Федеральное государственное бюджетное учреждение науки
Институт проблем передачи информации им. А.А. Харкевича
Российской академии наук (ИППИ РАН)

На правах рукописи
УДК 519.725

Жилин Игорь Витальевич

Разработка и анализ алгоритмов декодирования МПП- и ОЛО-кодов, допускающих распараллеливание и конвейеризацию

Специальность 05.13.17 —
«Теоретические основы информатики»

Диссертация на соискание учёной степени
кандидата технических наук

Научный руководитель:
доктор технических наук
Зяблов Виктор Васильевич

Москва — 2015

Оглавление

Введение

Одним из важных факторов укрепления российской экономики является приоритетное развитие связи и телекоммуникационных технологий. Информатизация государства и общества невозможна без создания эффективных систем передачи информации.

Одним из способов повышения надёжности и достоверности связи, а также снижения количества ошибок является помехоустойчивое кодирование. Развитие информационных технологий требует высоких скоростей для передачи больших объемов информации, необходимых для работы государственных и частных информационных систем, систем электронного документооборота, работы систем видеоконференцсвязи и других применений. Необходимость решения перечисленных задач ставит перед специалистами в области помехоустойчивого кодирования новые требования.

На данном этапе технологического развития потребность в высоких скоростях передачи данных растёт быстрее, нежели скорости работы вычислительных устройств, которые применяются для реализации кодеров и декодеров. Так, современные волоконно-оптические линии связи (ВОЛС) работают на скоростях до 100–400 Гбит/с, при этом рабочие частоты вычислительных устройств не превосходят единиц гигагерц. В некоторых случаях это вынуждает разработчиков применять простые, но неэффективные коды малой длины. Известно, что коды, дающие большой энергетический выигрыш, должны иметь большую длину [1]. В то же время эффективные коды большой длины не всегда допускают эффективное распараллеливание и по этой причине их применение для высокоскоростных линий связи неэффективно. Под эффективными кодами здесь понимаются коды, обладающие невысокой сложностью кодирования и имеющие хорошую корректирующую способность.

Таким образом, возникает потребность в кодах, которые позволили бы обеспечить невысокую сложность декодирования при больших длинах и эффективные распараллеливание и конвейеризацию. Последний пункт вытекает из требования обеспечения высоких скоростей их обработки на существующей технологической базе.

Всем этим требованиям в наибольшей степени удовлетворяют коды с малой плотностью проверок (МПП-коды) и обобщенные каскадные коды, которые также известны как обобщённые коды с локализацией ошибок (ОЛО-коды).

Коды с малой плотностью проверок были предложены Р. Галлагером в работе [2]. Данный класс кодов имеет минимальный рост сложности алгоритмов декодирования при увеличении длины кода, что позволяет строить длинные коды с хорошими корректирующим свойствами.

Впервые коды с локализацией ошибок были предложены в 1965 г. в работах [3, 4]. Затем в работе [5] были построены обобщённые коды с локализацией ошибок и преложены алгоритмы их кодирования и декодирования. Систематическое описание ОЛО-кодов приведено в работах [6, 7]. В работе [8] показано, что ОЛО-коды являются частным случаем обобщенных каскадных кодов. ОЛО-коды позволяют строить коды с большой скоростью и эффективными алгоритмами декодирования. Также конструкция ОЛО-кодов является очень гибкой, что позволяет подбирать оптимальный код в каждом конкретном случае.

Целью данной работы является исследование и разработка конструкций и алгоритмов декодирования МПП- и ОЛО-кодов, допускающих распараллеливание и конвейеризацию.

Для достижения поставленной цели необходимо было решить следующие **задачи**:

1. исследование и разработка широко используемых МПП-кодов с точки зрения анализа их корректирующих свойств и возможности распараллеливания и конвейеризации их декодирования как наиболее сложной операции;
2. исследование и разработка ОЛО-кодов с точки зрения анализа их корректирующих свойств и возможности распараллеливания и конвейеризации их декодирования;
3. разработка алгоритмов и программ расчёта и оптимизации ОЛО-кодов.

Научная новизна. В настоящей работе впервые:

1. Предложен способ векторизации алгоритма “распространения доверия” для q-ичных МПП-кодов.
2. Разработан метод применения алгоритма декодирования “распространения доверия” с мягким входом для каналов с жёстким решением.
3. Произведено сравнение различных алгоритмов декодирования МПП-кодов с единичной памятью с циклическим замыканием.
4. Предложен метод выбора структуры ОЛО-кода, оптимизации скоростей внешних кодов, обеспечивающий наибольшую возможною скорость ОЛО-кода при условии, что вероятность неправильного декодирования на кодовое слово не превышает заданную при заданной входной вероятности ошибки на символ. Также предложен алгоритм мягкого декодирования ОЛО-кодов, для которого предложена верхняя граница на вероятность неправильного декодирования.
5. Предложена нижняя граница на вероятность неправильного декодирования ОЛО-кода для заданного алгоритма декодирования, которая близка к верхней границе в области средних значений входной вероятности ошибки на символ.
6. Предложено семейство ОЛО-кодов, частная конструкция кода из которого обеспечивает энергетический выигрыш больше, чем применяемый в существующих ВОЛС код Боуза-Чоудхури-Хоквингема, и при этом обладает меньшей сложностью реализации.

7. Построены ОЛО-коды с использованием МПП-кодов в качестве внешних и разработан алгоритм мягкого декодирования для них.

Теоретическая и практическая значимость. Предложен метод выбора структуры ОЛО-кода, алгоритмы оптимизации скоростей внешних кодов, нижняя граница на вероятность неправильного декодирования для заданного алгоритма декодирования, а также реализующие их программы. Их применение позволяет обойтись без ресурсоёмкого моделирования и тем самым сократить время разработки физического уровня систем связи.

Разработан метод применения алгоритма декодирования “распространения доверия” с мягким входом для каналов с жёстким решением. Это позволяет улучшить эффективность работы МПП-кодов в системах, где недоступна информация о надёжностях принятых символов.

Предложен способ векторизации алгоритма “распространения доверия” для q-ичных МПП-кодов. Алгоритм был реализован на языке OpenCL, эта реализация позволяет при вычислениях на графическом ускорителе получить выигрыш в скорости моделирования до нескольких раз по сравнению с вычислениями на процессоре общего назначения.

Предложено семейство ОЛО-кодов, частная конструкция кода из которого обеспечивает энергетический выигрыш больше, чем применяемый в существующих ВОЛС код Боуза-Чоудхури-Хоквингема, и при этом обладает меньшей сложностью реализации. Коды из этого семейства могут быть использованы в качестве кодов коррекции ошибок в перспективных системах связи.

Положения, выносимые на защиту:

1. Векторизация алгоритма “распространения доверия” декодирования q-ичных МПП-кодов.

2. Декодер МПП-кодов для каналов с жёстким решением, основанный на алгоритме “распространения доверия”.

3. Алгоритмы и реализующие их программы оптимизации скорости ОЛО-кодов, позволяющие строить коды с максимальной скоростью для заданных входной вероятности ошибки и ограничения на выходную вероятность ошибки.

4. Разработка, построение и исследование ОЛО-кодов с внешними и внутренними кодами над одним алфавитом. Разработка алгоритма выбора структуры ОЛО-кода с учётом алгоритма декодирования, исправляющего ошибки и стирания.

5. Разработка, построение и исследование алгоритмов мягкого декодирования ОЛО-кодов с компонентными кодами Рида-Соломона и с компонентными МПП-кодами.

Апробация работы. Результаты диссертации неоднократно докладывались на семинарах по теории кодирования ИППИ РАН; на конференциях IEEE International Symposium on Information Theory, 2015; International Symposium on Problems of Redundancy in Information and Control Systems, 2012, 2014; 21th European Wireless (EW) Conference, 2015; XII International Workshop on Algebraic and Combinatorial Coding Theory, 2014; Информационные технологии и системы 2012,

2013, 2014. Основные результаты также докладывались на семинарах по теории кодирования в ИППИ РАН.

Личный вклад. Все основные научные положения и выводы, составляющие содержание диссертации, разработаны автором самостоятельно. Теоретические и практические исследования, а также вытекающие из них выводы и рекомендации проведены и получены автором лично. В совместных публикациях научному руководителю В. В. Зяблову принадлежат постановки задач и указания основных направлений исследований, а основные результаты, выкладки и численные расчеты выполнены диссертантом.

Публикации. Основные результаты по теме диссертации изложены в 6 печатных изданиях [9–14], из них 2 статьи в рецензируемых журналах, входящих в базы цитирований Web of Science или Scopus [9, 10], и 4 статьи в сборниках трудов конференций [9, 12–14].

Объем и структура работы. Диссертация состоит из введения, трёх глав, заключения и приложений. Полный объем диссертации **82** страницы текста с **41** рисунком и 2 таблицами. Список литературы содержит **44** наименования.

Глава 1

МПП-коды

1.1 Введение

Двоичные коды с малой плотностью проверок (МПП-коды) были предложены в 1960 году Галлагером [2]. Эти линейные блочные коды задаются с помощью проверочной матрицы **H**, характеризуемой относительно малым числом ненулевых элементов в строках и столбцах. Проверочную матрицу **H** МПП-кода можно представить в виде графа Таннера [15]. В главе 5 работы [2] Галлагер дал также краткое описание МПП-кодов, заданных над произвольным полем $GF(q)$, а также представил схему алгоритма их декодирования по апостериорным вероятностям на выходе канала (алгоритм “распространения доверия”).

Непосредственно после своего открытия МПП-коды не получили широкого распространения ввиду достаточно сложной реализации и были практически забыты на протяжении последующих 30 лет. К значимым теоретическим результатам, относящимся к этому периоду, следует отнести работу [16]. Тем не менее в конце 1990-х годов они были переоткрыты [17, 18], и с тех пор интерес к данному классу кодов все более и более усиливается. Однако в подавляющем большинстве работ, посвященных МПП-кодам, рассматриваются двоичные коды. Важные теоретические результаты, относящиеся к этому классу кодов, получены в работах [19–22].

В последнее время все в большем числе работ исследуются недвоичные МПП-коды. Важные теоретические результаты получены в [23, 24]. Также среди публикаций, в которых рассматриваются недвоичные МПП-коды, следует особо отметить работу [18], в которой подробно описан ансамбль МПП-кодов над полем $GF(q)$, а также обобщенный алгоритм их декодирования $q-ary\ SumProductAlgorithm (Q-SPA)$. В работе [25] представлено улучшение этого алгоритма, основанное на использовании быстрого преобразования Фурье, что позволяет существенно снизить вычислительную сложность декодера.

Помимо случайных недвоичных МПП-кодов, предложенных Галлагером, на практике часто применяют коды, проверочные матрицы которых построены конструктивно, например, с использованием матриц перестановок [26–28].

Данная глава состоит их нескольких пунктов. В пунктах 1.2 и 1.3 приведены описания рассматриваемых МПП-кодов и их алгоритмов декодирования. В п. 1.5 производится сравнение

различных декодеров для кодов с малой плотностью проверок, основанных на свёрточных кодах с единичной памятью. В пункте 1.4 сравниваются различные декодеры МПП-кодов для каналов с жёстким решением. В качестве таких декодеров выступают известный алгоритм мажоритарного декодирования, алгоритм с введением стираний [29], а также предложена адаптация алгоритма “распространения доверия”. В п. 1.6 предложен способ векторизации вычислений q-ичного алгоритма “распространения доверия”.

1.2 Рассматриваемые конструкции МПП-кодов

1.2.1 Ансамбль случайных двоичных МПП-кодов Галлагера

Рассмотрим построение проверочной матрицы $\mathbf{H}$ кода с малой плотностью проверок на четность (МПП-кода Галлагера). Проверочную матрицу $\mathbf{H}_0$ кода с проверкой на четность длины n_0 можно записать как $\mathbf{H}_0 = \underbrace{111\ldots1}_{n_0}$. Запишем диагональную блочную матрицу $\mathbf{H}_b$ с b проверочными матрицами $\mathbf{H}_0$ на главной диагонали:

$$\mathbf{H}_b = \begin{pmatrix} \underbrace{\begin{matrix} \mathbf{H}_0 & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{0} & \mathrm{H}_0 & \ldots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \ldots & \mathbf{H}_0 \end{matrix}}_{b} \end{pmatrix}, \tag{1.1}$$

где b очень велико. Поскольку размер матрицы $\mathbf{H}_0$ равен $1 \times n_0$, то размер матрицы $\mathbf{H}_b$ – $b \times bn_0$. Обозначим $\pi(\mathbf{H}_b)$ случайную перестановку столбцов матрицы $\mathbf{H}_b$. Тогда матрица, составленная из $\ell > 2$ таких перестановок в качестве слоев,

$$\mathbf{H} = \begin{pmatrix} \mathbf{H}_1 \\ \mathbf{H}_2 \\ \vdots \\ \mathbf{H}_\ell \end{pmatrix} = \begin{pmatrix} \pi_1(\mathbf{H}_b) \\ \pi_2(\mathbf{H}_b) \\ \vdots \\ \pi_\ell(\mathbf{H}_b) \end{pmatrix}, \tag{1.2}$$

является разреженной проверочной матрицей $\mathbf{H}$ размера $\ell b \times bn_0$, которая определяет ансамбль МПП-кодов Галлагера длины $n = bn_0$, где $n \gg n_0$. Обозначим этот ансамбль $\mathscr{E}(n_0,\ell,b)$.

О п р е д е л е н и е 1. Для компонентного кода с проверкой на четность с проверочной матрицей $\mathbf{H}_0$ независимо и равновероятно выбирая случайные перестановки π_l , $l = 1,2,\ldots,\ell$, определим ансамбль МПП-кодов Галлагера $\mathscr{E}(n_0,\ell,b)$.

Нижняя оценка на скорость R кода из $\mathscr{E}(n_0,\ell,b)$ определяется формулой:

$$R \geqslant 1 - \ell\,(1 - R_0)\,,$$

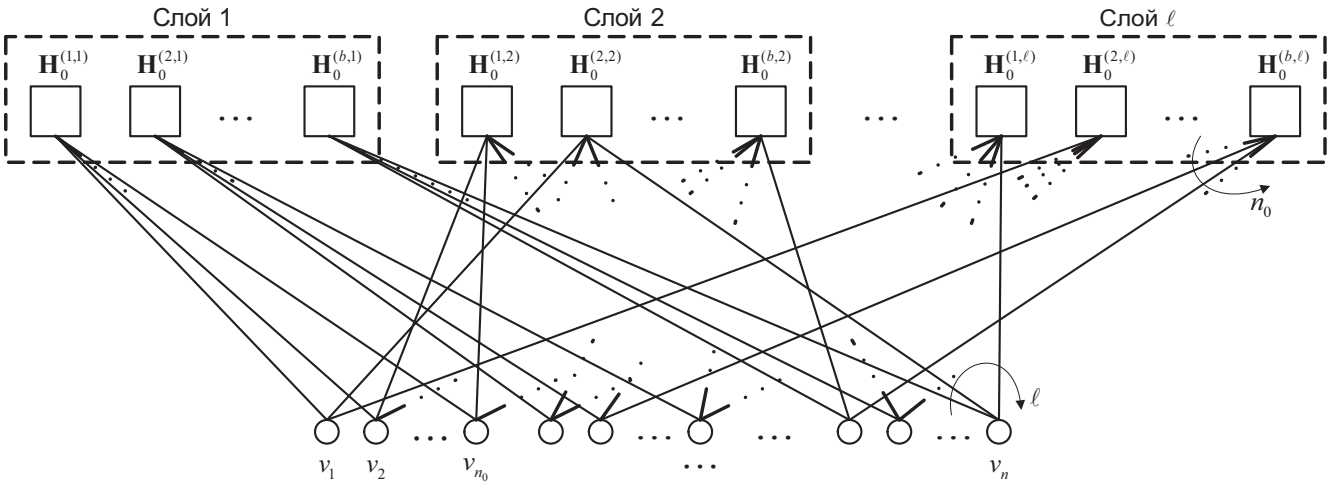


Рисунок 1.1. Двудольный граф Таннера, соответствующий проверочной матрице $\mathbf{H}$ МПП-кода Галлагера

где $R_0 = \frac{n_0-1}{n_0}$ – скорость кода с проверкой на четность. Равенство достигается только в случае, когда $\mathbf{H}$ имеет полный ранг.

Как следует из построения, МПП-код Галлагера из $\mathscr{E}(n_0,\ell,b)$ имеет $n = bn_0$ кодовых символов, которые распределены между ℓb компонентных кодов (b в каждом слое) с проверочной матрицей $\mathbf{H}_0$. Такие коды могут быть представлены с помощью двудольного графа Таннера [15] $G = (V_1 : V_2, E)$ с $n = bn_0$ вершинами-символами V_1 и ℓb вершинами-кодами V_2, как на рис. 1.1. Если символ входит в проверку кода-компонента, то в графе Таннера существует ребро из E, соединяющее соответствующую вершину-символ из V_1 с соответствующей вершиной-кодом из V_2. В соответствии с конструкцией проверочной матрицы МПП-кода Галлагера каждая вершина-код включает одно проверочное уравнение, представляющее собой сумму по модулю два n_0 входящих в данную проверку символов. Каждый кодовый символ входит в проверочное уравнение точно одного кода-компонента в каждом слое. Таким образом, соответствующий граф Таннера регулярен со степенью вершины-символа равной ℓ и степенью вершины-кода равной n_0.

1.2.2 Ансамбль случайных недвоичных МПП-кодов Галлагера

Вначале дадим описание структуры случайной проверочной матрицы МПП-кода Галлагера над полем $GF(q)$ [30].

Для построения проверочной матрицы МПП-кода над полем $GF(q)$ выберем натуральные n_0 и b, причем $b \gg n_0$, и рассмотрим блочную диагональную матрицу

$$\mathbf{H}_b = \underbrace{\begin{pmatrix} \mathbf{H}_0 & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{H}_0 & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{H}_0 \end{pmatrix}}_{b}, \tag{1.3}$$

на главной диагонали которой находятся b проверочных матриц $\mathbf{H}_0 = \underbrace{(11\ldots1)}_{n_0}$ кода с параметрами $(n,k,d) = (n_0,n_0-1,2)$, называемого кодом-компонентом. Матрица $\mathbf{H}_b$ имеет размер $b \times bn_0$.

Пусть $\phi(\mathbf{H}_b)$ обозначает матрицу, полученную произвольной перестановкой π столбцов матрицы $\mathbf{H}_b$ и умножением их на произвольные элементы c_j, $j = 1..bn_0$ из мультипликативной группы поля $GF^*(q)$. Тогда матрица

$$\mathbf{H} = \begin{pmatrix} \phi_1(H_b) \\ \phi_2(H_b) \\ \vdots \\ \phi_l(H_b) \end{pmatrix}, \tag{1.4}$$

размера $bl \times bn_0$, составленная из $l > 2$ таких матриц, как слоев, является проверочной матрицей МПП-кода над полем $GF(q)$.

Определим ансамбль $\mathcal{E}(b,l,n_0)$ МПП-кодов над полем $GF(q)$ следующим образом:

О п р е д е л е н и е 2. Элементы ансамбля $\mathcal{E}(b,l,n_0)$ получаются путем независимого выбора $l,n_0,b \in \mathbb{N}$: $l < n_0 < b$, перестановок π_1, ..., π_l, а также ненулевых констант $c_{ij} \in GF^*(q)$, $j = 1..bn_0$, $i = 1..l$.

Как уже было отмечено во введении, алгоритм декодирования подобных МПП-кодов подробно описывался в ряде работ, например, в [18].

В следующем разделе представлен иной подход к построению ансамбля недвоичных МПП-кодов, проверочные матрицы которых обладают более структурированной формой, позволяющей декодировать их с использованием векторных операций над полем $GF(q)$.

1.2.3 Ансамбль недвоичных МПП-кодов, основанных на матрицах перестановок

В данном разделе дадим описание ансамбля недвоичных кодов с малой плотностью проверок, основанных на матрицах перестановок.

Пусть $l,n_0,m \in \mathbb{N}$, причем $2 < l < n_0$, $ln_0 < m!$. Рассмотрим циклическую мультипликативную группу $GF^\star(q) = \{1,\alpha,\alpha^2,\ldots,\alpha^{q-2}\}$ поля $GF(q)$, $q = p^t$, $t \geq 1$, $p \geq 2$ – простое, α – примитивный элемент поля; а также симметрическую группу $\mathcal{P}_m$ перестановок столбцов размерности m, $|\mathcal{P}_m| = m!$.

В симметрической группе выберем ln_0 случайных перестановок $\{\boldsymbol{\pi}_{ji}\} \in \mathcal{P}_m$, $j = 1..l, i = 1..n_0$. Потребуем также, что если $\boldsymbol{\pi}_{ji} = \boldsymbol{\pi}_{ks}$, то $j = k$, $i = s$. Ясно, что такие условия выбора перестановок $\boldsymbol{\pi}_{ji}$ соответствуют урновой модели без возвращений.

Составим ln_0 случайных m-элементных векторов (возможно, с повторениями элементов внутри набора), состоящих из элементов группы $GF^\star(q)$ ($q = p^m$, $p \geq 2$ – простое, $m \geq 1$): $S_{ji} = (\beta_{ji}^{(1)}, \beta_{ji}^{(2)}, \ldots, \beta_{ji}^{(m)})$, $j = 1..l$, $i = 1..n_0$.

Каждую из ln_0 перестановок $\boldsymbol{\pi}_{ji}$ применим к диагональной матрице следующего вида:

$$diag(\mathbf{S}_{ji}) = \begin{pmatrix} \beta_{ji}^{(1)} & 0 & \dots & 0 \\ 0 & \beta_{ji}^{(2)} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \beta_{ji}^{(m)} \end{pmatrix}. \tag{1.5}$$

Полученные ln_0 матриц вида:

$$\mathbf{R}_{ji} = \boldsymbol{\pi}_{ji}(diag(\mathbf{S}_{ji})) \tag{1.6}$$

сделаем элементами блочной матрицы $\mathbf{H}$:

$$\mathbf{H} = \begin{pmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} & \dots & \mathbf{R}_{1n_0} \\ \mathbf{R}_{21} & \mathbf{R}_{22} & \dots & \mathbf{R}_{2n_0} \\ \dots & \dots & \dots & \dots \\ \mathbf{R}_{l1} & \mathbf{R}_{l2} & \dots & \mathbf{R}_{ln_0} \end{pmatrix}. \tag{1.7}$$

Указанный выше способ построения матрицы $\mathbf{H}$ гарантирует, что все матрицы в каждой строке и каждом столбце будут различны. Так как $\mathbf{R}_{ij}$ - квадратная $m \times m$ матрица, то размерность $\mathbf{H}$ — $ml \times mn_0$.

О п р е д е л е н и е 3. Проверочная матрица $\mathbf{H}$ определяет ансамбль $\mathcal{E}_P(l, n_0, m, q)$ регулярных q-ичных (l, n_0)-кодов с малой плотностью проверок длины $n = mn_0$. Элементы ансамбля $\mathcal{E}_P(l, n_0, m, q)$ задаются выбором без возвращений матриц перестановок $\mathbf{P}_{ji} \in \mathcal{P}_m$, $j = 1..l$, $i = 1..n_0$, а также ln_0 m-элементных наборов $S_{11}, \dots, S_{ln_0}$ из $GF(q)$.

О п р е д е л е н и е 4. Произвольный код $\mathcal{C} \in \mathcal{E}_P(l, n_0, m, q)$ назовём q-ичным кодом с малой плотностью проверок, основанным на матрицах перестановок.

Одним из наиболее распространенных на практике и простых по структуре подансамблей ансамбля $\mathcal{E}_P(l, n_0, m, q)$ является ансамбль недвоичных “квазициклических” МПП-кодов, который будем обозначать $\mathcal{E}_Q(l, n_0, m, q)$. Этот ансамбль получается, если в качестве матриц $\mathbf{P}_{ji}$ выбираются элементы из подгруппы $\mathcal{H} \in \mathcal{P}_m$. $\mathcal{H}$ состоит из всех циклических сдвигов столбцов единичной $m \times m$ матрицы $\mathbf{I}$. Очевидно, что $|\mathcal{H}| = m$.

Произвольный код $\mathcal{C} \in \mathcal{E}_Q(l, n_0, m, q)$ назовём недвоичным квазициклическим МПП-кодом.

Как уже было отмечено, ансамбль $\mathcal{E}_Q(l, n_0, m, q)$ является подансамблем ансамбля $\mathcal{E}_P(l, n_0, m, q)$. Поскольку каждая из матриц $\mathbf{P}_{ji}$ недвоичного квазициклического МПП-кода полностью определяется одним целым числом – величиной циклического сдвига $0 \le r_{ji} \le m - 1$, то для хранения всей проверочной матрицы $\mathbf{H}$ достаточно $ln_0(m \log_2 q + \log_2 m)$ бит. Легко заметить, что для хранения полной матрицы из ансамбля $\mathcal{E}_P(l, n_0, m, q)$ потребовалось бы $mln_0(\log_2 q + \log_2 m)$ бит. Таким образом, обеспечивается почти m -кратная экономия памяти по сравнению с произвольным кодом из ансамбля $\mathcal{E}_P(l, n_0, m, q)$.

Для хранения проверочной матрицы q-ичного МПП-кода Галлагера [2] требуется $mln_0(\log_2 q + \log_2 n)$ бит. Таким образом, использование кода из ансамбля $\mathcal{E}_Q(l, n_0, m, q)$ также приводит примерно к m-кратной экономии памяти.

1.3 Алгоритмы декодирования

Алгоритмы декодирования можно разделить на алгоритмы с жёстким входом и алгоритмы с мягким входом.

Первый класс алгоритмов, известный также как вероятностное декодирование (probabilistic decoding) – это алгоритмы, которые в качестве входа получают оценку вероятностного распределения символов, полученную из канала, и работают с численными значениями вероятностей. Этот класс включает в себя такие алгоритмы, как sum-product (алгоритм “распространения доверия”) и его упрощённый вариант min-sum.

Второй класс включает в себя методы декодирования, работающие непосредственно со значениями символов. Это такие алгоритмы, как мажоритарное декодирование и алгоритм декодирования с введением стираний.

Хочется подчеркнуть отличия алгоритмов декодирования с мягким и с жёстким входом. Алгоритмы с мягким входом имеют теоретически лучшую корректирующую способность, причём в пределе хороших соотношений сигнал/шум выигрыш составляет 3 дБ, а в пределе плохих они совпадают. В то же время в реальных условиях может отсутствовать информация о надёжностях принятых символов, и алгоритм с мягким входом не будет применим непосредственно.

В этом контексте отдельный интерес представляет возможность адаптации алгоритмов с мягким входом для каналов с жёстким выходом.

1.3.1 Общие черты декодеров

Все рассматриваемые декодеры МПП-кодов работают с представлением кода в виде факторграфа, также известного как граф Таннера.

Рассматриваемые алгоритмы являются итерационными. Каждая итерация состоит из последовательной обработки сначала данных вершин-проверок, а затем вершин-переменных.

В каждом декодере остановка итеративной части алгоритма производится, когда все проверки оказались выполнены или при достижении максимального числа итераций. Возможны также дополнительные критерии остановки.

1.3.2 Мажоритарное декодирование

Мажоритарное декодирование является простейшим методом декодирования. Алгоритм имеет жёсткий вход и выполняет следующие итерации:

1. Вычисление проверок. Результатом являются данные о том, выполняется ли данная проверка, хранящаяся в данной вершине.

2. Изменение символов, в которых не выполняется более половины проверок, на противоположные.

Кроме основных условий остановки возможны дополнительные. Алгоритм может завершать свою работу, если:

– не изменился вес синдрома,

– не изменился синдром,

– не изменился вектор символов.

Данный алгоритм лежит в основе алгоритмов, работающих с жёстким представлением данных.

Модификацией данного алгоритма является вариант, когда на втором шаге итерации производится инвертирование тех символов, которые имеют максимальное число невыполненных проверок.

Алгоритм отличается вычислительной простотой: в нём выполняются только логические операции.

1.3.3 Декодер с введением стираний

Усовершенствованием алгоритма мажоритарного декодирования является алгоритм с введением стираний. В данном алгоритме вместо того, чтобы инвертировать оцененный как ошибочный символ, производится сначала его пометка как “стёртого”, а затем повторное вычисление проверок с учётом стёртых символов и восстановление символов по вычисленным проверкам. Таким образом алгоритм на каждой итерации выполняет следующие действия:

1. Вычисление проверок. Результатом являются данные о том, выполняется ли данная проверка, хранящаяся в данной вершине.

2. Пометка символов, оцененных как ошибочные, стёртыми. Как ошибочные можно помечать, например, символы, в которых не выполняется более половины проверок, или символы, в которых не выполняется максимальное число проверок.

3. Вычисление проверок с учётом стёртых символов. В случае, если в проверке стёрт единственный символ, то её значение может быть использовано только в этом символе.

4. Мажоритарное вычисление стёртых символов по проверкам: выбирается то значение, которое является предпочтительным исходя из значений символов. В случае, если по итогам вычисления символов нет предпочтительного значения, то восстанавливается предыдущее.

Данный алгоритм также отличается простотой, он лишь несколько сложнее алгоритма мажоритарного декодирования.

1.3.4 Алгоритм “распространения доверия”

Алгоритм “распространения доверия” – это итеративный алгоритм вероятностного декодирования. Он работает на графе Таннера, передавая сообщения между вершинами-переменными и вершинами проверками. Точная версия этого алгоритма известна как Sum-Product. Детальное математическое описание алгоритма, включая вывод формул, может быть найдено в [2,31]. Здесь будет приведён итоговый результат, важный для понимания дальнейших частей работы.

Введём некоторые обозначения:

LLR – логарифм соотношения правдоподобия,

α_n – знак LLR n-той переменной из канала,

β_n – модуль LLR n-той переменной из канала,

$\alpha_{n'}$ – вычисленный знак LLR n-той переменной,

$\beta_{n'}$ – вычисленный модуль LLR n-той переменной из канала,

γ_{mn} – сообщение от m-й проверки к n-й переменной,

α_{mn} – знак сообщения от n-й переменной к m-й проверке, принимает значения $+1$ или -1,

β_{mn} – модуль сообщения от n-й переменной к m-й проверке,

$N(m)$ – набор переменных n_i, которые участвуют в проверке m,

$N(m)\backslash n$ – набор $N(m)$ кроме бита n,

$M(n)$ – набор проверок, в которых участвует n-тая переменная,

$M(n)\backslash m$ – набор $M(n)$ кроме бита m,

Алгоритм работает следующим образом:

Инициализация. Присвоим $\forall m : \alpha_{mn}\beta_{mn} = \alpha_n\beta_n$.

Горизонтальный шаг. Вычисление сообщений от вершин-проверок к вершинам-переменным. При использовании логарифмов отношения правдоподобия оно будет выглядеть следующим образом:

$$\gamma_{mn} = \Big(\prod_{n' \in N(m)\backslash n} \alpha_{mn'} \Big) \cdot \phi\Big(\sum_{n' \in N(m)\backslash n} \phi(\beta_{mn'}) \Big), \tag{1.8}$$

где функция $\phi(x)$:

$$\phi(x) = \ln \frac{e^x + 1}{e^x - 1}. \tag{1.9}$$

Выражение $\phi(\sum \phi(\beta_i))$ может рассматриваться как мягкий минимум между всеми величинами β_i.

Вертикальный шаг. Вычисление сообщений от вершин-переменных к вершинам-проверкам:

$$\alpha_{n'}\beta_{n'} = \alpha_n\beta_n + \sum_{m\in M(n)} \gamma_{mn}, \tag{1.10}$$

$$\alpha_{mn}\beta_{mn} = \alpha_n\beta_n + \sum_{m'\in M(n)\setminus m} \gamma_{m'n}. \tag{1.11}$$

Горизонтальный и вертикальный шаги выполняются ограниченное число раз. В случае, если все проверки оказались выполнены, алгоритм может быть остановлен досрочно.

1.3.5 Функция $\phi(x)$

Отдельным предметом изучения будет поведение алгоритма в зависимости от особенностей реализации функции $\phi(x)$, поэтому остановимся на ней подробней.

Как можно заметить, в данной функции вычисляется экспонента от входной величины логарифма отношения правдоподобия. Если эта величина окажется достаточно большой, она переполнит переменную, используемую при вычислении, таким образом, что единица, которая добавляется или вычитается к экспоненте окажется меньше, чем единица младшего разряда при вычислениях с плавающей точкой. В таком случае вычисленное отношение окажется строго единичным, а логарифм – нулевым. При повторном вычислении этой функции от полученного значения в знаменателе будет ноль, что даст бесконечность в ответе и неопределённости на следующих итерациях.

Для обхода данной проблемы в практических вычислениях функция $\phi(x)$ заменяется следующим её приближением:

$$\phi(x) = \begin{cases} 0, & x > m, \\ m, & x < \phi(m), \\ \ln\frac{e^x+1}{e^x-1}, & \phi(m) < x < m, \end{cases} \tag{1.12}$$

где m – это максимальное значение аргумента, при котором $\exp(m) \pm 1$ вычисляется без переполнения.

Выбор m, в первую очередь, зависит от доступной точности вычислений. Очевидно, чтобы вычислить $e^x \pm 1$ без потери единицы, требуется относительное разрешение, равное e^{-x}. В двоичном представлении это соответствует $\log_2(e^x) = x \cdot log_2 e \approx 1.443 \cdot x$ значащим битам. Это приводит к ограничениям для разных точностей вычислений, которые приведены в таблице 1.1

В таблице выделены две ключевые для компьютерной симуляции величины: $m = 16$ или $m = 36$, которые оказываются максимальными величинами, когда достаточно разрядности вычислений с помощью чисел с плавающей точкой стандарта IEEE-754 с использованием одинарной или двойной точности соответственно.

m	Точность, бит	Заметки
5	7	
7	10	
10	15	
16	23	максимум для IEEE-754 single
36	52	максимум для IEEE-754 double

Таблица 1.1. Требуемая точность вычислений для различных ограничений на максимальное значение функции $\phi(x)$

Следует отметить, что в пределе, когда алгоритм “распространения доверия” выполняется на фактор-графе без циклов, он сходится к оценке максимального правдоподобия. В то же время при разумных длинах кода (от сотен до сотен тысяч символов) это условие может быть выполнено только для небольшого числа итераций порядка 2–4, а далее сообщения становятся скоррелированными и несут смысл не вероятностей, а оценок надёжности символов. В то же время результативным является выполнение существенно большего числа итераций, порядка десятков.

Алгоритм “распространения доверия” минимизирует вероятность ошибки на бит. При этом вероятность ошибки кодового слова получается выше, чем у алгоритмов, минимизирующих вероятность ошибки на кодовое слово.

Sum-Product имеет высокую вычислительную сложность: на каждой итерации происходит вычисление сообщений, происходящее с целыми числами, а также вычисление нетривиального вида функции.

1.3.6 Min-Sum

Алгоритм Min-Sum (сумма наименьших) является упрощением алгоритма Sum-Product и обладает существенно меньшей вычислительной сложностью. Каждая итерация содержит два шага:

1. Вычисление сообщений от проверок к символам. В логарифмах отношения правдоподобия формулы выглядят следующим образом:

$$\gamma_{mn} = \prod_{n' \in N(m) \backslash n} \alpha_{mn'} \cdot \min_{n' \in N(m) \backslash n} \beta_{mn'}, \tag{1.13}$$

где γ_{mn} – сообщение от m-й проверки n-му символу, α_{mn} – знак сообщения от n-го символа для m-й проверки, β_{il} – модуль (абсолютное значение) сообщения от n-го символе для m-й проверки, $N(m)$ – множество символов в m-й проверке, $N(m) \backslash n$ – множество символов в m-й проверке кроме n.

2. Вычисление сообщений от символов проверкам. В логарифмах отношения правдоподобия формулы выглядят следующим образом:

$$\alpha_{n'}\beta_{n'} = \alpha_n\beta_n + \sum_{m\in M(n)} \gamma_{mn}, \tag{1.14}$$

$$\alpha_{mn}\beta_{mn} = \alpha_n\beta_n + \sum_{m'\in M(n)\setminus m} \gamma_{m'n}. \tag{1.15}$$

Min-Sum имеет меньшую вычислительную сложность по сравнению с Sum-Product благодаря отсутствию необходимости вычислять нетривиальные функции, однако остаётся существенно сложнее алгоритмов мажоритарного декодирования и алгоритма с введением стираний из-за необходимости производить арифметические операции.

1.4 Применение мягких алгоритмов декодирования для каналов с жёстким решением

1.4.1 Введение

Особенный интерес представляет использование алгоритмов с мягким входом совместно с каналом с жёстким выходом. В то время как обратное использование очевидно – декодеру с жёстким решением можно просто подать более вероятные значения входных символов канала с мягким выходом – обратное преобразование требует замены имеющихся на входе символов после жёсткого решения на некоторые оценки надёжности символов.

В данной части будет рассматриваться вопрос преобразования символов с жёсткого выхода канала для мягкого входа декодера.

Для начала сделаем некоторое отступление о разнице в работе итеративных алгоритмов с жёстким и с мягким представлением внутренней информации. Очевидно, что в итеративных алгоритмах на каждой итерации может быть посчитано значение числа ошибок. Эксперименты показывают, что при использовании декодера Sum-Product с мягким выходом из канала (в штатном режиме) число ошибок обычно монотонно падает от итерации к итерации. Интересной особенностью алгоритмов Sum-Product и Min-Sum является то, что кроме дискретного числа ошибок можно посчитать аналогичную непрерывную величину – сумму вероятностей ошибок по всем символам. Для этого нужно для каждого символа вычислить оценки надёжности в виде вероятностей p_i того, что символ нулевой, после чего для символов, которые при передаче были единичными посчитать величину $1 - p_i$ и просуммировать. В случае, если выполняется декодирование нулевого кодового слова, это будет просто сумма оценок надёжности p_i. В контексте алгоритма Sum-Product данная величина интересна тем, что позволяет получить более глубокое понимание происходящих процессов – она усредняет именно те оценки, с которыми работает алгоритм. Рассмотрение её поведения показывает, что в типичной ситуации (рис. 1.2)

успешного декодирования она практически совпадает с числом ошибок и при работе алгоритма от итерации к итерации меняется похожим образом. В случае, если на вход декодера подаётся мягкий выход канала, вероятности которого изменены в сторону увеличения или уменьшения надёжности, оценки суммы вероятностей ошибок соответственно изменяются, увеличивается вероятность отказа от декодирования, а при отсутствии отказа при занижении надёжностей время декодирования увеличивается (растёт число итераций), при завышении — уменьшается.

