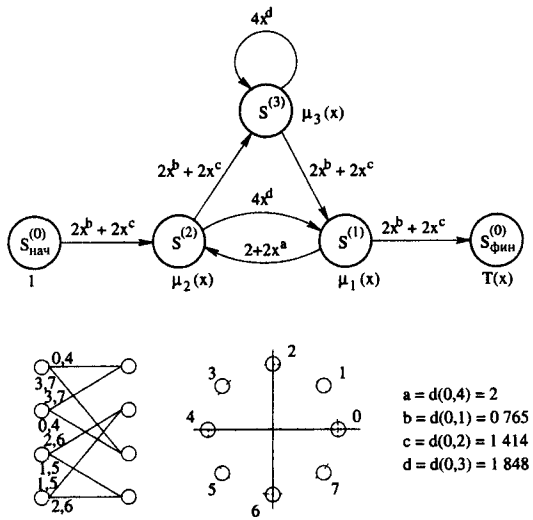


Рис. 107. Модифицированная диаграмма состояний конструкции TCM для 4-х состояний на сигналах 8-ФМ.

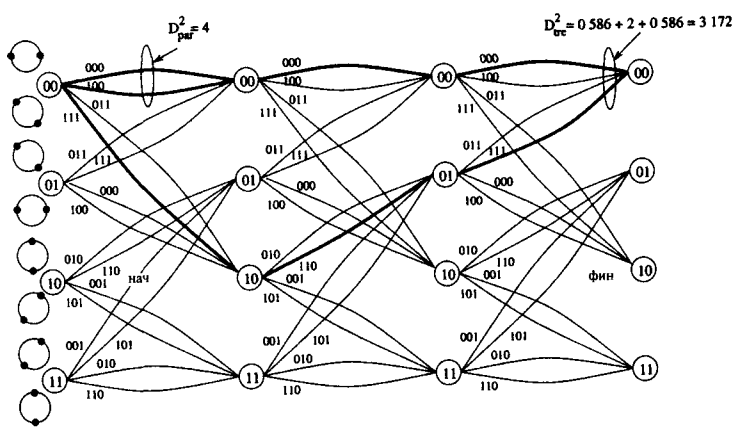


Рис. 108. Два пути с минимальным Евклидовым расстоянием между ними на решетке из Примера 96.

посредственно структуру решетки на Рисунке 108, можно определить, что квадрат минимального расстояния между кодовыми последовательностями равен

$$D_C^2 = \min\{D_{par}^2, D_{tre}^2\} = 3{,}172$$

Следовательно, по сравнению с системой 4-ФМ, имеющей спектральную эффективность тоже 2(бит/символ), асимптотический выигрыш от кодирования равен 2дБ.

9.2.4. Практические конструкции TCM и двухэтапное декодирование

Из практических соображений в работах [Vit4, ZW] было предложено такое разбиение созвездия 2^{ν} модуляционных точек, при котором смежные классы двух верхних уровней разбиения ассоциируются с выходами стандартного сверточного кодера памяти 6 и скорости 1/2. Такое отображение приводит к *практической конструкции TCM*. Это означает, что в общей структуре кодера, показанной на Рисунке 104, значение $k = 1$ фиксировано. Соответствующая структура кодера показана на Рисунке 109. В результате получаем структуру решетки практической TCM, которая не зависит от значения $\nu > 2$, в отличие от TCM конструкций, предложенных Унгербёком. Отличие состоит в том, что число параллельных ребер решетки $\nu - 2$ растет с числом бит на символ. Такая структура предполагает двухэтапное декодирование. На первом этапе параллельные ребра решетки объединяются в одно, после чего используется обыч-

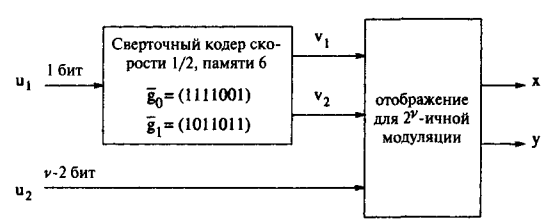


Рис. 109. Структурная схема кодера практической конструкции TCM.

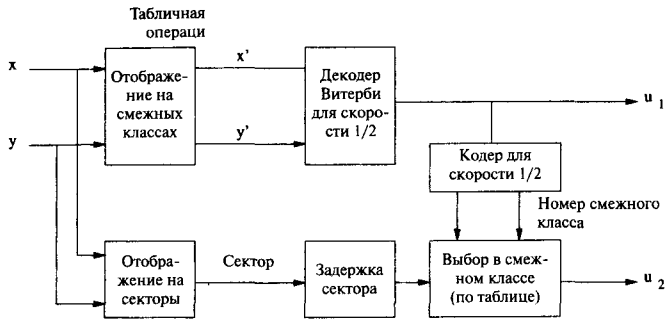


Рис. 110. Структурная схема двух этапного декодера практической конструкции TCM.

ный (с полки, из библиотеки) декодер Витерби для оценки кодированных символов, ассоциированных с двумя верхними уровнями разбиения. На втором этапе используются полученные оценки кодированных двоичных символов и значение (положение на плоскости) принятого сигнала для оценки не кодированных двоичных символов. На Рисунке 110 показана структурная схема двух этапного декодера практической TCM конструкции.

В работе [MM] принятые символы подвергаются преобразованию, которое позволяет использовать декодер Витерби без каких-либо изменений на этапе вычисления метрик ребер. Процедура декодирования соответствует процедуре, представленной в работе [CRKO, PS], с тем только отличием, что с учетом преобразования символов алгоритм Витерби может применяться таким же образом, как и для сигналов типа 2-ФМ (или 4-ФМ). Этот метод рассматривается ниже для случая M-ФМ.

Обозначим (x, y) значения I и Q координат принятого M-ФМ сигнала, амплитуда и фаза которого равны $r = (x^2 + y^2)^{1/2}$ и $\phi = \tan^{-1}(y/x)$, соответственно. Преобразование использует значение ϕ и применяется таким образом, что точки M-ФМ отображаются в точки «смежных классов», помеченных выходными (кодовыми) символами сверточного кодера скорости 1/2 и памяти 6.

Для TCM конструкции на сигналах M-ичной фазовой модуляции, $M = 2^{\nu}$, $\nu > 2$, введем параметр ξ, равный числу *кодированных бит на символ*[3], $\xi = 1, 2$. Тогда следующее ниже преобразование вращения, примененное к каждому принятому сигналу (x, y), дает входной символ (x', y') для декодера Витерби,

$$x' = r\cos[2^{\nu-\xi}(\phi - \Phi)],$$
$$y' = r\sin[2^{\nu-\xi}(\phi - \Phi)], \qquad (9.5)$$

где Φ некоторая фиксированная величина, действующая на все точки одинаково. После преобразования (9.5) $2^{\nu-\xi}$ точек смежного класса исходного 2^{ν}-ФМ созвездия схлопываются в одну точку 2^{ξ}-ФМ созвездия смежных классов на плоскости x'-y'.

Пример 98. Рассматривается решетчатая кодовая модуляция на сигналах 8-ФМ скорости 2/3 с двумя информационными битами на символ. Два информационных бита (u_1, u_2) вводятся в кодер, который формирует тройку (u_2, v_2, v_1), отображаемую на множество сигнальных точек 8-ФМ. Символы v_2 и v_1 представляют выход стандартного сверточного кодера скорости 1/2 и памяти 6[4]. Сигнальные точки нумеруются тройками (u_2, v_2, v_1), а каждая пара (v_2, v_1) является индексом смежного класса, содержащего две точки 2-ФМ, в созвездии 8-ФМ, как показано на Рисунке 111.

В данном случае в результате преобразования вращения с параметром $\phi' = 2\phi$ получаем, что подмножество 2-ФМ точек исходного множества 8-ФМ точек сводится в одну из точек 4-ФМ созвездия смежных классов на плоскости x'-y'. Это разбиение с преобразованием показано на Рисунке 111. Заметим, что обе точки любого смежного класса имеют одно и тоже значение фазы ϕ'. Это следует из того, что эти точки имеют фазы ϕ и $\phi+\pi$.

На выходе декодера Витерби получается оценка информационного *кодированного* бита u_1. Чтобы получить оценку *не кодированного* бита u_2 необходимо повторным кодированием оценки u_1 найти наиболее вероятный индекс смежного класса.

[3] Случай $\xi = 1$ соответствует конструкции TCM, в которой кодированные биты, распределены на два 8-ФМ сигнала. В результате имеем конструкцию скорости 5/6 на сигналах 8-ФМ, предложенную в спецификации DVB-DSNG [DVB].

[4] Выходы v_2 и v_1 соответствуют сверточному кодеру с генераторами 133 и 171 в восьмеричной записи.

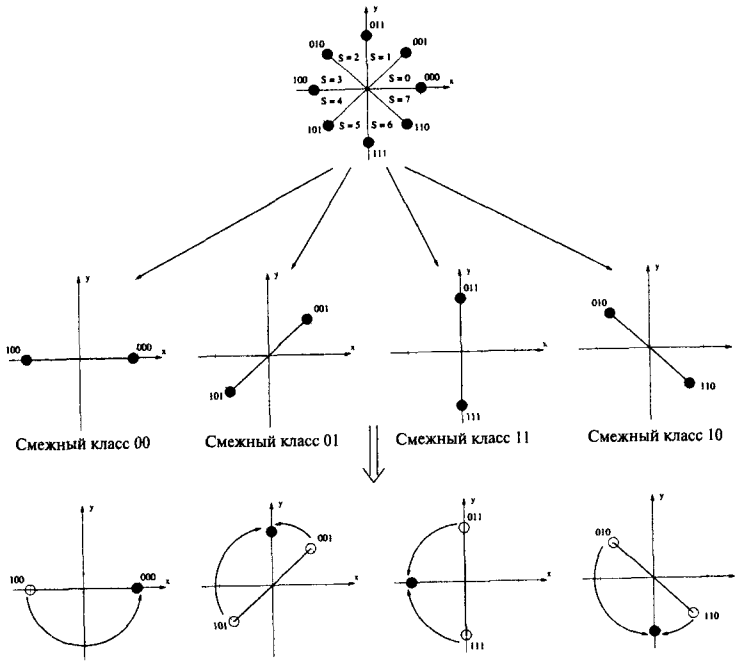


Рис. 111. Декомпозиция созвездия 8-ФМ ($\xi = 2$) и смежные классы.

Вычисленный индекс и *сектор*, в котором находится принятый символ 8-ФМ, используются для определения u_2. В заданном смежном классе сектор S указывает ближайшую к принятому символу 8-ФМ точку (с индексом u_2) из пары точек 2-ФМ. Например, если декодированный смежный класс есть (1, 1) и принятый символ находится в секторе S=3, то $u_2 = 0$. Это можно проверить на Рисунке 111.

На Рисунке 112 представлены результаты моделирования декодирования по максимуму правдоподобия (кривая с меткой TC8PSK_23_SSD) и двухэтапного декодирования практической TCM конструкции на сигналах 8-ФМ (кривая с меткой TC8PSK_23_TSD). Для сравнения показана кривая для 4-ФМ сигналов без кодирования. На представленных кривых двухэтапное декодирование уступает максимуму правдоподобия всего только 0,2дБ.

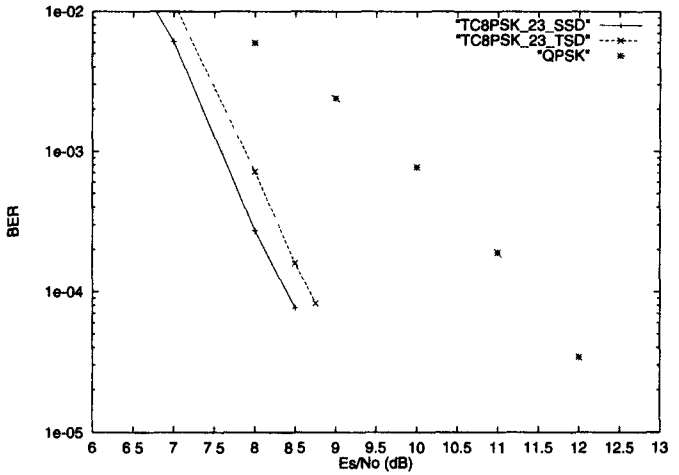


Рис. 112. Результаты сравнительного моделирования BER для декодера по максимуму правдоподобия и двухступенчатого декодера практической TCM конструкции на сигналах 8-ФМ.

Аналогичное преобразование может применяться и в случае M-КАМ (M-QAM). Отличие состоит в том, что преобразование выполняется раздельно для символов I-канала и Q-канала (напомним, что I и Q ортогональные каналы). Таким образом, нет необходимости вычислять фазу. На Рисунке 113 дан пример для TCM на сигналах 16-КАМ с ξ=2 кодированными битами на символ. Теперь кодовые биты являются индексами смежных классов 4-ФМ подмножеств. Преобразование для 16-КАМ (ξ=2) определено следующими операциями (напоминающими операции по модулю 4):

$$x' = \begin{cases} x, & |x| \le 2; \\ (x-4), & x \ge 2; \\ (x+4), & x < 2, \end{cases}$$

$$y' = \begin{cases} y, & |y| \le 2; \\ (y-4), & y \ge 2; \\ (y+4), & y < 2. \end{cases}$$

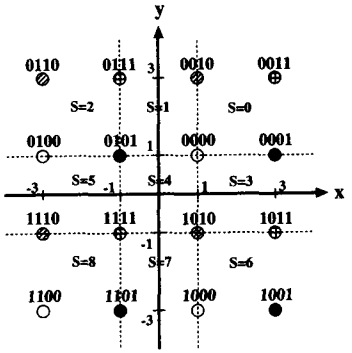


Рис. 113. 16-КАМ созвездие для практической TCM с двухуровневым декодированием.

Наконец, интересно заметить, что недавно в работе [WM] была предложена практическая конструкция TCM с турбо кодом в качестве компонентного кода.

9.3. Многоуровневая кодовая модуляция (MCM)

В многоуровневой схеме кодирования, предложенной Имаи-Хирокава [IH], для множества 2^ν сигнальных точек созвездия используется ν уровневая схема вложенных разбиений на два подмножества. Элементы кодовых слов ν двоичных компонентных кодов C_i, $1 \le i \le \nu$, используются для индексации (нумерации) смежных классов на каждом уровне разбиения. Одним из преимуществ MCM конструкций является гибкость в согласовании Евклидовых расстояний на подмножествах сигнальных точек, δ_i^2, $i = 1, 2, \ldots, \nu$, на каждом уровне разбиения с расстояниями Хемминга компонентных кодов. Уочмэн с соавторами [WFH] предложил несколько правил конструирования, основанных на соображениях пропускной способности (применил цепное неравенство для взаимной информации). Более того, как показано в работах [WFH, For8], многоуровневые коды с длинными компонентными кодами, такими как турбо коды или коды с низкой плотностью проверок, достигают пропускной способности канала.

Полезно так же отметить, что при выборе двоичных компонентных кодов разбиение на подмножества является дихотомическим, однако в общем случае компонентные коды могут быть выбраны над любым конечным полем соответственно схеме разбиения сигнального множества. Другим важным преимуществом многоуровневого кодирования является то, что декодирование (двоичного кода) может выполняться независимо на каждом уровне. Такое *многоуровневое* декодирование позволяет существенно снизить сложность по сравнению с оптимальным декодированием всего кода.

9.3.1. Конструкции и многоуровневое декодирование.

Обозначим C_i, $1 \le i \le \nu$, двоичный линейный блоковый (n, k_i, d_i) код. Обозначим $\mathbf{v}_i = (v_{i1}, v_{i2}, \ldots, v_{in})$ кодовое слово кода C_i. Рассмотрим код, образованный чередованием позиций подкодов, $\pi(|C_1|C_2|\ldots|C_\nu|)$, с кодовым словом вида

$$\mathbf{v} = (v_{11}v_{21}\cdots v_{\nu 1}\ v_{12}v_{22}\cdots v_{\nu 2}\cdots v_{1n}v_{2n}\cdots v_{\nu n})$$

Каждый блок из ν компонент вектора $\mathbf{v}$ является меткой (номером) сигнала на множестве 2^ν модуляционных точек S. Тогда

$$s(\mathbf{v}) = (s(v_{11}v_{21}\cdots v_{\nu 1}), s(v_{12}v_{22}\cdots v_{\nu 2}), \cdots, s(v_{1n}v_{2n}\cdots v_{\nu n}))$$

является последовательностью сигнальных точек в S.

Последовательности сигналов из множества S вида

$$\Lambda = \{s(\mathbf{v}): \mathbf{v} \in \pi(|C_1|C_2|\cdots|C_\nu|)\}$$

образуют *ν уровневый модуляционный код над сигнальным множеством S* или ν уровневую конструкцию кодовой модуляции на множестве 2^ν сигналов. Такое же определение справедливо и для сверточных компонентных кодов.

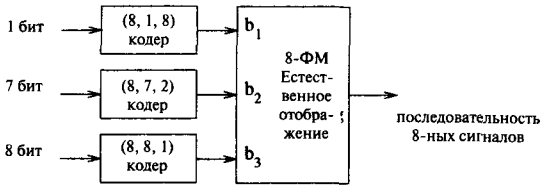


Рис. 114. Пример MCM системы на сигналах 8-ФМ.

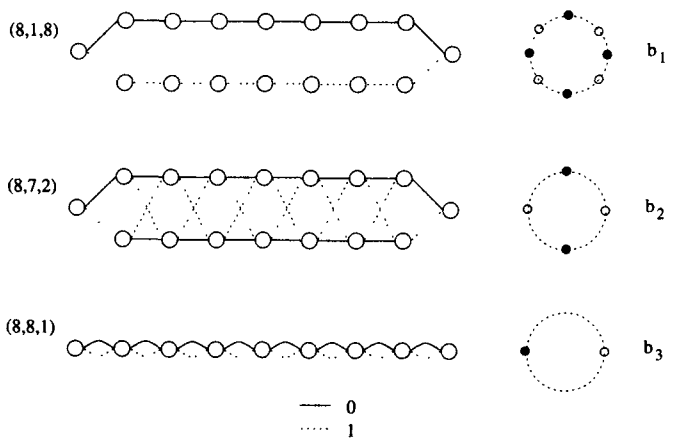


Рис. 115. Решетки компонентных кодов в примере MCM на сигналах 8-ФМ.

Скорость, или спектральная эффективность, этой многоуровневой конструкции равна $R = (k_1+k_2+\ldots+k_\nu)/n$ (бит/символ). Квадрат минимального Евклидова расстояния этой системы $D_C^2(\Lambda)$ удовлетворяет оценке [IH]

$$D_C^2(\Lambda) \geq \min_{1\leq i\leq \nu}\{d_i\delta_i^2\} \qquad (9.6)$$

Пример 99. В этом примере рассматривается система кодовой модуляции на сигналах 8-ФМ, на трех уровнях с блоковыми кодами. Структура кодера изображена на Рисунке 114. В предположении единичной энергии множества сигналов 8-ФМ и для схемы разбиения на Рисунке 103 находим, что квадраты минимальных Евклидовых расстояний равны $\delta_1^2 = 0{,}586$, $\delta_2^2 = 2$, $\delta_3^2 = 4$.

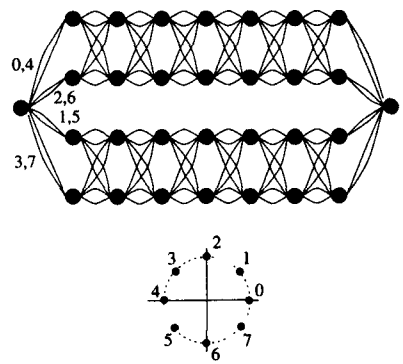


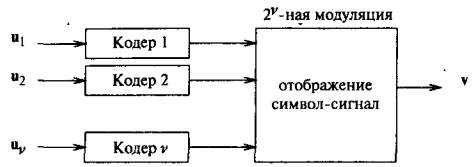
Рис. 116. Полная решетка для конструкции MCM на сигналах 8-ФМ.

Минимум квадрата расстояния Евклида для этой конструкции на сигналах 8-ФМ равен

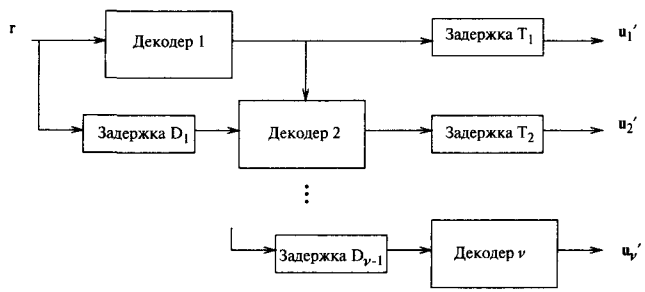
$$D_C^2(\Lambda) = \min\{d_1\delta_1^2, d_2\delta_2^2, d_3\delta_3^2\} = \min\{8\times 0{,}586,\ 2\times 2,\ 1\times 4\} = 4$$

Таким образом, выигрыш от кодирования по отношению к 4-ФМ (QPSK) сигналам без кодирования составляет 3дБ. Решетки компонентных кодов показаны на Рисунке 115. Полная решетка конструкции показана на Рисунке 116.

Как уже упоминалось, одним из преимуществ многоуровневого кодирования является возможность применения многоступенчатого декодирования. На Рисунках 117 (а) и (б) показаны основные структуры, используемые для кодирования и декодирования многоуровневых кодов. Многоступенчатое декодирование приводит к снижению сложности (измеряемой, например, числом ребер на решетке декодирования) по сравнению с декодированием по максимуму правдоподобия (реализуемым, например, алгоритмом Витерби на полной решетке многоуровневого кода). Однако при многоступенчатом декодировании декодеры ранних уровней считают, что на старших уровнях кодирование не применяется. Это приводит к увеличению количества кодовых слов на минимальном расстоянии. Иначе говоря, увеличивается *коэффициент ошибок или число ближайших соседей.* Величина связанных с этим эф-



(а) Многоуровневое кодирование



(б) Многоступенчатый декодер

Рис. 117. Базовая структура кодера и декодера для систем многоуровневой кодовой модуляции.

фектом потерь зависит от выбора компонентных кодов и отображения символов на сигналы. В диапазоне вероятности ошибки порядка $10^{-2} \sim 10^{-5}$ эта величина может достигать нескольких дБ.

Пример 100. В этом примере рассматривается трехуровневая система модуляции на сигналах 8-ФМ. Декодер первого уровня использует решетку первого компонентного кода C_1. Метриками ребер являются расстояния (корреляции) между подмножеством, выбранным на первом уровне декомпозиции, и принятой последовательностью сигналов, как показано на Рисунке 118.

Решение, принятое на первом уровне, передается на следующий уровень декодирования. Декодер на втором уровне использует решетку второго компонентного кода с информацией, полученной от первого уровня декодирования. В случае

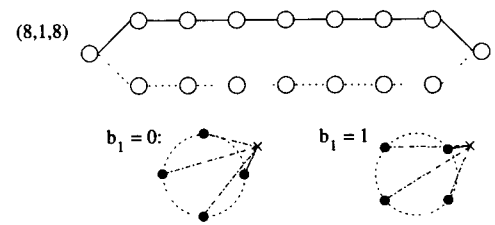


Рис. 118. Решетка и символы, используемые при вычислении метрик на первой ступени декодирования.

8-ФМ сигналов, если результат декодирования на первом уровне равен $b_1 = 0$, то принятая последовательность сигналов не изменена. Если же этот результат равен $b_1 = 1$, то принятый сигнал повернут на 45°. Далее, метрики ребер являются расстояниями (корреляциями) между подмножеством, выбранным на втором уровне декомпозиции (при условии, что решение на первом уровне декодирования уже принято), и принятой последовательностью сигналов.

Декодер третьего уровня (третьего компонентного кода) включается, когда уже известны решения первых двух уровней декодирования. Метрики ребер вычисляются как для двоичной модуляции. На этом уровне возможны четыре варианта 2-ФМ сигналов, отличающиеся поворотом, в зависимости от принятых решений на двух предыдущих уровнях декодирования. Таким образом, возможное решение состоит в дополнительном повороте принятого сигнала в зависимости от решений b_1 b_2 до некоторого стандартного положения и использовании одного и того же двоичного созвездия. Эта схема проиллюстрирована на Рисунке 119.

Для кодов средней и большой длины предельная эффективность многоуровневой кодовой модуляции может достигаться с помощью гибридного подхода, когда на первых уровнях используются мощные турбо коды, а на остальных уровнях используются двоичные коды с жестким декодированием. Такие комбинации могут достигать очень высокой помехоустойчивости [WFH].

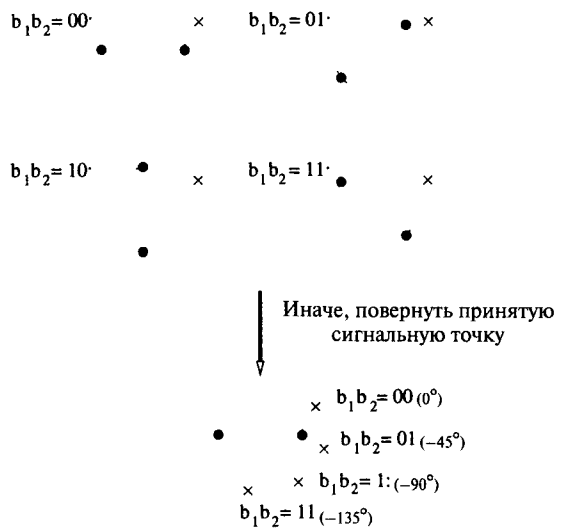


Рис. 119. Решетка и символы, используемые при вычислении метрик на третьей ступени декодирования.

9.3.2. Неравная защита в системах многоуровневой кодовой модуляции

Многоуровневая кодовая модуляция является удобной схемой создания *неравной защиты от ошибок* (UEP – unequal-error-protection), так как она обладает необходимой гибкостью в конструировании минимальных Евклидовых расстояний между кодовыми последовательностями на каждом уровне разбиений. Однако следует очень внимательно выбирать отображение между символами и сигналами с тем, чтобы не разрушить искомую способность к неравной защите. Эта тема исследовалась в работах [MFLI, IFMLI], где были предложены некоторые подходы, сочетающие обобщение правил блочного разбиения (декомпозиции) [WFH] и введенных Унгербёком в [Ung1].