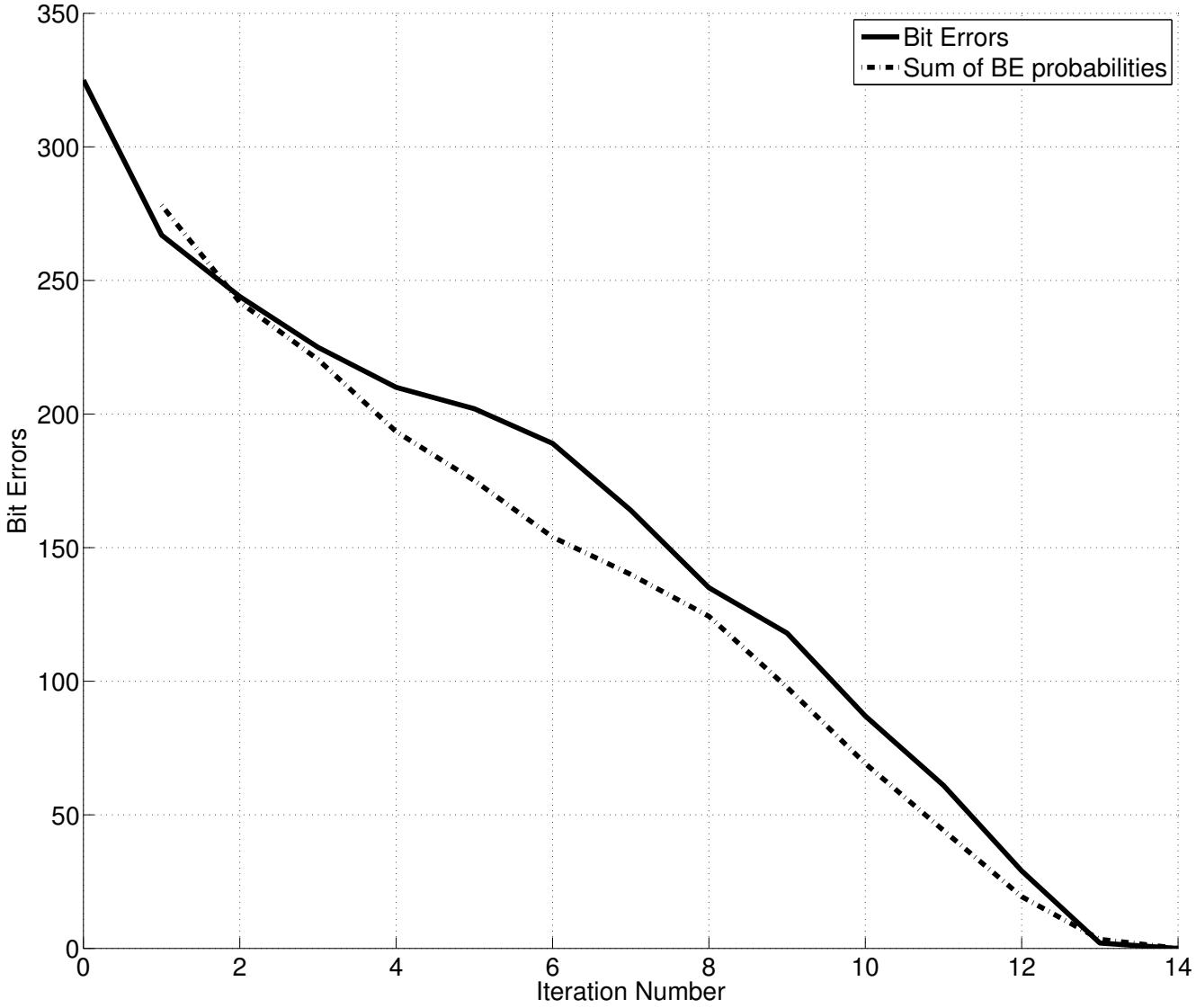


Рисунок 1.2. Пример зависимости числа ошибок (сплошная линия) и суммы вероятностей ошибок (штриховая линия) от номера итерации Sum-Product при соотношении сигнал/шум -0.7 дБ

Отсюда можно сделать вывод, что оптимальным может являться такой вход, который обеспечивает равенство суммы вероятностей ошибок, используемой декодером, реальному числу ошибок в кодовом слове.

Вариантом такого входа для двоичного кода является замена входных символов в виде 0 и 1 на вероятности p и $1-p$, где p – оценка вероятности ошибки в данном канале. Логарифмы отношения правдоподобия при этом выглядят как π и $-\pi$, где $\pi = \ln \frac{1-p}{p}$.

При использовании этого метода сумма вероятностей ошибок ведёт себя аналогично случаю работы с мягким выходом канала (рис. 1.3): практически совпадает с числом ошибок, от итерации к итерации меняется подобно ей. При этом при занижении оценки надёжности алгоритм сходится более медленно и наоборот.

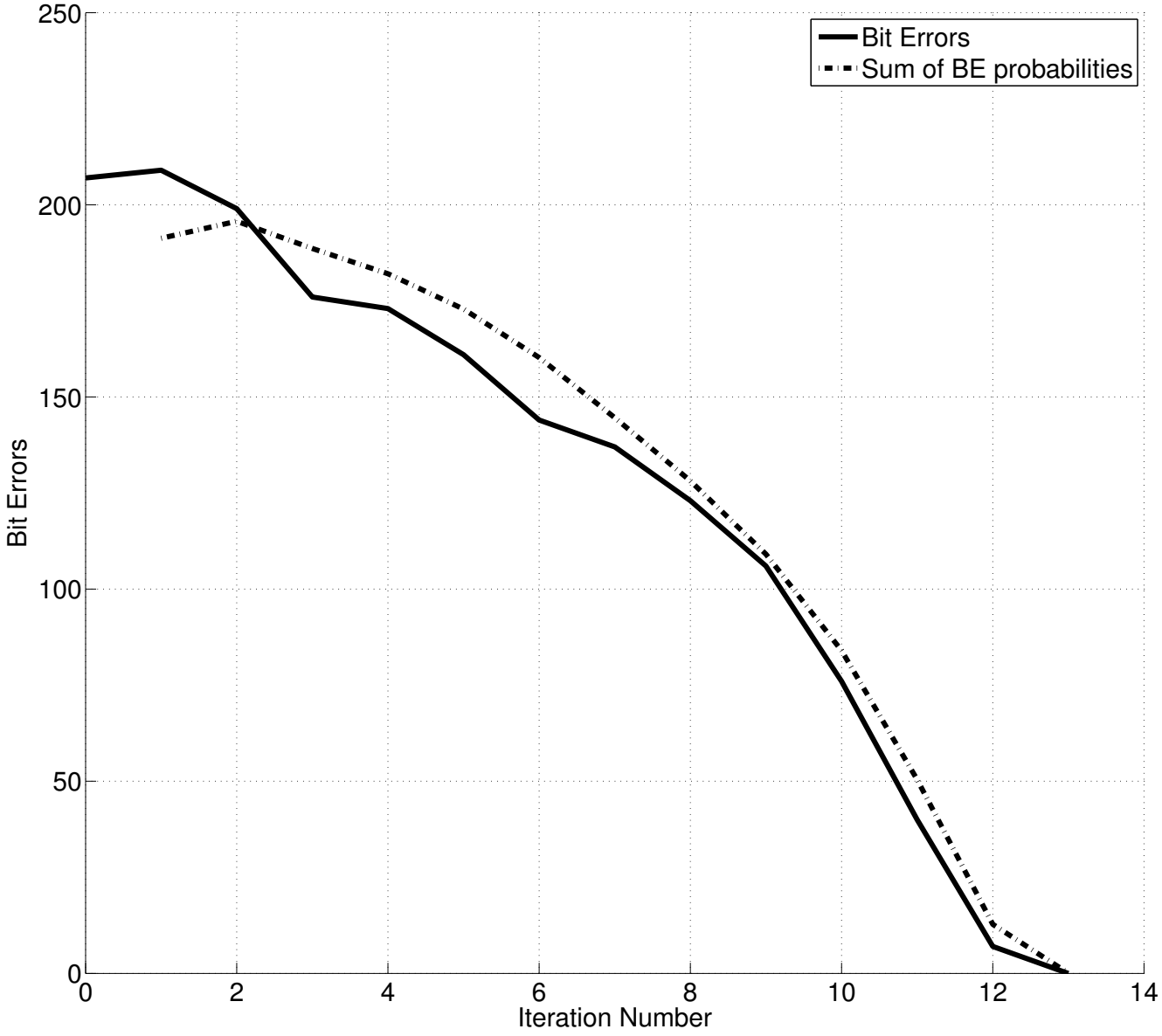


Рисунок 1.3. Пример зависимости числа ошибок (сплошная линия) и суммы вероятностей ошибок (штриховая линия) от номера итерации Sum-Product при подаче на вход оценок вида $\pm\pi = \ln\frac{1-p}{p}$ при соотношении сигнал/шум +0.6 дБ

Хочется отметить, что в данном случает с непосредственно жёсткими решениями алгоритм работает только на первой итерации – далее символам присваиваются оценки надёжности и алгоритм продолжает работу уже с ними. В отличие от алгоритмов с жёстким внутренним представлением оценок символов это позволяет сохранить и использовать существенную часть данных и тем самым улучшить эффективность работы.

1.4.2 Способы оценки надёжностей

Способы оценки вероятности ошибки p могут быть различны. Было проведено сравнение трёх способов:

1. Вычисление эмпирической вероятности как отношения числа ошибок в кодовом слове к его длине:
$$p = \frac{E}{n} = \frac{\sum_i |r_i - s_i|}{n}. \tag{1.16}$$
2. Расчёт коэффициента ошибок канала через известное соотношение сигнал/шум. Для двоичной фазовой манипуляции и канала с аддитивным белым гауссовским шумом он вычисляется следующим образом:
$$p = \frac{1}{\sqrt{\pi}} \int_1^\infty \exp[-\frac{t^2}{2\sigma^2}]dt, \sigma^2 = \frac{1}{2 \cdot SNR}. \tag{1.17}$$
3. Использование фиксированной константы, соответствующей работе при наихудшем соотношении сигнал/шум, при котором вероятность ошибки на блок становится меньше 10^{-1}–10^{-2}.

1.4.3 Результаты моделирования

Моделирование работы декодеров производилось методами имитационного моделирования, для чего была написана их реализация на языке Си в виде функции для MatLab. Для сравнения декодеров использовался двоичный МПП-код длины 3280 и скорости $1/2$. В качестве канала использовался канал с двоичной фазовой модуляцией с аддитивным белым гауссовским шумом.

Было произведено моделирование нескольких режимов работы декодеров.

Сравнение различных способов преобразования жёсткого выхода канала в мягкий вход декодера (непосредственное вычисление числа ошибок, расчёт на основе соотношения сигнал/шум, а также использование константы) не показало статистически значимого отличия для Sum-Product.

Результаты работы алгоритмов Sum-Product и Min-Sum приведены для случая подачи на него значений с фиксированной константой.

Декодеры с жёсткой логикой сравнивались в вариантах, когда символы считались ненадёжными в случае невыполнения наибольшего числа проверок среди всех символов, результаты приведены как для мажоритарного декодера, так и для декодера с введением стираний.

График зависимости числа ошибок от соотношения сигнал/шум показан на рисунке 1.4. Общее число экспериментов составило $2 \cdot 10^4$ кодовых слов или $6.56 \cdot 10^7$ бит. Результаты сравнивались по уровню вероятности ошибки, равному 10^{-5}.

На графике виден выигрыш порядка 1.1 дБ при применении алгоритма Sum-Product по сравнению с алгоритмами, работающими с жёстким представлением символов (соотношение сигнал/шум 1.2 дБ против 2.3 дБ). Видно, что алгоритм Min-Sum оказался практически неприме-

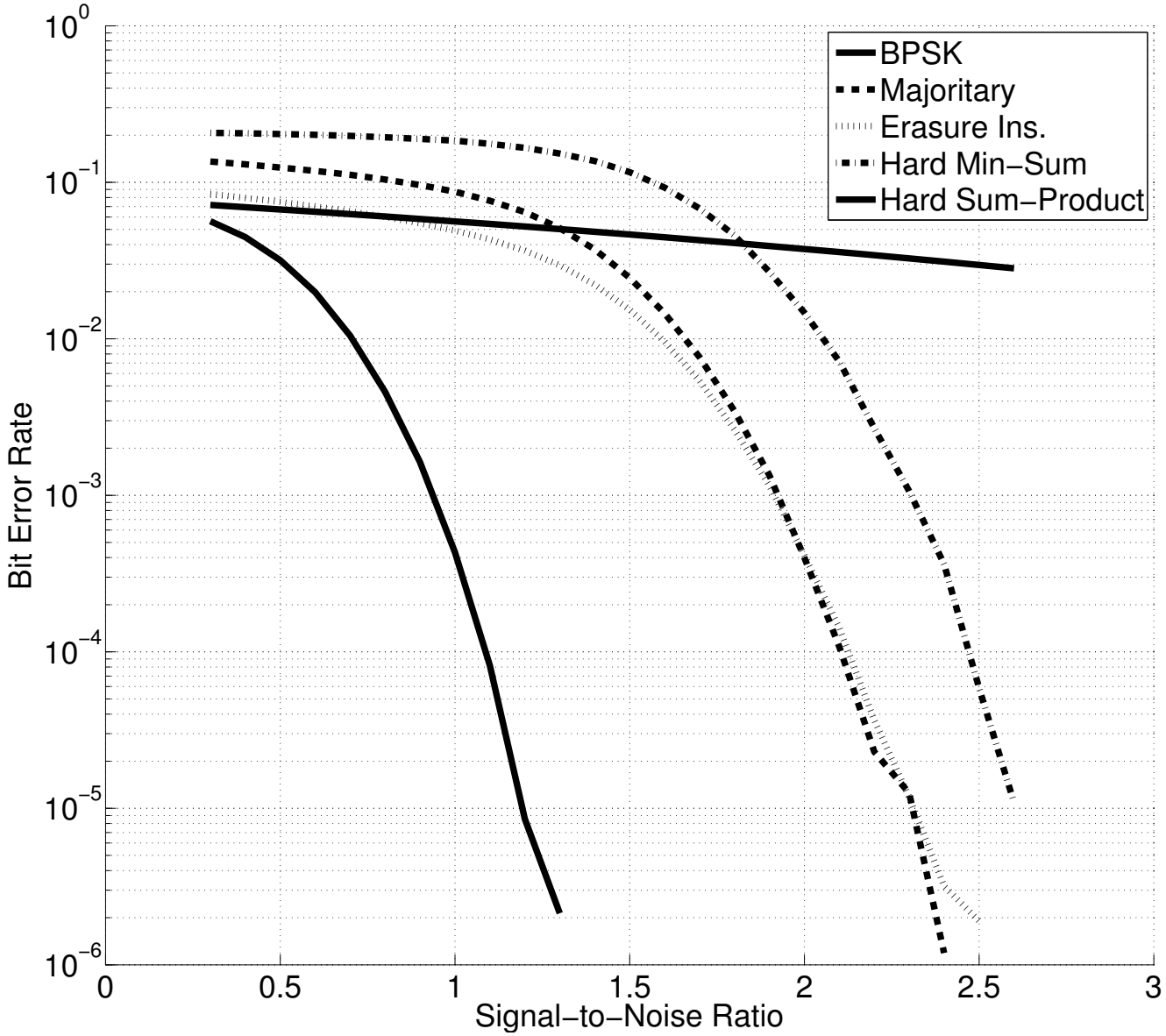


Рисунок 1.4. График зависимости числа ошибок от соотношения сигнал/шум для различных алгоритмов декодирования

ним для канала с жёстким выходом, показывая результаты на 0.3 дБ хуже, чем у мажоритарного декодирования.

1.5 ЕП-МПП-коды

1.5.1 Введение

Коды с единичной памятью (ЕП) были введены Ли в 1976 [32]. Это свёрточные коды со скоростью $R = k/n$, памятью $m = 1$ и длиной кодового ограничения $\nu \leq k$. В случае, когда $\nu < k$, эти коды называются кодами с частичной единичной памятью (ЧЕП). (Ч)ЕП коды строятся на основе блоковых кодов, например кодов Рида-Соломона [33, 34], кодов БЧХ [35], или

кодов с малой плотностью проверок [36]. Использования блоковых кодов делает возможным алгебраическое описание этих свёрточных кодов и упрощает их изучение.

В данной работе рассматриваются коды с единичной памятью с циклическим замыканием[1], основанные на блоковых МПП-кодах [2], и их декодирование. Назовём их (Ч)ЕП-МПП-кодами.

1.5.2 Конструкция

Рассмотрим конструкцию проверочной матрицы $\mathbf{H}'$ ЕП-МПП-кода, основанного на наборе МПП-кодов Галлагера. [2]. Обозначим проверочные матрицы этих компонентных кодов $\mathbf{H}_{i,0}, \mathbf{H}_{i,1}$, $i = \overline{1..t}$, где t — период. Тогда ЕП-МПП-код определяется его полубесконечной проверочной матрицей $\mathbf{H}'$:

$$\mathbf{H}' = \begin{pmatrix} \mathbf{H}_{1,1} & \mathbf{H}_{1,0} & & & & & \\ & \mathbf{H}_{2,1} & \mathbf{H}_{2,0} & & & & \\ & & \ddots & \ddots & & & \\ & & & \mathbf{H}_{t,1} & \mathbf{H}_{t,0} & & \\ & & & & \mathbf{H}_{1,1} & \mathbf{H}_{1,0} & \\ & & & & & \ddots & \ddots \end{pmatrix}, \tag{1.18}$$

где $\mathbf{H}_{i,0}$, $\mathbf{H}_{i,1}$ — матрицы $r \times n$, $r = n - k$. И для ЕП, и для ЧЕП кода блочная матрица $\mathbf{H}_{i,0}$ должна иметь полный ранг, а $\mathbf{H}_{i,1}$ может иметь меньший ранг, если это ЧЕП код: $rank\,(\mathbf{H}_{i,0}) = r$, $rank\,(\mathbf{H}_{i,1}) = r_1 \leq r$. Кодовая скорость R' построенного ЕП-МПП-кода равна кодовой скорости кода с проверочной матрицей $\mathbf{H}_{i,0}$.

Тогда ЕП-МПП-код с циклическим замыканием (ЦЗ-ЕП-МПП) длины $N = nt$ (период циклического замыкания равен t) имеет следующую проверочную матрицу $\mathbf{H}$:

$$\mathbf{H} = \begin{pmatrix} \mathbf{H}_{1,1} & \mathbf{H}_{1,0} & & & \\ & \mathbf{H}_{2,1} & \mathbf{H}_{2,0} & & \\ & & \ddots & \ddots & \\ \mathbf{H}_{t,0} & & & & \mathbf{H}_{t,1} \end{pmatrix} \tag{1.19}$$

размера $tr \times tn$ (согласно требованиям на ЕП-МПП-код).

Таким образом, кодовая скорость ЦЗ-ЕП-МПП-кода равна:

$$R \geq 1 - \frac{tr}{tn} = 1 - \frac{r}{n} = R_{i,\{0,1\}}. \tag{1.20}$$

Замечание 1. *Стоит отметить, что построенный ЦЗ-ЕП-МПП-код тоже является МПП-кодом со специальной конструкцией проверочной матрицы.*

[1] Tail-Biting

Также введём следующее обозначение:

$$\mathbf{H}_i = \begin{pmatrix} \mathbf{H}_{i,1} & \mathbf{H}_{i,0} \end{pmatrix}, \tag{1.21}$$

где $\mathbf{H}_i$ — проверочная матрица МПП-кода, построенная конкатенацией проверочных матриц $\mathbf{H}_{i,1}$ и $\mathbf{H}_{i,0}$.

1.5.3 Алгоритмы декодирования

Этот раздел посвящён описанию алгоритмов декодирования описанных выше ЕП-МПП-кодов, которые рассматриваются в этой работе. Вход этих алгоритмов — принятая последовательность $\mathbf{r}$, полученная после передачи кодового слова $\mathbf{c}$ через двоичный канал без памяти с аддитивным белым гауссовским шумом (АБГШ). В соответствии с конструкцией ЕП-МПП-кодов, принятая последовательность $\mathbf{r}$ может быть записана как $\mathbf{r} = \begin{pmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \dots & \mathbf{r}_t \end{pmatrix}$, где $\mathbf{r}_i$ — вектор длины n мягких решений для принятой последовательности (логарифмов отношения правдоподобий), который соответствует проверочным матрицам $\mathbf{H}_{i,1}$ и $\mathbf{H}_{((i-1) \bmod t)+1,0}$.

Алгоритм декодирования $\mathcal{A}$

Как было упомянуто выше, ЕП-МПП-код с циклическим замыканием тоже является МПП-кодом. Таким образом, первый алгоритм декодирования $\mathcal{A}$ — это хорошо известный алгоритм “распространения доверия” декодирования МПП-кодов. В этом случае проверочная матрица $\mathbf{H}$ ЕП-МПП-кода с циклическим замыканием рассматривается целиком как проверочная матрица некоторого МПП-кода и декодируется при помощи классического алгоритма “распространения доверия”. Обозначим

$$\mathbf{y} = D_{\mathbf{H}}^{(i_{max})}(\mathbf{r}) \tag{1.22}$$

операцию декодирования принятой последовательности $\mathbf{r}$ алгоритмом $\mathcal{A}$ с i_{max} итерациями ($\mathcal{A}(i_{max})$) для проверочной матрицы $\mathbf{H}$ и обновлённых величин $\mathbf{y}$.

Два следующих алгоритма используют не проверочную матрицу $\mathbf{H}$ целиком, а составляющие её проверочные матрицы $\mathbf{H}_i$, $1 \le i \le t$ на отдельных итерациях декодирования. Главное отличие этих алгоритмов — метод обмена решениями, сделанными при декодировании компонентных кодов, между этими компонентными кодами.

Алгоритм декодирования $\mathcal{B}$

Основная идея второго алгоритма декодирования $\mathcal{B}$ следующая. ЕП-МПП-код с циклическим замыканием рассматривается как сочетание нескольких компонентных кодов, соответствующих проверочным матрицам $\mathbf{H}_i$, $i \in \overline{1,t}$. Эти коды декодируются последовательно так же, как при декодировании свёрточных кодов. Решения, принятые при декодировании на первом шаге, используются как входные значения для перекрывающейся части предыдущего и текущего ком-

понентных кодов (для неперекрвающейся части используются значения из *неотдекодированной* части). Но из-за циклического замыкания кода число таких итераций фиксировано. Таким образом, процедура повторяется от первого до последнего шага. При описании этого алгоритма требуется отличать внутренние итерации декодирования (проводимые на каждом шагу) и внешние (число повторений последовательности шагов). Поэтому обозначим этот алгоритм $\mathcal{B}\left(i_{max}, j_{max}\right)$, где i_{max} — число внутренних итераций (число итераций декодера "распространения доверия"), а j_{max} — число внешних итераций (число раз, которое декодируются все t компонентных кодов).

Обозначим

$$\mathbf{y} = D_k^{(i_{max})}\left(\mathbf{x}\right) \tag{1.23}$$

декодирование *входной* последовательности $\mathbf{x}$ алгоритмом $\mathcal{A}\left(i_{max}\right)$ с проверочной матрицей $\mathbf{H}_k$, где $\mathbf{y}$ — обновлённые величины логарифмов отношения правдоподобия. Тогда алгоритм декодирования $\mathcal{B}\left(i_{max}, j_{max}\right)$ может быть записан следующим образом:

1: $\mathbf{r}_k^{(0)} \longleftarrow \mathbf{r}_k, \forall k : 1 \leq k \leq t$ # входные значения
2: $\mathbf{r}_1^{(1)} \longleftarrow \mathbf{r}_1$ # начальное рабочее значение
3: $\mathbf{\Delta}_k^{(0)} \longleftarrow 0, \forall k : 1 \leq k \leq t$ # изменения
4: **for** $j = 1$ **to** j_{max} **do** # j_{max} внешних итераций
5: **for** $k = 1$ **to** t **do** # для $\forall$ из t компонентных
6: $k_1 \longleftarrow k$, $k_2 \longleftarrow (k \bmod t) + 1$ # индексы "левой" и "правой" частей
7: $\mathbf{x}_{k_1} \longleftarrow \mathbf{r}_{k_1}^{(j)}$ # вычислена только что
8: $\mathbf{x}_{k_2} \longleftarrow \mathbf{r}_{k_2}^{(j-1)}$ # с предыдущей итерации
9: $\begin{pmatrix}\mathbf{y}_{k_1} & \mathbf{y}_{k_2}\end{pmatrix} \longleftarrow D_k^{(i_{max})}\left(\begin{pmatrix}\mathbf{x}_{k_1} & \mathbf{x}_{k_2}\end{pmatrix}\right)$
10: $\mathbf{\Delta}_{k_2}^{(j)} \longleftarrow \mathbf{y}_{k_2} - \mathbf{x}_{k_2}$ # разность для правой части
11: $\mathbf{r}_{k_2}^{(j)} \longleftarrow \mathbf{r}_{k_2}^{(j-1)} + \mathbf{\Delta}_{k_2}^{(j)}$ # выход для правой части
12: **end for**
13: **end for**
14: **return** $\mathbf{r}^{(j_{max})} = \left(\mathbf{r}_1^{(j_{max})}\mathbf{r}_2^{(j_{max})} \ldots \mathbf{r}_t^{(j_{max})}\right)$

Можно заметить, что на каждом шаге k текущей внешней итерации j обновляется правая часть $\mathbf{r}_{k_2}^{(j)}$ (которая соответствует проверочной матрице $\mathbf{H}_{k,0}$) текущей *декодируемой* последовательности $\begin{pmatrix}\mathbf{r}_{k_1}^{(j)} & \mathbf{r}_{k_1}^{(j)}\end{pmatrix}$ (которая соответствует проверочной матрице $\mathbf{H}_k$). После этого обновлённая правая часть, полученная на предыдущем шаге, используется как вход для левой части на текущем шаге $k+1$. Таким образом, правые части компонентных кодов постоянно обновляются в направлении слева направо.

Алгоритм декодирования $\mathcal{C}$

Третий алгоритм декодирования $\mathcal{C}$ — это модификация алгоритма декодирования $\mathcal{B}$, описанного выше. Согласно описанию алгоритма $\mathcal{B}$, он последовательно обновляет значения *промежуточной* последовательности LLR в направлении слева направо. Алгоритм $\mathcal{C}$ может быть

представлен как два параллельных алгоритма $\mathcal{B}$ с противоположным направлением обновления значений LLR.

Согласно конструкции ЕП-МПП-кода каждый символ вектора $\mathbf{r}_k$, $1 \leq k \leq t$ входит в два компонентных кода $H_{k,1}$ и $H_{(k \bmod t)+1,0}$. Обозначим как раньше

$$\mathbf{y} = D_k^{(i_{max})}(\mathbf{x}) \tag{1.24}$$

декодирование *рассматриваемой* последовательности $\mathbf{x}$ алгоритмом $\mathcal{A}(i_{max})$ с проверочной матрицей $\mathbf{H}_k$, где $\mathbf{y}$ — обновлённые значения LLR. И пусть $\mathbf{\Delta}_{k,1}$ и $\mathbf{\Delta}_{k,0}$ — изменения LLR, соответствующие проверочным матрицам $\mathbf{H}_{k,1}$ и $\mathbf{H}_{k,0}$ соответственно. Тогда описание алгоритма декодирования $\mathcal{C}(i_{max}, j_{max})$ выглядит следующим образом:

1: $\mathbf{r}_k^{(0)} \longleftarrow \mathbf{r}_k$, $\forall k : 1 \leq k \leq t$
2: $\mathbf{\Delta}_{k,\{0,1\}}^{(0)} \longleftarrow 0$, $\forall k : 1 \leq k \leq t$
3: **for** $j = 1$ **to** j_{max} **do**
4: **for** $k = 1$ **to** t **do**
5: $k_1 \longleftarrow k$, $k_2 \longleftarrow (k \bmod t) + 1$
6: $\mathbf{x}_{k_1} \longleftarrow \mathbf{r}_{k_1}^{(j-1)} - \mathbf{\Delta}_{k_1,1}^{(j-1)}$
7: $\mathbf{x}_{k_2} \longleftarrow \mathbf{r}_{k_2}^{(j-1)} - \mathbf{\Delta}_{k_2,0}^{(j-1)}$
8: $\begin{pmatrix} \mathbf{y}_{k_1} & \mathbf{y}_{k_2} \end{pmatrix} \longleftarrow D_k^{(i_{max})} \left(\begin{pmatrix} \mathbf{x}_{k_1} & \mathbf{x}_{k_2} \end{pmatrix} \right)$
9: $\mathbf{\Delta}_{k_1,1}^{(j)} \longleftarrow \mathbf{y}_{k_1} - \mathbf{x}_{k_1}$
10: $\mathbf{\Delta}_{k_2,0}^{(j)} \longleftarrow \mathbf{y}_{k_2} - \mathbf{x}_{k_2}$
11: **end for**
12: **for** $k = 1$ **to** t **do**
13: $\mathbf{r}_k^{(j)} \longleftarrow \mathbf{r}_k^{(j-1)} + \mathbf{\Delta}_{k,0}^{(j)} + \mathbf{\Delta}_{k,1}$
14: **end for**
15: **end for**
16: **return** $\mathbf{r}^{(j_{max})} = \left(\mathbf{r}_1^{(j_{max})} \mathbf{r}_2^{(j_{max})} \dots \mathbf{r}_t^{(j_{max})} \right)$

Другими словами на каждой итерации j декодируются все t компонентных кодов с проверочными матрицами $\mathbf{H}_k$, $1 \leq k \leq t$ и вычисляются соответсвующие изменения $\mathbf{\Delta}_{k_1,1}^{(j)}$ и $\mathbf{\Delta}_{k_1,1}^{(j)}$. Затем на следующей итерации сумма *рабочих* LLR предыдущей итерации и изменения от двух соседей используется как вход каждого из декодеров компонентных кодов на следующей итерации. При этом для левой половины используются изменения, вычисленные декодером слева для своей правой части, и наоборот.

1.5.4 Результаты моделирования

Результаты моделирования были получены для ЕП-МПП-кода с периодом $t = 4$, основанном на таких МПП-кодах (2,4) с проверочными матрицами $\mathbf{H}_{i,\{0,1\}}$, что проверочные матрицы $\mathbf{H}_i =$

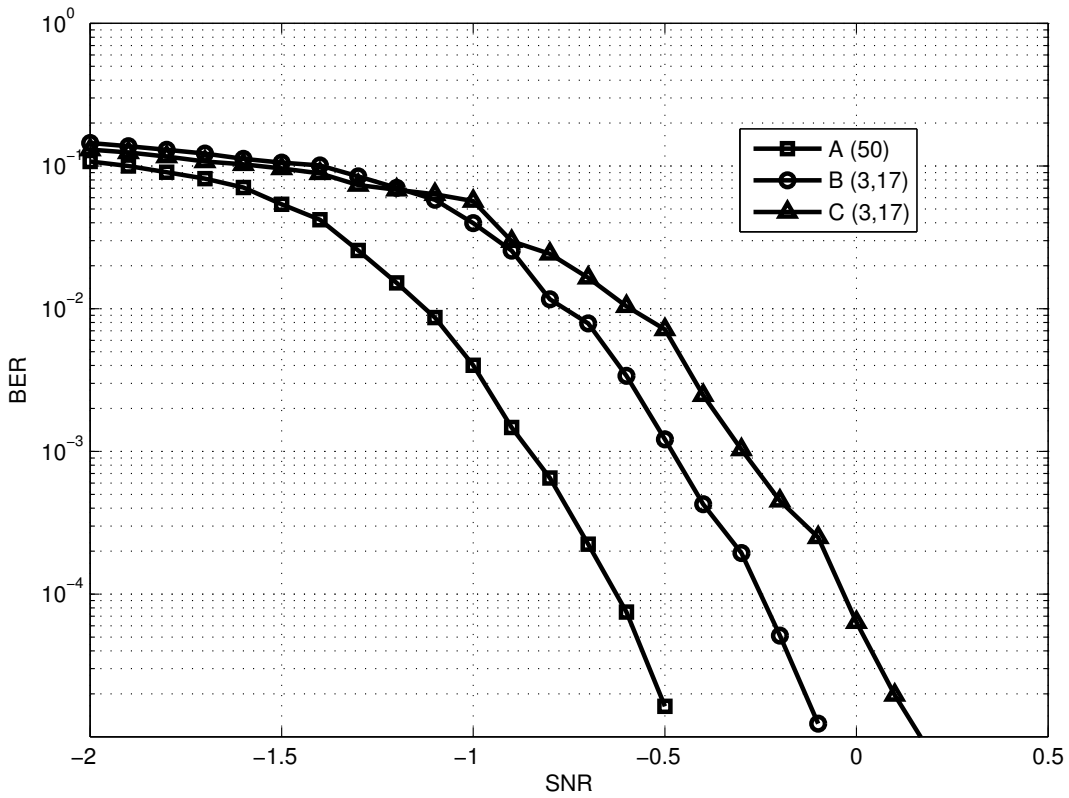


Рисунок 1.5. Результаты моделирования ЕП-МПП-кода скорости R=0.5, основанного на МПП-кодах (2,4), при декодировании алгоритмами $\mathcal{A}(50)$, $\mathcal{B}(3,17)$ and $\mathcal{C}(3,17)$

$\begin{pmatrix}\mathbf{H}_{i,1} & \mathbf{H}_{i,0}\end{pmatrix}$ не содержали циклов длины меньше 4,

$$\mathbf{H} = \begin{pmatrix} \mathbf{H}_{1,1} & \mathbf{H}_{1,0} & & \\ & \mathbf{H}_{2,1} & \mathbf{H}_{2,0} & \\ & & \mathbf{H}_{3,1} & \mathbf{H}_{3,0} \\ \mathbf{H}_{3,0} & & & \mathbf{H}_{3,1} \end{pmatrix}. \tag{1.25}$$

Кодовая скорость построенного ЕП-МПП-кода равна $R = 1 - \frac{2}{4} = 0.5$, код имеет длину $N = 2032$.

Число итераций алгоритма декодирования $\mathcal{A}(i_{max})$ было выбрано равным 50 ($i_{max} = 50$). Согласно описаниям алгоритмов декодирования $\mathcal{B}(i_{max}, j_{max})$ и $\mathcal{C}(i_{max}, j_{max})$ каждый компонентный код декодируется j_{max} алгоритмом $\mathcal{A}(i_{max})$. Таким образом, число внутренних i_{max} и внешних j_{max} итераций должно быть выбрано таким, чтобы $i_{max} j_{max} = 50$. В результате моделирования было выяснено, что для рассматриваемых ЕП-МПП-кодов наилучшая эффективность декодирования была получена при использовании алгоритмов $\mathcal{B}(3,17)$ и $\mathcal{C}(3,17)$. Этот факт может быть объяснён следующим наблюдением. Согласно конструкции компонентных МПП-кодов они не имеют циклов длины 4. Отсюда можно предположить, что число независимых итераций декодирования равно как минимум 3. Таким образом, число внутренних итераций должно быть выбрано равным 3.

На рис. 1.5 приведены результаты моделирования ЕП-МПП-кода с кодовой скростью $R = 0.5$, основанного на МПП-кодах (2,4), при декодировании алгоритмами $\mathcal{A}(50)$, $\mathcal{B}(3,17)$ и $\mathcal{C}(3,17)$.

Как можно видеть из рисунка 1.5, наилучшей эффективностью декодирования обладает алгоритм $\mathcal{A}(50)$. Алгоритм декодирования $\mathcal{B}(3,17)$ проигрывает алгоритму $\mathcal{A}(50)$ почти 0.4 дБ по уровню вероятности ошибки на бит, равной 10^{-5}. Эффективность алгоритма $\mathcal{C}(3,17)$ ещё на 0.2 дБ хуже, чем алгоритма $\mathcal{B}(3,17)$.

Алгоритмы $\mathcal{B}$ и $\mathcal{C}$ имеют такую же асимптотическую сложность, как и алгоритм $\mathcal{A}$ с точность до постоянного множителя, близкого к 1. Так как алгоритм $\mathcal{A}$ работает лучше, чем алгоритмы $\mathcal{B}$ и $\mathcal{C}$, применение последних двух алгоритмов является нецелесообразным.

1.6 Векторизация вычислений алгоритма “распространения доверия” для МПП-кодов, основанных на матрицах перестановок

1.6.1 Введение

Ниже предложена модификация алгоритма декодирования $Q-SPA$ для случая, когда проверочная матрица $\mathbf{H}$ недвоичного кода с малой плотностью проверок состоит из произвольных матриц перестановок, умноженных на произвольные диагональные матрицы с элементами из мультипликативной группы поля $GF^{\star}(q)$. Предложенный декодер имеет векторную реализацию, то есть все производимые им операции выполняются не над отдельными элементами поля $GF(q)$, а над векторами, заданными над этим полем. Данный подход позволяет существенно ускорить декодирование за счёт использования параллельных вычислителей, таких, как графический процессор общего назначения (видеоускоритель). Такой подход позволяет существенно увеличить локальность доступа к памяти.

1.6.2 Вычисление синдрома для недвоичного МПП-кода, основанного на матрицах перестановок

Пусть

$$\mathbf{H}=\begin{pmatrix} \boldsymbol{\pi}_{11}(diag(\mathbf{S}_{11})) & \boldsymbol{\pi}_{12}(diag(\mathbf{S}_{12})) & \dots & \boldsymbol{\pi}_{1n_0}(diag(\mathbf{S}_{1n_0})) \\ \boldsymbol{\pi}_{21}(diag(\mathbf{S}_{21})) & \boldsymbol{\pi}_{22}(diag(\mathbf{S}_{22})) & \dots & \boldsymbol{\pi}_{2n_0}(diag(\mathbf{S}_{2n_0})) \\ \dots & \dots & \dots & \dots \\ \boldsymbol{\pi}_{l1}(diag(\mathbf{S}_{l1})) & \boldsymbol{\pi}_{l2}(diag(\mathbf{S}_{l2})) & \dots & \boldsymbol{\pi}_{ln_0}(diag(\mathbf{S}_{ln_0})) \end{pmatrix} \tag{1.26}$$

— проверочная матрица регулярного q-ичного (l,n_0)-кода с малой плотностью проверок, причем размер матрицы $\boldsymbol{\pi}_{ji}(diag(\mathbf{S}_{ji}))$ равен $m\times m$.