При *смешанном разбиении* некоторые уровни разбиения являются нестандартными, тогда как другие выполняются по

правилам, предложенным Унгербёком[Ung1]. Этим способом достигается хороший обмен между коэффициентами ошибок и Евклидовыми расстояниями по уровням конструкции. Для достижения неравной защиты расстояния Евклида по уровням разбиения выбираются следующим образом

$$d_1\delta_1^2 \geq d_2\delta_2^2 \geq \cdots \geq d_\nu\delta_\nu^2 \tag{9.7}$$

Для $1 \leq i \leq \nu$ обозначим $\mathbf{v}_i(\mathbf{u}_i)$ кодовое слово кода C_i, соответствующее информационному вектору $\mathbf{u}_i$ размерности k_i бит. Обозначим $\mathbf{s} = \mathbf{s}(\mathbf{u})$ и $\mathbf{s}' = \mathbf{s}(\mathbf{u}')$ последовательности 2^ν-ных сигналов для информационных векторов $\mathbf{u} = (\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_\nu)$ и $\mathbf{u}' = (\mathbf{u}'_1, \mathbf{u}'_2, \ldots, \mathbf{u}'_\nu)$, соответственно. *Евклидово разделение* [YI] между кодовыми последовательностями на i-ом уровне декомпозиции для $i = 1, 2, \ldots, \nu$ определено как

$$s_i = \min\{d(\mathbf{s},\mathbf{s}'): \mathbf{u}_i \neq \mathbf{u}'_i, \mathbf{u}_j = \mathbf{u}'_j, j < i\} \tag{9.8}$$

где $s_1=d_1\delta_1^2$, $s_2=d_2\delta_2^2$, …, $s_\nu=d_\nu\delta_\nu^2$. В канале с АБГШ система неравенств (9.7) обеспечивает снижение уровня защиты от ошибок для компонентных сообщений меньшего уровня.

Известно [WFH], что правила разбиения Унгербёка [Ung1] не подходят к системам многоступенчатого декодирования МСМ последовательностей при низких и средних отношениях сигнал-шум. Это объясняется большим числом (NN) ближайших (соседей) кодовых слов на первых ступенях декодирования.

Пример 101. На Рисунке 120 представлены результаты моделирования трехуровневой кодовой модуляции на сигналах 8-ФМ с расширенными кодами БЧХ (64, 18, 22), (64, 57, 4) и (64, 63, 2) в качестве компонентных кодов C_i, $i = 1, 2, 3$, соответственно. Для этой схемы Евклидовы разделения равны $s_1 = 12{,}9$, $s_2 = s_3 = 8$ для 18 и 120 информационных символов, соответственно. Этому соответствует асимптотический энергетический выигрыш от кодирования 8,1дБ и 6дБ. Неприятный эффект от большого количества соседей NN (или коэффициента ошибок) на первых ступенях декодирования состо-

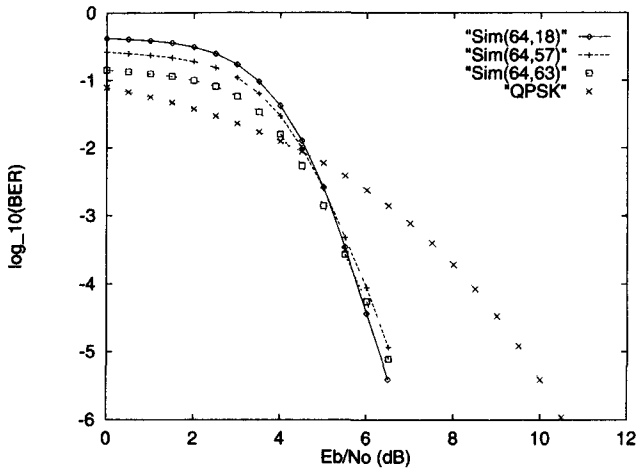


Рис. 120. Результаты моделирования трехуровневой кодовой модуляции на сигналах 8-ФМ с разбиениями по схеме Унгербёка.

ит в значительном снижении реального энергетического выигрыша.

Ниже рассматриваются схемы неравной защиты, основанные на нестандартном разбиении. Читатели могут найти дополнительные подробности конструирования схем с обычной (равной) и неравной защитой в работах [WFH, MFLI, IFMLI].

Нестандартное разбиение

На Рисунке 121 показано блочное разбиение [WFH], использованное для построения трехуровневой системы на сигналах 8-ФМ с неравной защитой. Здесь черным цветом представлены сигнальные точки с метками вида $0b_2b_3$, где b_2, $b_3 \in \{0, 1\}$. Аналогично, белым цветом выделены точки с метками $1\ b_2b_3$. Кружками отмечены точки с метками вида b_10b_3, где b_1, $b_3 \in \{0, 1\}$, и квадратиками выделены сигнальные точки с метками b_11b_3.

Из Рисунка 121 (б) видно, что для определения первого бита метки b_1 достаточно только координаты X. Если сигнальная точка находится в левой полуплоскости ($X < 0$), то $b_1 = 0$, ина-

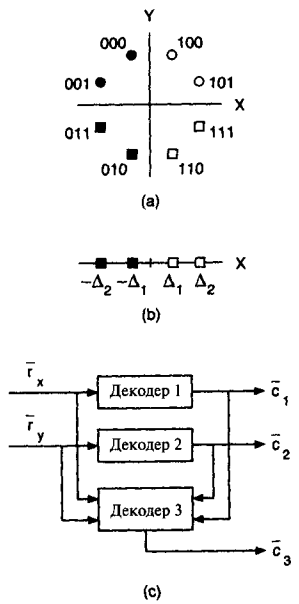


Рис. 121. 8-ФМ созвездие с блоковым разбиением: (а) нумерация, (б) проекции на X координату, (в) структура декодера (c_i обозначают оценки кодовых слов в коде C_i, $i = 1, 2, 3$.).

че $b_1 = 1$. Таким же образом, находим, что координаты Y достаточно для определения значения второго бита метки b_2. Если сигнальная точка находится в верхней полуплоскости ($Y > 0$), то $b_2 = 0$, иначе $b_2 = 1$. Это свойство блочного разбиения позволяет реализовать первые два уровня декодирования *независимо* или *параллельно*. Аналогичные соображения привели к созданию *параллельного декодирования* многоуровневых кодов с отображением Грея в работе [Schr].

Многоступенчатое декодирование

На первой и второй ступенях декодирования решение находится с помощью проекции принятой последовательности сигналов на ось X или Y, соответственно. На Рисунке 121(в) показана структурная схема многоступенчатого декодера для

трехуровневой кодовой модуляции на сигналах 8-ФМ с блочным разбиением. Декодеры первой и второй ступени работают независимо с синфазными и квадратурными компонентами принятой последовательности сигналов $\mathbf{r}_x$ и $\mathbf{r}_y$, соответственно. Решения, принятые относительно оценок соответствующих кодовых слов $\mathbf{v}_1$ и $\mathbf{v}_2$, сразу же передаются на третью ступень декодирования.

Пусть $\mathbf{v}_i^* = (v_{i1}^*, v_{i2}^*, \ldots, v_{in}^*) \in C_i$ декодированное кодовое слово на i-ой ступени декодирования, $i = 1, 2$. Перед декодированием третьей ступени каждая точка принятой последовательности $\mathbf{r} = (\mathbf{r}_x, \mathbf{r}_y)$ с координатами (r_{xj}, r_{yj}) проектируется из двумерного пространства на одномерное пространство с координатой r_{xj}'', $1 \le j \le n$. Значения r_{xj}'' являются входными переменными декодера C_3. Проекция зависит от декодированного квадранта, который индексируется парой (v_{1j}^*, v_{2j}^*), $1 \le j \le n$, как показано в следующей ниже таблице:

v_{1j}^*	v_{2j}^*	r_{xj}''
0	0	$-\sqrt{2}/2(r_{xj} - r_{yj})$
0	1	$-\sqrt{2}/2(r_{xj} + r_{yj})$
1	1	$\sqrt{2}/2(r_{xj} - r_{yj})$
1	0	$\sqrt{2}/2(r_{xj} + r_{yj})$

Эта операция является масштабированным поворотом $\mathbf{r}$ на угол $\pi/4$. Таким образом, *повернутая последовательность* $\mathbf{r}'' = (r_{x1}'', r_{x2}'', \ldots, r_{xn}'')$ может быть декодирована любой процедурой декодирования с мягким решением для компонентного кода C_3. Заметим, что, в отличие от декомпозиции Унгербёка, независимость между первыми двумя уровнями блочного разбиения приводит к отсутствию *размножения ошибок* при переходе от первой ступени ко второй.

Обозначим $A_w^{(i)}$ число кодовых слов в коде C_i веса w. Предполагая систематическое кодирование, можем записать границу объединения для вероятности ошибки на бит первой ступени декодирования в следующем виде [MFLI, IFMLI]

$$P_{b1}^{(NS)} \le \sum_{w=d_1}^{n} \frac{w}{n} A_w^{(1)} 2^{-w} \sum_{i=0}^{w} \binom{w}{i} Q\left(\sqrt{\frac{2RE_b}{N_0} d_P^2(i)}\right) \qquad (9.9)$$

где $d_P^2(i) = [i\Delta_1 + (w - i)\Delta_2]^2/w$. Граница (9.9) справедлива и для вероятности ошибки на бит второй ступени декодирования с теми же аргументами.

Границу (9.9) можно сравнить с аналогичной границей для разбиений по схеме Унгербёка:

$$P_{b1}^{(UG)} \le \sum_{w=d_1}^{n} \frac{w}{n} A_w^{(1)} 2^w Q\left(\sqrt{\frac{2RE_b}{N_0} w\Delta_1^2}\right) \qquad (9.10)$$

Из границ (9.9) и (9.10) следует, что для схемы разбиений Унгербёка влияние ближайших соседей возрастает экспоненциально на множитель 2^w , тогда как для блочного разбиения и $d^2{}_P(i) = w\Delta_1{}^2$ коэффициент ошибок, 2^{-w}, *убывает экспоненциально* с расстоянием компонентного кода первого уровня. В результате, для практических значений отношения сигнал-шум E_b/N_0, блочное разбиение может дать на первой ступени реальный энергетический выигрыш от кодирования *даже превышающий асимптотическое значение*. Это является весьма желательным эффектом в схемах неравной защиты.

В схемах нестандартного разбиения второй уровень конструируется обычно так, чтобы иметь энергетический выигрыш больше, чем на третьем уровне. При этом предположении хорошую аппроксимацию дает предположение о том, что результаты декодирования на первом и втором уровнях правильны. Тогда получаем следующую оценку

$$P_{b3}^{(NS)} \underset{\approx}{\le} \sum_{w=d_3}^{n} \frac{w}{n} A_w^{(3)} Q\left(\sqrt{\frac{2RE_b}{N_0} w\Delta_1^2}\right) \qquad (9.11)$$

Пример 102. Рассмотрим трехуровневую систему на сигналах 8-КАМ для неравной защиты с расширенными кодами БЧХ (64, 18, 22), (64, 45, 8) и (64, 63, 2) в качестве кодов первой, второй и третьей ступеней, соответственно. Эта схема кодирования имеет скорость 1,97(бит/символ) и ее можно срав-

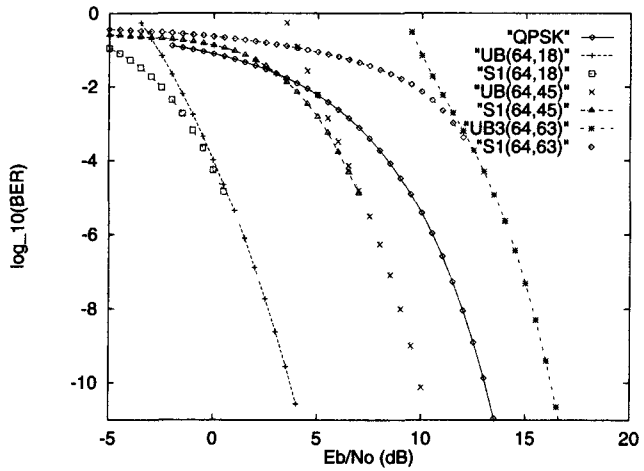


Рис. 122. Результаты моделирования трехуровневой кодовой модуляции на сигналах 8-ФМ с неравной защитой, компонентными кодами БЧХ и блочным разбиением.

нить с не кодированной передачей 4-ФМ сигналов (QPSK), которые дают примерно такую же скорость (разница составляет всего 0,06дБ). Результаты моделирования показаны на Рисунке 122. Метками S1(n, k) обозначены результаты моделирования, а UB(n, k) – верхние границы. Большой энергетический выигрыш 8,5дБ достигается при вероятности ошибки 10^{-5} для 18 наиболее защищенных двоичных символов (14,3%) на первом уровне. На втором и третьем уровнях соответствующие значения выигрыша составляют 2,5дБ и –4,0дБ[5].

9.4. Кодовая модуляция с побитовым перемешиванием (BICM)

В работах [CTB1, CTB2] представлена новейшая система практической кодовой модуляции. Эта система основана на двоич-

[3] Заметим, что при этом уровне BER полученный при моделировании энергетический выигрыш на первой ступени превышает асимптотический (8,1дБ) из-за меньших коэффициентов ошибок.

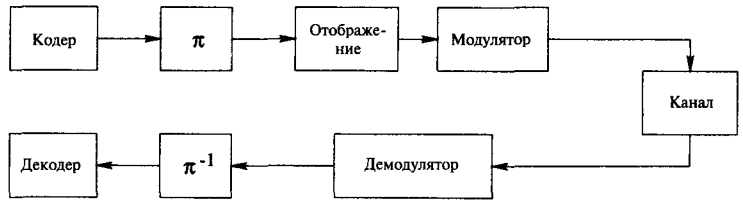


Рис. 123. Система кодовой модуляции с побитовым перемежением.

ном кодировании с последующим псевдослучайным *побитовым перемешиванием* (*перемежением*). Выход перемежителя группируется в блоки из ν бит, которыми через *отображение Грея* нумеруются точки 2^{ν}-ого модуляционного созвездия. Пропускная способность *кодовой модуляции с побитовым перемешиванием (BICM)* удивительно близка, как было показано, к пропускной способности TCM, использующей отображение Грея. Более того, в каналах с общими Релеевскими замираниями система BICM превосходит системы кодовой модуляции с посимвольным перемешиванием [CTB2]. Структурная схема этой системы показана на Рисунке 123.

Обозначим χ сигнальное созвездие 2^{ν} точек с минимальным расстоянием D_{min}. Пусть $\ell^i(x)$, $i = 1, 2, \ldots, \nu$, обозначает i-ый бит номера сигнальной точки x и пусть $\chi_b^i \subset \chi$ представляет подмножество сигнальных точек, номера (метки) которых на i-ой позиции имеют значение $b \in \{0, 1\}$.

9.4.1. Отображение Грея

Взаимно однозначное отображение множества двоичных векторов из $\{0, 1\}^{\nu}$ на множество точек χ является отображением Грея, если для всех $i = 1, 2, \ldots, \nu$, и $b \in \{0, 1\}$ каждый элемент $x \in \chi_b^i$ имеет самое большее одного ближайшего соседа $y \in \chi_{b'}^i$ на расстоянии D_{min}, где $b' = b \oplus 1$. Отображение Грея является ключевой частью системы BICM. Его главной задачей является, в идеале, создание эквивалентного канала, состоящего из ν параллельных, независимых, двоичных каналов без памяти.

Каждый двоичный канал соответствует некоторой позиции в номере сигнальной точки $x \in \chi$. Каждому кодовому слову на выходе двоичного кодера перемежитель приписывает случайно выбранную позицию метки сигнала для передачи кодовых символов.

9.4.2. Генерация метрик: обратное отображение

Перед описанием способа вычисления метрик для декодера по максимуму правдоподобия введем некоторые обозначения. Пусть r обозначает выход канала после передачи x. Предполагая равномерное распределение на входе, найдем условную вероятность r при условии, что $\ell^i(x) = b$,

$$p(r \mid \ell^i(x)=b)=\sum_{x\in\chi} p(r \mid x)p(x \mid \ell^i(x)=b)=2^{-(\nu-1)}\sum_{x\in\chi_b^i} p(r \mid x) \qquad (9.12)$$

Обозначим i позицию кодового символа v_j в метке сигнала $x_{\pi(j)}$. Для каждого момента j обозначим v_j кодовый символ и $x_{\pi(j)}$ сигнальную точку после перемежителя, которая была принята как $r_{\pi(j)}$ после передачи по каналу с шумом.

Приемник вычисляет *побитовые метрики*

$$\lambda^i(r_{\pi(j)},b)=\log\left(\sum_{x\in\chi_b^i} p(r_{\pi(j)} \mid x)\right) \qquad (9.13)$$

для $b = 0, 1$ и $i = 1, 2, \ldots, \nu$.

Декодер максимального правдоподобия, такой как алгоритм Витерби, использует введенные выше метрики и выбирает решение на основе следующего правила

$$v_i^* = \arg\max_{v\in C}\sum_j \lambda^i(r_{\pi(j)},v_j) \qquad (9.14)$$

Как и раньше возможна max-log аппроксимация (9.13), дающая приближенную битовую метрику,

$$\lambda_a^i(r_{\pi(j)},b)=\max_{x\in\chi_b^i}\log p(r_{\pi(j)} \mid x) \qquad (9.15)$$

9.4.3. Перемешивание

При передаче сигналов по каналу с АБГШ достаточно использовать короткий перемежитель. Его главная задача состоит в том, чтобы разбить корреляцию между битами, возникающую из-за модуляционного множества 2^ν сигнальных точек, ν бит на сигнал. Поэтому, для достижения хорошей эффективности, можно считать достаточным перемежитель, длина которого в несколько раз превышает ν [CTB2]. Заметим, что этот перемежитель не имеет ничего общего с перемежителем для турбо кода или произведения кодов.

9.5. Турбо кодовая модуляция на решетке (TTCM)

Существуют различные подходы к комбинированию турбо кодов или кодов-произведений с перемешиванием и цифровой модуляцией: практическая кодовая модуляция [LGB] , турбо кодовая модуляция на решетке с *посимвольным перемешиванием* [RW1, RW2], турбо кодовая модуляция на решетке с *побитовым перемешиванием* [BDMP2].

9.5.1. Практическая турбо TCM

Под впечатлением выдающейся эффективности турбо кодирования в 1994 в работе [LGB] была предложена другая практическая схема кодовой модуляции. Ее структурная схема показана на Рисунке 124. Главным элементом этой схемы является

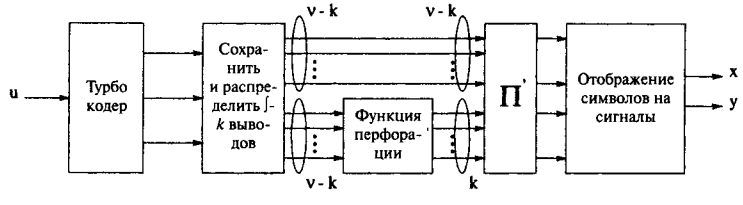


Рис. 124. Комбинация турбо кода и цифровой модуляции [LGB].

применение турбо кодирования и декодирования двоичных последовательностей, как и в других схемах практической TCM. Это обстоятельство требует внимательного отношения к вычислению *побитовых метрик*, как и в случае BICM.

9.5.2. Турбо TCM с посимвольным перемешиванием

В 1995 Робертсон и Фёрц (Robertson & Wörz) предложили использовать рекурсивные систематические сверточные коды, те же, что ранее предлагал использовать Унгербёк [Ung1], в качестве компонентных кодов в системе аналогичной турбо кодированию. Структурная схема этой системы показана на Рисунке 125. Из этой схемы видно, что перемешивание выполняется над

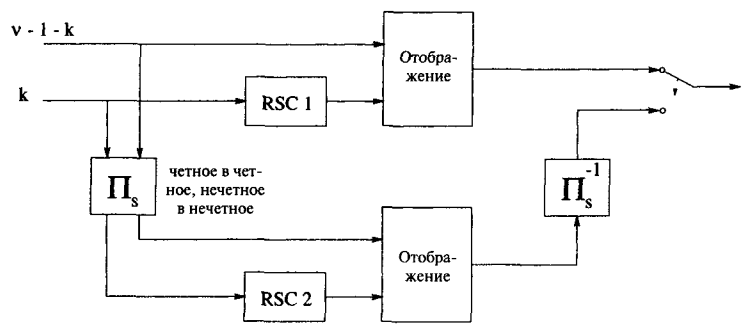


Рис. 125. Структура кодера турбо TCM с посимвольным перемешиванием.

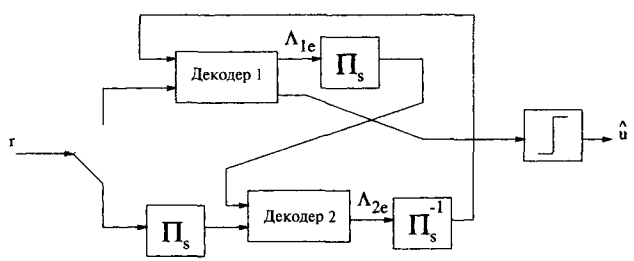


Рис. 126. Итеративный декодер для турбо TCM с посимвольным перемешиванием.

символами из v бит в отличие от перемешивания двоичных символов в двоичных турбо кодах. Здесь необходимо перфорировать *избыточные символы*, в связи с тем, что к каждой модуляционной точке идут два пути. Кроме того, требуется тщательный выбор компонентных кодов и конструкции перемежителя. Код не должен иметь параллельных переходов, а перемешивание должно выполняться так, что, либо четная позиция перемещается в четную и нечетная в нечетную, либо четная в нечетную и нечетная в четную. Относительно итеративного декодирования следует заметить, что здесь невозможно отделить систематические компоненты и внешние (поступающие извне), так как и те и другие передаются в одном символе. Тем не менее, вычисление LLR может быть разделено на априорную часть и систематическую-и-внешнюю часть. Следует позаботиться также и о том, чтобы эта информация использовалась компонентными декодерами не более одного раза. По этим причинам на выходе кодера появилось устройство в виде селектора для удаления избыточных символов [RW2]. На Рисунке 126 показана структурная схема итеративного декодера для турбо TCM.

9.5.3. Турбо TCM с побитовым перемешиванием

В 1996 Бенедетто с соавторами [BDMP2] предложили правило перфорации символов, которое обеспечивает появление информационного бита на выход кодера только один раз. Более того, в отличие от посимвольного перемешивания с перфорацией избыточных символов, они предложили использовать много *побитовых перемежителей*. Структурная схема кодера показана на Рисунке 127 для случая двух компонентных кодов.

MAP декодирование и битовые метрики

Структура декодера для системы турбо TCM с побитовым перемешиванием аналогична структуре декодера двоичного турбо кода. Главное отличие состоит в преобразовании величин LLR (логарифм отношения правдоподобия) из побитовых в посимвольные и, наоборот, из посимвольных в побитовые,

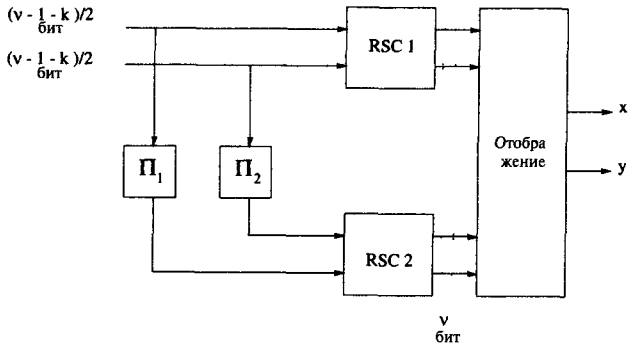


Рис. 127. Кодер для турбо TCM с побитовым перемешиванием.

которое должно выполняться между декодерами в итеративной процедуре [BDMP2, Vuc]. Для декодирования турбо TCM конструкций с побитовым перемешиванием величины LLR, вычисленные для отдельных битов, должны быть преобразованы в посимвольные априорные вероятности. И наоборот, априорные вероятности для символов должны быть преобразованы в побитовые внешние значения LLR.

Это делается следующим образом. Обозначим Ψ множество 2^{ν} сигнальных точек. Для символа $x(\mathbf{b}) \in \Psi$ с меткой $\mathbf{b} = (b_1, b_2, \ldots, b_{\nu})$ внешняя информация для символа b_i, $i = 1, 2, \ldots, \nu$, вычисляется по следующей формуле

$$\Lambda_e(b_i) = \log\left(\frac{\sum_{x(\mathbf{b}),b_i=1} e^{\Lambda_e(x(\mathbf{b}))}}{\sum_{x(\mathbf{b}),b_i=0} e^{\Lambda_e(x(\mathbf{b}))}}\right) \qquad (9.16)$$

Аналогично, априорная вероятность символа может быть вычислена из побитовых значений LLR с помощью следующего выражения

$$\Pr(\mathbf{b} = (b_1, b_2, \cdots, b_{\nu})) = \prod_{i=1}^{\nu} \frac{e^{b_i \Lambda_e(b_i)}}{1 + e^{\Lambda_e(b_i)}} \qquad (9.17)$$

Литература

[BCJR] L R Bahl, J Cocke, F Jelinek and J Raviv, «Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate, « *IEEE Trans Info Theory,* vol IT-20, pp 284-287, Mar 1974

[BP] A S Barbulescu and S S Pietrobon, «Interleaver Design for Turbo Codes, « *Elect Letters,* vol 30, no 25, pp 2107-2108, Dec 1994

[Bat] G Battail, «A Conceptual Framework for Understanding Turbo Codes, « *IEEEJ Sel Areas in Comm* , vol 16, no 2, pp 245-254, Feb 1998

[BB] S Benedetto and E Biglieri, *Principles of Digital Transmission,* Kluwer Academic/Plenum Publishers, 1999

[BM] S Benedetto and G Montorsi, «Unveiling Turbo Codes Some Results on Parallel Concatenated Coding Schenes, « *IEEE Trans Info Theory,* vol 42, no 2, pp 409-428, March 1996

[BDMP1] S Benedetto, D Divsalar, G Montorsi and F Pollara, «Serial Concatenation of Interleaved Codes Performance Analysis, Design, and Iterative Decoding, « *IEEE Trans Info Theory,* vol 44, no 3, pp 909-926, May 1998

[BDMP2] S Benedetto, D Divsalar, G Montorsi and F Pollara, «Parallel Concatenated Trellis Coded Modulation, « *Proc 1996 IEEE Int Conf Comm (ICC'96),* pp 974-978, 1996

[Berl] E R Berlekamp, *Algebraic Coding Theory,* rev ed , Aegean Park Press, 1984