Поскольку длина МПП-кода с проверочной матрицей $\mathbf{H}$ равна $n=mn_0$, то кодовое слово $\mathbf{c}=(c_1,c_2,\dots,c_n)$, $c_i\in GF(q)$, можно представить в следующем виде:

$$\mathbf{c}=(\bar{\mathbf{c}}_1,\dots,\bar{\mathbf{c}}_{n_0}), \tag{1.27}$$

где $\bar{\mathbf{c}}_i$ — q-ичный вектор длины m. Синдром $\mathbf{S}$ для принятого слова $\mathbf{y}$ вычисляется по формуле $\mathbf{S} = \mathbf{H}\mathbf{y}^{\mathbf{T}}$, причем $\mathbf{S} = \mathbf{0}$ тогда и только тогда, когда $\mathbf{y}$ является кодовым словом. Последнее соотношение может быть записано в виде однородной системы из l уравнений:

$$\begin{cases} \boldsymbol{\pi}_{11}(\mathbf{S}_{11} \cdot \bar{\mathbf{c}}_1) + \boldsymbol{\pi}_{12}(\mathbf{S}_{12} \cdot \bar{\mathbf{c}}_2) + \ldots + \boldsymbol{\pi}_{1n_0}(\mathbf{S}_{1n_0} \cdot \bar{\mathbf{c}}_{n_0}) = \mathbf{0} \\ \boldsymbol{\pi}_{21}(\mathbf{S}_{21} \cdot \bar{\mathbf{c}}_1) + \boldsymbol{\pi}_{22}(\mathbf{S}_{22} \cdot \bar{\mathbf{c}}_2) + \ldots + \boldsymbol{\pi}_{2n_0}(\mathbf{S}_{2n_0} \cdot \bar{\mathbf{c}}_{n_0}) = \mathbf{0} \\ \ldots \\ \boldsymbol{\pi}_{l1}(\mathbf{S}_{l1} \cdot \bar{\mathbf{c}}_1) + \boldsymbol{\pi}_{l2}(\mathbf{S}_{l2} \cdot \bar{\mathbf{c}}_2) + \ldots + \boldsymbol{\pi}_{ln_0}(\mathbf{S}_{ln_0} \cdot \bar{\mathbf{c}}_{n_0}) = \mathbf{0} \end{cases}, \tag{1.28}$$

где умножения векторов внутри перестановок π_{ij} выполняются поэлементно и по правилам группы $GF^{\star}(q)$. Сложения происходят по правилам поля $GF(q)$. Таким образом, доказано следующее

Утверждение 1. *Вектор* $\mathbf{y} = (\bar{\mathbf{y}}_1, \ldots, \bar{\mathbf{y}}_{n_0})$*, где* $\bar{\mathbf{y}}_i$ *— q-ичный вектор длины m, является кодовым словом кода с малой плотностью проверок длины $n = mn_0$, задаваемого проверочной матрицей* $\mathbf{H}$*, тогда и только тогда, когда выполняется l соотношений*

$$\sum_{i=1}^{n_0} \boldsymbol{\pi}_{ji}(\mathbf{S}_{ji} \cdot \bar{\mathbf{y}}_i) = \mathbf{0},\ j = 1..l. \tag{1.29}$$

Как следует из утверждения, для q-го МПП-кода, основанного на матрицах перестановок, вычисление синдрома ошибки имеет векторный характер: в вычислениях используются не отдельные q-ичные символы принятого слова, а блоки длины m.

1.6.3 Декодирование недвоичных МПП-кодов, основанных на матрицах перестановок

В этом разделе предложена модификация алгоритма $q - ary\ Sum\ Product\ Algorithm$ для недвоичных кодов с малой плотностью проверок, основанных на матрицах перестановок. Основная идея предложенной модификации заключается в одновременной обработке m символов принятого слова (т.е. алгоритм работает с векторами длины m), в то время как классический алгоритм $Q - SPA$ не предусматривает такую возможность. Векторный характер декодирования принятого слова, как будет показано, позволяет распараллелить процесс декодирования в m раз, что отразится на скорости обработки данных.

Мы будем использовать модель вычислителя, для которого выполняется следующее условие. Пусть он способен выполнить некоторую операцию над скалярными величинами за время τ. Тогда он способен за то же время τ выполнить такую же операцию (поэлементно) над вектором длины m, где m — любое заранее заданное число.

Отметим, что предложенный алгоритм можно рассматривать как обобщение алгоритма, приведённого в работе [37], на случай недвоичных кодов.

Для лучшего понимания работы алгоритма рассмотрим операции, которые используются в алгоритме декодирования, описанном ниже.

Предложенный алгоритм декодирования работает с мягкими значениями символов. Мягким значением символа будем называть дискретное распределение вероятностей значений этого символа в $GF(q)$. Для символа над полем $GF(q)$ распределение вероятности этого символа — это вектор размерности q, содержащий вероятности каждого из q возможных значений этого символа. Будем использовать для описания такого распределения (мягкого значения) векторы-столбцы. Таким образом, мягкое значение ℓ некоторого символа $x \in GF(q)$ будет записываться следующим образом:

$$\ell = (Pr\{x = 0\}, Pr\{x = 1\}, \ldots, Pr\{x = q - 1\})^T, \tag{1.30}$$

где $Pr\{x = \gamma\}$ обозначает вероятность того, что символ равен $\gamma \in GF(q)$.

Существует очевидное отображение жёстких значений из конечного поля на мягкие: если некоторый символ $x = a, a \in GF(q)$, то соответствующее ему мягкое значение будет выглядеть как $\chi = (0, \ldots, Pr\{x = a\} = 1, \ldots, 0)^T$, то есть соответствовать распределению детерминированной величины.

Естественным образом можно ввести смешанные операции, где один оператор — жёсткое значение, а другой — мягкое: для этого можно заменить жёсткое представление величины на мягкое. Сама замена производится естественным образом: если жёсткий символ равен a, то $\ell = (Pr\{x = a\} = 1, Pr\{x = j\} = 0, j \neq a)^T$.

Операции над мягкими значениями полностью соответствуют операциям над случайными величинами над полем. Рассмотрим результат произвольной операции, определённой в $GF(q)$, которую обозначим символом $\diamond$. Очевидно, что результатом её применения к мягким q-ичным значениям ξ и ζ является такое мягкое q-ичное значение $\eta = \xi \diamond \zeta$, компоненты которого являются вероятностями того, что результат операции будет иметь соответствующие жёсткие значения при заданных входных распределениях вероятностей ξ и ζ.

В предлагаемом алгоритме декодирования будет использоваться два типа операций: умножение мягкого q-ичного значения на ненулевое жёсткое q-ичное значение (на элемент группы $GF^\star(q)$), и сложение мягких q-ичных значений.

Умножение мягкого q-ичного значения ξ на жёсткое q-ичное значение $\alpha^k \in GF^\star(q)$ (α — примитивный элемент поля, $k = \overline{0, q-2}$), по сути, является перестановкой элементов ξ. Будем его обозначать как $\eta = \xi \odot \alpha^k$.

Сумму двух мягких q-ичных значений будем обозначать $\eta = \xi \oplus \zeta$. Она вычисляется как свёртка векторов распределений вероятностей ξ и ζ [25].

Далее по тексту под векторами будут подразумеваться векторы-строки. Под произведениями и суммами векторов в описании алгоритма подразумеваются поэлементные произведения и суммы их компонентов.

Как и при декодировании случайного q-ичного МПП-кода, на вход алгоритма передается оценка вероятностного распределения символов, полученная из канала [25]. Она представляет

из себя вектор мягких значений. Так как каждое мягкое значение — это вектор-столбец, то вектор этих значений можно записать как матрицу $\mathbf{L}$ размерности $q \times n$, где i-му столбцу $\mathbf{L}$ соответствует вектор-столбец вероятностей

$$L_i = (Pr\{x_i = 0\}, Pr\{x_i = 1\}, \ldots, Pr\{x_i = q-1\})^T, \tag{1.31}$$

где $Pr\{x_i = \gamma\}$ обозначает вероятность того, что i-й переданный символ равен $\gamma \in GF(q)$.

Далее абстрагируемся от того, что отдельные мягкие значения L_i являются векторами-столбцами, и будем рассматривать их как мягкие значения. Тогда под $\mathbf{L}$ будем понимать вектор, который имеет вид

$$\mathbf{L} = \left[L_1 \; L_2 \; \ldots \; L_n \right]. \tag{1.32}$$

Так как $n = mn_0$, то матрицу $\mathbf{L}$ можно разбить на n_0 векторов длины m:

$$\mathbf{L} = \left[\bar{\mathbf{L}}_1 \; \bar{\mathbf{L}}_2 \; \ldots \; \bar{\mathbf{L}}_{n_0} \right], \tag{1.33}$$

где

$$\bar{\mathbf{L}}_i = \begin{pmatrix} L_{(i-1)m+1} & \ldots & L_{im} \end{pmatrix}. \tag{1.34}$$

Введем множество $I(j)$ — набор переменных $\{v_1^{(j)}, v_2^{(j)}, \ldots, v_{n_0}^{(j)},\}$ участвующих в j-й проверке, и множество $J(i)$ — набор проверок $\{c_1^{(i)}, c_2^{(i)}, \ldots, c_l^{(i)},\}$ в которые входит i-я переменная. Рассмотрим произвольный вектор $\bar{\mathbf{L}}_i$. Так как размер $\bar{\mathbf{L}}_i$ равен $1 \times m$, а матрицы $\mathbf{R}_{1i}, \mathbf{R}_{2i}, \ldots, \mathbf{R}_{li}$ из (1.7) — $m \times m$ матрицы, содержащие ровно один ненулевой элемент в каждой строке и каждом столбце, то каждый элемент из $\bar{\mathbf{L}}_i$ участвует ровно в l различных проверках. Так как длина вектора $\bar{\mathbf{L}}_i$ равна m, то его элементы участвуют во всех ml проверках. Таким образом, при декодировании с использованием алгоритма $Q-SPA$ не требуется искать $J(\bar{\mathbf{L}}_i)$ для каждого вектора $\bar{\mathbf{L}}_i$. Проводя аналогичные рассуждения, легко заметить, что в каждом блоке из m проверок участвуют все mn_0 переменных, поэтому вычисление $I(j)$ для каждой j-й проверки тоже не требуется.

Введем необходимые обозначения:

1. $\mathbf{L} = \left[\bar{\mathbf{L}}_1 \; \bar{\mathbf{L}}_2 \; \ldots \; \bar{\mathbf{L}}_{n_0} \right]$ — принятый из канала вектор мягких значений;
2. $\mathbf{L}' = \left[\bar{\mathbf{L}}'_1 \; \bar{\mathbf{L}}'_2 \; \ldots \; \bar{\mathbf{L}}'_{n_0} \right]$ — вычисленный вектор мягких значений;
3. $\bar{\alpha}_{ji}$ — строка из m сообщений от переменных $\bar{\mathbf{L}}_i$ к j-й группе из m проверок;
4. $\bar{\gamma}_{ji}$ — строка из m сообщений от j-й группы из m проверок к $\bar{\mathbf{L}}_i$.

Изложенный ниже алгоритм декодирования применим для МПП-кодов, основанных на матрицах перестановок и работает с проверочной матрицей, представленной в форме (1.26).

Алгоритм 1 ((векторный Q-SPA)).

1. Начальная проверка: *по принятому из канала вектору* $\mathbf{L} = \left[\bar{\mathbf{L}}_1\ \bar{\mathbf{L}}_2\ \ldots\ \bar{\mathbf{L}}_{n_0}\right]$ *строится жёсткое решение* $\mathbf{x}$ *(для каждого мягкого символа берётся его наиболее вероятное значение), вычисляется синдром* $\mathbf{Hx^T}$ *согласно алгоритму, описанному в разделе 1.6.2. Если синдром равен нулевому вектору, то декодирование прекращается и* $\mathbf{x}$ *является результатом выполнения алгоритма, иначе переходим к шагу 2.*

2. Инициализация: *Строим матрицу* $\mathbf{A}$ *по правилу:*

$$\mathbf{A} = \begin{pmatrix} \bar{\alpha}_{11} & \bar{\alpha}_{12} & \ldots & \bar{\alpha}_{1n_0} \\ \bar{\alpha}_{21} & \bar{\alpha}_{22} & \ldots & \bar{\alpha}_{2n_0} \\ \ldots & \ldots & \ldots & \ldots \\ \bar{\alpha}_{l1} & \bar{\alpha}_{l2} & \ldots & \bar{\alpha}_{ln_0} \end{pmatrix}, \tag{1.35}$$

где

$$\bar{\alpha}_{ji} = \boldsymbol{\pi}_{ji}(\mathbf{S}_{ji} \odot \bar{\mathbf{L}}_i). \tag{1.36}$$

Операция $\bar{\kappa}_{ji} = \mathbf{S}_{ji} \odot \bar{\mathbf{L}}_i$ *представляет их себя поэлементное умножение вектора из* m *чисел из* $GF(q)$ *и* m *мягких значений из канала. Операция* $\bar{\alpha}_{ji} = \boldsymbol{\pi}_{ji}(\bar{\kappa}_{ji})$ *выполняет перестановку получившихся* m *мягких значений вектора* $\bar{\kappa}_{ji}$*, что даёт такой же* m*-элементный вектор.*

3. Горизонтальный шаг: *Строим матрицу* Γ *по правилу:*

$$\Gamma = \begin{pmatrix} \bar{\gamma}_{11} & \bar{\gamma}_{12} & \ldots & \bar{\gamma}_{1n_0} \\ \bar{\gamma}_{21} & \bar{\gamma}_{22} & \ldots & \bar{\gamma}_{2n_0} \\ \ldots & \ldots & \ldots & \ldots \\ \bar{\gamma}_{l1} & \bar{\gamma}_{l2} & \ldots & \bar{\gamma}_{ln_0} \end{pmatrix}, \tag{1.37}$$

где

$$\bar{\gamma}_{ji} = \bigoplus_{t=1, t\neq i}^{n_0} \bar{\alpha}_{jt}. \tag{1.38}$$

Здесь производится поэлементное сложение m*-элементных векторов* $\bar{\alpha}_{jt}$*.*

4. Вертикальный шаг: *Вычисление вектора апостериорных вероятностей и сообщений от переменных к проверкам:*

$$\bar{\mathbf{L}}'_i = \bar{\mathbf{L}}_i \odot \left[\bigodot_{j=1}^{l} \boldsymbol{\pi}_{ji}^{-1}(\bar{\gamma}_{ji}) \odot \mathbf{S}_{ji}^{(-1)}\right], \tag{1.39}$$

$$\bar{\alpha}_{ji} = \bar{\mathbf{L}}_i \odot \left[\bigodot_{t=1, t\neq j}^{l} \boldsymbol{\pi}_{ti}^{-1}(\bar{\gamma}_{ti}) \odot \mathbf{S}_{ti}^{(-1)}\right]. \tag{1.40}$$

Здесь $\boldsymbol{\pi}_{ji}^{-1}(\bar{\gamma}_{ji})$ — это обратная к $\boldsymbol{\pi}_{ji}$ перестановка m элементов вектора $\bar{\gamma}_{ji}$. Её результат поэлементно умножается на вектор $\mathbf{S}_{ji}^{(-1)}$, содержащий элементы, обратные к элементам $\mathbf{S}_{ji}$. Получающиеся в результате векторы $\bar{\mu}_{ti} = \boldsymbol{\pi}_{ti}^{-1}(\bar{\gamma}_{ti}) \odot \mathbf{S}_{ti}^{(-1)}$ из m элементов перемножаются поэлементно с друг с другом и с векторами $\bar{\mathbf{L}}_i$.

5. Проверка синдрома: *По вычисленным $\bar{\mathbf{L}}'_i$ строится жёсткое решение $\mathbf{x}$ и вычисляется синдром $\mathbf{S} = \mathbf{H}\mathbf{x}^{\mathbf{T}}$. Если $\mathbf{S} = \mathbf{0}$, то декодирование прекращается и $\mathbf{x}$ считается результатом выполнения алгоритма, иначе переходим к шагу 3.*

Шаги 3—5 выполняются ограниченное число раз. Если достигнуто максимальное число итераций, то алгоритм прерывается и блок считается принятым с ошибкой.

Описанный выше алгоритм оперирует векторами длины m. В пределах векторов операции над отдельными символами производятся независимо, благодаря чему процесс вычисления можно осуществлять параллельно для m символов.

1.6.4 Результаты моделирования

Целью моделирования является сравнение времени декодирования на современных вычислителительных устройствах с высоким распараллеливанием со временем декодирования на процессорах общего назначения. Это сравнение было проведено для ряда кодов с различными параметрами.

Представленный алгоритм декодирования был реализован на языке OpenCL. Такая реализация, с одной стороны, позволила убедиться в эффективности предложенного алгоритма для таких векторных вычислителей, как графические ускорители, и, с другой стороны, позволила использовать данный декодер для имитационного моделирования в дальнейшем.

Выбор языка OpenCL был связан с тем, что, в отличие от технологии nVidia CUDA, OpenCL является кросс-платформенным и имеет компиляторы для многих вычислительных устройств, включая процессоры AMD и Intel, видеоускорители AMD (ATI) и nVidia, а также гибридные процессорные устройства IBM Cell.

Для сравнения быстродействия использовались микропроцессор AMD Phenom II X6 1090T и графический процессор nVidia GeForce GTX 570.

AMD Phenom II X6 1090T представляет из себя шестиядерный процессор архитектуры x64, работающий на частоте 3.2 ГГц. Следует отметить, что этот процессор, как и все современные процессоры, имеет инструкции SIMD (single instruction, multiple data), позволяющие выполнять одну и ту же операцию сразу над несколькими операндами за один такт процессора (процессорного ядра). Одним из достоинств, следующих из переносимости OpenCL, является то, что программист может описывать алгоритм на более высоком уровне абстракции, не указывая в явном виде необходимость использования SIMD инструкций (что потребовало бы использования ассемблера), в то же время компилятор может сам объединить одинаковые операции, выполняемые над разными элементами массива в инструкции SIMD.

nVidia GeForce GTX 570 — это графический процессор с архитектурой nVidia Fermi. В GTX 570 находится 480 вычислительных ядер, работающих на частоте 1464 МГц. Быстродействие на задачах, связанных с активной работой с памятью, может быть ограничено малым объёмом локальной для отдельных ядер памяти и пропускной способностью шины памяти. Более подробные характеристики этого графического ускорителя можно посмотреть в информационных материалах (whitepapers) nVidia.

Следует подчеркнуть, что целью моделирования ставилась оценка выигрыша от использования предложенного векторного алгоритма декодирования при вычислениях с использованием графического процессора. Поэтому теоретический выигрыш в m раз не мог наблюдаться ввиду более высокой тактовой частоты и наличию нескольких ядер в процессоре общего назначения, а также ограниченному числу ядер в графическом процессоре. При моделировании не ставилась задача оптимизации алгоритма под конкретную архитектуру, поэтому задача распределения m отдельных вычислительных подзадач по ядрам процессора была переложена на компилятор и диспетчер задач.

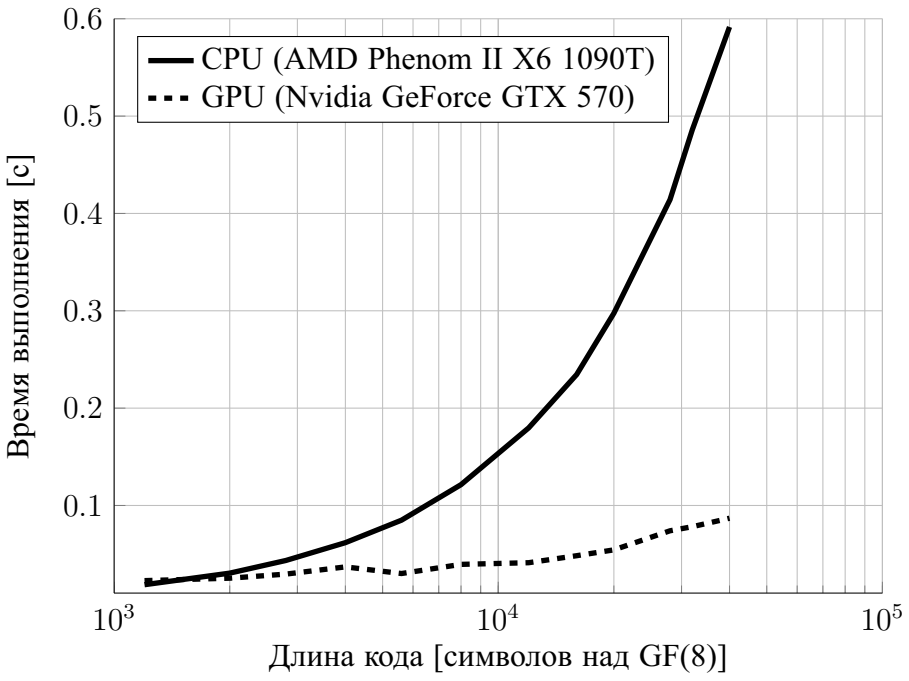


Рисунок 1.6. Время работы алгоритма на процессоре и графическом ускорителе в зависимости от длины кода с параметрами: $l = 3$, $n_0 = 4$, $R = 0.25$

На рис. 1.6, 1.7, 1.8, 1.9, 1.10, 1.11 приведено сравнение быстродействия декодера при его работе на графическом ускорителе и на процессоре в зависимости от длин и скоростей кодов. Максимальное число итераций декодирования ограничивалось 20.

Из рис. 1.6, 1.7, 1.8, 1.9, 1.10, 1.11 можно сделать вывод, что при достачно больших длинах кодов (от 1500 символов над полем $GF(8)$ для кодов скоростей $R = 0.25$ и $R = 0.5$ и от 11000 символов для кодов со скоростью $R = 0.8$) предложенный декодер, работающий на графическом процессоре, выигрывает по времени обработки данных перед декодером, выполняемом на

Рисунок 1.7. Отношение времени декодирования на процессоре ко времени декодирования на графическом ускорителе в зависимости от длины кода с параметрами: $l = 3$, $n_0 = 4$, $R = 0.25$

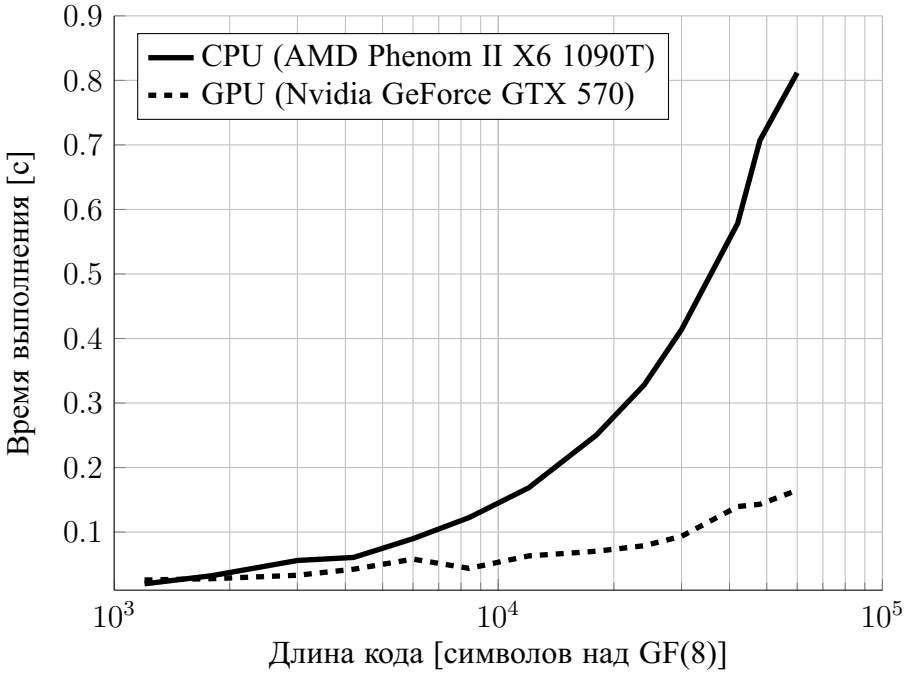


Рисунок 1.8. Время работы алгоритма на процессоре и графическом ускорителе в зависимости от длины кода с параметрами: $l = 3$, $n_0 = 6$, $R = 0.5$

обыкновенном процессоре. Заметно, что чем ниже скорость кода (и чем соответственно короче длина проверки), тем больший выигрыш наблюдается. Так, для кодов со скоростью $R = 0.25$ при длине $n = 12000$ выигрыш составил 6 раз, в то время как для кодов со скоростью $R = 0.5$

Рисунок 1.9. Отношение времени декодирования на процессоре ко времени декодирования на графическом ускорителе в зависимости от длины кода с параметрами: $l = 3$, $n_0 = 6$, $R = 0.5$

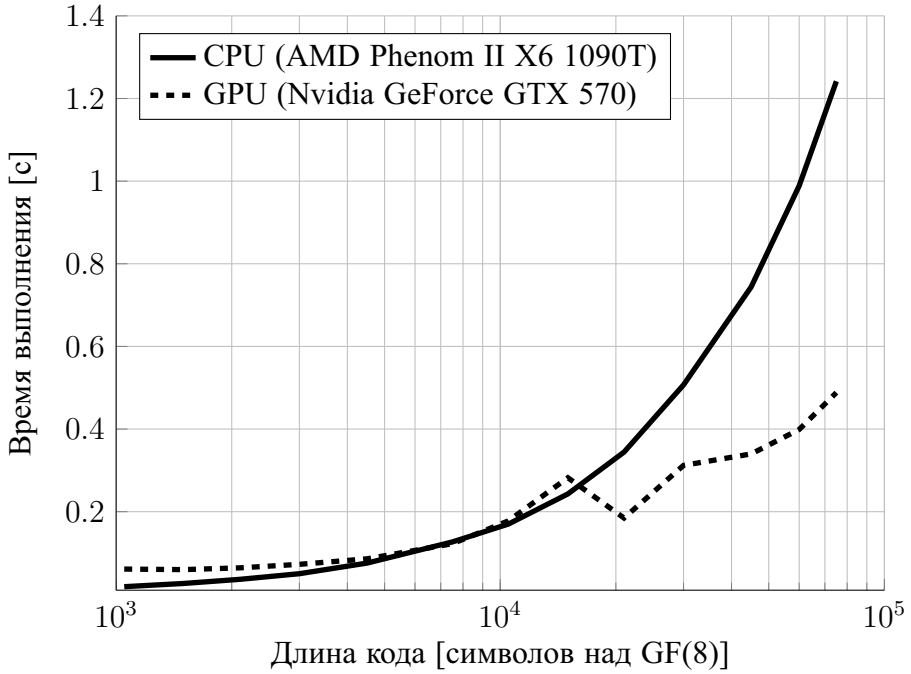


Рисунок 1.10. Время работы алгоритма на процессоре и графическом ускорителе в зависимости от длины кода с параметрами: $l = 3$, $n_0 = 15$, $R = 0.8$

при аналогичной длине, выигрыш составляет уже 4.5 раза, а при скорости $R = 0.8$ выигрыш имеет место только при длине более 20000 символов.

Рисунок 1.11. Отношение времени декодирования на процессоре ко времени декодирования на графическом ускорителе в зависимости от длины кода с параметрами: $l = 3$, $n_0 = 15$, $R = 0.8$

В то же время при небольших (до 1500 символов для скоростей $R = 0.25$ и $R = 0.5$, и до 20000 символов для $R = 0.8$) заметно, что декодирование с использованием процессора дает лучшие результаты, чем при использовании графического ускорителя.

Также особо следует отметить, что на рис. 1.6, 1.7, 1.8, 1.9, 1.10, 1.11 приведены сравнения быстродействия предложенного алгоритма при его реализации на графическом ускорителе и на всех шести ядрах процессора. Таким образом, в пересчете на одно процессорное ядро выигрыш увеличится приблизительно в 6 раз. Например, при $R = 0.25$ и $n = 12000$ символов поля $GF(8)$ выигрыш графического ускорителя по сравнению с одним процессорным ядром составит приблизительно 36 раз.

Отметим немонотонное поведение кривых времени вычислений на графическом ускорителе на рис. 1.6, 1.7, 1.8, 1.9, 1.10, 1.11. Это поведение было детерминированным, то есть при повторении эксперимента с теми же параметрами время выполнения с высокой точностью оставалось тем же. Можно предположить, что такой характер поведения кривых связан с наличием в графическом ускорителе нескольких “бутылочных горлышек”, которые могут влиять на производительность вычислений. К ним, в частности, относятся объёмы локальной, глобальной памяти и кеша, пропускные способности шин данных к этим типам памяти и особенности распределения задач планировщиком. В то же время из графиков видно, что эти факторы не оказывают влияния на общую закономерность.

Таким образом, предложенную векторную реализацию декодера можно использовать в случаях, когда либо длина кода достаточно велика, либо код не является высокоскоростным.

1.7 Выводы к главе

В первой главе:

– Разработан метод применения алгоритма декодирования “распространения доверия” с мягким входом для каналов с жёстким решением. Показано, что он даёт выигрыш около 1.1 дБ при применении алгоритма Sum-Product для канала с жёстким решением по сравнению с другими известными алгоритмами. Дан метод выбора фиксированного значения оценки надёжности, зависящего только от параметров кода, но не от вероятности ошибки на выходе из канала.

– Произведено сравнение различных алгоритмов декодирования МПП-кодов, основанных на свёрточных кодах с единичной памятью с циклическим замыканием. Показано, что наибольшей эффективностью обладает декодирование кода как блокового при помощи алгоритма “распространения доверия”.

– Предложен способ векторизации алгоритма “распространения доверия” для q-ичных МПП-кодов. Алгоритм был реализован на языке OpenCL, эта реализация позволила получить выигрыш в скорости моделирования, достигающий для некоторых параметров кода нескольких раз.

Глава 2

Обобщённые коды с локализацией ошибок

2.1 Введение

В настоящее время, в связи с ограниченностью частотного ресурса и возрастающими требованиями на скорость передачи данных по сетям связи, в аппаратуре передачи данных всё большее распространение получают сигнально-кодовые конструкции, основанные на модуляциях с большим индексом. В то же время коды коррекции ошибок, применявшиеся в широко распространённых системах, являлись двоичными и могли быть субоптимальны для соответствующих каналов.

В данной главе рассматривается известный тип кодов коррекции ошибок — коды с обобщённой локализацией ошибок [5]. Главное назначение этих кодов — получить очень малое значение вероятности неправильного декодирования.

Они подходят для оптики [38, 39] и позволяют работать уже от входной вероятности ошибки, равной 10^{-2}.

Коды с обобщённой локализацией ошибок — это класс линейных каскадных кодов [6, 7]. Было показано [8], что они являются частным случаем обобщённых каскадных кодов (ОКК).

Кодовое слово обобщённого каскадного кода обычно представляют в виде матрицы, высота которой равна длине внутренних кодов, а ширина — длине внешних кодов.

ОЛО-коды — это такой тип ОКК, в которых столбцы кодового слова не являются кодовыми словами какого-либо кода. Эта конструкция эффективна только в том случае, когда характер ошибок таков, что в принятой матрице кодового слова большинство столбцов не содержит ошибок. Отсюда легко увидеть, что при работе в дискретном симметричном канале без памяти для выполнения этого условия требуется иметь, по-возможности, более короткие внутренние коды.

Ключевая особенность ОЛО-кодов — специальный декодер, оптимизированный для декодирования кодов с малой избыточностью, работающих в относительно хороших каналах. Можно сказать, что этот декодер использует имеющуюся в коде избыточность только для тех частей кодового слова, в которых имеются ошибки.

2.2 ОЛО-коды с квадратичным расширением алфавита

2.2.1 Конструкция ОЛО-кодов

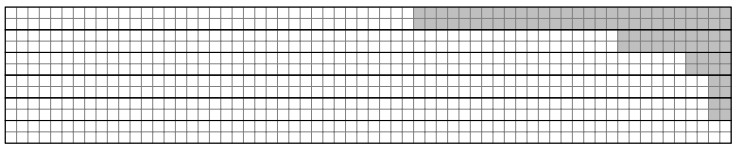
Опишем выбранную конструкцию кодов с обобщённой локализацией ошибок. Назовём его *нормальным* ОЛО-кодом.

Кодовым словом *нормального* q-ичного ОЛО-кода $\mathcal{C}$ назовём матрицу $\mathbf{C}$ над полем $GF(q)$ размеров $n_A \times n_B$, где n_A — длина внутренних кодов, а n_B — длина внешних кодов.

Обозначим $\mathbf{H}$ проверочную матрицу системы вложенных внутренних кодов $\{\mathcal{C}_j^A\}$. Её элементы принадлежат полю $GF(q)$, её размер $n_A \times n_A$. Она должна быть невырождена. Любые первые r_A строк этой матрицы (где r_A чётно), составляющие матрицу размера $r_A \times n_A$, будут являться матрицей внутреннего кода длины n_A с $k_A = n_A - r_A$ информационных символов. Будем нумеровать эти коды следующим образом: j-й внутренний код — это код, у которого $r_A = 2j$. Обозначим числа информационных символов, проверочных символов и расстояния этих кодов как k_j^A, r_j^A и d_j^A соответственно.

Назовём

$$\mathbf{S} = \mathbf{H} \cdot \mathbf{C} =$$



(2.1)

— матрицей синдромов внутренних кодов. Объединим её строки попарно. Пары строк назовём слоями, а их элементами будем считать подматрицы размера 2×1 и рассматривать их как символы над полем $Q = q^2$. Тогда слои будут представлять из себя вектор-строки $\mathbf{s}_j$ длины n_B над полем $GF(Q)$, $j = \overline{1,L}$. Число слоёв $L = n_A/2$ будем называть *порядком ОЛО-кода*.

Внешние коды *нормального* ОЛО-кода $\mathcal{C}_j^B$ — коды над полем $GF(Q)$ длины n_B и имеют числа информационных символов r_j^B, скорости $R_j^B = 1 - r_j^B/n_B$, числа информационных символов $k_j^B = n_B - r_j^B$, кодовые расстояния d_j^B.

О п р е д е л е н и е 5. ОЛО-код — это множество матриц $\mathbf{C}$ таких, что слои $\mathbf{s}_j$, $j = \overline{1,L}$ матрицы $\mathbf{S} = \mathbf{H} \cdot \mathbf{C}$ являются кодовыми словами внешних кодов.

Заданный таким образом ОЛО-код будет линейным кодом над полем $GF(q)$ длины $n = n_A n_B$, его избыточность $r = \sum_j 2r_j^B$. Скорость кода $R = \sum_j 2R_j^B/n_A$, то есть равна среднему арифметическому скоростей компонентных кодов. Кодовое расстояние ограничено снизу $d \geq \min_{j=\overline{2,L}}\{d_1^B, d_{j-1}^A d_j^B\}$ [6].

Следует отметить, что в кодовом слове $\mathbf{C}$ ОЛО-кода в явном виде не присутствуют кодовые слова ни внутренних, ни внешних кодов. Если быть точным, в столбцах кодового слова $\mathbf{C}$ ОЛО-кода находятся слова смежных классов внутренних кодов, в то время как выбор смежного класса

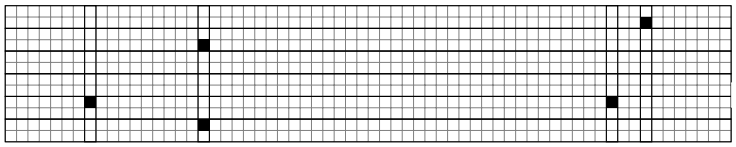
в i-м столбце $\mathbf{C}$ определяется теми символами слов внешних кодов, которые находятся в i-м столбце матрицы $\mathbf{S}$ (на i-х позициях вектор-строк $\mathbf{s}_j$).

При этом спектр расстояний смежного класса и, как следствие, его корректирующая способность, в точности совпадают с таковыми у внутреннего кода.

Ниже рассмотрим конструкцию, где качестве внутренних и внешних кодов используются коды с максимально достижимым расстоянием.

2.2.2 Декодирование

Пусть после передачи кодового слова $\mathbf{C}$ по каналу в нём произошли ошибки и было принято слово $\mathbf{V}$:

$\mathbf{V} =$  . (2.2)

В данном слове закрашенными квадратиками обозначены символы, в которых произошли ошибки; рамками обведены столбцы, которые содержат ошибочные символы.

Декодирование будет происходить итеративно, число итераций равно порядку кода L. На каждой итерации будет последовательно происходить сначала декодирование внешнего кода, а затем внутреннего. При этом декодирование на j-й итерации будет рассматриваться в предположении, что декодирование на всех предыдущих было правильным.

В нашем случае коды в составе системы вложенных кодов имеют максимально достижимое расстояние, то есть $d = n - k + 1 = r + 1$, откуда максимально число исправляемых ими ошибок $t_j^A = r_j^A/2$. Строки матрицы $\mathbf{S}$ объединены в слои по две, откуда следует, что $r_{j+1}^A = r_j^A + 2$. Отсюда следует, что количество ошибок в столбце, которое способен исправить каждый следующий внутренний код системы, растёт на 1: первый внутренний код ($j = 1$) имеет $r_1^A = 2$ и исправляет все одиночные ошибки в столбцах, второй — $r_2^A = 4$ и исправляет все двойные, и так далее.

Обозначим $\mathbf{H}_j$ проверочную матрицу j-го внутреннего кода, она будет содержать первые j слоёв ($2j$ строк) матрицы $\mathbf{H}$:

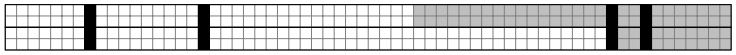
$\mathbf{H}_j =$  (2.3)

Тогда алгоритм декодирования на j-й итерации, $j = \overline{1,L}$, следующий:

– Перед декодированием $\mathbf{V}_0$ полагается равным принятому слову $\mathbf{V}$, $\mathbf{V}_0 = \mathbf{V}$.

– Полученное на предыдущей итерации слово $\mathbf{V}_{j-1}$ умножается на $\mathbf{H}_j$ слева, получается матрица синдромов j-го внутреннего кода:

$$\mathbf{S}'_j = \mathbf{H}_j \cdot \mathbf{V}_{j-1} = \quad (2.4)$$



Чёрным отмечены те столбцы, в которых синдромы могут содержать ошибки, то есть могут отличаться от переданных в этих же позициях (не отвечать истинным смежным классам внутренних кодов, которые передавались).

– Далее в строках вычисленной матрицы с помощью внешних кодов от 1 до j исправляются ошибки:

$$\mathbf{S}''_j = Dec_B\{\mathbf{S}'_j\}. \quad (2.5)$$

В результате, при условии, что внешние коды обладают достаточной корректирующей способностью, чтобы исправить все ошибки в $\mathbf{S}'_j$, столбцы матриц $\mathbf{S}''_j$ укажут правильные смежные классы для декодирования внутренних кодов. Если внешний код выдаёт отказ от декодирования, то алгоритм декодирования останавливается, и выдаётся отказ от декодирования всей кодовой конструкции. Если в результате декодирования внешнего кода произойдёт ошибка, то в результате декодирования ОЛО-кода либо будет получено неправильное кодовое слово, либо на одной из следующих итераций будет выдан отказ от декодирования.

– С использованием полученной на предыдущем шаге алгоритма матрицы $\mathbf{S}''_j$, с помощью j-х внутренних кодов можно исправить ошибки в столбцах матрицы $\mathbf{V}$:

$$\mathbf{V}_j = Dec_A[\mathbf{S}''_j]\{\mathbf{V}\}. \quad (2.6)$$

Учитывая конструкцию системы вложенных внутренних кодов, можно указать как минимум две возможности для исправления ошибок с их помощью. С одной стороны, можно вычислить синдромы ошибок как разности $\mathbf{S}''_j - \mathbf{S}'_j$, по ним найти представителей смежных классов и вычесть их из столбцов матрицы $\mathbf{V}$. Эта операция может быть записана в виде следующего выражения:

$$\mathbf{V}_j = Dec_A\{\mathbf{V} - \mathbf{H}_j^{-1}\mathbf{S}''_j\} + \mathbf{H}_j^{-1}\mathbf{S}''_j, \quad (2.7)$$

где $\mathbf{H}_j^{-1}$ — матрица, содержащая первые $2j$ столбцов матрицы $\mathbf{H}^{-1}$, а $Dec_A\{\}$ — операция декодирования внутренних кодов.

С другой стороны, можно декодировать эти смежные классы непосредственно, например, по решётке.

Таким образом, если j-й внутренний код способен исправить ошибки кратности t_j^A и менее, то по итогам декодирования внутренних кодов на этой итерации в матрице $\mathbf{V}_j$ останутся только столбцы без ошибок и столбцы с ошибками кратности более t_j^A.

– После исправления ошибок с помощью последнего, L-го слоя, декодирование останавливается. Матрица $\mathbf{V}_L$ является результатом декодирования.

2.2.3 Границы вероятности неправильного декодирования

Верхняя граница вероятности неправильного декодирования

Повторим ключевые особенности алгоритма декодирования, которые потребуются для того, чтобы написать верхнюю оценку вероятности *неправильного декодирования*[1] ОЛО-кода.

– Декодирование — процесс из L итераций.

– Внутренние коды на j-й итерации исправляют ошибки кратности $t_j^A = j$ и менее.

– Внешние коды имеют избыточности r_j^B и способны исправлять ошибки кратности $t_j^B \leq \lfloor r_j^B/2 \rfloor$.

Оценим вероятность ошибки декодирования ОЛО-кода для q-ичного канала без памяти с вероятностью ошибки на символ, равной p.

В таком случае, учитывая, что высота столбца равна n_A, вероятность того, что в столбце произойдёт не менее t ошибок, равна:

$$p_+\{t\} = \sum_{i=t}^{n_A} C_{n_A}^i p^i (1-p)^{n_A - i}. \tag{2.8}$$

В нашем случае $t_j^A = j$, то есть j-й внутренний код способен исправлять ошибки кратности j и менее. Это значит, что так как на $(j-1)$ итерации были исправлены ошибки кратности $(j-1)$ и менее, то на j-й итерации останутся только ошибки кратности j и более. Это значит, что число столбцов с неправильными синдромами в $\mathbf{S}'_j$, которые должны быть исправлены на j-й итерации декодирования, будет определяться вероятностью появления столбцов с j и более ошибок[2]. Вероятность появления не менее $t_j^B + 1$ ошибок в $\mathbf{s}_j$ можно оценить аналогично тому, как вычисляется вероятность появления j и более ошибок в столбце.

Обозначим

$$p_{Aj} \leq p_+\{j\} \tag{2.9}$$

— вероятность того, что на j-й итерации декодирования выбранный столбец матрицы $\mathbf{S}'_j$ будет содержать ошибку. Для всех столбцов эти вероятности одинаковы.

[1] То есть вероятность отказа от декодирования или перехода в неправильное кодовое слово.

[2] Некоторые такие столбцы дают правильные синдромы, но для целей получения оценки сверху это несущественно.

Тогда вероятность ошибки j-го внешнего кода можно оценить сверху величиной

$$p_{Bj} = \sum_{i=t_j^B+1}^{n_B} C_{n_B}^i p_{Aj}^i (1-p_{Aj})^{n_B-i}. \tag{2.10}$$

Вероятность ошибки кодовой конструкции в целом будет вероятностью того, что хоть один из внешних кодов декодируется неправильно. Сформулируем этот результат в виде теоремы:

Т е о р е м а 1. *Пусть p_{Bj} — оценка сверху вероятности неправильного декодирования j-го внешнего кода, $j = \overline{1,L}$. Тогда для вероятности p неправильного декодирования кода $\mathcal{C}$ справедлива граница:*

$$p \le p_B = \min\{\sum_{j=1}^{L} p_{Bj}, 1\}, \tag{2.11}$$

где L — число слоёв ОЛО-кода, p_B — оценка сверху вероятности неправильного декодирования.

Доказательство. Рассмотрим следующую систему событий.

Пусть $\bar{A}_j$ - событие “произошло $> t_j^B$ ошибок веса $\ge j$ в столбцах”, его вероятность вычисляется согласно формуле (2.10), $P\{\bar{A}_j\} = p_{Bj}$.

Вероятность $P\{\bar{A}_j A_{j-1} \dots A_1\}$ говорит о том, что на первых $j-1$ итерациях декодирования число столбцов с ошибками веса $\ge j-1$ не превосходило того количества, которое заведомо на соответствующих итерациях исправляется, и при этом число столбцов с ошибками веса $\ge j$ больше, чем то, которое гарантированно исправляется на j-й итерации декодирования. Как известно, она не больше любой из них, в частности, $P\{\bar{A}_j A_{j-1} \dots A_1\} \le P\{\bar{A}_j\}$.

Она будет являться оценкой сверху вероятности неправильного декодирования на j-й итерации $P\{\bar{D}_j\}$: $P\{\bar{D}_j\} \le P\{\bar{A}_j A_{j-1} \dots A_1\} \le P\{\bar{A}_j\}$

Пусть $\bar{D}$ — событие неправильного декодирования исходного ОЛО-кода $\mathcal{C}$. Его вероятность:

$$P\{\bar{D}\} = P\left\{\bigcup_{j=1}^{L} \bar{D}_j\right\} \le \sum_{j=1}^{L} P\{\bar{D}_j\} \le \sum_{j=1}^{L} P\{\bar{A}_j\}. \tag{2.12}$$

Из этой формулы непосредственно следует сумма в формуле (2.11).

Так как вероятность не может быть больше 1, остаётся ограничить эту сумму 1, что приводит к формуле (2.11), что и требовалось доказать. □

Нижняя граница вероятности неправильного декодирования

Для вычисления нижней границы неправильного декодирования требуется построить такие слова ошибок, которые, с одной стороны, будут обладать достаточно большой вероятностью, и, с другой, заведомо не смогут быть декодированы.

Ограничимся оценкой вероятности неправильного декодирования первого внешнего кода. Конструкция слова ошибки будет строиться следующим образом: пусть произошла ошибка $\mathbf{E} = \mathbf{V} - \mathbf{C}$, в которой есть не менее, чем $\lfloor (d_1^B - 1)/2 \rfloor$ столбцов веса 1. Любой столбец слова ошибки

веса 1 приводит к появлению на входе первого внутреннего кода ошибки в той же позиции, что и номер этого столбца.

Первый внутренний код не может иметь на входе стирания, так 0-й внутренний код не декодируется (он имеет скорость 1) и, как следствие, не даёт отказов от декодирования. Первый внешний код способен исправлять не более, чем $t_1^B = \lfloor (d_1^B - 1)/2 \rfloor$ ошибок. Это значит, что если столбцов веса 1 будет не менее, чем $t_1^B + 1$, то на входе декодера первого внутреннего кода появится не менее, чем $t_1^B + 1$ ошибок и он не будет декодирован правильно (то есть либо даст отказ от декодирования, либо декодирование будет ошибочно).

Вероятность появления такого слова легко может быть вычислена. Вероятность того, что в столбце произойдёт ровно 1 ошибка, равна:

$$p_{\{1\}} = n_A p (1-p)^{n_A - 1}. \tag{2.13}$$

Тогда вероятность ошибки j-го внешнего кода можно оценить снизу величиной

$$p'_{B1} = \sum_{i=t_B^{(j)}+1}^{n_B} C_{n_B}^i p_{\{1\}}^i (1 - p_{\{1\}})^{n_B - i}. \tag{2.14}$$

Сформулируем результат этого раздела в виде теоремы.

Т е о р е м а 2. *Для вероятности p неправильного декодирования кода $\mathcal{C}$ справедлива граница:*

$$p \geq p_B = p'_{B1} \tag{2.15}$$

Доказательство. Пусть $\bar{B}_1$ - событие “произошло $> t_1^B$ ошибок веса 1 в столбцах”, его вероятность вычисляется согласно формуле (2.14).

Вероятность $p'_{B1} = P\{\bar{B}_1\}$ говорит о том, что на первой итерации декодирования число столбцов с ошибками веса не меньше того количества, которое не может быть исправлено на первой итерации. Она будет являться оценкой снизу вероятности неправильного декодирования на 1-й итерации $P\{\bar{D}_1\}$: $P\{\bar{D}_1\} \geq P\{\bar{B}_1\}$

Пусть $\bar{D}$ — событие неправильного декодирования исходного ОЛО-кода $\mathcal{C}$, а $\bar{D}_j$ — событие неправильного декодирования на j-й итерации. Тогда:

$$P\{\bar{D}\} = P\left\{\bigcup_{j=1}^{L} \bar{D}_j\right\} \geq \min_j P\{\bar{D}_j\} \geq \min_j P\{\bar{B}_j\} \geq P\{\bar{B}_1\}. \tag{2.16}$$

Из этой формулы непосредственно следует формула (2.15). □

Поиск избыточностей кодов-компонентов, обеспечивающих заданную выходную вероятность ошибки при заданной входной

Пусть заданы базовые параметры кода (q, n_A, n_B, L, t_j^A), входная вероятность ошибки p_s и требуемая вероятность неправильного декодирования $p_f = p_B$, и требуется найти избыточности кодов-компонентов для того, чтобы обеспечить требуемую вероятность ошибки.

По описанию алгоритма декодирования выше можно построить процедуру поиска избыточностей внешних кодов:

- Для каждого слоя, зная корректирующие способности внутренних кодов, можно вычислить вероятности p_{Aj} появления столбцов с ошибками кратности j и более, которые потребуется исправить на j-й итерации алгоритма с помощью j-го внешнего кода.
- Далее требуется ограничить вероятность ошибки каждого из внешних кодов. Очевидно, что если $\forall j : p_{Bj} \leq p_B/L$, то вероятность ошибки кодовой конструкции не превысит требуемой p_B.

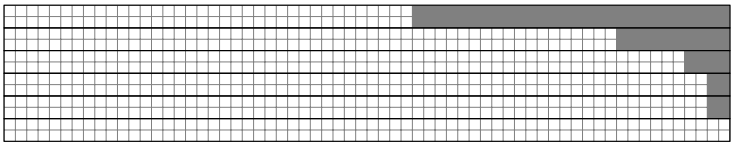
  Поэтому следует, с использованием формулы 2.10, произвести поиск таких t_j^B, которые обеспечат выполнение условия $p_{Bj} \leq p_B/L$.
- Количество проверочных символов в кодах вычислить как $r_j^B = \min(2t_j^B, n_B)$.
- На данном этапе известны параметры кода, которые обеспечивают требуемые характеристики выходной вероятности ошибки при заданной входной, но могут быть неоптимальны. В данном случае требуется провести оптимизацию параметров кода. Это можно сделать следующим образом:
  - Для набора r_j^B, полученного на предыдущем шаге, построить все альтернативные наборы $\{\hat{r}_j^B\}_k$ следующим образом:
    $\hat{r}_{jk}^B = r_j^B, j \neq k$;
    $\hat{r}_{jk}^B = r_j^B - 2, j = k$
    для $k = \overline{1,L}$
  - Для каждого из этих наборов вычислить вероятность ошибки в соответствии с алгоритмом, описанным в разделе 2.2.3.
  - Выбрать тот набор, вероятность ошибки которого наименьшая.
  - Если полученная вероятность меньше требуемой p_f, то повторить эту же операцию. Если больше, то выполнить аналогичную, но не с уменьшением, а с увеличением корректирующей способности кодов.
  - Алгоритм останавливается, если после двух шагов набор r_j^B не изменился.

2.2.4 Кодирование

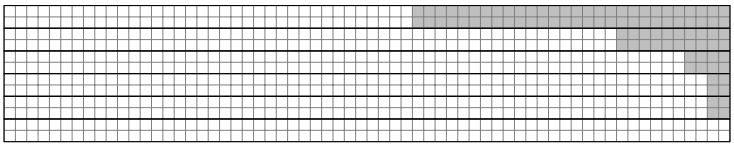
В данном разделе остановимся на способах кодирования ОЛО-кодов.

Несистематическое кодирование

Кодировать можно совсем просто. Пусть на входе алгоритма кодирования есть q-ичная матрица $\mathbf{I}$, имеющая такой же размер и такое же разбиение на слои $\mathbf{I}_j$, как и матрица $\mathbf{S}$. В каждом из слоёв $\mathbf{I}_j$ первые k_j^B символов — информационные, а последние r_j^B — нулевые (в формуле 2.17 элементы с информационными символами обозначены белым цветом, с нулевыми — тёмно-серым):

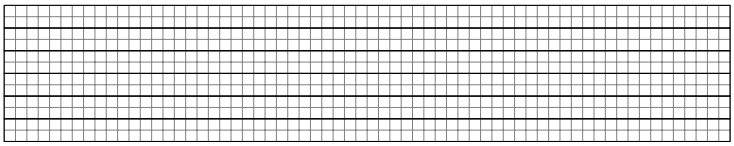
$\mathbf{I} =$  (2.17)

Для того, чтобы получить кодовое слово ОЛО-кода, требуется сначала закодировать внешними кодами строки (слои) матрицы $\mathbf{I}$. Кодеры внешних кодов используют в качестве информационных те символы, которые стоят на соответствующих местах в вектор-строках $\mathbf{I}_j$, и возвращают вектор-строки соответствующих им кодовых слов $\mathbf{s}_j$. В свою очередь, $\mathbf{s}_j$ являются компонентами матрицы синдромов внешних кодов $\mathbf{S}$:

$\mathbf{S} =$  (2.18)

Обозначим эту операцию следующим образом: $\mathbf{S} = Enc_B\{\mathbf{I}\}$. После выполнения этой операции элементы на местах информационных символов в матрицах $\mathbf{I}$ и $\mathbf{S}$ совпадут, а на местах проверочных символов в матрице $\mathbf{S}$ будут проверочные символы внешних кодов.

Затем для того, чтобы получить кодовое слово, нужно умножить полученную матрицу на матрицу, обратную $\mathbf{H}$: $\mathbf{C} = \mathbf{H}^{-1} \cdot \mathbf{S}$. Как уже было сказано, матрица $\mathbf{H}$ должна быть невырождена.

$\mathbf{C} = \mathbf{H}^{-1} \cdot \mathbf{S} =$  (2.19)

Таким образом, можно сформулировать

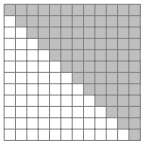
Т е о р е м а 3. *Для несистематического кодирования матрицы* $\mathbf{I}$ *с информационными символами требуется выполнить следующую операцию:*

$$\mathbf{C} = \mathbf{H}^{-1} \cdot Enc_B\{\mathbf{I}\} \tag{2.20}$$

Доказательство. Легко показать, что она будет удовлетворять определению кодового слова ОЛО-кода: $\mathbf{H} \cdot \mathbf{C} = \mathbf{H} \cdot \mathbf{H}^{-1} \cdot \mathbf{S} = \mathbf{S}$, и в слоях $\mathbf{S}$ по построению находятся кодовые слова внешних кодов. □

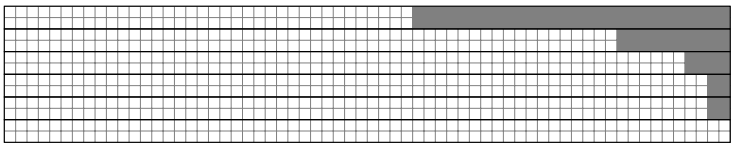
Систематическое кодирование

Выше на матрицу $\mathbf{H}$ не накладывалось никаких ограничений кроме невырожденности. Ниже для реализации систематического кодирования та же матрица потребуется в верхней треугольной форме и будет обозначена $\mathbf{U}$. [3]

$\mathbf{U} =$ 

(2.21)

В таком случае матрица $\mathbf{U}^{-1}$, обратная к $\mathbf{U}$, тоже будет иметь верхний треугольный вид. Для того, чтобы закодировать матрицу с информационными символами (2.17), для начала умножим её на $\mathbf{U}$:

$\mathbf{R} = \mathbf{U} \cdot \mathbf{I} =$ 

(2.22)

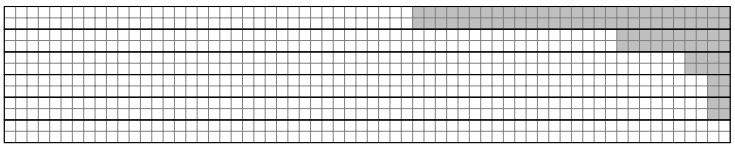
Благодаря верхней треугольной форме матрицы $\mathbf{U}$ структура матрицы $\mathbf{R}$ будет совпадать со структурой матрицы $\mathbf{I}$. После этого $\mathbf{R}$ нужно закодировать внешними кодами аналогично кодированию матрицы $\mathbf{I}$ в несистематическом случае.

Так же, как выше, обозначим эту операцию следующим образом: $\mathbf{S} = Enc_B\{\mathbf{R}\}$. После выполнения этой операции элементы на местах информационных символов в матрицах $\mathbf{R}$ и $\mathbf{S}$ совпадут, а на местах проверочных символов в матрице $\mathbf{S}$ будут проверочные символы внешних кодов.

[3] Если быть точным, достаточно, чтобы матрица была блочно-верхне-треугольной, где высоты блоков будут равны высотам слоёв матрицы $\mathbf{S}$.

После этого получившуюся матрицу остаётся умножить на $\mathbf{U}^{-1}$:

$$\mathbf{C} = \mathbf{U}^{-1} \cdot \mathbf{S} = \mathbf{U}^{-1} \cdot Enc_B\{\mathbf{U} \cdot \mathbf{I}\} =$$



(2.23)

Таким образом, можно сформулировать

Т е о р е м а 4. *Для систематического кодирования матрицы* $\mathbf{I}$ *с информационными символами требуется выполнить следующую операцию:*

$$\mathbf{C} = \mathbf{U}^{-1} \cdot Enc_B\{\mathbf{U} \cdot \mathbf{I}\} \tag{2.24}$$

Доказательство. Легко показать, что она будет удовлетворять определению кодового слова ОЛО-кода: $\mathbf{U} \cdot \mathbf{C} = \mathbf{U} \cdot \mathbf{U}^{-1} \cdot Enc_B\{\mathbf{U} \cdot \mathbf{I}\} = \mathbf{S}$, и в слоях $\mathbf{S}$ по построению находятся кодовые слова внешних кодов.

Благодаря верхней треугольной форме матрицы $\mathbf{U}$ в результате получившейся процедуры на местах информационных символов матрицы $\mathbf{C}$ будут те же символы, что и в матрице $\mathbf{I}$. □

2.2.5 Анализ параметров кодовых конструкций и их влияние на эффективность

Исходя из конструкции кода, алгоритма декодирования и метода расчёта избыточностей кодов-компонентов, можно сделать несколько выводов.

При фиксированной длине кодовой конструкции является целесообразным использование, по-возможности, коротких внутренних кодов и длинных внешних (малых n_A и больших n_B). Это будет способствовать улучшению производительности кодов двумя путями.

Во-первых, при увеличении длины внутренних кодов n_A при заданной входной вероятности ошибки p_s увеличивается вероятность ошибки в столбце. При малых p_s она примерно равна $p_i \approx n_A p_s$, то есть пропорциональна n_A.

Во-вторых, увеличение длины внешних кодов позволяет естественным образом уменьшить долю ошибок, которую требуется исправить. Под долей ошибок τ подразумевается отношение числа ошибок t к длине кода n, $\tau = t/n$. Это легко объяснимо на примере предельного случая, когда $n \to \infty$: при фиксированной вероятности ошибки p средняя доля ошибок будет равна p, а среднеквадратичное отклонение — $\sigma = \sqrt{p(1-p)/n}$. Для обеспечения малой вероятности неправильного декодирования требуется заложить $\tau = p + x\sigma$, где x зависит от требуемой вероятности ошибки. В случае малых n существенное влияние имеет слагаемое, отвечающее за среднеквадратичное отклонение. Это продемонстрировано на рис. 2.1, где показаны кривые вероят-

ности того, что случайная величина с биномиальным распределением с параметрами $p = 10^{-1}$ и $n \in \{50, 110, 250\}$ будет иметь относительное значение больше, чем значение на оси абсцисс. Из кривых видно, что если требуется достичь вероятности ошибки не более 10^{-15}, то коду длины $n = 50$ придётся исправлять долю ошибок, равную $\tau = 0.54$, а при $n = 250$ только $\tau = 0.28$.

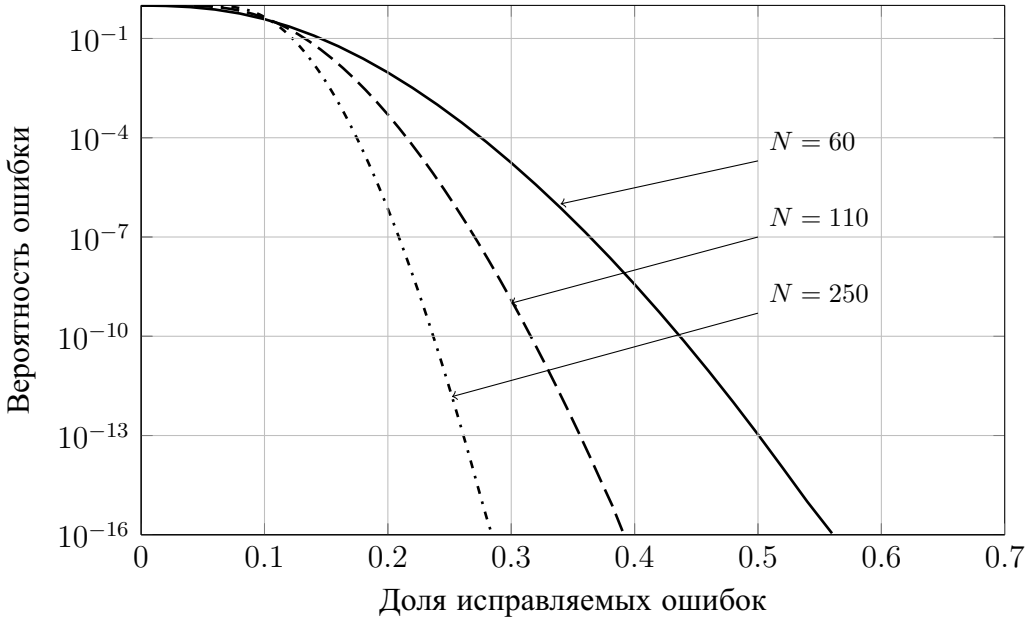


Рисунок 2.1. Вероятность события, что доля ошибок не превысит заданный порог, для трёх разных длин кодов

2.2.6 Примеры построения кодов

Построим, применив алгоритм из раздела 2.2.3, коды со следующими параметрами: базовое поле $GF(16)$, длина внутренних кодов $n_A = 16$, длина внешних $n_B = 256$. Полная длина кода получается $n = 16384$ бита. Зададим уровень входной вероятности ошибки, равный $p_s = 10^{-2}$, и построим конструкции для трёх требуемых выходных вероятностей ошибки $p_f \in \{10^{-12}, 10^{-15}, 10^{-18}\}$.

Скорости полученных кодов равны $R \in \{0.881, 0.867, 0.854\}$, оценка на расстояние $d_{min} \geq d \in \{39, 45, 45\}$. Графики вероятностей ошибки для этих кодов приведены на рис. 2.2.

По серии графиков на рис. 2.2 легко видеть, что точка перегиба кривой вероятности ошибки всегда находится в области целевых параметров входной и выходной вероятностей ошибки.

На рис. 2.3 отдельные кривые соответствуют вероятностям ошибочного декодирования отдельных слоёв; из них легко видеть, что вероятность ошибочного декодирования конструкции в целом, как правило, обусловлена вероятностью ошибки наихудшего слоя. При этом следует отметить, что при вероятностях ошибки больше целевых таким слоем, как правило, является первый, а при меньших — один из промежуточных.

По поведению серии кривых наиболее хорошо видно, что при заданной выходной вероятности ошибки предложенный алгоритм проводит оптимизацию параметров внешних кодов таким

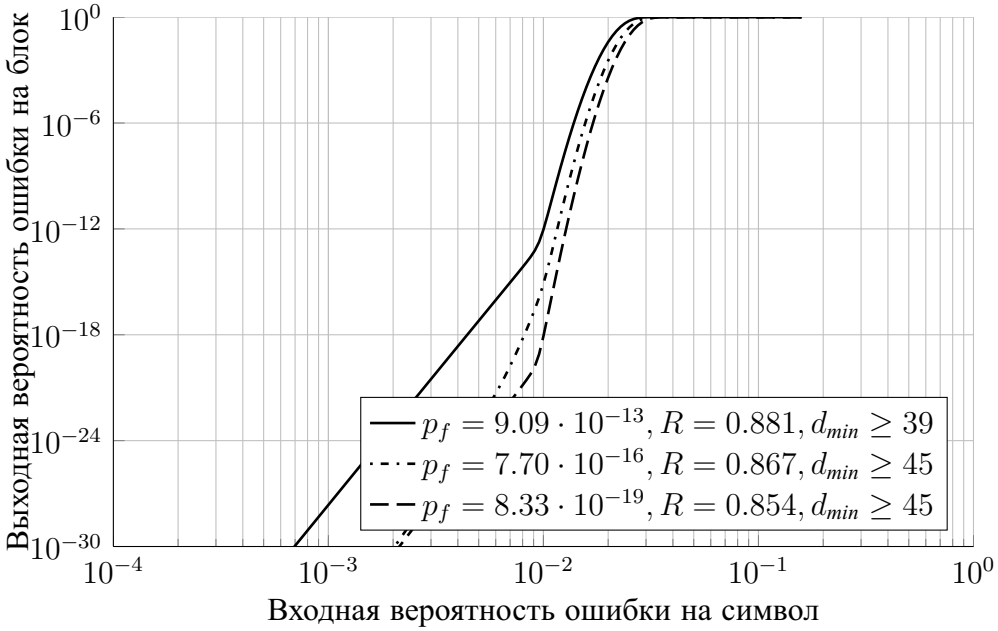


Рисунок 2.2. Вероятности ошибки отдельных внешних кодов в зависимости от входной вероятности ошибки для кода, построенного с целевыми $p_s = 10^{-2}$, $p_f \in \{10^{-12}, 10^{-15}, 10^{-18}\}$; параметры конструкции: $q = 16$, $n_A = 16$, $n_B = 256$, $n = 4096$

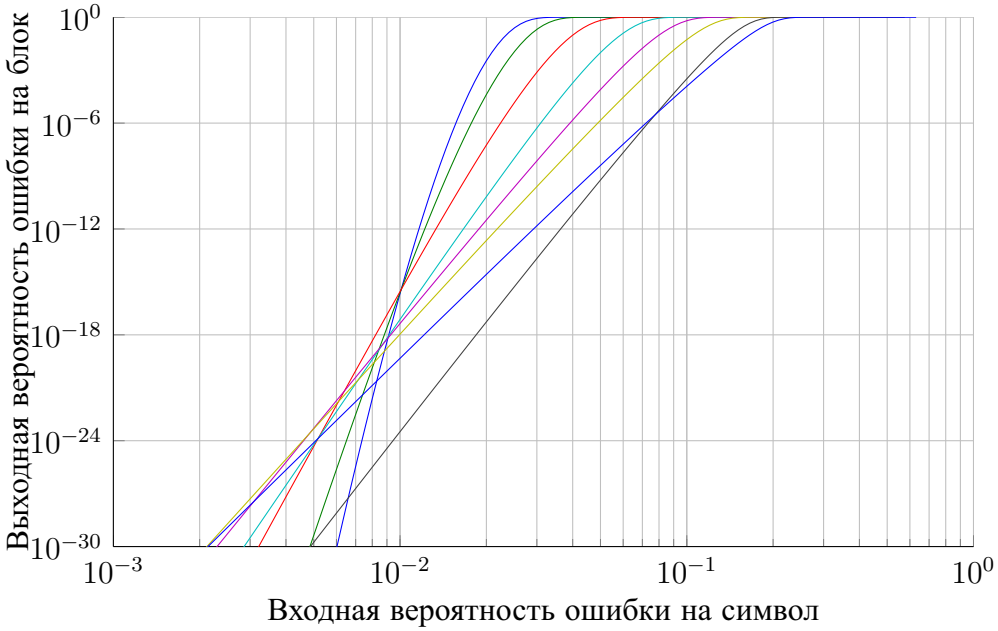


Рисунок 2.3. Вероятности ошибки отдельных внешних кодов в зависимости от входной вероятности ошибки для кода, построенного с целевыми $p_s = 10^{-2}$, $p_f = 10^{-15}$, итоговые $R = 0.867$, $p_f = 7.70 \cdot 10^{-16}$, $d_{min} \geq 45$; параметры конструкции: $q = 16$, $n_A = 16$, $n_B = 256$, $n = 4096$

образом, чтобы каждый из них достигал требуемую вероятность неправильного декодирования при минимальной его избыточности. Благодаря этому в районе целевой вероятности ошибки видно пересечение нескольких кривых, остальные кривые проходят ниже. При уменьшении входной вероятности ошибки (движении справа налево по оси абсцисс графика) до p_s вероятность неправильного декодирования, как правило, мажорируется вероятностью неправильного

декодирования первого внешнего кода, а после неё — одного из промежуточных внешних кодов. Как правило, это тот код, у которого наихудшая скорость убывания вероятности неправильного декодирования. В самой точке наблюдается перегиб. Следует отметить, что после точки перегиба вероятность неправильного декодирования достаточно быстро падает при уменьшении входной вероятности ошибки.

Предложенный алгоритм даёт код с наилучшей скоростью при заданных входной и выходной вероятностях. Коды, построенные для других входных и выходных вероятностей и обеспечивающие ту же p_f при той же p_s, дадут худшую скорость.

Это легко показать, построив несколько кодов, оптимизированных под различные параметры, но дающие одинаковые вероятности ошибки в выбранной точке. Пример такого построения показан на рис. 2.4. В данном случае ставилась задача построить коды, которые будут обладать одинаковыми вероятностями неправильного декодирования $p_f = 10^{-15}$ при входной вероятности ошибки $p_s = 10^{-2}$, но оптимизировать их под различные целевые $p_f \in \{10^{-10}, 10^{-15}, 10^{-20}\}$, подбирая параметр p_s. Получившиеся параметры кодов равны $p_s = 10^{-2}$, $p_f = 7.7 \cdot 10^{-16}$, $R = 0.867$, $d_{min} \geq 45$ для оптимального, а также $p_s = 8.2 \cdot 10^{-3}$, $p_f = 9.4 \cdot 10^{-21}$, $R = 0.862$, $d_{min} \geq 45$ и $p_s = 1.49 \cdot 10^{-2}$, $p_f = 8.17 \cdot 10^{-11}$, $R = 0.855$, $d_{min} \geq 39$. Очевидно, что код, построенный сразу для целевых параметров, имеет наибольшую скорость. Если ОЛО-код построен для других параметров, то дополнительная избыточность внешних кодов используется по-разному. Если требуется более эффективная работа в области больших входных вероятностей ошибки, то она идёт на увеличение избыточности первых внешних кодов. В противном случае, если требуется достижение более низких вероятностей неправильного декодирования, то эта избыточность используется для улучшения промежуточных кодов.

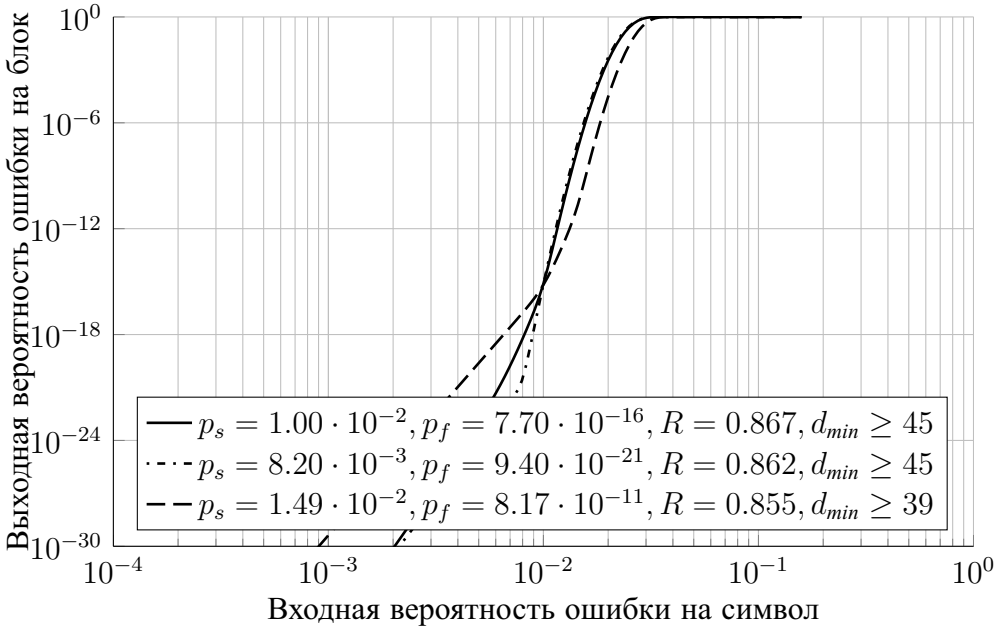


Рисунок 2.4. Вероятности ошибки кодов с одинаковой вероятностью неправильного декодирования $p_f \leq 10^{-15}$ при входной вероятности ошибки $p_s = 10^{-2}$, построенных с разными целевыми p_s и p_f; параметры конструкции: $q = 16$, $n_A = 16$, $n_B = 256$, $n = 4096$

Тот факт, что для наилучшей работы кодовой конструкции требуется наименьшая длина внутренних кодов и наибольшая — внешних, можно проиллюстрировать на примере. Нами были построены три кода с одинаковой полной длиной $n_A \times n_B = 1024$ 16-ричных символа с различными соотношениями длин внутренних и внешних кодов, $n_A \times n_B \in \{4 \times 256, 8 \times 128, 16 \times 64\}$. Целевой выходной вероятностью ошибки была задана $p_f = 10^{-15}$, входные вероятности ошибки $p_s \in \{9.6 \cdot 10^{-3}, 6.8 \cdot 10^{-3}, 5.1 \cdot 10^{-3}\}$ были подобраны таким образом, чтобы скорость построенных кодов была равна $R = 0.8 \pm 0.002$. Результат представлен на рис. 2.5. Из этой серии графиков легко видно, что наилучшими характеристиками обладает именно конструкция с параметрами $n_A \times n_B = 4 \times 256$.

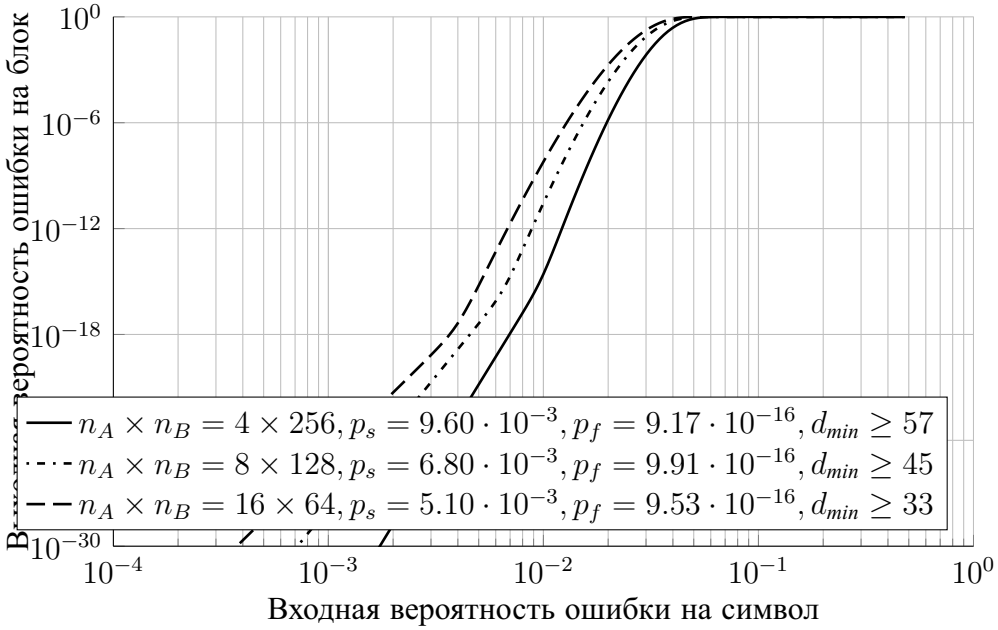


Рисунок 2.5. Вероятности ошибки кодов с одинаковой полной длиной и различной длиной внутренних кодов. Целевые выходные вероятности ошибки $p_f = 10^{-15}$, целевые входные вероятности ошибки подбирались таким образом, чтобы получить равные скорости кодов $R = 0.8 \pm 0.002$; параметры конструкции: $q = 16$, $n = 1024$

В то же время, если построить набор кодов с одинаковыми параметрами входной и выходной вероятностей ошибки, но разными длинами кодов-компонентов, то их скорости будут существенно отличаться, а поведение будет практически идентично. Серия таких кривых показана на рис. 2.6.

Пример построения нижней границы вероятности неправильного декодирования показан на рис. 2.7. Видно, что при вероятностях выше точки с целевыми параметрами нижняя граница близка к верхней, т. к. в данном случае неправильное декодирование происходит, как правило, именно из-за неправильного декодирования первого внешнего кода.