[Ber2] E R Berlekamp, «Bounded Distance + 1 Soft-Decision Reed-Solomon Decoding,» *IEEE Trans Info Theory,* vol 42, no 3, pp 704-720, May 1996

[BG1] C Berrou and A Glavieux, «Reflections on the Prize Paper 'Near Optimum Error-Correcting Coding and Decoding Turbo Codes', « *IEEE Info Theory Soc News* , vol 48, no 2, June 1998

[BG2] C Berrou and A Glavieux, «Near Optimum Error Correcting Coding and Decoding,» *IEEE Trans Comm* , vol 44, no 10, pp 1261-1271, Oct 1996

[BGT] C Berrou, A Glavieux and P Thitimajshima, «Near Shannon Limit Error-Correcting Coding and Decoding Turbo-Codes,» Proc *1993 IEEE Int Conf Comm (ICC'93), pp* 1064-1070, Geneve, Switzerland, May 1993

[BDMS] E Biglieri, D Divsalar, P J McLane and M K Simon, *Introduction to Trellis-Coded Modulation with Applications,* Macmillan Publishing, 1991

[Blah] R E Blahut, *Theory and Practice of Error Control Codes,* Addison-Wesley, 1984

[BL] M Blaum, «A Family of Efficient Burst-Correcting Array Codes,» *IEEE Trans Info Theory,* vol 36, no 3, pp 671-675, May 1990

[BZ] E L Blokh and V V Zyablov, «Coding of Generalized Concatenated Codes,» *Probl Pered Informatsii,* vol 10, no 1, pp 45-50, 1974

[BABSZ] M Boo, F Arguello, J D Bruguera, R Doallo and E L Zapata, «High-Performance VLSI Architecture for the Viterbi Algorithm, «*IEEE Trans Comm* , vol 45, no 2, pp 168-176, Feb 1997

[Bos] M Bossert, *Channel Coding for Telecommunications,* John Wiley & Sons, 1999

[BH] M Breiling and J B Huber, «Combinatorial Analysis of the Minimum Distance of Turbo Codes, «*IEEE Trans Info Theory,* vol 47, no 7, pp 2737-2750, Nov 2001

[Bro] A E Brouwer and T Verhoeff, «An Updated Table of Minimum-Distance Bounds for Binary Linear Codes, «*IEEE Trans Info Theory,* vol 39, no 2, pp 662-677, Mar 1993

[BW] H O Burton and E J Weldon, Jr, «Cyclic Product Codes, «*IEEE Trans Info Theory,* vol IT-11, no 3, pp 433-439, July 1965

[CCG] J B Cain, G C Clark and J M Geist, «Punctured Convolutional Codes of Rate $(n-1)/n$ and Simplified Maximum Likelihood Decoding, «*IEEE Trans Info Theory,* vol IT-25, pp 97-100, Jan 1979

[CTB1] G Caire, G Taricco and E Biglieri, «Capacity of Bit-Interleaved Channels, « *Elect Letters,* vol 32, pp 1060-1061, June 1996

[CTB2] G Caire, G Taricco and E Biglieri, «Bit-Interleaver Coded Modulation, « *IEEE Trans Info Theory,* vol 44, no 3, pp 927-946, May 1998

[CRKO] F Garden, M D Ross, B T Kopp and W P Osborn, «Fast TCM Decoding Phase Quantization and Integer Weighting, « *IEEE Trans Comm*, vol 42, no 4, pp 808-812, Apr 1994

[CY] C -C Chao and Y -L Yao, «Hidden Markov Models for the Burst Error Statistics of Viterbi Decoding, « *IEEE Trans Comm*, vol 44, no 12, pp 1620-1622, Dec 1996

[Cha] D Chase, «A Class of Algorithms for Decoding Block Codes with Channel Measurement Information, « *IEEE Trans Info Theory,* vol IT-18, pp 170-182, 1972

[CK] P Chaudhari and A K Khandani, «Using the Fourier Transform to Compute the Weight Distribution of a Binary Linear Block Code, « *IEEE Comm Let*, vol 5, no 1, pp 22-24, Jan 2001

[CFRU] S -Y Chung, G D Forney, Jr, T J Richardson and R Urbanke, «On the Design of Low-Density Parity-Check Codes within 0 0045 dB of the Shannon Limit, « *IEEE Comm Let,* vol 5, no 2, pp 58-60, Feb 2001

[Cla] G Clark and J Cain, *Error-Correction Coding for Digital Communications,* Plenum Press, 1981

[Col] O M Collins, «The Subtleties and Intricacies of Building a Constraint Length 15 Convolutional Decoder, « *IEEE Trans Comm*, vol 40, no 12, pp 1810-1819, Nov 1992

[CoT] T M Cover and J A Thomas, *Elements of Information Theory,* John Wiley & Sons, 1991

[DFK1] Y Desaki, T Fujiwara and T Kasami, «A Method for Computing the Weight Distribution of a Block Code by Using Its Trellis Diagram, « *IEICE Tran Fundamentals,* vol E77-A, pp 1230-1237, Aug 1994

[DFK2] Y Desaki, T Fujiwara and T Kasami, «The Weight Distribution of Extended Binary Primitive BCH Codes of Length 128, « *IEEE Trans Info Theory,* vol 43, no 4, pp 1364-1371, July 1997

[DYS] U Dettmar, G Yan and U K Sorger, «Modified Generalized Concatenated Codes and Their Application to the Construction and Decoding of LUEP Codes, « *IEEE Trans Info Theory,* vol 41, no 5, pp 1499-1503, Sept 1995

[Dho] A Dholakia, *Introduction to Convolutional Codes With Applications,* Kluwer, 1994

[DDP] D Divsalar, S Dolinar and F Pollara, «Iterative Turbo Decoder Analysis Based on Density Evolution, « *IEEE J Sel Areas in Comm*, vol 19, no 5, pp 891-907, May 2001

[DJM] D Divsalar, H Jin and R J McEliece, «Coding Theorems for Turbo-Like Codes, «*Proc 1998 Allerton Conf on Commun, Control, and Computing,* pp 210-219, Univ Illinois, Urbana-Champaign, 1998

[Div] D Divsalar and F Pollara, «Turbo Codes for Deep-Space Communications, « *JPL TDA Progress Report 42-120,* pp 29-39, Feb 1995

[DP] D Divsalar and F Pollara, «Turbo Codes for PCS Applications, «*Proc 1993 IEEE Int Conf Comm (ICC'93),* pp 1064-1070, Geneva, Switzerland, May 1993

[DVB] EN 310 210, ETSI, «Digital Video Broadcasting (DVB) Framing Structure, Channel Coding and Modulation for Digital Satellite News Gathering (DSNG) and Other Contribution Applications by Satellite, « Mar 1997

[EH] H El Gamal and A R Hammons, Jr, «Analyzing the Turbo Decoder Using the Gaussian Approximation, «*IEEE Trans Info Theory,* vol 47, no 2, pp 671-686, Feb 2001

[Eli1] P Elias, «Error-Free Coding, « *IRE Trans*, vol PGIT-4, pp 29-37, 1954

[Eli2] P Elias, «Coding for Noisy Channels, « *IRE Com Rec*, vol 3, pt 4, pp 37-46, 1955 Also in E R Berlekamp, ed, *Key Papers in the Development of Coding Theory,* pp 48-55, IEEE Press, 1974

[Eli3] P Elias, «Error-Correcting Codes for List Decoding, « *IEEE Trans Info Theory,* vol 37, no 1, pp 5-12, Jan 1991

[FV] W Feng and B Vucetic, «A List Bidirectional Soft Output Decoder of Turbo Codes, « *Proc Int Symp Turbo Codes and Rel Top*, pp 288-292, Brest, France, Sept 3-5, 1997

[For1] G D Forney, Jr, *Concatenated Codes,* MIT Press Reseach Monograph 37, 1966

[For2] G, D, Forney, Jr, «On Decoding BCH Codes, « *IEEE Trans Info Theory,* vol IT-11, pp 393-403, Oct 1965 Also in E R Berlekamp, ed, *Key Papers in the Development of Coding Theory,* pp 149-155, IEEE Press, 1974

[For3] G D Forney, «Generalized Minimum Distance Decoding, « *IEEE Trans Info Theory,* vol IT-12, pp 125-131, April 1966

[For4] G D Forney, «Convolutional Codes I Algebraic Structure, « *IEEE Trans Info Theory,* vol IT-16, no 6, pp 720-738, Nov 1970

[For5] G D Forney, Jr, «Burst Correcting Codes for the Classic Bursty Channel, « *IEEE Trans Comm Tech*, vol COM-19, pp 772-281, 1971

[For6] G D Forney, Jr, «Coset Codes II Binary Lattices and Related Codes, « *IEEE Trans Info Theory,* vol 24, no 5, pp 1152-1187, Sept 1988

[For7] G D Forney, «Codes on Graphs Normal Realizations, « *IEEE Trans Info Theory,* vol 47, no 2, pp 520-548, Feb 2001

[For8] G D Forney and G Ungerboeck, «Modulation and Coding for Linear Gaussian Channels, « *IEEE Trans Info Theory, Special Commemorative Issue,* vol 44, no 6, pp 2384-2415, Oct 1998

[FBLH] M P C Fossorier, F Burkert, S Lin and J Hagenauer, «On the Equivalence Between SOVA and Max-Log-MAP Decodings, « *IEEE Comm Letters,* vol 2, no 5, pp 137-139, May 1998

[FLC] M P C Fossorier, S Lin and D J Costello, Jr, «On the Weight Distribution of Terminated Convolutional Codes, « *IEEE Trans Info Theory,* vol 45, no 5, pp 1646-1648, July 1999

[FL1] M P C Fossorier and S Lin, «Soft-Decision Decoding on Linear Block Codes Based on Ordered Statistics, « *IEEE Trans Info Theory,* vol 41, no 5, pp 1379-1396, Sept 1995

[FL2] M P C Fossorier and S Lin, «Differential Trellis Decoding of Convolutional Codes, «*IEEE Trans Info Theory,* vol 46, no 3, pp 1046-1053, May 2000

[FL3] M P C Fossorier and S Lin, «Some Decomposable Codes The $|a + x|b + x|a + b + x|$ Construction, « *IEEE Trans Info Theory,* vol 43, no 5, pp 1663-1667, Sept 1997

[FL4] M P C Fossorier and S Lin, «Complementary Reliability-Based Soft-Decision Decoding of Linear Block Codes Based on Ordered Statistics, « *IEEE Trans Info Theory,* vol 42, no 5, pp 1667-1672, Sep 1997

[FL5] M P C Fossorier and S Lin, «Soft-Input Soft-Output Decoding of Linear Block Codes Based on Ordered Statistics, « *Proc 1998 IEEE Global Telecomm Conf (GLOBECOM'98),* pp 2828-2833, Sydney, Australia, Nov 1998

[FLR] M P C Fossorier, S Lin and D Rhee, «Bit-Error Probability for Maximum-Likelihood Decoding of Linear Block Codes and Related Soft-Decision Decoding Methods, « *IEEE Trans Info Theory,* vol 44, no 7, pp 3083-3090, Nov 1998

[FMH] M P C Fossorier, M Mihaljevic and H Imai, «Reduced Complexity Iterative Decoding of Low-Density Parity Check Codes Based on Belief Propagation, « *IEEE Trans Comm*, vol 47, no 5, pp 673-680, May 1999

[FK] B J Frey and F R Kschischang, «Early Detection and Trellis Splicing Reduced-Complexity Iterative Decoding, « *IEEE J Sel Areas in Comm*, vol 16, no 2, pp 153-159, Feb 1998

[FKKL] T Fujiwara, T Kasami, A Kitai and S Lin, «On the Undetected Error Probability of Shortened Hamming Codes, « *IEEE Trans Comm*, vol COM-33, pp 570-574, June 1985

[Gal] R G Gallager, «Low-Density Parity-Check Codes, « *IRE Trans Info Theory,* vol 8, no 1, pp 21-28, Jan 1962

[GPB] R Garello, P Perleoni and S Benedetto, «Computing the Free Distance of Turbo Codes and Serially Concatenated Codes with Interleavers Algorithms and Applications, « *IEEE J Sel Areas in Comm*, vol 19, no 5, pp 800-812, May 2001

[GG] J von zur Gathen and J Gerhard, *Modern Computer Algebra,* Cambridge University Press, 1999

[Gol] M J E Golay, «Notes on Digital Coding» *Proc IRE*, vol 37, p 657, June 1949 Also in *Key Papers in the Development of Coding Theory,* E R Berlekamp, ed, p 13, IEEE Press, 1974

[Glc] J D Golic, «Iterative Optimum Symbol-by-Symbol Decoding and Fast Correlation Attacks, « *IEEE Trans Info Theory,* vol 47, no 7, pp 3040-3049, Nov 2001

[GZ] D Gorenstein and N Zierler, «A Class of Error Correcting Codes in p^m Symbols, « *J SIAM,* vol 9, pp 207-214, June 1961

[Gur] V Guruswami and M Sudan, «Improved Decoding of Reed-Solomon and Algebraic-Geometry Codes, « *IEEE Trans Info Theory,* vol 45, no 6, pp 1757-1767, Sep 1999

[Hag] J Hagenauer, «Rate-Compatible Punctured Convolutional Codes (RCPC Codes) and their Applications, « *IEEE Trans Comm*, vol 36, no 4, pp 389-400, April 1988

[HH] J Hagenauer and P Hoher, «A Viterbi Algorithm with Soft-Decision Outputs and Its Applications, « *Proc 1989 IEEE Global Telecomm Conf (GLOBECOM'89),* pp 47 1 1-47 1 7, Dallas, Texas, 1989

[HOP] J Hagenauer, E Offer and L Papke, «Iterative Decoding of Binary Block and Convolutional Codes, « *IEEE Trans Info Theory,* vol 42, no 2, pp 429-445, Mar 1996

[HaW] E K Hall and S G Wilson, «Stream-Oriented Turbo Codes, « *IEEE Trans Info Theory,* vol 47, no 5, pp 1813-1831, July 2001

[Ham] R W Hamming, «Error Detecting and Error Correcting Codes, « *Bell Syst Tech J* , vol 29, pp 147-160, 1950 Also in *Key Papers in the Development of Coding Theory,* E R Berlekamp, ed , pp 9-12, IEEE Press, 1974

[Hay] S Haykin, *Digital Communications,* John Wiley and Sons, 1988

[HeW] C Heegard and S B Wicker, *Turbo Coding,* Kluwer Academic Press, 1999

[HEK] A P Hekstra, «An Alternative to Metric Rescaling in Viterbi Decoder, « *IEEE Trans Comm* , vol 37, no 11, pp 1220-1222, Nov 1989

[Her] I N Herstein, *Topics in Algebra,* 2nd ed , John Wiley and Sons, 1975

[HEM] J Hokfelt, O Edfors and T Maseng, «A Turbo Code Interleaver Design Criterion Based on the Performance of Iterative Decoding, « *IEEE Comm Let* , vol 5, no 2, pp 52-54, Feb 2001

[HM] B Honay and G S Markarian, *Trellis Decoding of Block Codes A Practical Approach,* Kluwer, 1996

[HMF] B Honary, G S Markarian and P G Farell, «Generalised Array Codes and Their Trellis Structure, « *Elect Letters,* vol 28, no 6, pp 541-542, Mar 1993

[Hsia] M Y Hsiao, «A Class of Optimal Minimum Odd-Weight-Column SEC-DED Codes, « *IBM J Res and Dev* , vol 14, July 1970

[IH] H Imai and S Hirakawa, «A New Multilevel Coding Method Using Error-Correcting Codes, « *IEEE Trans Info Theory,* vol IT-23, no 3, pp 371-377, May 1977

[IFMLI] M Isaka, M P C Fossorier, R H Morelos-Zaragoza, S Lin and H Imai, «Multilevel Coded Modulation for Unequal Error Protection and Multistage Decoding Part II Asymmetric Constellations, « *IEEE Trans Comm* , vol 48, no 5, pp 774-784, May 2000

[ISTC1] *Proceedings of the International Symposium on Turbo Codes and Related Topics,* Brest, France, September 3-5, 1997

[ISTC2] *Proceedings of the Second International Symposium on Turbo Codes and Related Topics,* Brest, France, September 4-7, 2000

[ITCG] Special Issue on Codes and Graphs and Iterative Decoding Algorithms, *IEEE Trans Info Theory,* vol 47, no 2, Feb 2001

[Jer] M C Jeruchim, P Balaban and K S Shanmugan, *Simulation of Communication Systems,* Plenum Press, 1992

[Joh] R Johannesson and K S Zigangirov, *Fundamentals of Convolutional Coding,* IEEE Press, 1999

[JSAC1] Special Issue on the turbo principle from theory to practice Part I, *IEEE J Sel Areas in Comm* , vol 19, no 5, May 2001

[JSAC2] Special Issue on the turbo principle from theory to practice Part II, *IEEEJ Sel Areas in Comm* , vol 19, no 9, Sept 2001

[Kam] N Kamiya, «On Algebraic Soft-Decision Decoding Algorithms for BCH Codes, « *IEEE Trans Info Theory,* vol 47, no 1, pp 45-58, Jan 2001

[KNIH] T Kaneko, T Nishijima, H Inazumi and S Hirasawa, «An Efficient Maximum-, Likelihood Decoding Algorithm for Linear Block Codes with Algebraic Decoder, « *IEEE Trans Info Theory,* vol 40, no 2, pp 320-327, Mar 1994

[Kasl] M Kasahara, Y Sugiyama, S Hirasawa and T Namekawa, «A New Class of Binary Codes Constructed on the Basis of BCH Codes, « *IEEE Trans Info Theory,* vol IT-21, pp 582-585, 1975

[Kas2] M Kasahara, Y Sugiyama, S Hirasawa and T Namekawa, «New Classes of Binary Codes Constructed on the Basis of Concatenated and Product Codes, « *IEEE Trans Info Theory,* vol IT-22, no 4, pp 462-468, July 1976

[Kasm] T Kasami, «A Decoding Procedure for Multiple-Error-Correcting Cyclic Codes, « *IEEE Trans Info Theory,* vol IT-10, pp 134-138, 1964

[KLP] T Kasami, S Lin and W W Peterson, «Polynomial Codes, « *IEEE Trans Info Theory,* vol IT-14, no 6, pp 807-814, Nov 1968

[KKTFL] T Kasami, T Koumoto, T Takata, T Fujiwara and S Lin, «The Least Stringent Sufficient Condition on the Optimality of Suboptimally Decoded Codewords, « *Proc 1995 IEEE Int Symp Info Theory (ISIT'95),* p 470, Whistler, Canada, 1995

[Kaz] P Kazakov, «Fast Calculation of the Number of Minimum-Weight Words of CRC Codes, «*IEEE Trans Info Theory,* vol 47, no 3, pp 1190-1195, March 2001

[KLF] Y Kuo, S Lin and M P C Fossorier, «Low-Density Parity-Check Codes Based on Finite Geometries A Rediscovery and New Results, « *IEEE Trans Info Theory,* vol 47, no 7, pp 2711-2736, Nov 2001

[KFL] R R Kschischang, B J Frey and H -A Loeliger, «Factor Graphs and the Sum-Product Algorithm, « *IEEE Trans Info Theory,* vol 47, no 2, pp 498-519, Feb 2001

[KV] R Koetter and A Vardy, «Algebraic Soft-Decision Decoding of Reed-Somon Codes, «*Proc 2000 IEEE Int Symp Info Theory (ISIT'OO),* p 61, Sorrento, Italy, June 25-30, 2000

[Lee] L H C Lee, *Convolutional Coding Fundamentals and Applications,* Artech House, 1997

[LGB] S Le Goff, A Glavieux and C Berrou, «Turbo-Codes and High-Spectral Efficiency Modulation, « *Proc 1994 IEEE Int Conf Comm (ICC'94),* pp 645-649, 1994

[LC] S Lin and D J Costello, Jr , *Error Control Coding Fundamentals and Applications,* Prentice-Hall, 1983

[LKFF] S Lin, T Kasami, T Fujiwara and M Fossorier, *Trellises and Trellis-Based Decoding Algorithms for Linear Block Codes,* Kluwer Academic Press, 1998

[LYHH] J Lodge, R Young, P Hoeher and J Hagenauer, «Separable MAP 'Filters' for the Decoding of Product and Concatenated Codes, « *Proc 1993 IEEE Int Conf Comm (ICC'93),* pp 1740-1745, May 1993

[Mac] D J C MacKay, «Good Error-Correcting Codes Based on Very Sparse Matrices, « *IEEE Trans Info Theory,* vol 45, no 2, pp 399-432, Mar 1999

[MN] D J C MacKay and R M Neal, «Near Shannon Limit Performance of Low Density Parity Check Codes, « *Electronics Letters,* vol 32, pp 1645-1646, 1996

[MS] F J MacWilliams and N J A. Sloane, *The Theory of Error Correcting Codes,* North-Holland, 1977

[Man] D Mandelbaum, «Decoding Beyond the Designed Distance of Certain Algebraic Codes, «*Info and Control,* vol 35, pp 209-228, 1977

[MT] P A Martin and D P Taylor, «On Adaptive Reduced-Complexity Iterative Decoding, «*Proc 2000 IEEE Global Telecomm Conf (GLOBECOM'OO),* pp 772 -776, 2000

[Masl] J L Massey, *Threshold Decoding,* MIT Press, 1963

[Mas2] J L Massey, «Shift Register Synthesis and BCH Decoding, « *IEEE Trans Info Theory,* vol IT-15, no 1, pp 122-127, Jan 1969

[Mas3] J L Massey, «Coding and Modulation in Digital Communications, « *Proc Int Zurich Seminar on Dig Comm* , pp E2(1)-E2(4), Zurich, Switzerland, 1974

[Mas4] J L Massey, «The How and Why of Channel Coding, « *Proc Int Zurich Seminar on Dig Comm* , pp 67-73, Zurich, Switzerland, 1984

[MW] B Masnick and J Wolf, «On Linear Unequal Error Protection Codes, « *IEEE Trans Info Theory,* vol IT-13, no 4, pp 600-607, July 1967

[McE] R J McEliece, *The Theory of Information and Coding,* Addison-Wesley, 1977

[MMC] R J McEliece, D J C MacKay and J -F Cheng, «Turbo Decoding as an Instance of Pearls' 'Belief Propagation' Algorithm, « *IEEE J Sel Areas in Comm* , vol 16, no 2, pp 140-152, Feb 1998

[Meg] J E Meggit, «Error Correcting Codes for Correcting Bursts of Errors, « *IBM J Research Develop* , no 4, pp 329-334, 1960

[ML] A M Michelson and A H Levesque, *Error-Control Techniques for Digital Communications,* John Wiley and Sons, 1985

[MG] M J Mihaljevic and J D Golic, «A Comparison of Cryptoanalitic Principles Based on Iterative Error Correction, « *Advances in Cryptology — EUROCRYPT'91 (Lecture Note in Computer Science),* vol 547, pp 527-531, Springer-Verlag, 1991

[MFKL] R H Morelos-Zaragoza, T Fujiwara, T Kasami and S Lin, «Constructions of Generalized Concatenated Codes and Their Trellis-Based Decoding Complexity, « *IEEE Trans Info Theory,* vol 45, no 2, pp 725-731, Mar 1999

[MI] R H Morelos-Zaragoza and H Imai, «Binary Multilevel Convolutional Codes with Unequal Error Protection Capabilities, « *IEEE Trans Comm*, vol 46, no 7, pp 850-853, July 1998

[MM] R H Morelos-Zaragoza and A Mogre, «A Two-Stage Decoder for Pragmatic Trellis-Coded *M*-PSK Modulation Using a Symbol Transformation, « *IEEE Trans Comm*, vol 49, no 9, pp 1501-1505, Sept 2001

[MFLI] R H Morelos-Zaragoza, M P C Fossorier, S Lin and H Imai, «Multilevel Coded Modulation for Unequal Error Protection and Multistage Decoding Part I Symmetric Constellations, « *IEEE Trans Comm*, vol 48, no 2, pp 204-213, Feb 2000

[Mor] J M Morris, «Burst Error Statistics of Simulated Viterbi Decoded BPSK on Fading and Scintillating Channels, « *IEEE Trans Comm*, vol 40, no 1, pp 34-41, Jan 1992

[OCC] I M Onyszchuk, K -M Cheung and O Collins, «Quantization Loss in Convolutional Decoding, « *IEEE Trans Comm*, vol 41, no 2, pp 261-265, Feb 1993

[Prl] J Pearl, *Probabilistic Reasoning in Intelligent Systems Networks of Plausible Inference*, Morgan Kaufmann, 1988

[PSC] L C Perez, J Seghers and D J Costello, Jr, «A Distance Spectrum Interpretation of Turbo Codes, « *IEEE Trans Info Theory*, vol 42, no 6, pp 1698-1709, Nov 1996

[PW] W W Peterson and E J Weldon, Jr, *Error Correcting Codes*, 2nd ed, MIT Press, 1972

[Pet] W W Peterson, «Encoding and Error-Correction Procedures for the Bode-ChaudhuriCodes, « *IRE Trans Info Theory*, vol IT-6, pp 459-470, Sept 1960 Also in E R Berlekamp, ed, *Key Papers in the Development of Coding Theory*, pp 109-120, IEEE Press, 1974

[PP] A Picart and R M Pyndiah, «Adapted Iterative Decoding of Product Codes, «*Proc 1995 IEEE Global Telecomm Conf (GLOBECOM'95)*, pp 2357-2362, 1995

[Plo] M Plotkin, «Binary Codes with Specified Minimum Distances, «*IEEE Trans Info Theory*, vol IT-6, pp 445-450, 1960

[Pre] O Pretzel, *Codes and Algebraic Curves*, Oxford Lecture Series in Mathematics and Its Applications, 8, 1998

[Pro] J G Proakis, *Digital Communications*, 3rd ed, McGraw-Hill, 1995

[PS] M B Pursley and J M Shea, «Bit-by-Bit Soft-Decision Decoding of Trellis-Coded *M*-DPSK Modulation, « *IEEE Comm Letters*, vol 1, no 5, pp 133-135, Sept 1997

[Pyn] R M Pyndiah, «Near-Optimum Decoding of Product Codes Block Turbo Codes, « *IEEE Trans Comm*, vol 46, no 8, pp 1003-1010, Aug 1998

[PGPJ] R M Pyndiah, A Glavieux, A Picart and S Jacq, «Near Optimum Decoding of Product Codes, « *Proc 1994 IEEE Global Telecomm Conf (GLOBECOM'94)*, vol 1, pp 339-343, San Francisco, CA, Dec 1994