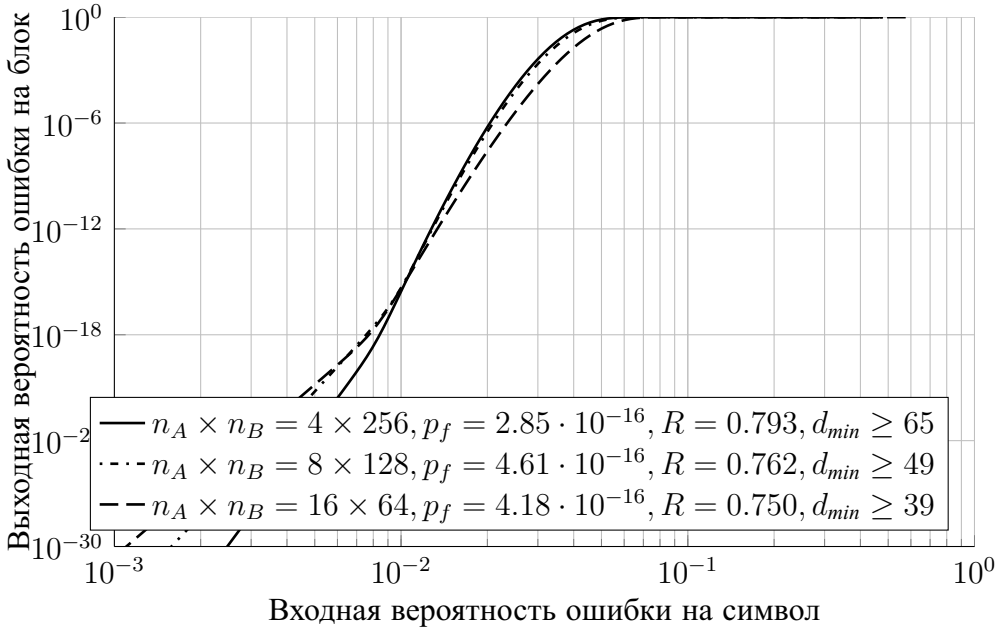


Рисунок 2.6. Вероятности ошибки кодов с одинаковой полной длиной и различной длиной внутренних кодов, построенных с целевыми входной вероятностью ошибки $p_s = 10^{-2}$ и выходной $p_f = 10^{-15}$; параметры конструкции: $q = 16$, $n = 1024$

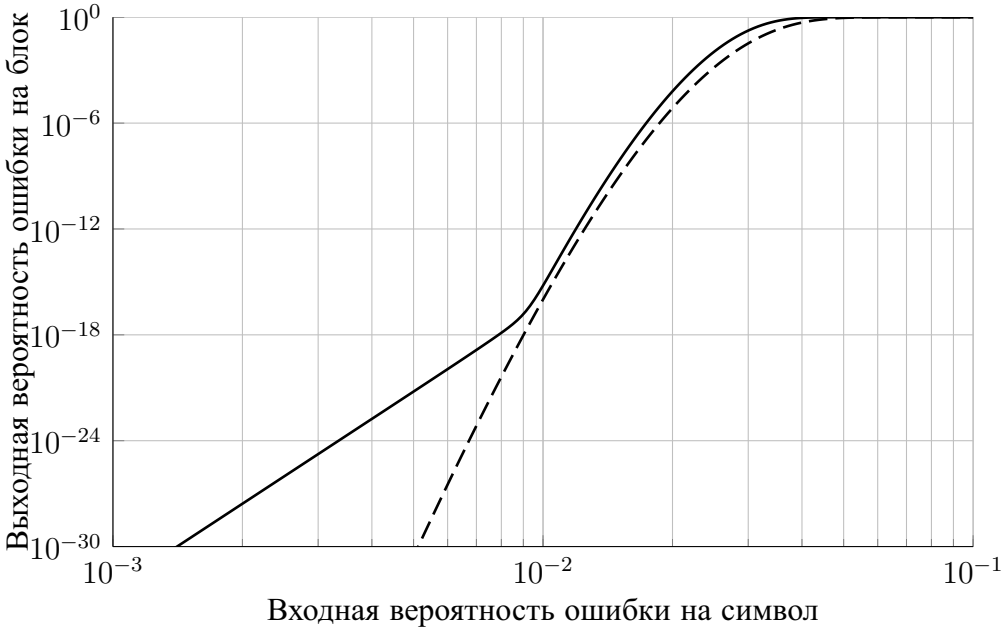


Рисунок 2.7. Верхняя и нижняя границы вероятности неправильного декодирования входной вероятности ошибки для кода, построенного с целевыми $p_s = 10^{-2}$, $p_f = 10^{-15}$; полученная вероятность неправильного декодирования при входной вероятности ошибки $p_s = 10^{-2}$ ограничена $1.01 \cdot 10^{-16} \leq p_f \leq 6.18 \cdot 10^{-16}$; параметры конструкции: $q = 16$, $n_A = 8$, $n_B = 256$, $n = 2048$

2.3 ОЛО-коды над одним алфавитом

2.3.1 Конструкция ОЛО-кодов

Опишем выбранную конструкцию кодов с обобщённой локализацией ошибок. Назовём его *равнопольным* ОЛО-кодом.

Кодовым словом *равнопольного* q-ичного ОЛО-кода $\mathcal{C}$ назовём матрицу $\mathbf{C}$ над полем $GF(q)$ размеров $n_A \times n_B$, где n_A — длина внутренних кодов, а n_B — длина внешних кодов.

Введём обозначения: $\mathcal{T}_j(\mathbf{M})$ — первые j строк матрицы $\mathbf{M}$, $\mathcal{R}_j(\mathbf{M})$ — j-я строка матрицы $\mathbf{M}$, где $\mathbf{M}$ — произвольная матрица.

Обозначим $\mathbf{H}$ проверочную матрицу системы вложенных внутренних кодов $\{\mathcal{C}_j^A\}$. Её элементы принадлежат полю $GF(q)$, её размер $n_A \times n_A$. Эта матрица должна быть невырождена. Любая матрица $\mathcal{T}_j(\mathbf{H})$ размера $j \times n_A$, будет являться проверочной матрицей внутреннего кода длины n_A с $k_A = n_A - j$ информационных символов.

$$\mathcal{T}_j(\mathbf{H}) = \quad . \tag{2.25}$$

Будем нумеровать эти коды естественным образом: j-й внутренний код — это код, у которого число информационных символов, проверочных символов и расстояние равно $k_j^A = n_A - j$, $r_j^A = j$ и $d_j^A = j + 1$ соответственно.

Назовём

$$\mathbf{S} = \mathbf{H} \cdot \mathbf{C} = \tag{2.26}$$

— матрицей синдромов внутренних кодов.

Внешние коды *равнопольного* ОЛО-кода $\mathcal{C}_j^B$ — коды над полем $GF(q)$ длины n_B и имеют числа информационных символов r_j^B, скорости $R_j^B = 1 - r_j^B/n_B$, числа информационных символов $k_j^B = n_B - r_j^B$, кодовые расстояния d_j^B.

О п р е д е л е н и е 6. ОЛО-код — это множество матриц $\mathbf{C}$ таких, что строки $\mathbf{s}_j = \mathcal{R}_j(\mathbf{S})$, $j = \overline{1,n_A}$ матрицы $\mathbf{S} = \mathbf{H} \cdot \mathbf{C}$ являются кодовыми словами внешних кодов.

Заданный таким образом ОЛО-код будет линейным кодом над полем $GF(q)$ длины $n = n_A n_B$, число его проверочных символов $r = \sum_j r_j^B$. Скорость кода $R = \sum_j R_j^B/n_A$, то есть равна среднему арифметическому скоростей внешних кодов. Кодовое расстояние ограничено снизу $d \geq \min_{j=\overline{2,n_A}}\{d_1^B, d_{j-1}^A d_j^B\}$ [6]. На рис. 2.8 показаны эта нижняя граница для кодов с $q = n_A = n_B = 256$ и граница Варшамова-Гилберта для $q = 256$.

Следует отметить, что в кодовом слове $\mathbf{C}$ ОЛО-кода в явном виде не присутствуют кодовые слова ни внутренних, ни внешних кодов. Если быть точным, в столбцах кодового слова $\mathbf{C}$ ОЛО-кода находятся слова смежных классов внутренних кодов, в то время как выбор смежного класса

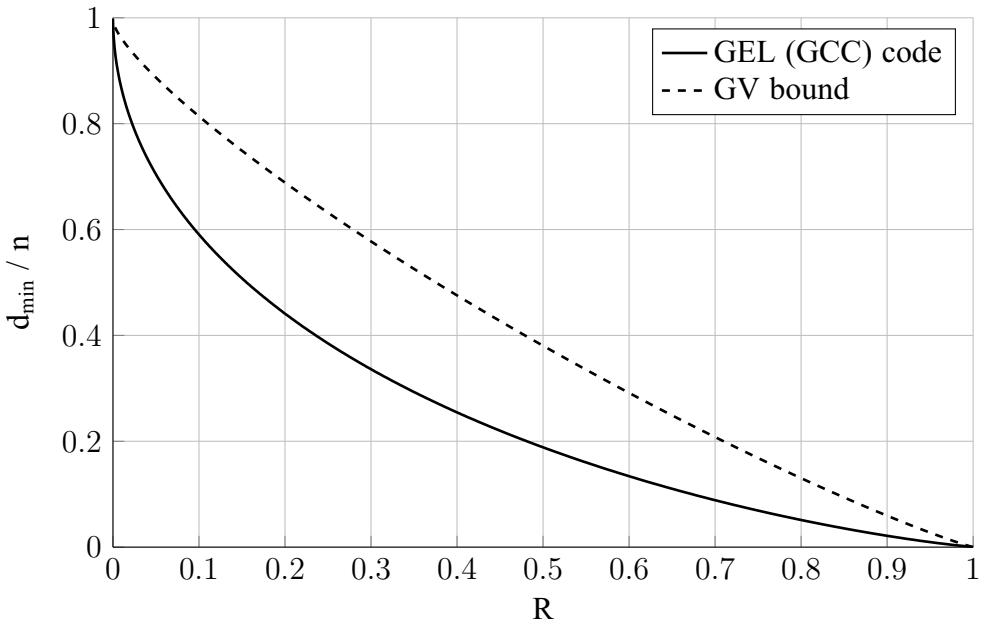


Рисунок 2.8. Нижние границы на кодовое расстояние: граница Варшамова-Гилберта и нижняя граница для ОЛО (ОКК) кодов [6]

в i-м столбце $\mathbf{C}$ определяется теми символами слов внешних кодов, которые находятся в i-м столбце матрицы $\mathbf{S}$ (на i-х позициях вектор-строк $\mathbf{s}_j$).

При этом спектр расстояний смежного класса и, как следствие, его корректирующая способность, в точности совпадают с таковыми у внутреннего кода.

Ниже рассмотрим конструкцию, где качестве внутренних и внешних кодов используются МДР-коды.

2.3.2 Кодирование

Несистематическое кодирование

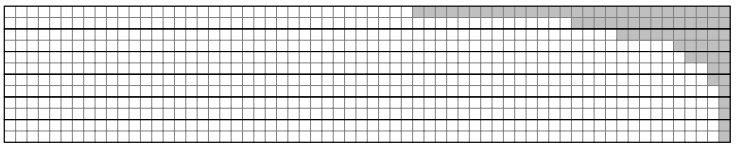
Несистематическое кодирование ОЛО-кодов выписывается наиболее просто. Пусть на входе алгоритма кодирования есть q-ичная матрица $\mathbf{I}$, имеющая такой же размер, как и матрица $\mathbf{S}$. В каждой из строк $\mathbf{I}_j = \mathcal{R}(\mathbf{I})$ первые k_j^B символов — информационные, а последние r_j^B — нулевые (в (2.27) элементы с информационными символами обозначены белым цветом, с нулевыми — тёмно-серым):

$$\mathbf{I} = \quad . \tag{2.27}$$

Для того, чтобы получить кодовое слово ОЛО-кода, требуется сначала закодировать внешними кодами строки матрицы $\mathbf{I}$. Кодеры внешних кодов используют в качестве информационных те символы, которые стоят на соответствующих местах в вектор-строках $\mathbf{I}_j$, и возвращают

вектор-строки соответствующих им кодовых слов $\mathbf{s}_j$. В свою очередь, $\mathbf{s}_j$ являются компонентами матрицы синдромов внешних кодов $\mathbf{S}$:

$$\mathbf{S} = \quad .$$

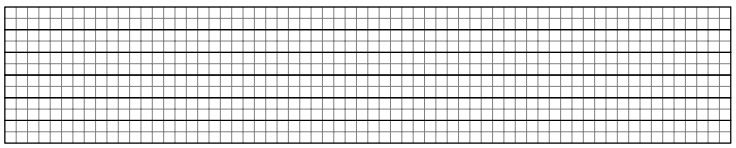


(2.28)

Обозначим эту операцию следующим образом: $\mathbf{S} = Enc_B\{\mathbf{I}\}$. После выполнения этой операции элементы на местах информационных символов в матрицах $\mathbf{I}$ и $\mathbf{S}$ совпадут, а на местах проверочных символов в матрице $\mathbf{S}$ будут проверочные символы внешних кодов.

Затем для того, чтобы получить кодовое слово, нужно умножить полученную матрицу на матрицу, обратную $\mathbf{H}$: $\mathbf{C} = \mathbf{H}^{-1} \cdot \mathbf{S}$. Как уже было сказано, матрица $\mathbf{H}$ должна быть невырождена.

$$\mathbf{C} = \mathbf{H}^{-1} \cdot \mathbf{S} = \quad .$$



(2.29)

Таким образом, отсюда следует

Т е о р е м а 5. *Для несистематического кодирования матрицы* $\mathbf{I}$ *с информационными символами требуется выполнить следующую операцию:*

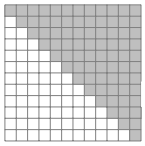
$$\mathbf{C} = \mathbf{H}^{-1} \cdot Enc_B\{\mathbf{I}\}. \tag{2.30}$$

Доказательство. Легко показать, что она будет удовлетворять определению кодового слова ОЛО-кода: $\mathbf{H} \cdot \mathbf{C} = \mathbf{H} \cdot \mathbf{H}^{-1} \cdot \mathbf{S} = \mathbf{S}$, и в строках $\mathbf{S}$ по построению находятся кодовые слова внешних кодов. □

Систематическое кодирование

Выше на матрицу $\mathbf{H}$ не накладывалось никаких ограничений кроме невырожденности. Ниже для реализации систематического кодирования та же матрица потребуется в верхней треугольной форме и будет обозначена $\mathbf{U}$:

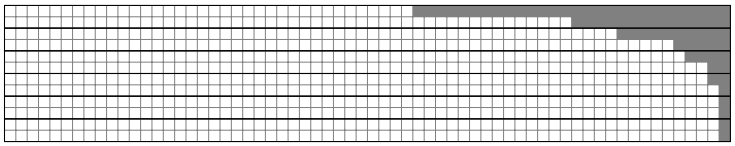
$$\mathbf{U} = \quad .$$



(2.31)

В таком случае матрица $\mathbf{U}^{-1}$, обратная к $\mathbf{U}$, также будет иметь верхний треугольный вид. Для того, чтобы закодировать матрицу с информационными символами (2.27), для начала умножим её на $\mathbf{U}$:

$$\mathbf{R} = \mathbf{U} \cdot \mathbf{I} = \quad .$$



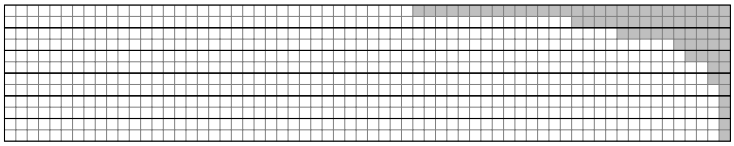
(2.32)

Благодаря верхней треугольной форме матрицы $\mathbf{U}$ структура матрицы $\mathbf{R}$ будет совпадать со структурой матрицы $\mathbf{I}$. После этого $\mathbf{R}$ нужно закодировать внешними кодами аналогично кодированию матрицы $\mathbf{I}$ в несистематическом случае.

Так же, как выше, обозначим эту операцию следующим образом: $\mathbf{S} = Enc_B\{\mathbf{R}\}$. После выполнения этой операции элементы на местах информационных символов в матрицах $\mathbf{R}$ и $\mathbf{S}$ совпадут, а на местах проверочных символов в матрице $\mathbf{S}$ будут проверочные символы внешних кодов.

После этого получившуюся матрицу остаётся умножить на $\mathbf{U}^{-1}$:

$$\mathbf{C} = \mathbf{U}^{-1} \cdot \mathbf{S} = \mathbf{U}^{-1} \cdot Enc_B\{\mathbf{U} \cdot \mathbf{I}\} = \quad .$$



(2.33)

Таким образом, может быть сформулирована

Т е о р е м а 6. *Для систематического кодирования матрицы* $\mathbf{I}$ *с информационными символами требуется выполнить следующую операцию:*

$$\mathbf{C} = \mathbf{U}^{-1} \cdot Enc_B\{\mathbf{U} \cdot \mathbf{I}\}. \tag{2.34}$$

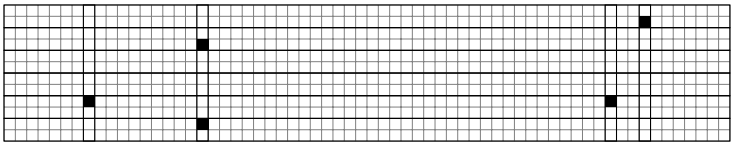
Доказательство. Легко показать, что она будет удовлетворять определению кодового слова ОЛО-кода: $\mathbf{U} \cdot \mathbf{C} = \mathbf{U} \cdot \mathbf{U}^{-1} \cdot Enc_B\{\mathbf{U} \cdot \mathbf{I}\} = \mathbf{S}$, и в строках $\mathbf{S}$ по построению находятся кодовые слова внешних кодов.

Благодаря верхней треугольной форме матрицы $\mathbf{U}$ в результате получившейся процедуры на местах информационных символов матрицы $\mathbf{C}$ будут те же символы, что и в матрице $\mathbf{I}$. □

2.3.3 Алгоритм декодирования и верхняя граница вероятности неправильного декодирования

Пусть после передачи кодового слова $\mathbf{C}$ по каналу в нём произошли ошибки и было принято слово $\mathbf{V}$:

$$\mathbf{V} = \mathbf{C} + \mathbf{E} = \quad , \tag{2.35}$$



где $\mathbf{E}$ — матрица ошибок, внесённых каналом. Рассмотрим канал с независимыми ошибками с вероятностью ошибки, равной p_s. При этом нас не будет интересовать характер ошибок, иными словами, единственным требованием к каналу является вероятность правильной передачи символа, равная $1 - p_s$.

В данном слове чёрным обозначены символы, в которых произошли ошибки; рамками обведены столбцы, которые содержат ошибочные символы.

Декодирование будет происходить в n_A шагов, на каждом из которых будет последовательно происходить сначала декодирование внешнего кода, а затем внутреннего. При этом будем рассматривать декодирование на k-м шаге в предположении, что декодирование на всех предыдущих было правильным.

Рассмотрим первые четыре шага декодирования в явном виде.

Первый шаг

Перед декодированием единственным известным является принятая матрица $\mathbf{V}$. Для унификации обозначим её $\mathbf{V}_0$, то есть $\mathbf{V}_0 = \mathbf{V}$.

1. На первом шаге декодирования слово $\mathbf{V}_0$ умножается слева на первую строку матрицы $\mathbf{H}$:

$$\mathbf{S}'_1 = \mathcal{R}_1(\mathbf{H}) \cdot \mathbf{V}_0 = \quad \cdot \tag{2.36}$$

Результатом этого умножения является строка длины n_B. Если передача по каналу не привела к появлению ошибок и $\mathbf{V}$ совпадает с переданным словом $\mathbf{C}$, то $\mathbf{S}'_1$ равна первой строке матрицы $\mathbf{S}$, то есть $\mathbf{S}'_1 = \mathcal{R}_1(\mathbf{S})$. В таком случае $\mathbf{S}'_1$ будет являться кодовым словом первого внешнего кода. Ошибки в $\mathbf{V}$ будут приводить к появлению ошибок в $\mathbf{S}'_1$. Такие символы $\mathbf{S}'_1$ отмечены на рисунке чёрным.

Рассмотрим строку $\mathbf{\Delta}_1 = \mathcal{R}_1(\mathbf{H}) \cdot \mathbf{E}$. Очевидно, что $\mathbf{\Delta}_1 = \mathcal{R}_1(\mathbf{H}) \cdot (\mathbf{V} - \mathbf{C}) = \mathbf{S}'_1 - \mathcal{R}_1(\mathbf{S})$. Эта строка является отображением вектора $\mathbf{E}$ в вектор ошибки для первого внешнего кода.

Те столбцы $\mathbf{E}$ (и, как следствие, $\mathbf{V}$), которые приведут к появлению ненулевых символов в строке $\mathbf{\Delta}_1$, приведут к ошибкам к $\mathbf{S}'_1$. Иными словами, это те столбцы из $\mathbf{E}$, ошибки в которых способен обнаружить первый внутренний код. Так как ошибки в символах $\mathbf{E}$ независимы, то и столбцы $\mathbf{E}$ тоже будут независимы.

Нас интересует оценка сверху на вероятность неправильного декодирования. Для её вычисления для начала оценим сверху вероятность того, что символы $\mathbf{S}'_1$ будут содержать ошибки.

Расстояние первого внутреннего кода равно 2, он гарантированно обнаруживает ошибки кратности 1, но также может обнаруживать ошибки и более высокой кратности (те, которые не являются словами этого кода). Для получения оценки сверху предположим, что первый внутренний код обнаруживает любые комбинации ошибок в столбце. Это допущение приведёт к завышению реального числа ошибок, появляющихся в $\mathbf{S}'_1$, но не может приводить к недооценке этого числа. В таком случае вероятность появления одной и более ошибки в отдельном символе может быть вычислена как дополнение к вероятности того, что в столбце не произойдёт ни одной ошибки, то есть $p_+\{1\} = 1 - (1 - p_s)^{n_A}$.

Так как столбцы $\mathbf{E}$ независимы, то ошибки в символах $\mathbf{S}'_1$ в данном случае тоже являются независимыми.

2. После вычисления $\mathbf{S}'_1$ естественным является использование первого внешнего кода для исправления в ней ошибок. Обозначим эту операцию следующим образом:

$$\mathbf{S}''_1 = Dec_B\{\mathbf{S}'_1\}. \tag{2.37}$$

При условии, что первый внешний код обладает достаточной корректирующей способностью, чтобы исправить все ошибки в $\mathbf{S}'_1$, $\mathbf{S}''_1$ совпадёт с первой строкой матрицы $\mathbf{S}$, то есть будет выполнено $\mathbf{S}''_1 = \mathcal{R}_1(\mathbf{S})$.

Введём также обозначение $\mathbf{W}_1 = \mathbf{S}''_1 - \mathbf{S}'_1$. *Оно является вектором ошибки в* $\mathbf{S}'_1$, *который оказывается найден первым внешним кодом.*

Так как у нас есть оценка сверху на вероятность ошибки в отдельных символах слова внешнего кода, мы можем оценить сверху вероятность его неправильного декодирования. Вероятность ошибки на входе декодера оценим сверху как $p_t\{1\} = p_+\{1\}$.

Тогда вероятность ошибки первого внешнего кода можно оценить сверху величиной

$$p_{B1} = \sum_{m=\lfloor (r_1^B)/2 \rfloor + 1}^{n_B} \binom{n_B}{m} p_t^m (1 - p_t)^{n_B - m}, \tag{2.38}$$

где под p_t понимается $p_t\{1\}$.

Если внешний код выдаст отказ от декодирования, то алгоритм декодирования останавливается, и выдаётся отказ от декодирования всей кодовой конструкции. Если в результате декодирования внешнего кода произойдёт ошибка (переход в неправильное слово внешнего кода), то возможны два варианта: или ошибка декодирования конструкции в целом, или отказ на одном из дальнейших шагов.

3. После декодирования первого внешнего кода нам известна строка $\mathbf{S}''_1$, содержащая исправленные синдромы первого внутреннего кода. Первый внутренний код имеет расстояние 2, поэтому он гарантированно обнаружит ошибки веса 1. Это значит, что столбцы, содержащие ошибки веса 1, будут отмечены.

Обозначим операцию декодирования внутренних кодов следующим образом:

$$\mathbf{V}_1 = Dec_A[\mathbf{S}''_1]\{\mathbf{V}\}. \tag{2.39}$$

На данном шаге внутренний код не исправляет ошибки, поэтому декодирование в привычном понимании не требуется. Рассмотрим эту операцию подробнее.

После декодирования первого внешнего кода нам известна строка $\mathbf{W}_1$*, ненулевые элементы которой указывают на столбцы, которые содержат ошибки. Декодирование сводится к замене этих столбцов на стирания. Обозначим* $\mathbf{V}_1$ *матрицу, которая равна матрице* $\mathbf{V}$ *во всех столбцах за исключением тех, в которых обнаружены ошибки. Столбцы* $\mathbf{V}_1$*, которые соответствуют столбцам с обнаруженными в* $\mathbf{V}$ *ошибками, будут содержать стёртые символы.*

Здесь следует обратить внимание на то, что декодеры внутренних кодов исправляют ошибки каждый раз в матрице $\mathbf{V}$, принятой из канала. Кроме матрицы $\mathbf{V}$ они в качестве параметра получают матрицу $\mathbf{S}''_1$, указывающую на смежные классы внутренних кодов, в которых производится декодирование.

Как было отмечено выше, первые внутренние коды обладают расстоянием 2, и способны только обнаруживать, но не исправлять ошибки. Поэтому в результате матрица $\mathbf{V}_1$ будет иметь столбцы либо совпадающие с $\mathbf{V}$, либо стёртые.

На этом первый шаг декодирования заканчивается, его результатами являются матрицы $\mathbf{S}''_1$ (размера $1 \times n_B$) и $\mathbf{V}_1$. В дальнейшем, после j-го шага результатом алгоритма будут матрицы $\mathcal{T}_j(\mathbf{S}'')$ (размера $j \times n_B$) и $\mathbf{V}_j$. *Также результатом декодирования является матрица* $\mathbf{W}_1$.

Второй шаг

1. Полученное на первом шаге слово $\mathbf{V}_1$ умножается слева на вторую строку матрицы $\mathbf{H}$:

$$\mathbf{S}'_2 = \mathcal{R}_2(\mathbf{H}) \cdot \mathbf{V}_1 = \quad \cdot \tag{2.40}$$

Результатом этого умножения, как и на первом шаге, является строка длины n_B. Так как в матрице $\mathbf{V}_1$ часть столбцов стёрты, будем считать результат умножения стёртого столбца на $\mathcal{R}_2(\mathbf{H})$ равным стёртому символу. Если некоторый столбец матрицы $\mathbf{V}$ после приёма из канала содержал ошибку, которая не была обнаружена и стёрта первым внутренним кодом, то соответствующий столбец $\mathbf{V}_1$ будет содержать ту же ошибку.

Кодовое расстояние внутреннего кода на предыдущей итерации нам гарантирует обнаружение ошибок кратности 1. Для оценки сверху вероятностей ошибки и стирания в $\mathbf{S}'_2$ естественным является предположить, что ошибки веса 1 перейдут в стирания, а ошибки веса 2 и более — в ошибки. В отличие от первого шага, эти вероятности не могут быть выписаны столь же тривиальной формулой, поэтому напишем формулы в общем случае.

Вероятность того, что в столбце произойдёт t ошибок, равна:

$$p\{t\} = \binom{n_A}{t} p_s^i (1-p_s)^{n_A - t}. \tag{2.41}$$

Вероятность того, что ошибок будет не менее, чем t, равна:

$$p_+\{t\} = \sum_{i=t}^{n_A} p\{i\} = \sum_{i=t}^{n_A} \binom{n_A}{i} p_s^i (1-p_s)^{n_A - i}. \tag{2.42}$$

2. Далее можно использовать второй внешний код для исправления ошибок и стираний в строке $\mathbf{S}'_2$:

$$\mathbf{S}''_2 = Dec_B\{\mathbf{S}'_2\}. \tag{2.43}$$

При условии, что второй внешний код обладает достаточной корректирующей способностью, чтобы исправить все ошибки и стирания в $\mathbf{S}'_2$, $\mathbf{S}''_2$ совпадёт со второй строкой матрицы $\mathbf{S}$.

Так как у нас есть оценка сверху на вероятности ошибки и стирания в отдельных символах слова, мы можем оценить сверху вероятность неправильного декодирования первого внешнего кода. Вероятности ошибки и стирания на входе его декодера оценим как $p_t\{2\} = p_+\{2\}$, $p_e\{2\} = p\{1\}$. В данном случае оценка вероятности стирания не является верхней, так как часть стираний мы учитываем как ошибки.

Тогда вероятность ошибки второго внешнего кода можно оценить сверху величиной

$$p_{B2} = \sum_{i=0}^{r_2^B} \sum_{m=\lfloor (r_2^B - i)/2 \rfloor + 1}^{n_B - i} \binom{n_B}{i, m, n_B} p_e^i p_t^m (1 - p_e - p_t)^{n_B - i - m} + \sum_{i=r_2^B+1}^{n_B} \binom{n_B}{i} p_e^i (1-p_e)^{n_B - i}, \tag{2.44}$$

где под p_t и p_e понимаются $p_t\{2\}$ и $p_e\{2\}$ соответственно.

При декодировании будет строиться матрица $\mathbf{S}''$, составленная из результатов декодирования внешних кодов. По сути, она будет являться оценкой матрицы $\mathbf{S}$ декодером и будет совпадать с ней по размеру. На j-м шаге нас будут интересовать только первые j её строк $\mathcal{T}_j(\mathbf{S}'')$, при этом её j-я строка будет получаться исправлением ошибок в $\mathbf{S}'_j$ на текущем шаге: $\mathcal{R}_j(\mathbf{S}'') = \mathbf{S}''_j$. Очевидно, что строки с номерами от 1 до $j-1$ будут получены на предыдущих шагах. Хотя все эти строки являются выходами внешних кодов, следует отметить, что для их получения используются, в том числе и результаты декодирования внутренних кодов на предыдущих шагах.

Также будем строить матрицу $\mathbf{W}$*, содержащую в своих строках разности* $\mathbf{S}''_j - \mathbf{S}'_j$*, то есть* $\mathcal{R}_j(\mathbf{W}) = \mathbf{S}''_j - \mathbf{S}'_j$.

При условии, что внешний код обладает достаточной корректирующей способностью, чтобы исправить все ошибки в $\mathbf{S}'_2$, и на предыдущих итерациях декодирование было правиль-

ным, первые 2 строки матрицы $\mathbf{S}''$ укажут правильные смежные классы для декодирования внутренних кодов. Как и на первом шаге, на этом и на всех последующих, если внешний код выдаёт отказ от декодирования, то алгоритм декодирования останавливается, и выдаётся отказ от декодирования всей кодовой конструкции. Если в результате декодирования внешнего кода произойдёт ошибка, то в результате декодирования ОЛО-кода либо будет получено неправильное кодовое слово, либо на одной из следующих итераций может быть выдан отказ от декодирования.

3. С использованием полученной на предыдущем шаге алгоритма матрицы $\mathcal{T}_2(\mathbf{S}'')$, с помощью вторых внутренних кодов можно исправить ошибки в столбцах матрицы $\mathbf{V}$:

$$\mathbf{V}_2 = Dec_A[\mathcal{T}_2(\mathbf{S}'')]\{\mathbf{V}\}. \tag{2.45}$$

Эти смежные классы можно декодировать непосредственно, например, по синдромной решётке.

Альтернативой является использование для декодирования в смежном классе декодера самого кода. Для этого требуется вычесть из принятого слова представителя того смежного класса, в котором требуется декодировать. Полученная разность будет равна сумме некоторого кодового слова и слова ошибки. Если после исправления ошибок в таком слове к результату обратно прибавить того же представителя смежного класса, будет получен результат декодирования в смежном классе.

Обозначим $\mathbf{H}_j^{-1}$ матрицу, состоящую из первых j столбцов матрицы $\mathbf{H}^{-1}$, обратной к $\mathbf{H}$. Тогда представителя смежного класса можно получить как $\mathbf{b} = \mathbf{H}_j^{-1} \cdot \mathcal{T}_j(\mathbf{S}'')$. Операцию декодирования внутренних кодов тогда можно записать как:

$$\mathbf{V}_j = Dec_A\{\mathbf{V} - \mathbf{b}\} + \mathbf{b}. \tag{2.46}$$

Рассмотрим матрицу $\mathcal{T}_j(\mathbf{W})$, содержащую первые строки матрицы $\mathbf{W}$. Столбцы этой матрицы содержат отличия между $\mathcal{T}_j(\mathbf{S}'')$, которая получена декодированием внешних кодов и при условии правильного декодирования совпадает с $\mathcal{T}_j(\mathbf{S})$, и принятой из канала $\mathcal{T}_j(\mathbf{S}')$. По сути, такой столбец является проекцией матрицы ошибок $\mathbf{E}$. Рассмотрим декодер, который по синдрому внутреннего кода возвращает соответсвующий этому синдрому вектор-столбец ошибки. Обозначим такую операцию как $\mathbf{E}_j = Dec_A(\mathcal{T}_j(\mathbf{W}))$, где подразумевается, что на каждый столбец матрицы $\mathcal{T}_j(\mathbf{W})$ декодер возвращает соответствующий столбец ошибки в матрице $\mathbf{E}_j$.

В таком случае декодирование смежных классов в столбцах $\mathbf{V}$ может быть выполнено как поиск $\mathbf{E}_j$ и вычисление $\mathbf{V}_j = \mathbf{V} - \mathbf{E}_j$. В случае отказов от декодирования внутренних кодов $\mathbf{E}_j$ будет содержать столбцы со стираниями. Разность столбца из $\mathbf{V}$ со стёртым столбцом из $\mathbf{E}_j$ будем считать равной стёртому столбцу.

Вторые внутренние коды имеют расстояние 3, поэтому они гарантируют исправление ошибок кратности 1, а также могут обнаруживать некоторые ошибки более высокой кратности.

Это означает, что столбцы веса 1, содержавшиеся в $\mathbf{V}$, будут исправлены. Про столбцы, содержащие ошибки большего веса, нельзя сказать ничего определённого. Возможные варианты включают в себя стирание столбцов, внесение ошибок внутренними кодами или неизменность. В то же время можно с уверенностью утверждать, что $\mathbf{V}_2$ не может содержать ошибок в столбцах, которые были правильными в $\mathbf{V}$.

Таким образом, $\mathbf{V}_2$ не будет содержать ошибок в столбцах, в которых $\mathbf{V}$ содержала ошибки веса 1 и менее. Часть столбцов $\mathbf{V}_2$, в которых $\mathbf{V}$ содержала ошибки, могут быть обнаружены и перейти в стирания.

Третий шаг

1. Слово $\mathbf{V}_2$ умножается слева на третью строку матрицы $\mathbf{H}$:

$$\mathbf{S}'_3 = \mathcal{R}_3(\mathbf{H}) \cdot \mathbf{V}_2 = \quad \cdot \tag{2.47}$$

Имеющиеся в $\mathbf{V}_2$ ошибки будут приводить к появлению ошибок в $\mathbf{S}'_3$. Как было показано выше, в результате декодировании на предыдущих шагах матрица $\mathbf{V}_2$ не будет содержать ошибок в столбцах, в которых матрица $\mathbf{V}$ содержала ошибки веса 1 и меньше. Некоторая часть ошибок веса 2 и более может перейти либо в стирания, либо в ошибки, которые не обнаруживаются внутренним кодом на текущем шаге. Это будет приводить к тому, что для исправления ошибок и стираний в $\mathbf{S}'_3$ третьему внешнему коду потребуется не большая корректирующая способность, чем если бы число ошибок было равно числу столбцов с ошибками веса 2 и более в $\mathbf{V}$. Для целей получения верхней границы на вероятность неправильного декодирования будет предполагаться, что на входе третьего внешнего кода есть только ошибки.

Оценим вероятности появления ошибки в $\mathbf{S}'_3$ как вероятность появления в $\mathbf{V}$ ошибки веса 2 и более, тогда $p_t\{3\} = p_+\{2\}$, $p_e\{3\} = 0$. $\mathbf{V}_2$ может содержать стирания на месте ошибок, обнаруженных внутренними кодами, но в данном случае эти стирания учитываются как ошибки.

2. После вычисления $\mathbf{S}'_3$ ошибки в ней исправляются третьим внешним кодом:

$$\mathbf{S}''_3 = Dec_B\{\mathbf{S}'_3\}. \tag{2.48}$$

При условии, что этот код обладает достаточной корректирующей способностью, чтобы исправить все ошибки и стирания в $\mathbf{S}'_3$, $\mathbf{S}''_3$ совпадёт с третьей строкой матрицы $\mathbf{S}$.

Теперь можно оценить сверху вероятность неправильного декодирования третьего внешнего кода:

$$p_{B3} = \sum_{m=\lfloor (r_3^B)/2 \rfloor + 1}^{n_B} \binom{n_B}{m} p_t^m (1-p_t)^{n_B - m}, \tag{2.49}$$

где под p_t понимается $p_t\{3\}$.

3. После декодирования третьего внешнего кода нам известна матрица $\mathcal{T}_3(\mathbf{S}'')$, содержащая исправленные синдромы третьего внутреннего кода. Он имеет расстояние 4. Воспользуемся им для гарантированного исправления ошибок веса 1 и обнаружения ошибок веса 2:

$$\mathbf{V}_3 = Dec_A[\mathcal{T}_3(\mathbf{S}'')]\{\mathbf{V}\}. \tag{2.50}$$

Хотя ошибки веса 1 уже один раз были исправлены при вычислении $\mathbf{V}_2$, эти ошибки будут исправляться в матрице $\mathbf{V}$ повторно. Это связано с тем, что при исправлении ошибок в $\mathbf{V}$ кодом с расстоянием 3 ошибки веса 2 и более могли перейти в ошибки большего веса. Например, ошибка веса 2 может перейти в ошибку, равную слову веса 3 этого кода. В то же время внутренний код с расстоянием 4 гарантирует обнаружение ошибок веса 2.

Матрица $\mathbf{V}_3$ не будет содержать ошибок в тех столбцах, в которых матрица $\mathbf{V}$ содержала ошибки веса не более 1, и будет содержать стирания в тех столбцах, в которых матрица $\mathbf{V}$ содержала ошибки веса 2.

Четвёртый шаг

1. Слово $\mathbf{V}_3$ умножается слева на четвёртую строку матрицы $\mathbf{H}$:

$$\mathbf{S}'_4 = \mathcal{R}_4(\mathbf{H}) \cdot \mathbf{V}_3 = \quad \boxed{} \cdot \tag{2.51}$$

Кодовое расстояние внутреннего кода на предыдущей итерации нам гарантирует исправление ошибок кратности 1 и обнаружение ошибок кратности 2. Для оценки сверху вероятностей ошибки и стирания в $\mathbf{S}'_4$ естественным является предположить, что ошибки веса 2 перейдут в стирания. Ошибки веса 3 и более тоже могут перейти в стирания, но мы не будем это учитывать и будем рассматривать их как ошибки.

2. Далее можно использовать четвёртый внешний код для исправления ошибок и стираний в строке $\mathbf{S}'_4$:

$$\mathbf{S}''_4 = Dec_B\{\mathbf{S}'_4\}. \tag{2.52}$$

При условии, что второй внешний код обладает достаточной корректирующей способностью, чтобы исправить все ошибки в $\mathbf{S}'_4$, $\mathbf{S}''_4$ совпадёт с четвёртой строкой матрицы $\mathbf{S}$.

Так как у нас есть оценка сверху на вероятности ошибки и стирания в отдельных символах слова, можно оценить сверху вероятность неправильного декодирования первого внешнего кода. Вероятности ошибки и стирания на входе его декодера оценим как $p_t\{4\} = p_+\{3\}$, $p_e\{4\} = p\{2\}$.

Тогда вероятность ошибки четвёртого внешнего кода можно оценить сверху величиной

$$p_{B4} = \sum_{i=0}^{r_4^B} \sum_{k=\lfloor (r_4^B - i)/2 \rfloor + 1}^{n_B - i} \binom{n_B}{i,k,n_B} p_e^i p_t^k (1-p_e-p_t)^{n_B-i-k} + \sum_{i=r_4^B+1}^{n_B} \binom{n_B}{i} p_e^i (1-p_e)^{n_B-i}, \quad (2.53)$$

где под p_t и p_e понимаются $p_t\{4\}$ и $p_e\{4\}$ соответственно.

При условии, что внешний код обладает достаточной корректирующей способностью, чтобы исправить все ошибки в $\mathbf{S}'_4$, и на предыдущих итерациях декодирование было правильным, первые 4 строки матрицы $\mathbf{S}''$ укажут правильные смежные классы для декодирования внутренних кодов.

3. С использованием полученной на предыдущем шаге алгоритма матрицы $\mathcal{T}_4(\mathbf{S}'')$, с помощью вторых внутренних кодов можно исправить ошибки в столбцах матрицы $\mathbf{V}$:

$$\mathbf{V}_4 = Dec_A[\mathcal{T}_4(\mathbf{S}'')]\{\mathbf{V}\}. \quad (2.54)$$

Четвёртые внутренние коды имеют расстояние 5, поэтому они гарантируют исправление ошибок кратности 2 и менее, а также могут обнаруживать некоторые ошибки более высокой кратности.

Это означает, что столбцы веса 2 и менее, содержавшиеся в $\mathbf{V}$, будут исправлены. Про столбцы, содержащие ошибки большего веса, нельзя сказать ничего определённого.

Произвольный шаг

Выпишем теперь алгоритм декодирования для j-го шага в общем виде, $j = \overline{1,n_A}$.

1. Перед декодированием $\mathbf{V}_0$ полагается равным принятому слову $\mathbf{V}$, $\mathbf{V}_0 = \mathbf{V}$.

2. Полученное ранее слово $\mathbf{V}_{j-1}$ умножается на $\mathcal{R}_j(\mathbf{H})$ слева, получается строка:

$$\mathbf{S}'_j = \mathcal{R}_j(\mathbf{H}) \cdot \mathbf{V}_{j-1} = \quad \cdot \quad (2.55)$$

На нечётном шаге с номером $j = 2l-1$ матрица $\mathbf{V}_{j-1}$ не будет содержать столбцов с ошибками веса меньше l. На чётном шаге с номером $j = 2l$ матрица $\mathbf{V}_{j-1}$ не будет содержать столбцов с ошибками веса меньше l, а столбцы с ошибками веса l в ней будут отмечены как стёртые.

Учитывая, что высота столбца равна n_A, вероятность того, что в столбце произойдёт t ошибок, равна:

$$p\{t\} = \binom{n_A}{t} p_s^i (1-p_s)^{n_A - t}. \tag{2.56}$$

Вероятность того, что ошибок будет не менее, чем t, равна:

$$p_+\{t\} = \sum_{i=t}^{n_A} p\{i\} = \sum_{i=t}^{n_A} \binom{n_A}{i} p_s^i (1-p_s)^{n_A - i}. \tag{2.57}$$

3. При декодировании строится матрица $\mathbf{S}''$, составленная из результатов декодирования внешних кодов. По сути, она будет являться оценкой декодером матрицы $\mathbf{S}$ и будет совпадать с ней по размеру. На j-м шаге нас будут интересовать только первые j её строк $\mathcal{T}_j(\mathbf{S}'')$, строки с номерами от 1 до $j-1$ будут получены на предыдущих шагах, а j-я строка будет получаться исправлением ошибок в $\mathbf{S}'_j$ на текущем шаге:

$$\mathcal{R}_j(\mathbf{S}'') = Dec_B\{\mathbf{S}'_j\}. \tag{2.58}$$

Также будем строить матрицу $\mathbf{W}$*, содержащую в своих строках разности* $\mathbf{S}''_j - \mathbf{S}'_j$*, то есть* $\mathcal{R}_j(\mathbf{W}) = \mathbf{S}''_j - \mathbf{S}'_j$.

При условии, что j-й внешний код обладает достаточной корректирующей способностью, чтобы исправить все ошибки и стирания в $\mathbf{S}'_j$, $\mathcal{R}_j(\mathbf{S}'')$ совпадёт с j-й строкой матрицы $\mathbf{S}$, то есть с $\mathcal{R}_j(\mathbf{S})$.

Так как у нас есть оценка сверху на вероятности ошибки и стирания в отдельных символах слова, мы можем оценить сверху вероятность неправильного декодирования первого внешнего кода. Вероятности ошибки и стирания на входе его декодера оценим следующим образом: на нечётном шаге с номером $j = 2l-1$ положим $p_t\{j\} = p_+\{l\}$, $p_e\{j\} = 0$; на чётном шаге с номером $j = 2l$ положим $p_t\{j\} = p_+\{l+1\}$, $p_e\{j\} = p\{l\}$. Оценки вероятности стирания не являются верхними, так как часть стираний учитываются как ошибки.

Тогда вероятность ошибки второго j-го кода можно оценить сверху величиной

$$p_{Bj} = \sum_{i=0}^{r_j^B} \sum_{m=\lfloor (r_j^B - i)/2 \rfloor + 1}^{n_B - i} \binom{n_B}{i,m,n_B} p_e^i p_t^m (1-p_e-p_t)^{n_B - i - m} + \sum_{i=r_j^B+1}^{n_B} \binom{n_B}{i} p_e^i (1-p_e)^{n_B - i}, \tag{2.59}$$

где под p_t и p_e понимаются $p_t\{j\}$ и $p_e\{j\}$ соответственно.

При условии, что внешний код обладает достаточной корректирующей способностью, чтобы исправить все ошибки в $\mathbf{S}'_j$, и на предыдущих итерациях декодирование было правильным, столбцы матрицы $\mathcal{T}_j(\mathbf{S}'')$ укажут правильные смежные классы для декодирования внутренних кодов.

Если внешний код выдаёт отказ от декодирования, то алгоритм декодирования останавливается, и выдаётся отказ от декодирования всей кодовой конструкции. Если в результате декодирования внешнего кода произойдёт ошибка, то в результате декодирования ОЛО-кода либо будет получено неправильное кодовое слово, либо на одной из следующих итераций может быть выдан отказ от декодирования.

4. С использованием полученной на предыдущем шаге алгоритма матрицы $\mathcal{T}_2(\mathbf{S}'')$, с помощью вторых внутренних кодов можно исправить ошибки в столбцах матрицы $\mathbf{V}$:

$$\mathbf{V}_j = Dec_A[\mathcal{T}_j(\mathbf{S}'')]\{\mathbf{V}\}. \tag{2.60}$$

$$\mathbf{V}_j = Dec_A\{\mathbf{V} - \mathbf{b}\} + \mathbf{b}, \tag{2.61}$$

где $\mathbf{b} = \mathbf{H}_j^{-1} \cdot \mathcal{T}_j(\mathbf{S}'')$, а $\mathbf{H}_j^{-1}$ — матрица, состоящая из первых j столбцов матрицы $\mathbf{H}^{-1}$, обратной к $\mathbf{H}$

Рассмотрим матрицу $\mathcal{T}_j(\mathbf{W})$, содержащую первые строки матрицы $\mathbf{W}$. Столбцы этой матрицы содержат отличия между $\mathcal{T}_j(\mathbf{S}'')$, которая получена декодированием внешних кодов и при условии правильного декодирования совпадает с $\mathcal{T}_j(\mathbf{S})$, и принятой из канала $\mathcal{T}_j(\mathbf{S}')$. По сути, такой столбец является проекцией матрицы ошибок $\mathbf{E}$. Рассмотрим декодер, который по синдрому внутреннего кода возвращает соответсвующий этому синдрому вектор-столбец ошибки. Обозначим такую операцию как $\mathbf{E}_j = Dec_A(\mathcal{T}_j(\mathbf{W}))$, где подразумевается, что на каждый столбец матрицы $\mathcal{T}_j(\mathbf{W})$ декодер возвращает соответствующий столбец ошибки в матрице $\mathbf{E}_j$.
В таком случае декодирование смежных классов в столбцах $\mathbf{V}$ может быть выполнено как поиск $\mathbf{E}_j$ и вычисление $\mathbf{V}_j = \mathbf{V} - \mathbf{E}_j$. В случае отказов от декодирования внутренних кодов $\mathbf{E}_j$ будет содержать столбцы со стираниями. Разность столбца из $\mathbf{V}$ со стёртым столбцом из $\mathbf{E}_j$ будем считать равной стёртому столбцу.

Декодеры внутренних кодов могут выдавать как кодовые слова, так и отказы от декодирования. В случае возникновения отказа от декодирования столбец считается стёртым. В таком случае на следующем шаге декодирования при выполнении умножения в пункте 2 стёртый столбец перейдёт в стёртый символ внешнего кода, и декодер внешнего кода далее будет исправлять как ошибки, так и стирания.

5. После исправления ошибок с помощью последнего, n_A-го внешнего кода, декодирование останавливается. Матрица $\mathbf{V}_{n_A}$ является результатом декодирования.

Декодирование будет правильным, если был правильно отдекодирован каждый внешний код. В таком случае вероятность неправильного декодирования можно оценить следующим

образом:

$$p \leq p_B = \min\{\sum_{j=1}^{n_A} p_{Bj}, 1\}, \tag{2.62}$$

где n_A — число вложенных внутренних кодов ОЛО-кода, p_B — оценка сверху вероятности неправильного декодирования всей конструкции.

2.3.4 Нижняя граница вероятности неправильного декодирования

Для вычисления нижней границы неправильного декодирования требуется построить такие слова ошибок, которые, с одной стороны, будут обладать достаточно большой вероятностью и, с другой, заведомо не смогут быть декодированы.

Ограничимся оценкой вероятности неправильного декодирования первого внешнего кода. Конструкция слова ошибки будет строиться следующим образом: пусть произошла ошибка $\mathbf{E} = \mathbf{V} - \mathbf{C}$, в которой есть не менее, чем $\lfloor (d_1^B - 1)/2 \rfloor + 1$ столбцов веса 1. (В отличие от верхней границы, здесь нас интересуют столбцы веса ровно 1, а не 1 и более.) Любой столбец слова ошибки веса 1 приводит к появлению на входе первого внутреннего кода ошибки в той же позиции, что и номер этого столбца.

Первый внутренний код не может иметь на входе стирания, так как 0-й внутренний код не декодируется (он имеет скорость 1) и, как следствие, не даёт отказов от декодирования. Первый внешний код способен исправлять не более, чем $t_1^B = \lfloor (r_1^B)/2 \rfloor$ ошибок. Это значит, что если столбцов веса 1 будет не менее, чем $t_1^B + 1$, то на входе декодера первого внутреннего кода появится не менее, чем $t_1^B + 1$ ошибок и он не будет декодирован правильно (то есть либо даст отказ от декодирования, либо декодирование будет ошибочно).

Вероятность появления такого слова легко может быть вычислена. Вероятность того, что в столбце произойдёт ровно 1 ошибка, равна:

$$p_t\{1\} = n_A p_s (1 - p_s)^{n_A - 1}. \tag{2.63}$$

Тогда вероятность ошибки j-го внешнего кода можно оценить снизу величиной

$$p'_{B1} = \sum_{i=t_B^{(j)}+1}^{n_B} \binom{n_B}{i} p_t\{1\}^i (1 - p_t\{1\})^{n_B - i}. \tag{2.64}$$

Вероятность p неправильного декодирования кода $\mathcal{C}$ не меньше, чем вероятность неправильного декодирования любого из компонентных кодов. Отсюда следует граница:

$$p \geq p_B = p'_{B1} \tag{2.65}$$

2.3.5 Поиск избыточностей кодов-компонентов, обеспечивающих заданную выходную вероятность ошибки при заданной входной

Пусть нам заданы базовые параметры кода (q, n_A, n_B, $r_j^A = j$), входная вероятность ошибки p_s, требуемая вероятность неправильного декодирования $p_f = p_B$ и требуется найти избыточности кодов-компонентов для того, чтобы обеспечить требуемую вероятность ошибки.

По описанию алгоритма декодирования выше можно построить процедуру поиска избыточностей внешних кодов:

- Для каждого шага мы, зная корректирующие способности внутренних кодов, можем вычислить вероятности появления столбцов $p_t\{j\}$ и $p_e\{j\}$, которые приведут к появлению ошибок и стираний на входе внешнего кода на j-й итерации алгоритма.

- Далее требуется ограничить вероятность ошибки каждого из внешних кодов. Очевидно, что если $\forall j : p_{Bj} \leq p_B/n_A$, то вероятность ошибки кодовой конструкции не превысит требуемой p_B.

  Поэтому следует, с использованием формулы 2.59, произвести поиск таких r_j^B, которые обеспечат выполнение условия $p_{Bj} \leq p_B/n_A$.

- На данном этапе нам известны параметры кода, которые обеспечивают требуемые характеристики выходной вероятности ошибки при заданной входной, но могут быть неоптимальны. В данном случае требуется провести оптимизацию параметров кода. Это можно сделать следующим образом:

  - Для набора r_j^B, полученного на предыдущем шаге, построить все альтернативные наборы $\{\hat{r}_j^B\}_k$ следующим образом:
    $\hat{r}_{jk}^B = r_j^B, j \neq k$;
    $\hat{r}_{jk}^B = r_j^B - 1, j = k$
    для $k = \overline{1,n_A}$
  - Для каждого из этих наборов вычислить вероятность ошибки в соответствии с алгоритмом, описанным в разделе 2.3.3.
  - Выбрать тот набор, вероятность ошибки которого наименьшая.
  - Если полученная вероятность меньше требуемой p_f, то повторить эту же операцию. Если больше, то выполнить аналогичную, но не с уменьшением, а с увеличением корректирующей способности кодов.
  - Алгоритм останавливается, если после двух шагов набор r_j^B не изменился.

2.3.6 Примеры построения кодов

Построим, применив алгоритм из раздела 2.3.5, коды со следующими параметрами: поле $GF(256)$, длина внутренних кодов $n_A = 8$, длина внешних $n_B = 256$. Полная длина кода получается $n = 16384$ бита. Зададим уровень входной вероятности ошибки, равный $p_s = 10^{-2}$ и построим конструкции для трёх требуемых выходных вероятностей ошибки $p_f \in \{10^{-12}, 10^{-15}, 10^{-18}\}$.

Скорости полученных кодов равны $R \in \{0.884, 0.871, 0.857\}$, оценка на расстояние $d_{min} \geq d \in \{40, 40, 56\}$. Графики верхних границ на вероятности ошибки для этих кодов приведены на рис. 2.9.

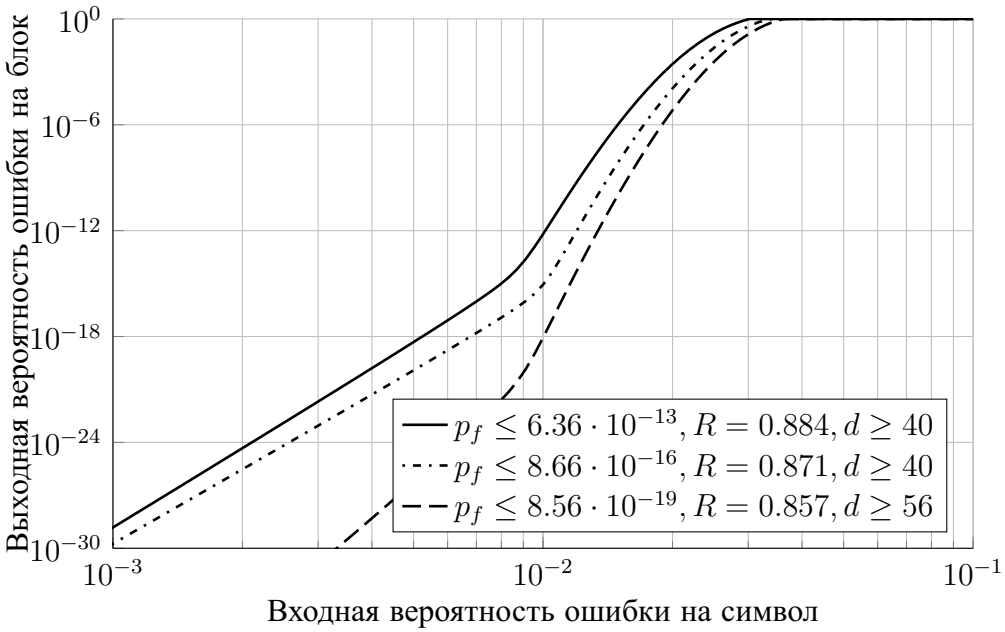


Рисунок 2.9. Вероятности ошибки отдельных внешних кодов в зависимости от входной вероятности ошибки для кода, построенного с целевыми $p_s = 10^{-2}$, $p_f \in \{10^{-12}, 10^{-15}, 10^{-18}\}$; параметры конструкции: $q = 256$, $n_A = 8$, $n_B = 256$, $n = 2048$

По серии графиков на рис. 2.9 легко видеть, что кривая вероятности ошибки изменяет наклон в области целевых параметров входной и выходной вероятностей ошибки.

На рис. 2.10 кривые соответствуют вероятностям ошибочного декодирования внешних кодов. Из них легко видеть, что вероятность ошибочного декодирования конструкции в целом, как правило, обусловлена вероятностью ошибки одного или двух внешних кодов, которые имеют наибольшие вероятности ошибки. Следует отметить, что при входных вероятностях ошибки больше целевых такими кодами являются первые два. Интересно, что в этой области вероятности неправильного декодирования идут парами, в которые входят вероятности на шаге, на котором обнаруживаются ошибки заданного веса и на следующем шаге, на котором они исправляются как стирания.

По поведению серии кривых наиболее хорошо видно, что при заданной выходной вероятности ошибки предложенный алгоритм проводит оптимизацию параметров внешних кодов таким образом, чтобы каждый из них достигал требуемую вероятность неправильного декодирова-

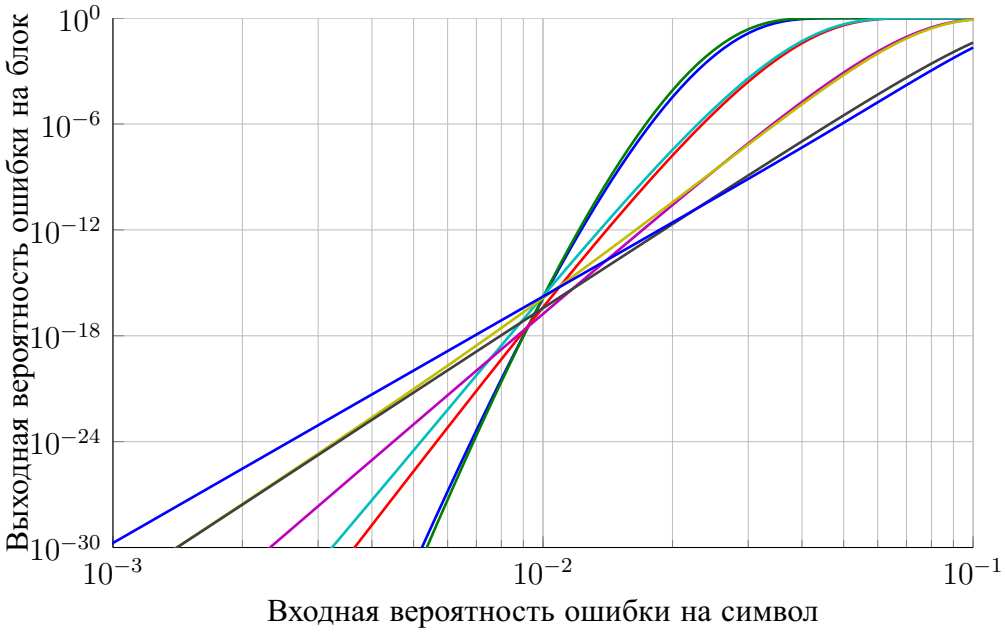


Рисунок 2.10. Вероятности ошибки отдельных внешних кодов в зависимости от входной вероятности ошибки для кода, построенного с целевыми $p_s = 10^{-2}$, $p_f = 10^{-15}$, итоговые $R = 0.871$, $p_f \leq 8.66 \cdot 10^{-16}$, $d_{min} \geq 40$; параметры конструкции: $q = 256$, $n_A = 8$, $n_B = 256$, $n = 2048$

ния при минимальной его избыточности. Благодаря этому в районе целевой вероятности ошибки видно пересечение нескольких кривых, остальные кривые проходят ниже. При уменьшении входной вероятности ошибки (движении справа налево по оси абсцисс графика) до p_s вероятность неправильного декодирования, как правило, мажорируется вероятностью неправильного декодирования первых внешних кодов, а после неё — одного из промежуточных. Как правило, это тот код, у которого наихудшая скорость убывания вероятности неправильного декодирования. В самой точке наблюдается изменение наклона кривой. Следует отметить, что после целевой точки вероятность неправильного декодирования всё равно достаточно быстро падает при уменьшении входной вероятности ошибки.

Предложенный алгоритм даёт код с наилучшей скоростью при заданных входной и выходной вероятностях. Коды, построенные для других входных и выходных вероятностей и обеспечивающие ту же p_f при той же p_s, дадут худшую скорость.

Это легко показать, построив несколько кодов, оптимизированных под различные параметры, но дающие одинаковые вероятности ошибки в выбранной точке. Пример такого построения показан на рис. 2.11. В данном случае ставилась задача построить коды, которые будут обладать одинаковыми вероятностями неправильного декодирования $p_f = 10^{-15}$ при входной вероятности ошибки $p_s = 10^{-2}$, но оптимизировать их под различные целевые $p_f \in \{10^{-10}, 10^{-15}, 10^{-20}\}$, подбирая параметр p_s. Получившиеся параметры кодов равны $p_s = 10^{-2}$, $p_f = 8.66 \cdot 10^{-16}$, $R = 0.871$, $d_{min} \geq 40$ для оптимального, а также $p_s = 7.90 \cdot 10^{-3}$, $p_f = 8.60 \cdot 10^{-21}$, $R = 0.865$, $d_{min} \geq 56$ и $p_s = 1.57 \cdot 10^{-2}$, $p_f = 8.28 \cdot 10^{-11}$, $R = 0.859$, $d_{min} \geq 40$. Очевидно, что код, построенный сразу для целевых параметров, имеет наибольшую скорость. Дополнительная

избыточность внешних кодов, если ОЛО-код построен для других параметров, используется по-разному. Если требуется более эффективная работа в области больших входных вероятностей ошибки, то она идёт на увеличение избыточности первых внешних кодов. В противном случае, если требуется достижение более низких вероятностей неправильного декодирования, то эта избыточность используется для улучшения промежуточных кодов.

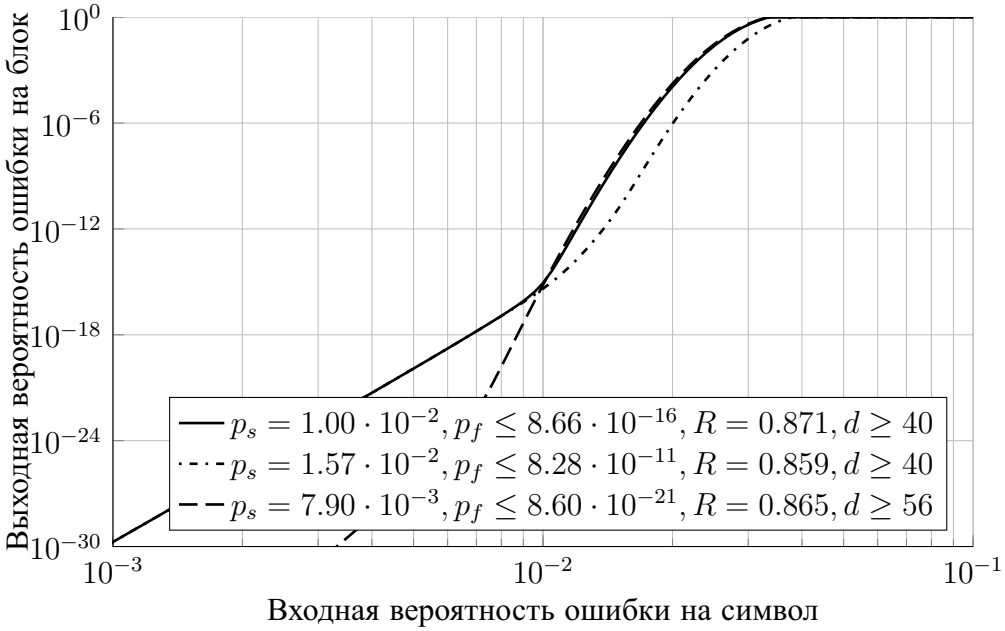


Рисунок 2.11. Вероятности ошибки кодов с одинаковой вероятностью неправильного декодирования $p_f \leq 10^{-15}$ при входной вероятности ошибки $p_s = 10^{-2}$, построенных с разными целевыми p_s и p_f; параметры конструкции: $q = 256$, $n_A = 8$, $n_B = 256$, $n = 2048$

Тот факт, что для наилучшей работы кодовой конструкции требуется наименьшая длина внутренних кодов и наибольшая длина внешних, можно проиллюстрировать на примере. Нами были построены три кода с одинаковой полной длиной $n_A \times n_B = 2048$ 256-ичных символа с различными соотношениями длин внутренних и внешних кодов, $n_A \times n_B \in \{8 \times 256, 16 \times 128, 32 \times 64\}$. Целевой выходной вероятностью ошибки была задана $p_f = 10^{-15}$, входные вероятности ошибки $p_s \in \{6.0 \cdot 10^{-3}, 4.5 \cdot 10^{-3}, 3.7 \cdot 10^{-3}\}$ были подобраны таким образом, чтобы скорость построенных кодов была равна $R = 0.9$. Результат представлен на рис. 2.12. Из этой серии графиков видно, что наилучшими характеристиками обладает именно конструкция с параметрами $n_A \times n_B = 8 \times 256$.

В то же время, если построить набор кодов с одинаковыми параметрами входной и выходной вероятностей ошибки, но разными длинами кодов-компонентов, то их скорости будут существенно отличаться, а поведение будет практически идентично. Серия таких кривых показана на рис. 2.13.

Пример построения нижней границы вероятности неправильного декодирования показан на рис. 2.14, где она обозначена штриховой линией. Сплошной линией для сравнения нанесена верхняя граница. Видно, что при вероятностях выше точки с целевыми параметрами нижняя граница достаточно близка к верхней, т. к. в данном случае неправильное декодирование происходит, как правило, именно из-за неправильного декодирования первого внешнего кода.

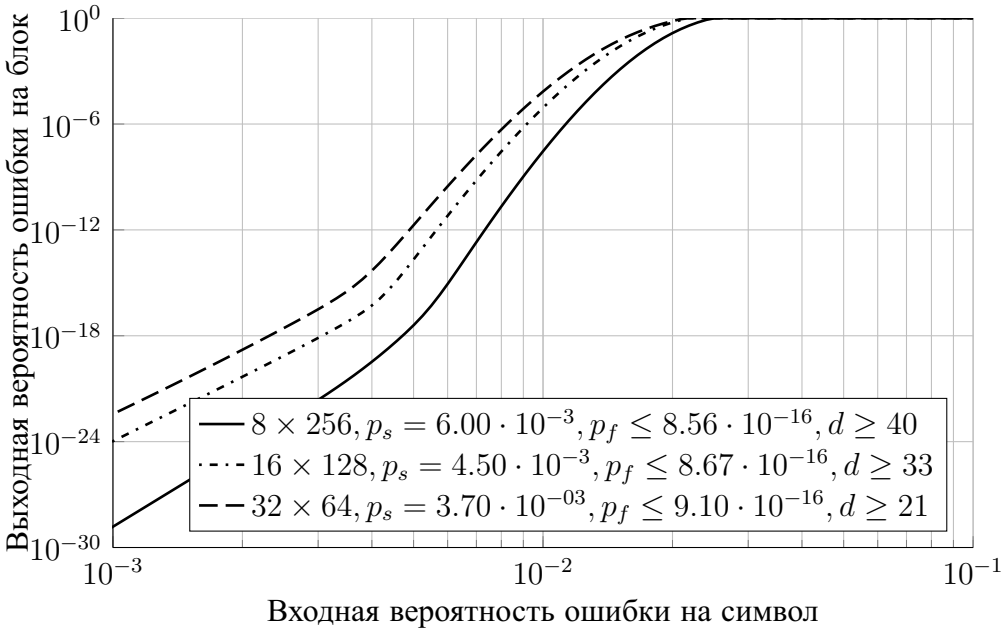


Рисунок 2.12. Вероятности ошибки кодов с одинаковой полной длиной и различной длиной внутренних кодов; целевые выходные вероятности ошибки $p_f = 10^{-15}$, целевые входные вероятности ошибки подбирались таким образом, чтобы получить равные скорости кодов $R = 0.9$; параметры конструкции: $q = 256$, $n = 2048$

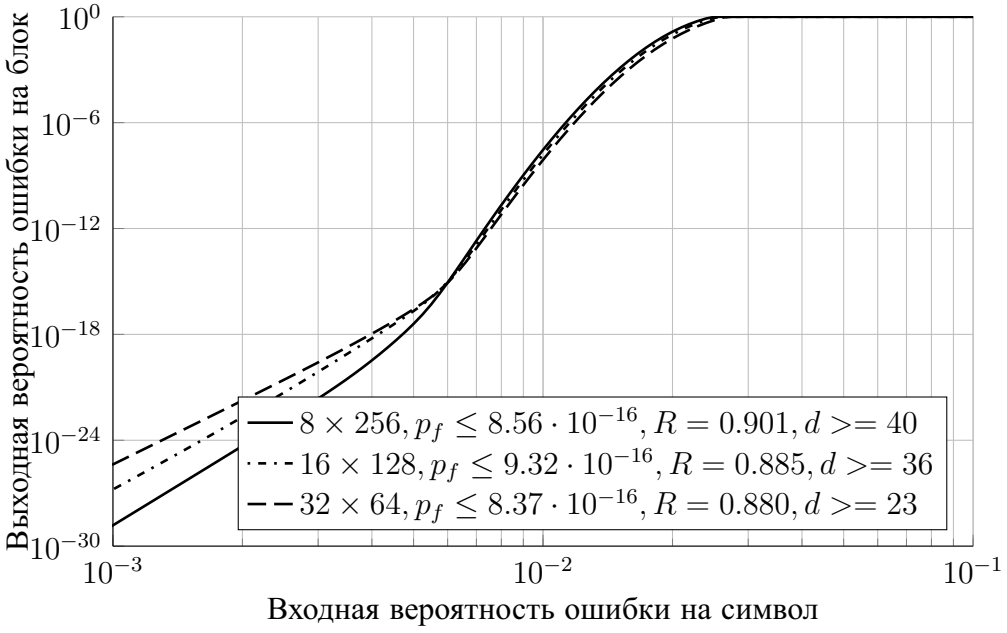


Рисунок 2.13. Вероятности ошибки кодов с одинаковой полной длиной и различной длиной внутренних кодов, построенных с целевыми входной вероятностью ошибки $p_s = 6 \cdot 10^{-3}$ и выходной $p_f = 10^{-15}$; параметры конструкции: $q = 256$, $n = 2048$

На том же графике показана вероятность появления ошибки веса более $\lfloor (d-1)/2 \rfloor$ (самая левая кривая), при декодировании до половины кодового расстояния она являлась бы кривой вероятности неправильного декодирования соответствующего декодера. Видно, что она далеко отстоит от верхней границы для алгоритма декодирования, описанного выше. Это говорит о

том, что этот декодер исправляет большое количество ошибок за пределами половины кодового расстояния.

Там же приведена вероятность появления ошибки веса более $\lfloor (d_{GV}-1)/2 \rfloor$ (самая правая кривая), где d_{GV} — кодовое расстояние кода с той же скоростью, лежащего на границе Варшамова-Гилберта. Видно, что границы вероятности ОЛО-кода сравнимы с вероятностью для кода на границе Варшамова-Гилберта.

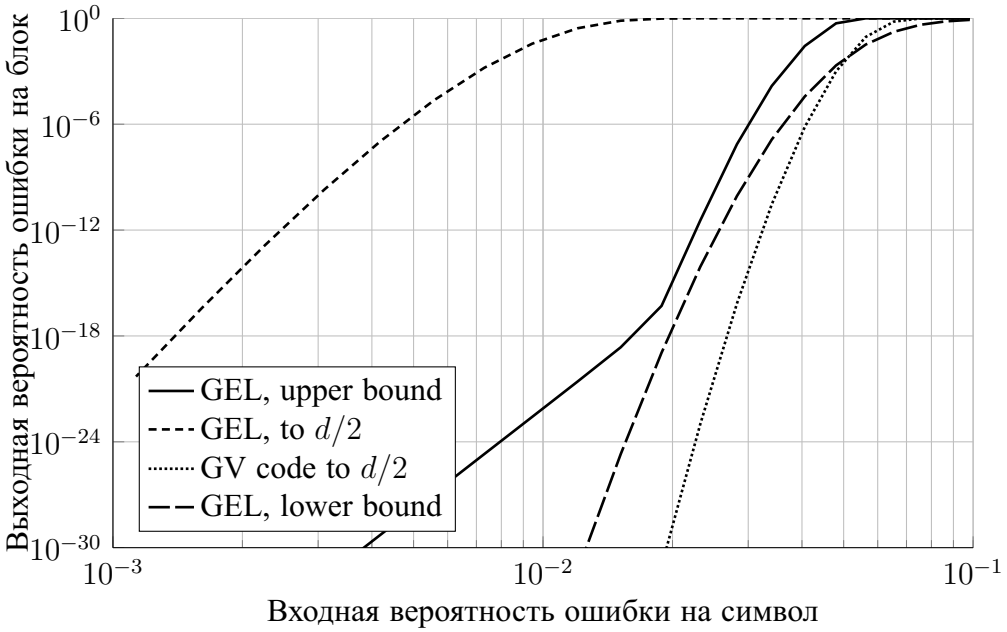


Рисунок 2.14. Верхняя и нижняя границы вероятности неправильного декодирования входной вероятности ошибки для кода, построенного с целевыми $p_s = 10^{-2}$, $p_f = 10^{-15}$, а также вероятности появления ошибки веса более $\lfloor (d-1)/2 \rfloor$ (самая левая кривая) и веса более $\lfloor (d_{GV}-1)/2 \rfloor$ (самая правая кривая), где d_{GV} — кодовое расстояние кода с той же скоростью, лежащего на границе Варшамова-Гилберта; полученная вероятность неправильного декодирования при входной вероятности ошибки $p_s = 2 \cdot 10^{-2}$ ограничена $3 \cdot 10^{-18} \leq p_f \leq 8.42 \cdot 10^{-16}$, $R = 0.805$, $d \geq 56$; параметры конструкции: $q = 256$, $n_A = 8$, $n_B = 256$, $n = 2048$

2.4 Обобщение границ на ОЛО-коды с другими внешними кодами

2.4.1 ОЛО-коды с различными внутренними и внешними кодами

Описанные выше методы расчёта верхней и нижней границ вероятности неправильного декодирования и алгоритм выбора избыточностей внешних кодов можно применить и для других конструкций.

Сначала опишем такую конструкцию, используя обозначения, принятые в [6].

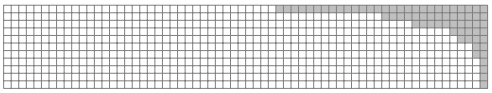
2.4.2 Описание конструкции

Кодовым словом q-ичного ОЛО-кода назовём матрицу $\mathbf{C}$ над полем $GF(q)$ размеров $n_A \times n_B$, где n_A — длина внутренних кодов, а n_B — длина внешних кодов.

Обозначим $\mathbf{H}$ проверочную матрицу системы вложенных внутренних кодов. Её элементы принадлежат полю $GF(q)$, её размер $n_A \times n_A$. Любые первые $r_j = \sum_{i=1}^{j} m_i$ строк этой матрицы, составляющие матрицу размера $r_j \times n_A$, будут являться матрицей внутреннего кода длины n_A с $k_j = n_A - r_j$ информационных символов.

Пусть

$$\mathbf{S} = \mathbf{H} \cdot \mathbf{C} =$$



(2.66)

— матрица синдромов внутренних кодов. Матрица $\mathbf{C}$ будет являться кодовым словом ОЛО-кода тогда и только тогда, когда строки матрицы $\mathbf{S}$ будут являться кодовыми словами внешних кодов. Рассмотрим последнее требование более подробно.

Строки матрицы $\mathbf{S}$ сгруппированы в слои, каждый из которых будет соответствует одному внешнему коду. Матрица $\mathbf{S}$ состоит их L слоёв, в каждом слое m_i строк, $i = \overline{1, L}$. (Очевидно, что $\sum_{i=1}^{L} m_i = n_A$.) Такой код мы будем называть кодом L-го порядка.

Тогда q-ичная подматрица $\mathbf{S}_i$, являющаяся кодовым словом внешнего кода, может рассматриваться с разных точек зрения:

1. Столбцы $\mathbf{S}_i$ могут быть рассмотрены как символы над полем $GF(q^{m_i})$. В таком случае в качестве внешнего кода может быть использован q^{m_i}-ичный код длины n_B.

2. Символам $\mathbf{S}_i$ могут быть поставлены в соответствие символы кода над полем $GF(q)$ длины $m_i n_B$. Примерами таких соответствий может быть запись символов кодового слова внешнего кода в $\mathbf{S}_i$ по строкам, или по столбцам, или в случайном порядке.

Единственным ограничением на отображение подматрицы на кодовое слово внешнего кода состоит в том, что требуется расположение проверочных символов внешнего кода в правых столбцах $\mathbf{S}_i$, при этом столбец $\mathbf{S}_i$ является либо целиком проверочным, либо целиком информационным. Это ограничение требуется для того, чтобы кодирование можно было производить так же, как и в разделе 2.3.2.

В частности, внешние коды разных слоёв могут принадлежать различным классам кодов.

Легко видеть, что коды из раздела 2.2 могут быть получены из описания выше, если положить $m_i = 2, i = \overline{1, n_A/2}$ и выбран первый вариант из альтернативы выше, а коды из раздела 2.3 — если положить $m_i = 1, i = \overline{1, n_A}$.