[Ram] J L Ramsey, «Realization of Optimum Interleavers, « *IEEE Trans Info Theory*, vol IT-16, no 3, pp 338-345, May 1970

[RR] S M Reddy and J P Robinson, «Random Error and Burst Correction by Iterated Codes, «*IEEE Trans Info Theory*, vol IT-18, no 1, pp 182-185, Jan 1972

[RC] I S Reed and X *Chen, Error-Control Coding for Data Networks*, Kluwer Academic Press, 1999

[RS] I Reed and G Solomon, «Polynomial Codes over Certain Finite Fields, « *SIAM J Appl Math*, Vol 8, pp 300-304, 1960 Also in E R Berlekamp, ed, *Key Papers in the Development of Coding Theory*, pp 70-71, IEEE Press, 1974

[RSU] T J Richardson, M A Shokrollahi and R L Urbanke, «Design of Capacity-Approaching Irregular Low-Density Parity-Check Codes, « *IEEE Trans Info Theory*, vol 47, no 2, pp 619-637, Fe 2001

[RU] T J Richardson and R L Urbanke, «The Capacity of Low-Density Parity-Check Codes Under Message-Passing Decoding, « *IEEE Trans Info Theory*, vol 47, no 2, pp 599-618, Feb 2001

[Rob] P Robertson, «Improving Decoder and Code Structure of Parallel Concatenated Recursive Systematic (Turbo) Codes, « *Proc IEEE Int Conf Univ Per Comm (ICUPC'94)*, pp 183-187, 1994

[RVH] P Robertson, E Villebrun and P Hoeher, «A Comparison of Optimal and Sub-Optimal MAP Decoding Algorithms Operating in the Log Domain, «*Proc 1995 IEEE Int Conf Comm (ICC'95)*, pp 1009-1013, 1995

[RW1] P Robertson and T Worz, «Coded Modulation Scheme Employing Turbo Codes, «*Electronics Letters*, vol 31, pp 1546-1547, Aug 1995

[RW2] P Robertson and T Worz, «Bandwidth-Efficient Turbo Trellis-Coded Modulation Using Punctured Component Codes, « *IEEE J Sel Areas in Comm*, vol 16, no 2, pp 206-218, Feb 1998

[SSSN] H R Sdajadpour, N J A Sloane, M Salehi and G Nebe, «Interleaver Design for Turbo Codes, « *IEEE] Sel Areas in Comm*, vol 19, no 5, pp 831-837, May 2001

[SB] G Schnabl and M Bossert, «Soft-Decision Decoding of Reed-Muller Codes as Generalized Multiple Concatenated Codes, « *IEEE Trans Info Theory*, vol 41, no 1, pp 304-308, Jan 1995

[Schr] P Schramm, «Multilevel Coding with Independent Decoding on Levels for Efficient Communication on Static and Interleaved Fading Channels, « *Proc IEEE 1997 Int Symp Per, Ind and Mob Radio Comm (PIMRC'97)*, pp 1196-1200, 1997

[Sha] C Shannon, «A Mathematical Theory of Communication, « *Bell Syst Tech J*, vol 27, pp 379-423 and 623-656, 1948 Also in *Key Papers in the Development of Information Theory*, D Slepian, ed, pp 5-29, IEEE Press, 1974

[SLF] R Shao, S Lin and M P C Fossorier, «Two Simple Stopping Criteria for Turbo Decoding, « *IEEE Trans Comm*, vol 47, no 8, pp 1117-1120, Aug 1999

[Sin] R C Singleton, «Maximum Distance *q-nary* Codes, « *IEEE Trans Info Theory*, vol IT-10, pp 116-118, 1964

[SS] M Sipser and D A Spielman, «Expander Codes, « *IEEE Trans Info Theory*, vol 42, no 6, pp 1710-1722, Nov 1996

[Sle] D Slepian, «A Class of Binary Signaling Alphabets, « *Bell Sys Tech J*, vol 35, pp 203-234, Jan 1956

[Slo] N J A Sloane, S M Reddy and C -L Chen, «New Binary Codes, « *IEEE Trans Info Theory*, vol IT-18, no 4, pp 503-510, July 1972

[Sud] M Sudan, «Decoding of Reed-Solomon Codes Beyond the Error-Correction Bound, « *J Complexity*, vol 12, pp 180-193, Dec 1997

[SKHN] Y Sugiyama, Y Kasahara, S Hirasawa and T Namekawa, «A Method for Solving Key Equation for Goppa Codes, « *Info, and Control*, vol 27, pp 87-99, 1975

[Sug] Y Sugiyama, M Kasahara, S Hirasawa and T Namekawa, «Further Results on Goppa Codes and Their Applications to Constructing Efficient Binary Codes, « *IEEE Trans Info Theory*, vol IT-22, pp 518-526, 1978

[TP] D J Taipale and M B Pursley, «An Improvement to Generalized Minimum Distance Decoding, « *IEEE Trans Info Theory*, vol 37, no 1, pp 167-172, Jan 1991

[TYFKL] T Takata, Y Yamashita, T Fujiwara, T Kasami and S Lin, «On a suboptimum decoding of decomposable block codes, « *IEEE Trans Info Theory*, vol 40, no 5, pp 1392-1406, Sept 1994

[TC] O Y Takeshita and D J Costello, Jr, «New Deterministic Interleaver Designs for Turbo Codes, « *IEEE Trans Info Theory*, vol 46, no 6, pp 1988-2006, Sept 2000

[TLFL] H Tang, Y Liu, M P C Fossorier and S Lin, «On Combining Chase-2 and GMD Decoding Algorithms for Nonbinary Block Codes, « *IEEE Comm Letters*, vol 5, no 5, pp 209-211, May 2001

[Tan] R M Tanner, «A Recursive Approach to Low Complexity Codes, « *IEEE Trans Info Theory*, vol IT-27, no 5, pp 533-547, Sept 1981

[tBrl] S ten Brink, «Convergence of iterative decoding, « *Electronics Letters*, vol 35, pp 806-808, May 1999

[tBr2] S ten Brink, «Convergence Behaviour of Iteratively Decoded Parallel Concatenated Codes, « submitted to *IEEE Trans Comm*, Mar 2000

[TKK] H Tokushige, T Koumoto and T Kasami, «An Improvement to GMD-like Decoding Algorithms, « *Proc 2000 IEEE Int Symp Info Theory (ISIT'OO)*, p 396, Sorrento, Italy, 2000

[Ungl] G Ungerboeck, «Channel Coding with Multilevel/Phase Signals, « *IEEE Trans Info Theory*, vol IT-28, no 1, pp 55-67, Jan 1982

[Ung2] G Ungerboeck, «Trellis-Coded Modulation with Redundant Signal Sets I Introduction, «*IEEE Comm Mag*, vol 25, no 2, pp 5-11, Feb 1987

[Ung3] G Ungerboeck, «Trellis-Coded Modulation with Redundant Signal Sets II State of the Art, « *IEEE Comm Mag*, vol 25, no 2, pp 12-21, Feb 1987

[Van] W J van Gils, «Two Topics on Linear Unequal Error Protection Codes Bounds on Their Length and Cyclic Code Classes, « *IEEE Trans Info Theory,* vol IT-29, no 6, pp 866-876, Nov 1983

[VO] S A Vanstone and P C van Oorschot, *An Introduction to Error Correcting Codes with Applications,* Kluwer Academic Publishers, 1989

[Vit1] A J Viterbi, «Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm, « *IEEE Trans Info Theory,* vol IT-13, pp 260-269, April 1967

[Vit2] A J Viterbi, «Convolutional Codes and Their Performance in Communication Systems, «*IEEETrans Comm*, vol COM-19, no 4, pp 751-772, 1971

[Vit3] A J Viterbi, «An Intuitive Justification and a Simplified Implementation of the MAP Decoder for Convolutional Codes, « *IEEE J Sel Areas in Comm*, vol 16, no 2, pp 260-264, Feb 1998

[Vit4] A J Viterbi, J K Wolf, E Zehavi and R Padovani, «A Pragmatic Approach to Trellis-Coded Modulation, « *IEEE Comm Mag*, pp 11-19, July 1989

[ViOm] A J Viterbi and J K Omura, *Principles of Digital Communication and Coding,* McGraw-Hill, 1979

[Vuc] B Vucetic and J Yuan, *Turbo Codes Principles and Applications,* Kluwer Academic, 2000

[WFH] U Wachsmann, R F H Fischer and J B Huber, «Multilevel Codes Theoretical Concepts and Practical Design Rules, « *IEEE Trans Info Theory,* vol 45, no 5, pp 1361-1391, July 1999

[WM] B E Wahlen and C Y Mai, «Turbo Coding Applied to Pragamtic Trellis-Coded Modulation, « *IEEE Comm Letters,* vol 4, no 2, pp 65-67, Feb 2000

[Wel] E J Weldon, Jr, «Decoding Binary Block Codes on Q-ary Output Channels, « *IEEE Trans Info Theory,* vol IT-17, no 6, pp 713-718, Nov 1971

[Wib] N Wiberg, «Codes and Decoding on General Graphs, « Ph D dissertation, Dept Elect Eng, Linkoping Univ, 1996

[Wic] S B Wicker, *Error Control Systems for Digital Communication and Storage,* Prentice-Hall, 1995

[WB] S B Wicker and V K Bhargava, *Reed-Solomon Codes and Their Applications,* IEEE Press, 1994

[Wil] S G Wilson, *Digital Modulation and Coding,* Prentice-Hall, 1996

[Wol] J K Wolf, «Efficient Maximum-Likelihood Decoding of Linear Block Codes Using a Trellis, « *IEEE Trans Info Theory,* vol IT-24, no 1, pp 76-80, Jan 1978

[WV] J K Wolf and A J Viterbi, «On the Weight Distribution of Linear Block Codes formed From Convolutional Codes, « *IEEE Trans Comm*, vol 44, no 9, pp 1049-1051, Sep 1996

[WJ] J M Wozencraft and I M Jacobs, *Principles of Communication Engineering,* John Wiley and Sons, 1965

[YI] K Yamaguchi and H Imai, «A New Block Coded Modulation Scheme and Its Soft Decision Decoding, « *Proc 1993 IEEE Inter Symp Info Theory (ISIT'93),* p 64, 1993

[YKH] Y Yasuda, K Kashiki and Y Hirata, «High-Rate Punctured Convolutional Codes for Soft Decision Viterbi Decoding, « *IEEE Trans Comm*, vol COM-32, no 3, pp 325-328, March 1984

[ZW]E Zehavi and J K Wolf, «P^2 Codes Pragmatic Trellis Codes Utilizing Punctured Convolutional Codes, « *IEEE Communications Magazine,* pp 94-99, Feb 1995

[ZP] V V Zyablov and M S Pinsker, «Estimation of the Error-Correction Complexity for Gallager Low-Density Codes, « *Prob Pered Inform*, vol 11, no 1, pp 26-36, 1975

[Zin] V A Zinov'ev, «Generalized Cascade Codes, « *Probl Pered Informatsii,* vol 12, no 1, pp 5-15, 1976

ЛИТЕРАТУРА (добавленная при переводе)

Э Берлекэмп, *Алгебраическая теория кодирования*, «Мир», Москва 1971

Р Блейхут, *Теория и практика кодов, исправляющих ошибки*, «Мир», Москва, 1986

R E Blahut, *Algebraic methods for signal processing and communications coding*, Springer-Verlag, 1992

Э Л Блох и В В Зяблов, *Обобщенные каскадные коды*, «Связь», Москва, 1976

Э Л Блох и В В Зяблов, *Линейные каскадные коды*, «Наука», Москва 1982

Кларк и Кейн,

Г С Евсеев, «О сложности декодирования линейных кодов», Проблемы передачи информации, т 19, №1, стр 3-8, 1983

Д Форни, *Каскадные коды*, «Мир», Москва, 1970

Р Галлагер, *Коды с малой плотностью проверок на четность*, «Мир», Москва, 1966

Р Галлагер, *Теория информации и надежная связь*, «Советское радио», Москва 1974

Р Лидл и Г Нидерайтер, *Конечные поля*, т 1, 2, «Мир», Москва 1988

Ф Дж Мак-Вильямс и Н Дж Слоэн, *Теория кодов, исправляющих ошибки*, «Связь», Москва 1979

Дж Мэсси, *Пороговое декодирование*, «Мир», Москва 1966

Дж Проакис, *Цифровая связь*, Пер с англ /Под ред Д Д Кловского – М Радио и связь 2000 – 800 с ил ISBN 5-256-01434-X

У Питерсон и Э Уэлдон, *Коды, исправляющие ошибки,* Пер с англ / Под ред Р Л Добрушина и С Н Самойленко – МИР, Москва, 1976 –594 с ил

К Шэннон, *Работы по теории информации и кибернетике*, «Иностранная литература», Москва 1963

C Schlegel, *Trellis coding*, IEEE PRESS, 1997

А Д Витерби и Дж Омура, *Принципы цифровой связи и кодирования*, «Радио и связь», Москва 1982

Дж Возенкрафт и И Джекобс, *Теоретические основы техники связи*, «Мир», Москва 1969

Приложение А

РАСПРЕДЕЛЕНИЕ ВЕСОВ РАСШИРЕННЫХ КОДОВ БЧХ

В этом приложении представлено распределение весов всех расширенных БЧХ кодов длины до 128. включительно. В первой строчке таблицы указаны параметры кода «n, k, d» (они же являются именем файла, содержащего распределение весов на ECC веб сайте.) В следующих ниже строчках таблицы дается список весов w и числа кодовых слов данного веса A_w. Эти коды имеют симметричное распределение весов, а именно, справедливо равенство $A_w = A_{n-w}$, for $0 < w < n/2$. С учетом этого приводится только половина распределения весов.