2.4.3 ОЛО-коды с расширенными кодами БЧХ в качестве внутренних

Можно заметить, что алгоритм расчёта из раздела 2.3.3 требует знания минимума характеристик о свойствах внутреннего кода: количество гарантированно обнаруживаемых и количество гарантированно исправляемых ошибок. В случае кодов Рида-Соломона эти характеристики определяются кодовым расстоянием: коды с расстоянием $2t+1$ используются для исправления t ошибок, а коды с расстоянием $2t+2$ — для исправления t и обнаружения $t+1$ ошибки.

Этих свойств достаточно для написания верхних границ вероятности неправильного декодирования внешних кодов: знание о корректирующих свойствах внутреннего кода используется для того, чтобы приписать символам внешних кодов вероятности ошибок и стираний. При этом сначала делается оценка снизу вероятности стирания, а все остальные ошибки и стирания учитываются как ошибки.

Кроме этого для написания границы нужно иметь возможность оценить сверху вероятность неправильного декодирования внешних кодов при известных вероятностях ошибки и стирания на их входах. Для некоторых кодов, в частности, для кодов Рида-Соломона при их декодировании до половины кодового расстояния, это может быть сделано комбинаторными методами, применёнными в разделах 2.2 и 2.3.

Этот метод расчёта можно применить, например, для конструкций с кодами Боуза-Чоудхури-Хоквингема в качестве внутренних кодов. Такая конструкция, например, рассматривалась Fahrner и др. в [40]. Авторы работы рассматривали конструкцию с кодами БЧХ в качестве внутренних кодов и кодами РС в качестве внешних, коды строились над полем GF(64). При этом для выбора скоростей внешних кодов использовалось моделирование внутренних кодов.

Рассмотрим другую конструкцию, основанную на кодах БЧХ. Как видно из границы из раздела 2.3.3 и результатов расчёта требуемых скоростей внешних кодов, наибольшая избыточность требуется от первого внешнего кода, так как у него на входе нет стираний, только ошибки.

Это наблюдение даёт возможность предложить простой метод оптимизации конструкции. Будем рассматривать систему вложенных расширенных кодов БЧХ над полем GF(Q). В таком случае первым кодом этой системы будет проверка на чётность, обнаруживающая любые ошибки веса 1, вторым — код с расстоянием 4, исправляющий ошибки веса 1 и обнаруживающий ошибки веса 2, и так далее. Внешние коды будут кодами Рида-Соломона над полем GF(Q) за исключением первого кода, который относится к самой проверке на чётность. Будем рассматривать ситуацию, когда этот код является кодом скорости $R=0$, то есть, по сути, отсутствует: проверка на чётность используется безусловно, без внешнего кода.

Иными словами, в этой конструкции $q=2$, $m_1=1$, $m_i=\log_2(Q), i=\overline{2,L}$.

В таком случае мы имеем систему вложенных внутренних кодов, в которой первый внутренний код имеет расстояние $d_{min}=2$, а последующие идут с шагом 2.

Преимущество этой конструкции в том, что проверка на чётность позволяет заменить преобладающие в принятом слове столбцовые ошибки веса 1 на стирания, что заметно снижает требуемую избыточность второго внешнего кода. В то же время объём добавленной избыточно-

сти сравнительно невелик, так как защищающий проверку на чётность внешний код скорости 0 имеет длину в битах в $\log_2(Q)$ раз меньше, чем внешние коды Рида-Соломона.

2.4.4 Построение ОЛО-кода для ВОЛС

Построим на основе этой конструкции код, который может быть применён в волоконно-оптической линии связи.

Будем сравнивать предлагаемый код с кодом БЧХ $(9200,8192)$, предложенным в [41], применяемом с четырёхкратным интерливингом. При входной вероятности ошибки 0.0037 он имеет вероятность ошибки на блок, равную $4.26 \cdot 10^{-9}$, и вероятность ошибки на бит, равную $3.38 \cdot 10^{-11}$.

Поставим цель получить код, обеспечивающий при тех же или лучших параметрах длины и числа информационных символов лучшую надёжность передачи (меньшую вероятность неправильного декодирования в том же канале) и при этом обеспечивающий возможность распараллеливания и конвейеризации декодирования.

Для этого рассмотрим конструкцию выше, где возьмём $Q = 256$, $n_B = 252$, $n_A = 73$. Если задаться целевой вероятностью неправильного декодирования на блок $p_f \leq 10^{-13}$ при входной вероятности ошибки на символ $p_s = 3.7 \cdot 10^{-3}$, то можно получить ОЛО-код $(18396, 16408)$ со скоростью передачи $R = 0.8919$. Этот код менее, чем в два раза длиннее используемого в ВОЛС кода БЧХ $(9200,8192)$, но содержит большее число информационных символов, чем два слова этого кода БЧХ. При декодировании при входной вероятности ошибки на бит 0.0037 вероятность ошибки на блок не более $7 \cdot 10^{-14}$, а вероятность ошибки на бит оценивается как $5 \cdot 10^{-16}$. Вероятностные характеристики декодирования представлены на рис. 2.15.

Таким образом, эта конструкция превосходит применяемый в волоконно-оптических линиях связи код БЧХ.

Отметим, что все коды и внутренние и внешние декодируется над $GF(2^8)$. Таблицы над этим полем в 64 раза меньше, чем таблицы при декодировании кода BCH (9200,8192).

Декодер такого каскадного кода может быть выполнен в виде конвейера из 2x9 ступеней, у которого каждая нечётная ступень представляет собой параллельное декодирование 252 внутренних кодов, а четная ступень — декодирование соответствующего внешнего кода Рида-Соломона. Блок-схема i-й пары кодов представлена на рис. 2.16.

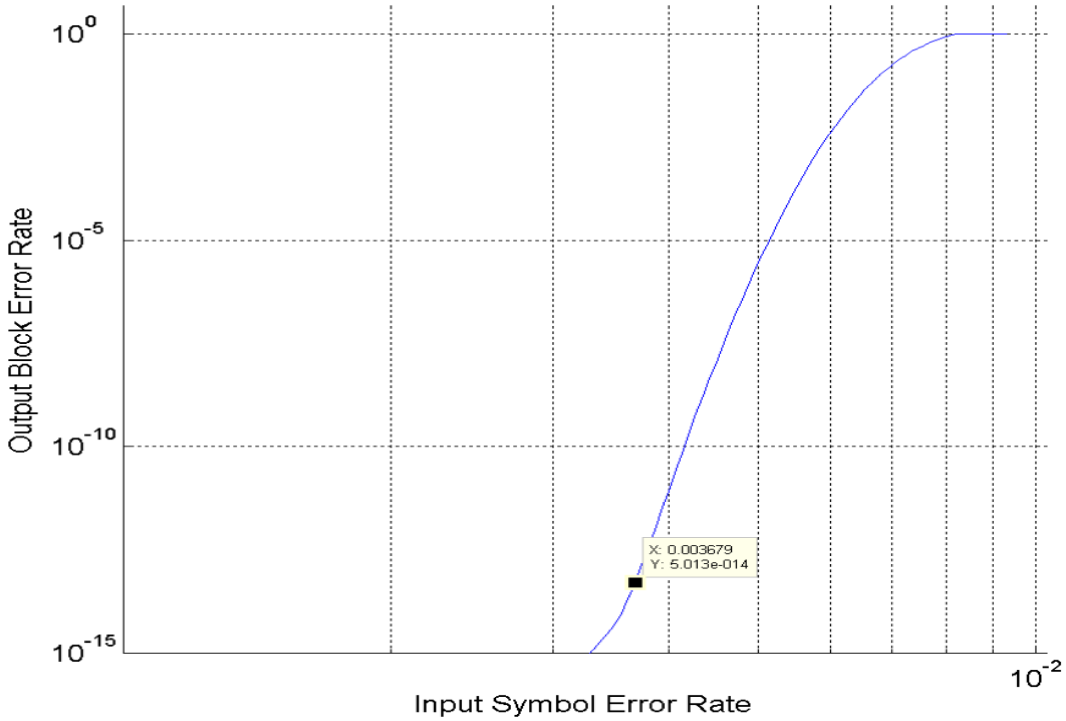


Рисунок 2.15. Оценка сверху на вероятность ошибки на блок предложенного ОЛО-кода с внутренними кодами БЧХ

2.5 Выводы к главе

– В главе рассмотрены конструкции обобщённых кодов с локализацией ошибок, имеющих компонентные коды с квадратичным расширением алфавита и с компонентными кодами над одним алфавитом. Приведено описание верхних границ вероятности неправильного декодирования предложенных конструкций. Для этих конструкций:

  – Предложены методы выбора структуры ОЛО-кода и оптимизации скоростей внешних кодов, обеспечивающие наибольшую возможною скорость ОЛО-кода при условии, что вероятность неправильного декодирования на кодовое слово не превышает заданную при заданной входной вероятности ошибки на символ.

  – Предложены нижние границы на вероятность неправильного декодирования ОЛО-кодов.

– Дано описание конструкций в общем виде, позволяющее расширить понятие ОЛО-кодов на различные компонентные коды.

– Предложено семейство ОЛО-кодов с расширенными кодами БЧХ в качестве внутренних и кодами Рида-Соломона в качестве внешних. Предложена частная конструкция из этого семейства, которая обеспечивает энергетический выигрыш больше, чем применяемый в существующих ВОЛС код БЧХ, и при этом обладает меньшей сложностью реализации.

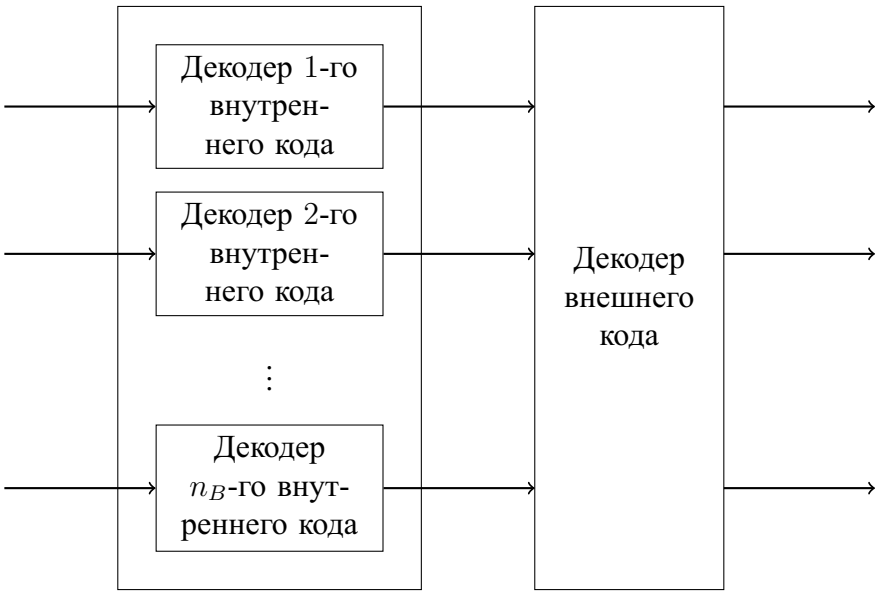


Рисунок 2.16. Блок-схема, показывающая метод распараллеливания и конвейеризации декодирования предложенного кода

– Стоит отметить, что важным достоинством рассмотренных конструкций является также и то, что они основаны на широко используемых кодах Рида-Соломона или Боуза-Чоудхури-Хоквингема, которые реализованы в составе программных библиотек для цифровых сигнальных процессоров или ядер ИС (IP core) для программируемых логических интегральных схем.

  Применение кодеров и декодеров внешних кодов с динамически настраиваемой избыточностью позволит динамически изменять параметры конструкции в зависимости от качества канала, в котором на данный момент работает система передачи данных.

Глава 3

Проблемы мягкого декодирования ОЛО-кодов

3.1 Введение

Известно, что энергетический выигрыш при мягком декодировании существенно превосходит выигрыш жесткого декодера, и разница может достигать 3 дБ. Таким образом, коды, которые возможно декодировать жесткими и мягкими алгоритмами декодирования одновременно, могут рассматриваться как хорошие кандидаты для перспективных систем передачи информации.

Так как энергия в каналах передачи информации ограничена, то декодирование перспективных сигнально-кодовых конструкций должно приводить к большому энергетическому выигрышу (порядка 11 дБ). Требования современных систем связи на высокую спектральную эффективность, выражаемую в бит/с на герц занимаемой полосы частот, приводят к использованию модуляций высокого порядка и кодов с высокой скоростью. Поэтому такая сигнально-кодовая конструкция должна быть основана на согласованном с модуляцией недвоичном коде с малой избыточностью, алгоритмы кодирования и декодирования которого должны иметь малую сложность, а также должны иметь возможность быть распараллелены. Этим требованиям удовлетворяют коды с малой плотностью проверок и обобщенные коды с локализацией ошибок.

В данной главе рассматриваются ОЛО-коды, допускающие мягкое декодирование. В разделе 3.2 предложена конструкция ОЛО-кодов с короткими двоичными кодами в качестве внутренних и МПП-кодами в качестве внешних. В разделе 3.3 предложен алгоритм мягкого декодирования ОЛО-кодов, описанных в разделе 2.2, и для него построена верхняя граница на вероятность неправильного декодирования и предложен алгоритм выбора избыточностей внешних кодов.

3.2 ОЛО-коды, построенные на основе МПП-кодов

В данной главе рассматриваются обощённые коды с локализацией ошибок, построенные на основе МПП-кодов. В них в качестве внутренних кодов выбраны двоичные коды длины 3, а в качестве внешних — МПП-коды Галлагера.

Для этих конструкций выведены нижние границы на кодовое расстояние.

Ниже предложен алгоритм мягкого декодирования этой кодовой конструкции, который основан на алгоритме декодирования ОЛО-кодов.

Особенностью этих ОЛО-кодов является то, что они являются кодами с малой плотностью проверок. Это позволяет произвести сравнение различных методов мягкого декодирования: стандартного алгоритма “распространения доверия”, предложенного алгоритма декодирования и их комбинации, которая описана ниже.

Главная цель, которая ставится в этой главе — сравнение эффективности декодирования различных декодеров для фиксированного кода. Оптимизации конструкции для максимизации корректирующей способности и энергетического выигрыша не производилось. Такая оптимизация является отдельной нетривиальной задачей, связанной с выбором оптимальных параметров кодов для сравниваемых алгоритмов, решение которой может отличаться для различных декодеров. В то же время задачей исследования являлось сравнение алгоритмов декодирования в равных условиях.

3.2.1 Описание кодовой конструкции

Описание как ОЛО-кода

Зададим кодовое слово этой конструкции как двоичную матрицу $\mathbf{C}$, которая имеет $n_A = 3$ строки и $n_B = n/3$ столбцов, где n — длина всей конструкции. Назовём её конструкцией B.

Каждое слово кода должно удовлетворять следующему равенству:

$$\mathbf{H}_0 \cdot \mathbf{C} = \mathbf{0}, \tag{3.1}$$

где

$$\mathbf{S} = \mathbf{H}_0 \cdot \mathbf{C} = \begin{array}{|c|c|c|}\hline & S^{(1)} & \\ \hline & S^{(2)} & \\ \hline & S^{(3)} & \\ \hline \end{array} \tag{3.2}$$

— матрица синдромов внутренних кодов, которая также является матрицей, содержащей символы слов внешних кодов. Серая часть матрицы соответствует расположению проверочных символов.

Система вложенных внутренних кодов задаётся следующей проверочной матрицей:

$$\mathbf{H}_0 = \left[\begin{array}{ccc} 1 & 0 & 1 \\ \hdashline 0 & 1 & 1 \\ \hline 0 & 0 & 1 \end{array}\right]. \tag{3.3}$$

Система состоит из двух кодов: кода $(3,1,3)$ и кода $(3,0,\infty)$.

Каждый столбец матрицы $\mathbf{S}$ — это синдром внутреннего кода, вычисленный с использованием соответствующей строки $\mathbf{C}$ как входа декодера. Нижняя строка этой матрицы — это кодовое слово второго внешнего кода; очевидно, что он имеет длину $n/3$. Две верхних строки этой матрицы ($\mathbf{C}^{(12)}$) могут быть заданы тремя различными способами, дающими различные конструкции:

1. Первый случай — рассмотрение $\mathbf{S}^{(1)}$ и $\mathbf{S}^{(2)}$ как кодовых слов двух независимых кодов длины $n/3$. Из-за симметричной структуры двух верхних строк матрицы $\mathbf{H}_0$ характеристики этих кодов должны совпадать.

2. Второй случай — использование одного двоичного кода длины $\frac{2}{3}n$. В данной работе рассматривалась запись такого кодового слова по столбцам, но возможно любое соответствие между символами кода и синдрома.

3. Третий случай — рассмотрение столбцов подматрицы $\mathbf{S}^{(12)}$ как символов над полем $GF(4)$ и использование кода длины $n/3$ над $GF(4)$.

Назовём эти конструкции B1, B2 и B3.

В данной диссертации рассматриваются коды с двоичными кодами Галлагера [2] в качестве внешних кодов, а именно случай 2 из списка выше.

Можно отметить, что из-за случайной структуры проверочных матриц кода Галлагера, которая включает случайные перестановки столбцов, соответствие между символами внешнего кода и символами синдрома в конструкции B2 не имеет значения.

Проверочная матрица кода как целого

Опишем, как построить проверочную матрицу кода как целого. Объясним, как она строится для конструкций B1 и B2.

Проверочная матрица может быть записана следующим образом:

$$\mathbf{H} = \begin{pmatrix} \mathbf{H}_0^{(12)} \otimes \mathbf{H}_E^{(12)} \\ \mathbf{H}_0^{(3)} \otimes \mathbf{H}_E^{(3)} \end{pmatrix}, \tag{3.4}$$

где $\mathbf{H}_0^{(12)}$ — две верхних строки $\mathbf{H}_0$, $\mathbf{H}_0^{(3)}$ — нижняя строка $\mathbf{H}_0$, $\mathbf{H}_E^{(12)}$ — проверочная матрица первого внешнего кода (или двух кодов), $\mathbf{H}_E^{(3)}$ — проверочная матрица второго внешнего кода.

Произведение $\mathbf{H}_0^{(12)} \cdot \mathbf{H}_E^{(12)}$ — это специальное тензорное произведение: оно заменяет каждую пару элементов строки $\mathbf{H}_E^{(12)}$ на произведение этих элементов на $\mathbf{H}_0^{(12)}$, выполняя замену подматриц размера 2-на-1 на подматрицы 3-на-1.

Произведение $\mathbf{H}_0^{(3)} \cdot \mathbf{H}_E^{(3)}$ — это обыкновенное произведение Кронекера, где каждый элемент $\mathbf{H}_E^{(3)}$ заменяется матрицей $\mathbf{H}_0^{(3)}$, домноженной на этот элемент.

Получившаяся матрица будет разреженной и может быть декодирована алгоритмом “распространения доверия”.

3.2.2 Теоретические границы на кодовое расстояние

Получим границы на кодовое расстояние рассматриваемых конструкций.

Введём некоторые обозначения:

$\delta_q(R)$ — относительное кодовое расстояние q-ичного кода со скоростью R, лежащего на границе Варшамова-Гилберта.

R_i — скорость внешнего кода i-го слоя.

Так как два нижних кода конструкции B1 должны иметь одинаковые скорости, а каждая из конструкций B2 и B3 имеют только один код в своих нижних строках, обозначим все эти скорости одной величиной R_3.

Четыре конструкции этих кодов имеют следующие нижние границы на кодовое расстояние.

Для конструкции B1:

$$\begin{aligned}\delta = \min\big\{&\delta_2(R_1), \frac{1}{3}\delta_2(R_3),\\ &\frac{2}{3}\delta_2(R_3) - \frac{1}{3}\delta_2(R_1)\big\}.\end{aligned} \tag{3.5}$$

Для конструкции B2:

$$\begin{aligned}\delta = \min\big\{&\delta_2(R_1), \frac{2}{3}\delta_2(R_3),\\ &\frac{2}{3}\delta_2(R_3) - \frac{1}{3}\delta_2(R_1)\big\}.\end{aligned} \tag{3.6}$$

И для конструкции B3:

$$\begin{aligned}\delta = \min\big\{&\delta_2(R_1), \frac{1}{3}\delta_4(R_3),\\ &\frac{2}{3}\delta_4(R_3) - \frac{1}{3}\delta_2(R_1)\big\}.\end{aligned} \tag{3.7}$$

Идея доказательства границ следующая. Рассматривается каждая конфигурация ошибок в отдельном столбце $\mathbf{C}$ (для отдельного внутреннего кода) и соответствующую конфигурацию ошибок в столбце $\mathbf{S}$. Эти конфигурации, складываясь, дают суммарный вес кодовых слов внешнего кода. Существует несколько вариантов таких конфигураций; они соответствуют отдельным компонентам, входящим в минимум в формуле ниже. Некоторые из этих конфигураций могут

быть опущены так как они мажорируются другими при любых параметрах конструкции. Поэтому рассматриваются наихудшие параметры.

Очевидно, что для конструкций B $R_1 = (3R - R_3)/2$. При построении границ для каждой скорости конструкции R были подобраны оптимальные скорости компонентных кодов, которые максимизируют кодовое расстояние конструкции.

Также возможно использовать границы на кодовое расстояние МПП-кодов. В данной диссертации для вычисления нижних границ на кодовые расстояния компонентных МПП-кодов используются методы, разработанные в [2,42]. Известно, что эти коды достигают границу Варшамова-Гилберта при росте веса столбца. С другой стороны, для того, чтобы эти коды могли быть эффективно декодированы алгоритмом “распространения доверия”, требуется чтобы этот вес был по возможности мал.

Эти границы были построены с использованием численных методов. Границы для конструкций B2, B3 с внешними кодами, лежащими на границе Варшамова-Гилберта показаны на рис. 3.1. Результаты для конструкции B2 с внешними МПП-кодами Галлагера с весом столбца проверочной матрицы, равным 4, 7 или 10, показаны на рис. 3.2. Для сравнения на последний график нанесена кривая “VG”, соответствующая той же конструкции с внешними кодами на границе Варшамова-Гилберта.

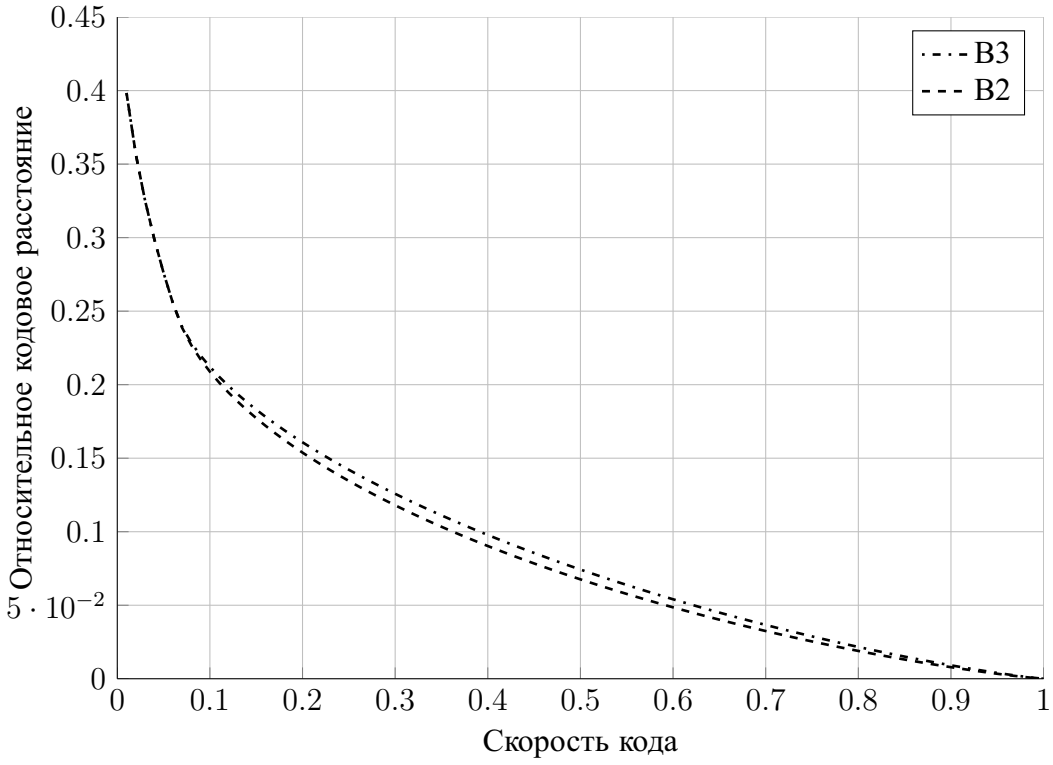


Рисунок 3.1. Нижние границы на кодовые расстояния при условии, что компонентные коды лежат на границе Варшамова-Гилберта

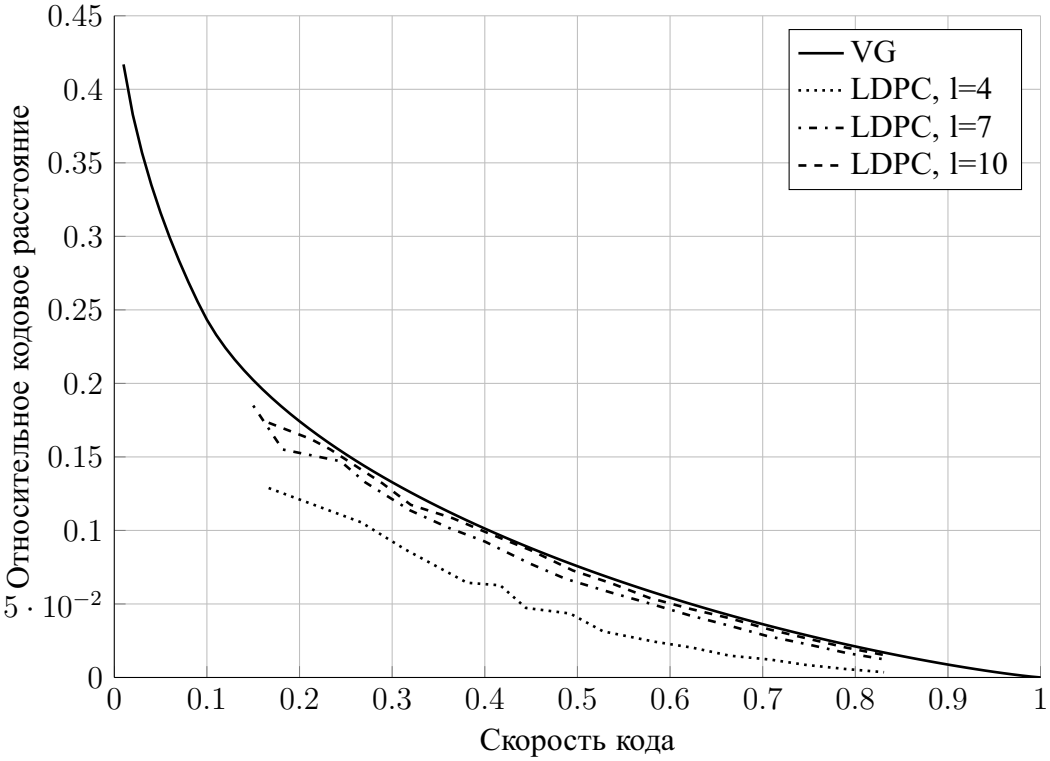


Рисунок 3.2. Нижние границы на кодовое расстояние конструкции B2 с компонентными МПП-кодами с различным весом столбца проверочной матрицы

3.2.3 Декодирование

Мягкий каскадный декодер

Основной рассматриваемый алгоритм декодирования этих конструкций — это мягкое декодирование этой конструкции алгоритмом, являющимся адаптацией жёсткого алгоритма декодирования ОЛО-кодов. Это мягкий алгоритм декодирования, который работает следующим образом.

Алгоритм принимает на вход от канала матрицу апостериорных вероятностей $Pr\{x_i = 1|y_i\}$, где x_i и y_i — входы и выходы канала соответственно. Эти величины фактически являются распределениями вероятностей (РВ) символов над $GF(2)$. Обозначим матрицу принятых значений $\mathbf{C}$. Алгоритм включает в себя следующие шаги:

1. Вычисляем $\mathbf{S}$ с использованием $\mathbf{C}$ как входа. Обозначим эту операцию как матричное произведение $\mathbf{S} = \mathbf{H}_0 \cdot \mathbf{C}$. Она сходна с операцией в определении кода, но имеет существенное отличие: элементы матриц $\mathbf{S}$ и $\mathbf{C}$ — это не символы, а вероятностные распределения над полем $GF(2)$. Обозначим

$$\mathbf{S}_i = \begin{pmatrix} s_1 \\ s_2 \\ s_3 \end{pmatrix}, \mathbf{C}_i = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix}, \tag{3.8}$$

где $\mathbf{S}_i$ и $\mathbf{C}_i$ — произвольные столбцы матриц $\mathbf{S}$ и $\mathbf{C}$ соответственно. Тогда согласно структуре матрицы $\mathbf{H}_0$:

$$\begin{pmatrix} s_1 \\ s_2 \\ s_3 \end{pmatrix} = \begin{pmatrix} c_1 \oplus c_3 \\ c_2 \oplus c_3 \\ c_3 \end{pmatrix}. \tag{3.9}$$

Операция суммирования здесь соответствует вычислению РВ суммы двух величин, имеющих РВ, заданные как аргументы оператора.

Запись выше означает, что третья строка $\mathbf{S}$ равна первой строке $\mathbf{C}$, а две первые строки $\mathbf{S}$ — синдромы кодов, проверочные матрицы которых заданы двумя нижними строками матрицы $\mathbf{H}_0$.

Это код длины 3 и скорости $1/3$. Рассмотрим вычисление этих символов синдрома двумя способами.

Первый — это независимое вычисление РВ каждого символа синдрома по отдельности с использованием тех символов кодового слова, от которых зависит этот символ синдрома. Например, $s_1 = c_1 \oplus c_3 = c_1(1 - c_3) + (1 - c_1)c_3$.

Рассмотрим второй способ. Синдром $(s_1, s_2)^T$ этого кода — это двоичный вектор длины 2, и существует всего $2^2 = 4$ возможных его значения. Принятое слово $(c_1, c_2, c_3)^T$ — это двоичный вектор длины 3, с $2^3 = 8$ возможными значениями, каждое из которых соответствует определенному значению синдрома. Зная вектор $\mathbf{C}_i = (c_1, c_2, c_3)^T$, можно вычислить все восемь этих вероятностей, после чего суммы вероятностей, соответствующих определённым значениям синдромов, дадут значения четырёх возможных вероятностей значений синдрома. От этих четырёх вероятностей легко перейти обратно к РВ отдельных битов: для того, чтобы получить вероятность s_i того, что i-й бит равен 1, нужно просуммировать вероятности синдромов из подмножества, где этот бит равен 1.

Оба метода дают одинаковый результат. Первый вычислительно проще. Второй здесь приведён с целью показать, что у рассматриваемого кода $(3,1,3)_2$ есть четыре возможных значения синдрома и восемь соответствующих им слов. Вероятностные распределения над этими множествами будут необходимы на шаге 3.

2. После получения $\mathbf{S}$ используем декодер первого внешнего кода для обновления $\mathbf{S}_{(12)}$:

$$\mathbf{S}'_{(12)} = \mathrm{Dec}\{\mathbf{H}_E^{(12)}\}(\mathbf{S}_{(12)}). \tag{3.10}$$

3. Теперь после того, как внешний код изменил символы синдрома первого внутреннего кода, требуется обновить с помощью внутреннего кода символы матрицы $\mathbf{C}$, чтобы они соответствовали обновлённым синдромам. Эта операция производится следующим образом. Четыре вероятности синдромов соответствуют четырём парам вероятностей слов. Обновлённая $\mathbf{S}'_{(12)}$ даёт информацию о новых вероятностях синдромов и, соответственно, пар слов, но

не даёт никакой информации о вероятностных распределениях внутри этих пар. Поэтому восемь новых вычисленных вероятностей слов должны быть таковы, чтобы вероятности пар символов (суммарные) были равны вероятностям, заданным новыми синдромами, а внутри пар соотношение вероятностей символов сохранялось бы.

4. После вычисления $\mathbf{C}'$ можно получить вход второго внешнего кода $\mathbf{S}'_{(3)}$, который равен третьей строке $\mathbf{C}'$.

5. После этого декодер второго внешнего кода обновляет $\mathbf{S}_{(3)}$:

$$\mathbf{S}''_{(3)} = \mathrm{Dec}\{\mathbf{H}_E^{(3)}\}(\mathbf{S}'_{(3)}). \tag{3.11}$$

6. Получение $\mathbf{C}''$ также является тривиальным: используем результат декодирования второго внешнего кода ($\mathbf{C}''_{(3)} = \mathbf{S}''_{(3)}$) и нижнюю часть матрицы из шага 3:

$$\mathbf{C}'' = \begin{pmatrix} \mathbf{C}''_{12} \\ \mathbf{C}'_{3} \end{pmatrix}. \tag{3.12}$$

7. Алгоритм останавливается, если достигнуто максимальное число итераций или оба внешних декодера вернули кодовые слова внешних кодов. В противном случае алгоритм переходит к шагу 1, где в качестве нового значения $\mathbf{C}$ используется полученная на предыдущем шаге $\mathbf{C}''$.

Декодер “распространения доверия” и гибридный декодер

Как было показано выше, существуют другие способы декодирования этой конструкции.

Во-первых, можно воспользоваться алгоритмом “распространения доверия” [31] для декодирования кода по построенной проверочной матрице.

Во-вторых, возможно совместить декодирование с помощью алгоритма “распространения доверия” и с помощью предложенного декодера. Назовём этот метод декодирования “гибридным”.

Гибридное декодирование выполнялось следующим образом. Сначала к символам, полученным из канала применялось пять итераций декодера “распространения доверия”. Если в результате этого оказывалось получено кодовое слово, то алгоритм останавливался, в противном же случае выход алгоритма использовался как вход предложенного каскадного декодера.

Сравнение предложенных алгоритмов декодирования

Предложенные алгоритмы декодирования были реализованы на языках Си и Матлаб для имитационного моделирования. Эффективность декодеров рассматривалась в канале с двоичной фазовой модуляцией и аддитивным белым гауссовским шумом.

На рис. 3.3 и рис. 3.4 представлены результаты сравнения трёх рассматриваемых алгоритмов декодирования, применяемых для одного и того же кода. Длина рассматриваемой конструкции

$n = 3000$, скорость $R = 0.5$. Параметры компонентных кодов выбраны равными $n_L = 2000$, $R_L = 0.4$ для первоговнешнего кода и $n_U = 1000$, $R_U = 0.7$ для второго внешнего кода.

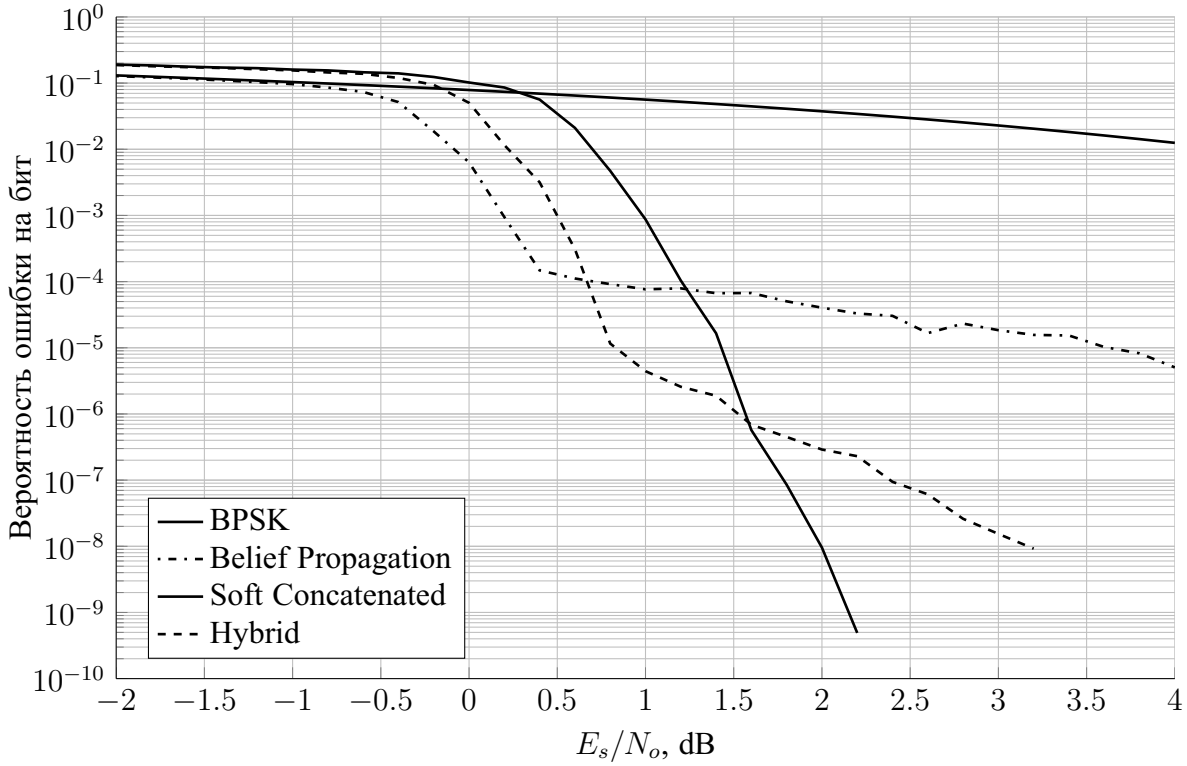


Рисунок 3.3. Вероятности ошибки на бит в зависимости от соотношения сигнал/шум для рассматриваемых декодеров

Из этих результатов легко видеть, что алгоритм “распространения доверия” показывает на этих кодах весьма высокую “полку” вероятности ошибки. Она находится на уровне вероятности ошибки на блок, равной 10^{-1}, и начинается практически сразу после того, как входное соотношение сигнал/шум становится достаточным для работы декодера.

Мягкий каскадный алгоритм декодирования не показывает признаков “полки” вероятности ошибки на тех вероятностях, на которых было проведено моделирование, то есть 10^{-9} на бит или 10^{-7} на блок. (Моделирование заняло около недели на процессоре с 6 ядрами по 3.2 ГГц.) В то же время точка, с которой этот алгоритм начинает свою работу почти на 1 дБ хуже, чем у алгоритма “распространения доверия”.

Как можно видеть из графиков, гибридный алгоритм показал промежуточное поведение как относительно точки, в которой он начал работу, так и относительно “полки” вероятности ошибки. Можно видеть промежуточную область, где он выигрывает у каждого из первых двух алгоритмов.

Можно предположить, что возникновение “полки” вероятности ошибки у алгоритма “распространения доверия” связано с существенной ролью коротких циклов в графе Таннера проверочной матрицы кода, что, однако, не сказывается на мягком каскадном декодере.