A.1 Длина 8

wd.8.1.8
8 1
wd.8.4.4
4 14
8 1
wd.8.7 2
2 28
4 70

A.2 Длина 16

wd.16.05.08
8 30
wd.16.07.06
6 48
8 30
wd.16.11.04
4 140
206
6 448
8 870
wd.16.15.02
2 120
4 1820
6 8008
8 12870

A.3 Длина 32

wd.32.06.16
16 62
wd.32.11.12
12 496
16 1054
wd.32.16.08
8 620
12 13888
16 36518
wd.32.21.06
6 992
8 10540
10 60512
12 228160
14 446400
16 603942
wd.32.26.04
4 1240
6 27776
8 330460
10 2011776
12 7063784
14 14721280
16 18796230
wd.32.31.02
2 496
4 35960
6 906192
8 10518300
10 64512240
12 225792840
14 471435600
16 601080390

A.4 Длина 64

wd.64.07.32
32 126
wd.64.10.28
28 448
32 126
wd.64.16.24
24 5040
28 12544
32 30366
wd.64.18.22
22 4224
24 5040
26 24192
28 12544
30 69888
32 30366
wd.64.24.16
16 2604
18 10752
22 216576
24 291648
26 1645056
28 888832
30 4419072
32 1828134
wd.64.30.14
14 8064
16 30828
18 631680
20 1128960
22 14022144
24 14629440
26 105057792
28 65046016
30 282933504
32 106764966
wd.64.36.12
12 30240
14 354816
16 3583020
18 27105792
20 145061280
22 603113472
208
24 1853011776
26 4517259264
28 8269968448
30 12166253568
32 13547993382
wd.64.39.10
10 13888
12 172704
14 2874816
16 29210412
18 214597824
20 1168181280
22 4794749760
24 14924626752
26 35889146496
28 66620912960
30 96671788416
32 109123263270
wd.64.45.08
8 27288
10 501760
12 12738432
14 182458368
16 1862977116
18 13739292672
20 74852604288
22 306460084224
24 956270217000
26 2294484111360
28 4268285380352
30 6180152832000
32 6991765639110
wd.64.51.06
6 20160
8 1067544
10 37051840
12 801494400
14 11684617344
16 119266575708
18 879321948288
20 4789977429888
22 19616032446528
24 61193769988008
26 146864398476096
28 273137809339136
30 395577405119232
32 447418802536902
wd.64.57.04
4 10416
6 1166592
8 69194232
10 2366570752
12 51316746768
14 747741998592
16 7633243745820
18 56276359749120
20 306558278858160
22 1255428754917120
24 3916392495228360
26 9399341113166592
28 17480786291963792
30 25316999607653376
32 28634752793916486
wd.64 63.02
2 2016
4 635376
6 74974368
8 4426165368
10 151473214816
12 3284214703056
14 47855699958816
16 488526937079580
18 3601688791018080
20 19619725782651116
22 80347448443237936
24 250649105469666110
26 601557853127198720
28 1118770292985240200
30 1620288010530347000
32 1832624140942591500

A.5 Длина 128

wd.128.008.064
64 254
wd.128.015.056
56 8128
64 16510
wd.128 022.048
48 42672
56 877824
64 2353310
wd 128.029.044
210
44 373888
48 2546096
52 16044672
56 56408320
60 116750592
64 152623774
wd.128.036.032
32 10668
36 16256
40 2048256
44 35551872
48 353494848
52 2028114816
56 7216135936
60 14981968512
64 19484794406
wd.128.043.032
32 124460
36 8810752
40 263542272
44 4521151232
48 44899876672
52 262118734080
56 915924097536
60 1931974003456
64 2476672341286
wd.128.050.028
28 186944
32 19412204
36 113839296
40 337238522888
44 579267441920
48 5744521082944
52 33558415333632
56 117224663972352
60 247312085243776
64 31699236111910
wd.128.057.024
24 597408
28 24579072
32 2437776684
36 141621881856
40 4315318568736
44 74150180302848
48 73528925007168
52 4295496356229120
56 15004724612905792
211
60 31655991621445632
64 4574965317267238
wd.128.064.022
22 243840
24 6855968
26 107988608
28 1479751168
30 16581217536
32 161471882796
34 1292241296640
36 9106516329984
38 53383279307904
40 278420690161824
42 1218666847725184
44 4782630191822848
46 15858705600596992
48 47425684161326912
50 120442185147493376
52 277061634654099456
54 543244862505775360
56 967799721857135168
58 1473287478189735168
60 2041819511308530688
62 2421550630907043328
64 2617075886216910118
wd.128.071.020
20 2674112
22 37486336
24 839699616
26 13825045248
28 188001347136
30 2140095182336
32 20510697927468
34 166689980438016
36 1156658661471040
38 6886497209935616
40 35363776220195360
42 157207798773129984
44 607468163067994304
46 2045773679068686336
48 6023796954778012480
50 15537040516548126720
52 35191124114633006464
54 70078589269156969984
56 122925566952088660288
58 190054082758956107264
60 259342737902840355456

62 312380032198035579904
64 332409207867786543910
212
wd.128.078 016
16 387096
18 5462016
20 213018624
22 539859840
24 107350803840
26 1766071867392
28 24074650400768
30 273932927993856
32 2625267567169884
34 2133648518951040
36 14805286631892608
38 881470039149213696
40 4526561735332554624
42 20122606565844068352
44 77755925658495682560
46 261859003134276581376
48 771046023044966543784
50 1988741249124011372544
52 4504463828911859699712
54 8970059328813665832960
56 15734472710169831412480
58 2432692269013718774169 6
60 3319587221944924483584
62 3998464407989233708646 4
64 4254837887630253151495 0
wd.128 085 014
14 341376
16 22121368
18 856967552
20 27230880768
22 680417833472
24 13721772977024
26 226128254847488
28 3081454360189952
30 35064826913355520
32 336014520825141340
34 2731238665152128768
36 189496122805134118 4
38 1128349932260321039 36
40 5793648467052949968 64
42 25758496166314862044 16
44 9952155728071153882112
46 3351998240451222340160 0
48 9868791466657342836484 0
50 2545742962488001599228 16
52 5765364656040619165149184
54 1148237129819878789497856
56 2138905488918250206574 08
58 3114034684742715393815552
60 4248814088020530790422528
62 5118344408749492898411 52
64 5445862703373444517825478
wd.128.092.012
12 1194816
14 45646848
16 2751682584
18 110071456768
20 3484410778688
22 8709939355008
24 1756359917165952
26 28944450656120832
28 394426389988237184
30 448829772766317158 4
32 4300984271589669308 4
34 3495987175785875312 64
36 2425549189872597678976
38 14442886028067639783424
40 7415866532060415580416
42 3297089066350487847690 24
44 1273875338627254559097 6
46 4290559778009132197764096
48 1263204709961981875163997 6
50 3258552533730703659129 1392
52 7379663146924327761511104
54 1469744221488665142430842 88
56 2577778683068002369247232
58 3985966282327258318952371 2
60 5438479459612333934726545 92
62 6551483932680756588722380 80
64 6977009624641314914571309 4
wd 128.099 010
10 796544
12 90180160
14 6463889536
16 347764539928
18 14127559573120
20 44575475469248
22 1114968526546777 6
24 224811690627712384
26 37048953778219110 4
28 5048655617312167360 0
30 5745021767305712555 52
32 5505259786944679990620
34 4474863572027338314 368
36 3104702962799943929753 6
38 1848689417301349247899904
40 9492309123731911851566976
214
42 4220274021289462404510374 4
44 1635604174238999188223251 2
46 5491916536029199089614841 60
48 1616902022803263350264149928
50 4170947258582865019960480640
52 9445968792041391795950926784
54 1881272610465984668145312896
56 3299556702053516278222434304
58 5102036860227828704471599232
60 6961253682581943211726121216
62 8385899464831778035255231 5392
64 8922497198963119451267798 6758
wd 128.106 008
8 774192
10 105598976
12 11361676032
14 828626841600
16 44515013174520
18 1808265733435392
20 57056968214853376
22 1427159096213901312
24 28775892186952836240
26 4742266424066961162 24
28 6462279071735110418944
30 7353627881643377292902 4
32 7046732528807792356874 52
34 5727825370458099461038080
36 3974019792802806306390 4768
38 2366322454142038380819496 96
40 1215015567801313175697152304
42 5401950747433627456981266432
44 2087117334236687285956601 4720
46 7029653166324768481637872 8448
48 2069634589128910262771981 68776
50 5338812491138407971152234 61888
52 1209084005346591905941683436800
54 2408028941464856710061855682560
56 4223432578506218555128558121488
58 6530607181280659655017851666432
60 8910404713446325250255943109632
62 1073395131529430117449184 1282048
64 1142079641434358842413615868 9350
wd.128.113.006
6 341376
8 87288624
10 13842455424
12 1448180487936
14 106141978256640
16 5697211389035256
18 231462916338818304
20 7303265469631124224
23 182676478544670888576
24 3683313800412335283600
26 6070101136622999342092 8
28 8271717185445875657630 72
30 9412643693444880033139200
32 9019817636126063611266870 0
34 7331616474282651537211008 00
36 5086745334774298795535505920
38 3028892741304486246612513728 0
40 1555219926784991759059415071 20
42 6914496956716930874586344620 80
44 2671510187822394963671294035200
46 8997956052897506127123622265600
48 2649132274084454855472046144 0200
50 6833679988658651159231793719 5776
52 1547627526843289352309216571 51744
54 3082277045075720326820005075 10400
56 5405993700486723632424736843 64048
58 8359177192041145268889081549 32352
60 1140531803320872201314876100 099072
62 1373945768357978846215074177 297408
64 1461861941035652013200273232 486470
wd.128.120.004
4 85344
6 42330624
8 11170182384
10 1772228014592
12 185359804775712
14 13586256544975872
16 729242357526446712
18 29627257927486958592
20 9348179550929226293 44
22 2338258936574936642918 4
24 4714641660340593022 12704
26 7769729456174562056216064
28 1058779799704768692755 385504
30 1204818392766796825789 470720
32 1154536657423705241877721*7820
34 9384469087079854005243 460320
36 6511034028512200825869 31517920
38 3876982708869397190103 809681920
40 1990681506284869946214 005860480
42 8850556104597527515220 031406080
44 3419533040412683458478 46829061280
46 1151738374770880217441 839661716480
48 3390889310828097487679 807613566280
50 8747110385483091255323 050018747392
52 1980963234359406110564 038479579552
54 3945314617696930206006 77131106155 52
56 6919671936622992781986 372056678992
58 1069974680581268544419 56301420662272
60 1459880708250713896331 196542663972 16
62 1758650583498215857154 113923579125 76
64 1871183284525631492099 91044344449600
wd.128 127 002
2 8128
4 10668000
6 5423611200
8 1429702652400
10 226846154180800
12 23726045489546400
14 1739040916651367936
16 9334302120126219878 4
18 3792289018215984005120
20 1196566982326569884712 96
22 2992971438910354793431040
24 6034741325194250007504486 4
26 9945253703920120562649661 44
28 1355238143621496448603704 5248
30 1542167542741701579874175 54944
32 1477806921502279921677734 248448
34 1201212043146238773004850 9542400
36 8334123556495567822018198 0577792
38 4962537867352840085871279 26292480
40 2548072328044630786408938 247028736
42 1132871181388483483204671 1017308160
44 4377002291728235884501768 9455329280
46 1474225119706727508434852 96833593344
48 4340338317859967634879942 52512722944
50 1119630129341835894935101 4497936998 40
52 2535632939980039741724790 850645393408
54 5050002710652071717329822 398986321920
56 8857180078877432500571052 147864502272
58 1369567591144023701353247 469658439680 0
60 1868647306560914396571225 5676040871936
62 2251072746877716714367117 2081479843840
64 2395114604192810393768871 0428982509568

Заявки на книги присылайте по адресу:
125319 Москва, а/я 594
Издательство «Техносфера»
e-mail: knigi@technosphera.ru
sales@technosphera.ru
факс: (095) 956 33 46

В заявке обязательно указывайте
свой почтовый адрес!

Подробная информация о книгах на сайте
http://www.technosphera.ru

Р. Морелос-Сарагоса
Искусство помехоустойчивого кодирования.
Методы, алгоритмы, применение

Компьютерная верстка – М. Ю. Бирюков
Дизайн книжных серий – С.Ю. Биричев
Ответственный за выпуск – Л.Ф. Соловейчик

Формат 84 x 108/32. Печать офсетная.
Гарнитура Ньютон
Печ.л 20. Тираж 2000 экз. Зак. № 2993.
Бумага офсет №1, плотность 65 г/м^2.

Издательство «Техносфера»
Москва, ул. Тверская, дом 10 строение 3

Диапозитивы изготовлены ООО «Европолиграфик»
Отпечатано на ГУРПП
г. Ржев, ул. Урицкого, дом 91