Можно с уверенностью сказать, что полка вероятности ошибки связана с типом применяемого декодера, но не с самим кодом.

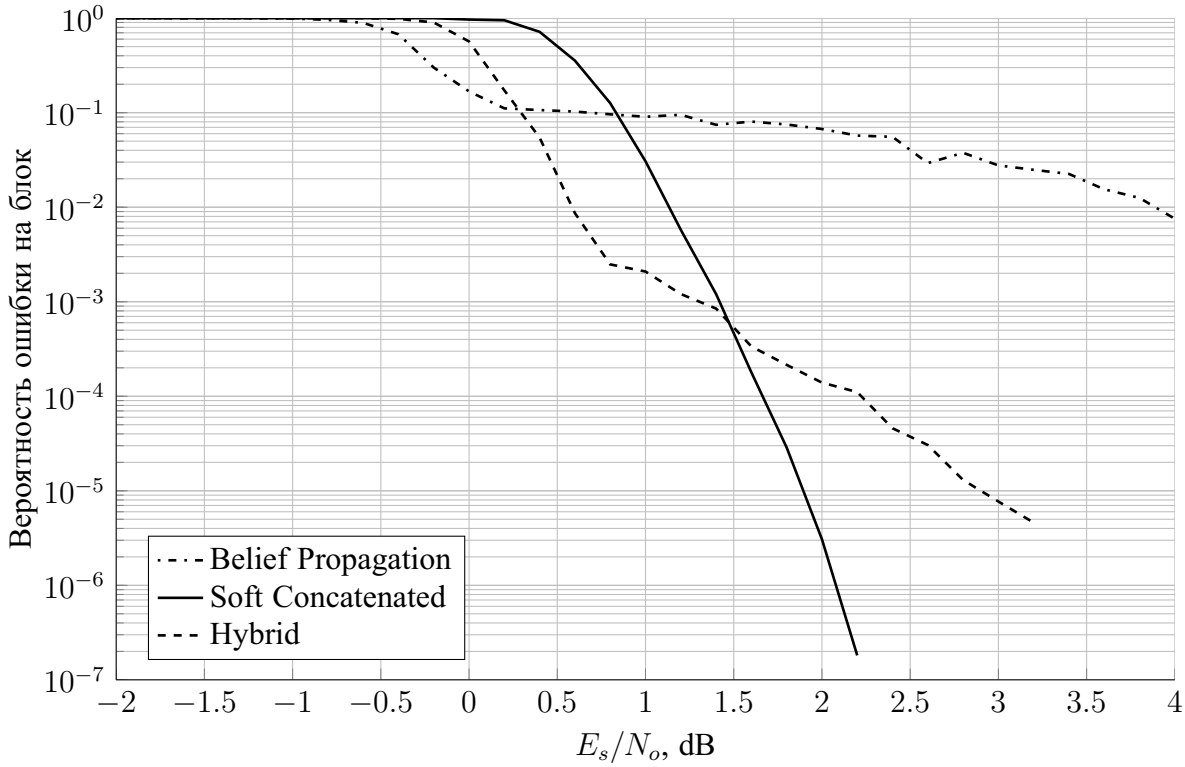


Рисунок 3.4. Вероятности ошибки на блок в зависимости от соотношения сигнал/шум для рассматриваемых декодеров

3.3 ОЛО-коды на основе кодов Рида-Соломона с мягким декодированием внутренних кодов

3.3.1 Введение

В данном разделе рассматривается мягкое декодирование кодов, описанных в разделе 2.2. Для них предложен алгоритм декодирования, в котором декодирование внутренних кодов происходит по максимуму правдоподобия, а внешних — с помощью классического декодера, исправляющего ошибки до половины кодового расстояния. Для этого алгоритма построена верхняя граница на вероятность неправильного декодирования и с использованием этой границы построен алгоритм выбора параметров конструкции.

3.3.2 Мягкое декодирование внутренних кодов

Рассмотрим передачу j-ого внутреннего кода через канал без памяти и с мягким выходом. Каждый символ c_i переданного слова $\mathbf{c}_j$ будет принят как мягкое значение. Это значение может быть представлено в виде вещественного вектора над $\mathbb{R}^q$:

$$\mathbf{v}_i = \big(Pr\{c_i = 0\}, Pr\{c_i = 1\}, \ldots, Pr\{c_i = q-1\}\big), \tag{3.13}$$

где $Pr\{c_i = k\}$, $k = 0..q-1$ — вероятность того, что $c_i = k$. Ясно, что

$$\sum_{k=0}^{q-1} Pr\{c_i = k\} = 1. \tag{3.14}$$

Таким образом, принятое слово будем рассматривать как вектор мягких значений:

$$\mathbf{v} = (\mathbf{v}_1 \mathbf{v}_2 \dots \mathbf{v}_{n_A}). \tag{3.15}$$

С этого момента и далее будем оперировать только мягкими значениями, не используя их внутреннюю структуру.

Так как переданные слова являются представителями смежных классов (включая само кодовое пространство), декодер внутреннего кода должен знать смежный класс, в котором будет декодироваться принятое слово. Цель декодера — найти слово из смежного класса, которое будет ближайшим к переданному. Таким образом, декодер имеет два входных параметра: первый — номер смежного класса, и второй — принятое слово. Он возвращает слово из заданного смежного класса, ближайшее к принятому. Обозначим операцию декодирования j-го внутреннего кода следующим образом:

$$\mathbf{v}_j = Dec_A[\mathbf{s}_j]\{\mathbf{v}\}, \tag{3.16}$$

где $\mathbf{s}_j$ — номер смежного класса, $\mathbf{v}$ — принятое слово, и $\mathbf{v}_j$ — результат декодирования (вектор над $GF(q)$). Номер смежного класса $\mathbf{s}_j$, очевидно, является синдромом принятого слова $\mathbf{c}_j$: $\mathbf{s}_j = \mathbf{H}_j \mathbf{c}_j$.

В данной главе рассматривается декодирование коротких внутренних кодов по максимуму правдоподобия. Можно отметить, что в зависимости от скорости кода декодирование по максимуму правдоподобия может происходить разными способами. При малой скорости кода более выгодным является перебор кодовых слов, обеспечивающий сложность $O(nq^k)$. При высоких скоростях оптимальным является декодирование по решётке, сложность которого ограничена $O(nq^r)$, где $r = n - k$ — число проверочных символов. Границы сложности декодирования кодов по решётке можной найти в [43]. Декодирование по решётке q-ичных кодов описано в [44].

Поскольку требуется декодировать по решётке не сам код, а его смежный класс, алгоритм декодирования будет иметь небольшие отличия. Опишем алгоритм.

Для работы декодера требуется построение синдромной решётки. Она представляет из себя граф, который строится следующим образом. Для q-ичного кода (n,k) решётка содержит $(n+1) \times q^r$ вершин. Вершины разбиты на q^r ярус, каждый из которых пронумерован определённым значением синдрома. Также она разбита на $(n+1)$ секций, i-я секция решётки, $i \in \overline{1,n}$, соответствует состоянию декодера после приёма (обработки) i-го кодового символа. Секция с $i = 0$ является начальным состоянием.

Между вершинами решётки идут рёбра, соответствующие возможным изменениям состояния синдрома. Очевидно, что если синдром имеет заданное состояние, то после приёма очередного

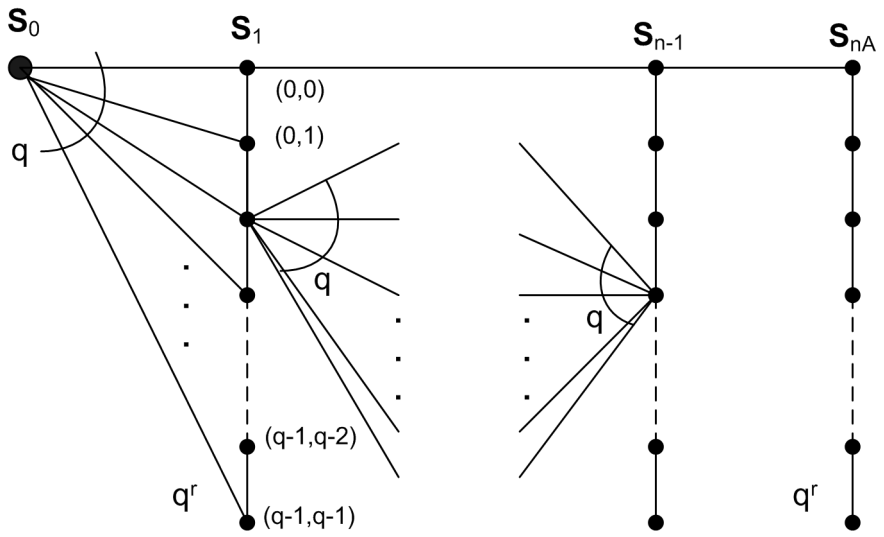


Рисунок 3.5. Схематичное изображение синдромной решётки

символа возможен переход в одно из q возможных состояний. Значение состояния после приёма i-го символа v_i определяется как $\mathbf{s}_i = \mathbf{s}_{i-1} + \mathcal{C}_i\{\mathbf{H}\}v_i$, где $\mathcal{C}_i\{\mathbf{H}\}$ — i-й столбец проверочной матрицы рассматриваемого кода.

В случае мягкого декодирования вместо детерминированных значений речь идёт о вероятностном распределении по возможным значения синдрома на каждом состоянии. Пришедшее из канала мягкое значение задаёт распределение вероятностей q возможных переходов из каждого из q^r состояний $\mathbf{s}_{i-1}$ в q^r состояний $\mathbf{s}_i$. Отсюда можно сделать наблюдение, что из каждой вершины выходит и в каждую из них входит по q рёбер. Это нарушается только в начале, т. к. в начальное состояние ничего не приходит, и оно задаётся как детерминированное и равное нулю.

Схематичное изображение решётки приведено на рис. 3.5.

Для декодирования по максимуму правдоподобия требуется пройти по решётке, запоминая соответствующие пройденному пути значения принятых символов. Из q приходящих в вершину ребер оставляется только то, которое имеет наибольшую вероятность.

Если требуется декодировать сам код, итоговое слово соответствует пути, приходящему на n-й секции в нулевое состояние, то есть соответствующее нулевому синдрому. Так как нас интересует декодирование в смежных классах, требуется выбирать путь, приходящий не в нулевое состояние, а в соответствующее смежному классу, заданному на входе декодера.

Рассмотрим два примера системы вложенных внутренних кодов.

Пусть требуется построить ОЛО-код порядка $L = 2$. Так как $L = n_A/2$, то внутренние коды имеют длину $n_A = 4$. В этом случае система вложенных внутренних кодов состоит из первого кода с параметрами $(4,2,3)$ и второго кода с параметрами $(4,0,\infty)$. Под последним кодом будем понимать код нулевой скорости (т.е. код, состоящий из единственного нулевого слова). Код РС $(4,2,3)$ будем декодировать по мягкому максимуму правдоподобия.

На рисунке 3.6 представлены результаты моделирования для кода РС $(4,2,3)$. Представлены две кривые. Первая (сплошная) — это зависимость между отношением сигнал/шум на кодо-

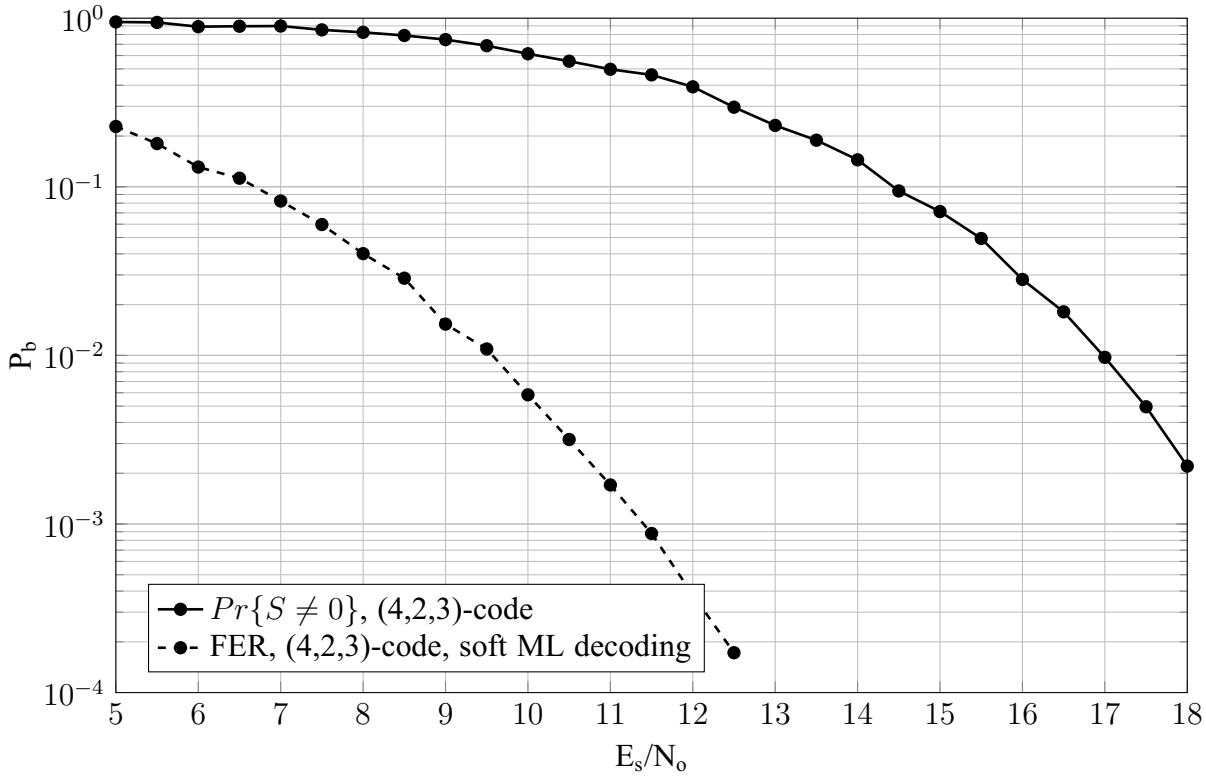


Рисунок 3.6. Входные вероятности внешних кодов, $L = 2$

вый символ и вероятностью того, что синдром жесткого решения от принятого слова не равен нулевому вектору. Данную вероятность можно трактовать как вероятность обнаруженной ошибки. Данная величина равна вероятности ошибки на входе первого внешнего кода. Вероятность ошибки на блок при декодировании по мягкому максимуму правдоподобия для кода РС $(4,2,3)$ представлена пунктирной линией. Данная вероятность совпадает с вероятностью на входе второго внешнего кода.

Теперь рассмотрим второй пример. Построим ОЛО-код порядка $L = 3$. Система внутренних кодов состоит из трех укороченных РС-кодов длины $n_A = 6$. Данные коды имеют следующие параметры:

– $(n_A,k,d) = (6,4,3)$,

– $(n_A,k,d) = (6,2,5)$,

– $(n_A,k,d) = (6,0,\infty)$.

Первый внутренний код декодируется по решетке, как описано выше, а второй — по перебором слов смежного класса. На рисунке 3.7 представлены входные вероятности для внешних кодов в зависимости от входного отношения сигнал/шум на кодовый символ.

Как и в предыдущем примере, кривая справа — это зависимость между отношением сигнал/шум на кодовый символ и вероятностью того, что синдром жесткого решения от принятого слова укороченного кода РС $(6,4,3)$ не равен нулевому вектору. Данную вероятность можно трак-

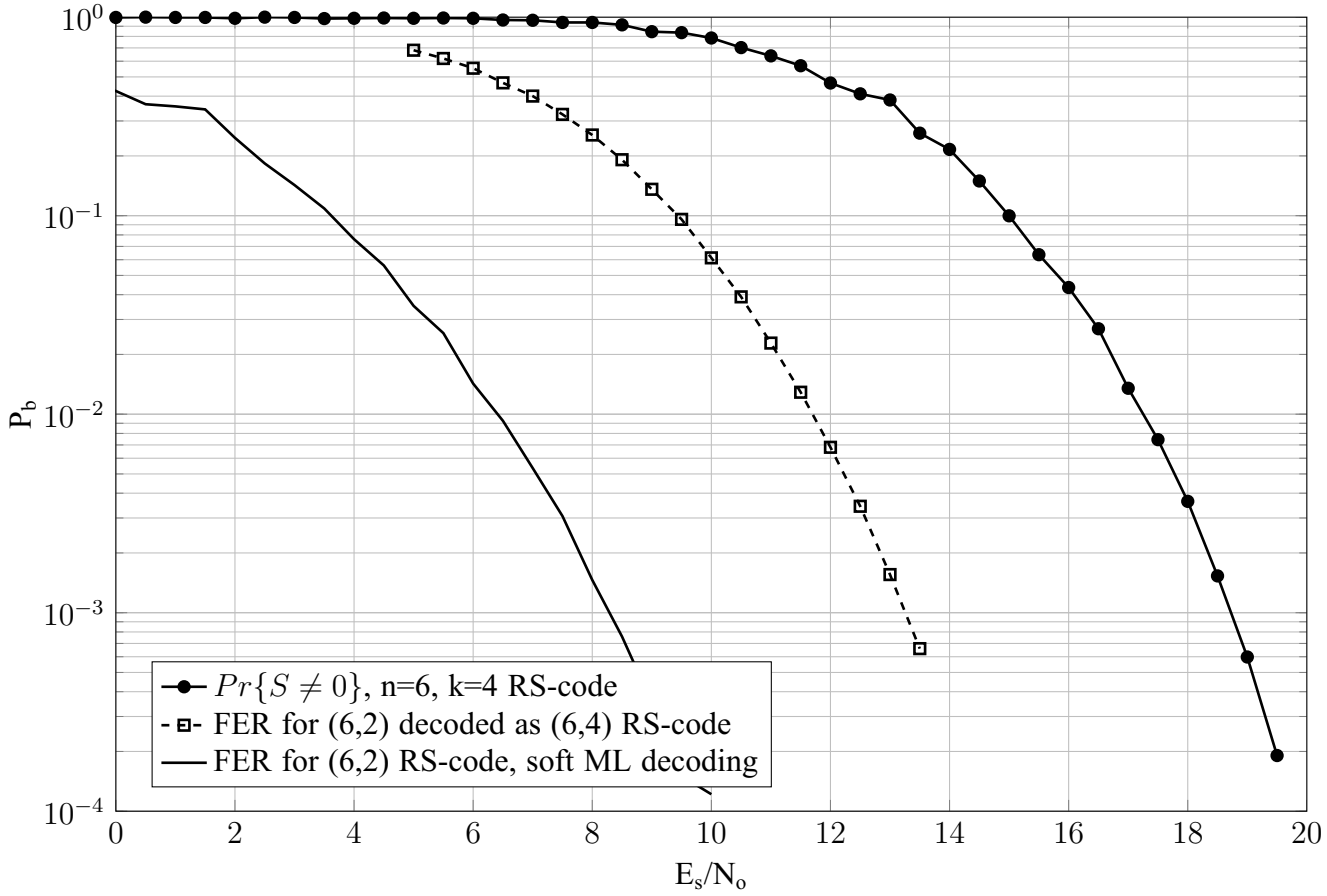


Рисунок 3.7. Вероятности на входе внешних кодов для ОЛО-кода с $L = 3$

товать как вероятность обнаруженной ошибки. Данная величина равна вероятности ошибки на входе первого внешнего кода, который имеет наибольшую избыточность.

Пунктирная кривая отражает вероятность ошибки на входе второго внешнего кода. Она получена следующим образом: слова укороченного кода РС $(6,2,5)$, мягко принятые из канала, декодируются по максимуму правдоподобия декодером кода РС $(6,4,3)$, после чего вычисляется синдром ошибки для $(6,2,5)$ кода. Очевидно, что ненулевой синдром соответствует ошибке в соответствующем символе на входе второго внешнего кода.

Результаты декодирования (вероятность ошибки на блок) для кода РС $(6,2,5)$ представлены сплошной линией (слева). Эта вероятность совпадает с вероятностью на входе третьего внешнего кода, обладающего наименьшей избыточностью.

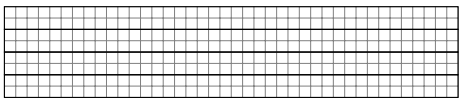
Можно заметить, что для оценки корректирующих свойств ОЛО-кода, не важен конкретный тип декодера внутреннего кода. Единственное, что существенно — это вероятность ошибки каждого кода системы вложенных внутренних кодов для заданного канала. Определим p_{Aj} как вероятность неправильного декодирования j-го внутреннего кода. Эти вероятности впоследствии будут использованы для вывода верхней границы вероятности неправильного декодирования всей конструкции ОЛО-кода целиком.

3.3.3 Мягкое декодирование ОЛО-кода

Рассмотрим передачу кодового слова $\mathbf{C}$ ОЛО-кода порядка L через канал без памяти и с мягким выходом.

В этом случае, на приемном конце будет принята матрица мягких значений размера $n_A \times n_B$:

$$\mathbf{V} = \quad . \tag{3.17}$$



Алгоритм декодирования ОЛО-кода является итеративным, причем количество итераций совпадает с порядком кода L. Каждая итерация включает в себя два основных шага: декодирование внешних и декодирование внутренних кодов. Рассмотрим j-ую итерацию декодирования при условии, что декодирование на предыдущих $j-1$ итерациях было безошибочным.

Таким образом, j-й ($j = \overline{1,L}$) шаг декодирования представим следующим образом:

– Перед декодированием построим матрицу $\mathbf{V}_0$, которая является жестким решением от матрицы $\mathbf{V}$ мягких значений (можно считать, что $\mathbf{V}_0$ — результат декодирования 0-го внутреннего кода.)

– Слово $\mathbf{V}_{j-1}$, полученное на предыдущей итерации, умножается слева на $\mathbf{H}_j$. В результате, получим матрицу синдромов j-го внутреннего кода:

$$\mathbf{S}'_j = \mathbf{H}_j \cdot \mathbf{V}_{j-1} = \quad . \tag{3.18}$$

Полученная матрица возможно будет содержать некоторые столбцы, которые отличаются от переданных. Это означает, что они содержат ошибки. В формуле 3.18 обозначены примеры таких столбцов (отмечены черным).

– Декодеры кодов $\mathcal{C}_1^B, \ldots, \mathcal{C}_j^B$ исправляют ошибки в $\mathbf{S}'_j$:

$$\mathbf{S}''_j = Dec_B\{\mathbf{S}'_j\}. \tag{3.19}$$

В результате при условии, что внешние коды обладают достаточной корректирующей способностью, чтобы исправить все ошибки в $\mathbf{S}'_j$, столбцы матриц $\mathbf{S}''_j$ укажут правильные смежные классы для декодирования внутренних кодов.

Если внешний код выдаёт отказ от декодирования, то алгоритм декодирования останавливается, и выдаётся отказ от декодирования всей кодовой конструкции.

– Полученная матрица $\mathbf{S}''_j$ используется j-ым внутренним кодом для того, чтобы исправить ошибки в $\mathbf{V}$:

$$\mathbf{V}_j = Dec_A[\mathbf{S}''_j]\{\mathbf{V}\}. \tag{3.20}$$

– После декодирования последним, L-ым слоем, декодирование заканчивается. Матрица $\mathbf{V}_L$ считается результатом декодирования.

3.3.4 Верхняя граница вероятности неправильного декодирования

Рассмотрим вероятность неправильного декодирования ОЛО-кода порядка L для описанного выше алгоритма декодирования.

Повторим ключевые особенности алгоритма декодирования, которые потребуются для того, чтобы написать верхнюю оценку вероятности неправильного декодирования ОЛО-кода:

– Декодирование — процесс из L итераций.

– Вероятность ошибки на j-ой итерации для j-ого внутреннего кода — p_{Aj}.

– Внешние коды имеют избыточности r_j^B и способны исправлять ошибки кратности $t_j^B \leq \lfloor r_j^B/2 \rfloor$.

Декодирование $(j-1)$-ого внутреннего кода может привести к ошибке в столбце матрицы $\mathbf{V}_{j-1}$. После умножения этой матрицы на $\mathbf{H}_j$, появится ошибка в $\mathbf{S}'_j$, которая должна быть исправлена внешним кодом. Это означает, что вероятность ошибки на входе декодера j-ого внешнего кода ограничена сверху вероятностью ошибки $(j-1)$-ого внутреннего кода.

Обозначим через

$$p_j \leq p_{A(j-1)} \tag{3.21}$$

— вероятность ошибки в столбце матрицы $\mathbf{S}'_j$ для j-ого шага. Эти вероятности совпадают для всех столбцов.

Вероятность появления более чем t_j^B ошибок в $\mathbf{s}_j$ и, соответственно, вероятность неправильного декодирования внешнего кода, ограничена сверху следующей величиной:

$$p_{Bj} = \sum_{i=t_j^B+1}^{n_B} \binom{n_B}{i} p_j^i (1-p_j)^{n_B-i}. \tag{3.22}$$

Вероятность ошибочного декодирования ОЛО-кода определяется вероятностью ошибки любого внешнего кода. Это замечание позволяет сформулировать следующее утверждение:

Т е о р е м а 7. *Пусть p_{Bj} — верхняя граница вероятности ошибки для j-ого внешнего кода, $j = \overline{1,L}$. Тогда, для вероятности ошибки p ОЛО-кода C выполняются следующие соотношения:*

$$p \leq p_B = \min\{\sum_{j=1}^{L} p_{Bj}, 1\}, \tag{3.23}$$

где L — число слоев ОЛО-кода, p_B — верхняя граница на вероятность неправильного декодирования.

3.3.5 Поиск избыточностей кодов-компонентов, обеспечивающих заданную выходную вероятность ошибки при заданной входной

Пусть заданы базовые параметры кода (q, n_A, n_B, L, t_j^A), вероятности P_{Aj} и требуемая вероятность неправильного декодирования $p_f = p_B$. Требуется найти избыточности внешних кодов-компонентов для того, чтобы обеспечить требуемую вероятность ошибки.

Алгоритм поиска оптимальных скоростей компонентных кодов состоит из следующих шагов:

- Зная P_{Aj}, вычислим верхнюю границу на p_j — вероятности появления ошибочных символов в S_j', которые должны быть исправлены на j-ом шаге декодирования j-ым внешним кодом.

- Далее требуется ограничить вероятность ошибки каждого из внешних кодов. Очевидно, что если $\forall j : p_{Bj} \leq p_B/L$, то вероятность ошибки кодовой конструкции не превысит требуемой p_B.

  Поэтому следует найти такие t_j^B, которые обеспечат выполнение условия $p_{Bj} \leq p_B/L$.

- Количество проверочных символов в кодах вычислить как $r_j^B = \min(2t_j^B, n_B)$.

На данном этапе известны параметры кода, которые обеспечивают требуемые характеристики выходной вероятности ошибки при заданной входной, но могут быть неоптимальны. В данном случае требуется провести оптимизацию параметров кода. Это можно сделать следующим образом:

- Для набора r_j^B, полученного на предыдущем шаге, построить все альтернативные наборы $\{\hat{r}_j^B\}_k$ следующим образом:

  $\hat{r}_{jk}^B = r_j^B, j \neq k$;

  $\hat{r}_{jk}^B = r_j^B - 2, j = k$

  для $k = \overline{1,L}$

- Для каждого из этих наборов вычислить вероятность ошибки в соответствии с алгоритмом, описанным в разделе 3.3.4.

- Выбрать тот набор, вероятность ошибки которого наименьшая.

- Если полученная вероятность меньше требуемой p_f, то повторить эту же операцию. Если больше, то выполнить аналогичную, но не с уменьшением, а с увеличением корректирующей способности кодов.

- Алгоритм останавливается, если после двух шагов набор r_j^B не изменился.

3.3.6 Численные результаты

В данном разделе используются описанный выше метод оптимизации и полученные оценки для построения ОЛО-кода для заданной выходной вероятности ошибки и входном отношении сигнал/шум на кодовый символ. Расчеты проведены для канала с аддитивным белым гауссовским шумом с амплитудной модуляцией КАМ-16. Рассмотрим эффективность (с точки зрения максимально достижимой скорости для заданной выходной вероятности) для предложенной конструкции и сравним полученные результаты с конструкцией ОЛО-кода, предложенного в [10].

Как уже было отмечено ранее, в данной главе используется мягкое декодирование по максимуму правдоподобия для внутренних кодов длин $n_A = 4$ и $n_A = 6$. Внутренние коды — укороченные коды Рида-Соломона над $GF(2^4)$. В обоих случаях длина внешнего кода $n_B = 256$. Внешние коды также являются кодами Рида-Соломона (расширенными), но заданы над полем $GF(2^8)$. Таким образом, длины полученных ОЛО-кодов равны 4096 и 6144 бит соответственно. Для случая, когда $n_A = 4$, количество слоев равно $L = 4$, а для $n_A = 6$ имеем $L = 3$.

E_s/N_0	p_s	R (s,2)	R (h,2)	R (s,3)	R (h,3)
5	$5.37 \cdot 10^{-1}$	0.0469	0	0.2292	0
6	$4.80 \cdot 10^{-1}$	0.1758	0	0.263	0
7	$4.19 \cdot 10^{-1}$	0.2539	0	0.2865	0
8	$3.54 \cdot 10^{-1}$	0.332	0	0.3099	0
9	$2.87 \cdot 10^{-1}$	0.3906	0	0.4219	0.0286
10	$2.22 \cdot 10^{-1}$	0.4258	0.0625	0.5104	0.1198
11	$1.62 \cdot 10^{-1}$	0.4492	0.1836	0.5677	0.2083
12	$1.09 \cdot 10^{-1}$	0.4648	0.2891	0.6042	0.3672
13	$6.75 \cdot 10^{-2}$	0.5078	0.3906	0.6276	0.4844
14	$3.72 \cdot 10^{-2}$	0.6523	0.5781	0.6745	0.6094
15	$1.78 \cdot 10^{-2}$	0.7617	0.7227	0.7917	0.75
16	$7.15 \cdot 10^{-3}$	0.8477	0.8281	0.8672	0.849
17	$2.32 \cdot 10^{-3}$	0.9023	0.8906	0.9167	0.9089
18	$5.73 \cdot 10^{-4}$	0.9297	0.9297	0.9505	0.9427

Таблица 3.1. Скорости ОЛО-кодов в зависимости от входного отношения сигнал/шум на кодовый символ E_s/N_0 для АГБШ с модуляцией КАМ-16

В таблице 3.1 представлены параметры найденных ОЛО-кодов в зависимости от входного отношения сигнал/шум на кодовый символ для канала с модуляцией КАМ-16. В таблице использованы следующие обозначения:

– $R(s,L)$ — скорость ОЛО-кода с мягким декодированием внутреннего кода,

– $R(h,L)$ — скорость ОЛО-кода с жестким декодированием внутреннего кода,

– L — число слоев.

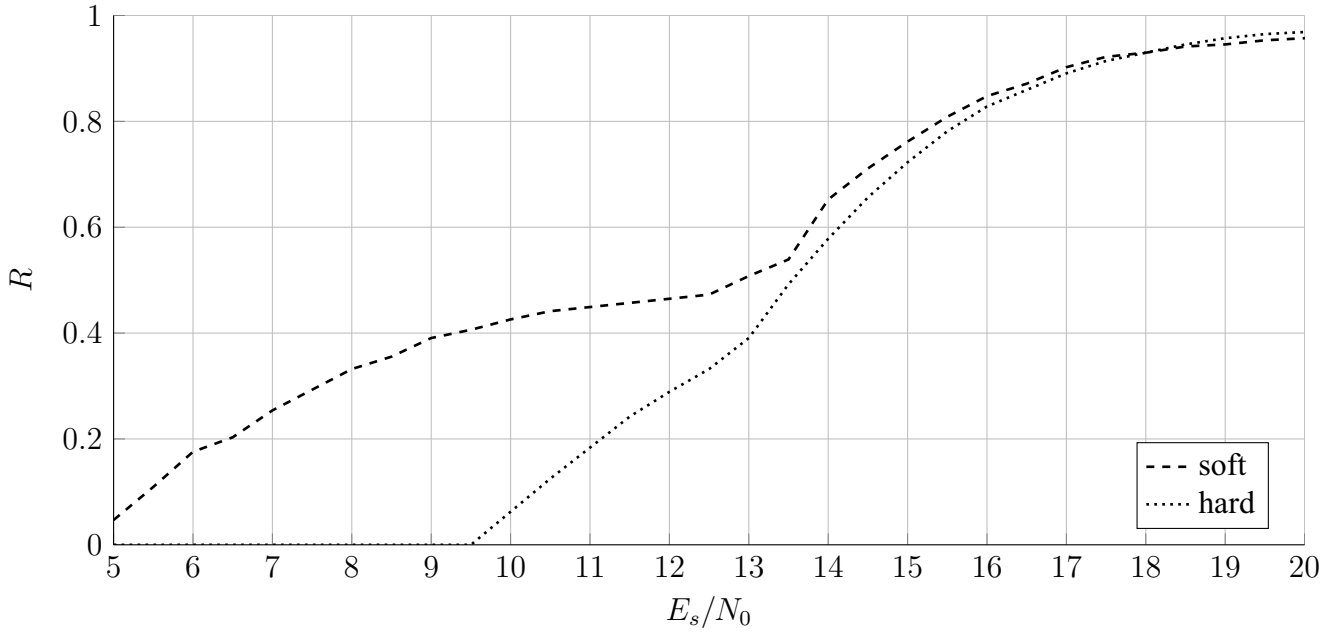


Рисунок 3.8. Скорости ОЛО-кодов в зависимости от входного отношения сигнал/шум на кодовый символ E_s/N_0 для АГБШ с модуляцией КАМ-16, код длины 4096; коды построены для одинаковой целевой вероятности неправильного декодирования, равной 10^{-15}

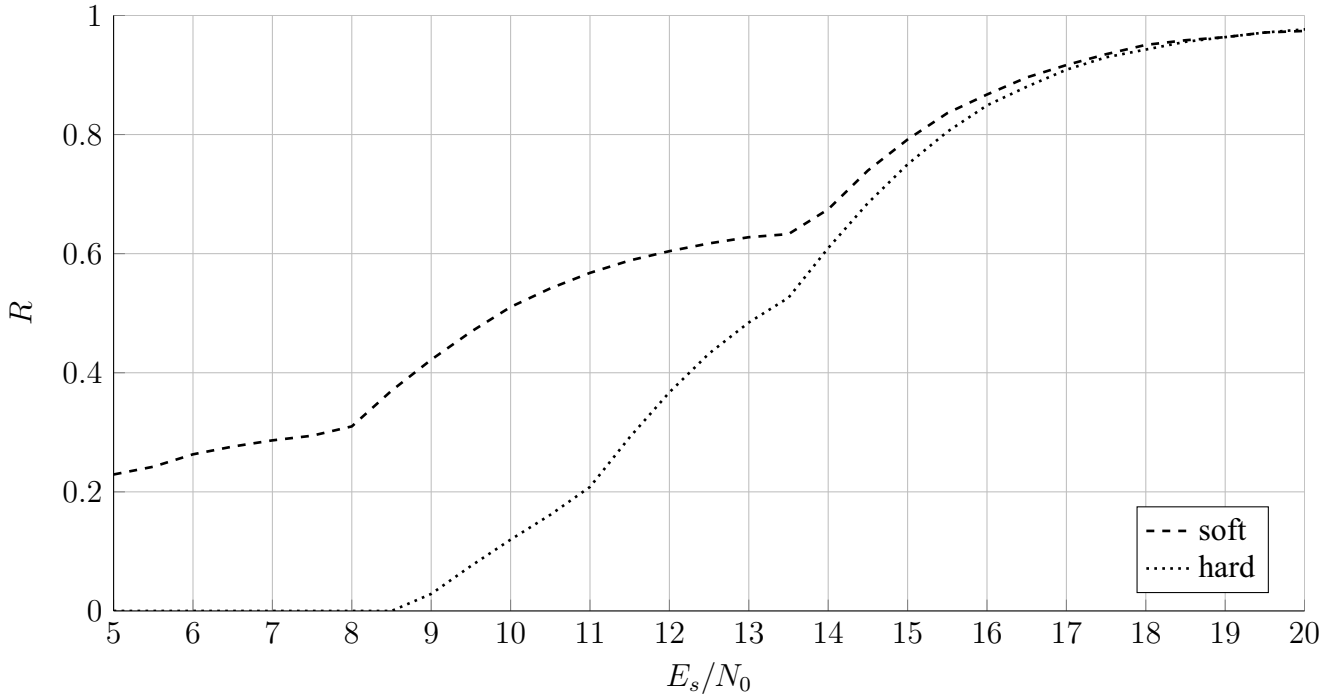


Рисунок 3.9. Скорости ОЛО-кодов в зависимости от входного отношения сигнал/шум на кодовый символ E_s/N_0 для АГБШ с модуляцией КАМ-16, код длины 6144; коды построены для одинаковой целевой вероятности неправильного декодирования, равной 10^{-15}

Требуемая выходная вероятность: $p_f = 10^{-15}$. Графическая иллюстрация данной таблицы приведена на рисунках 3.8 и 3.9.

Следует отметить, что при построении кода с малой скоростью скорость первых внешних кодов обращается в ноль, то есть эти коды не содержат информационных символов. Это эквивалентно отсутствию этих внешних кодов. В этом случае в столбцах передаются слова не смежных классов, а самих внутренних кодов. В этом случае декодировать внешний код не требуется, и декодирование внутренних кодов может происходить сразу же. Такой код уже не является обобщённым кодом с локализацией ошибок; по-сути, речь идёт о применении описанных выше алгоритма декодирования и методов расчёта к обобщённым каскадным кодам.

Переходы, где скорость одного из компонентных кодов обращается в ноль при ухудшении отношения соотношения сигнал/шум, видны на рис. 3.8 и 3.9 как точки, в которых резко изменяется наклон кривой. Так, на рис. 3.8 код для мягкого декодирования является ОЛО-кодом при $E_s/N_0 > 13.5$ дБ и обобщённым каскадным в противном случае. На рис. 3.9 наблюдается аналогичное поведение, а при $E_s/N_0 < 8$ дБ в ноль обращается скорость не только первого, но и второго внешнего кода.

Стоит также отметить, что при низких соотношениях сигнал/шум и, как следствие, высоких вероятностях ошибки на символ, есть широкая область, где построение кода для жёсткого декодирования невозможно (кривая скорости при этих значениях ложится на ось абсцисс). В качестве примера можно привести $E_s/N_0 = 9$ дБ для конструкции длины 4096 (см. рис. 3.8), где скорость кода с мягким декодированием внутренних кодов около 0.4, а с жёстким равна нулю.

Зафиксируем описанную конструкцию ОЛО-кода, когда $L = 2$, а также входную вероятность ошибки (которая является функцией от отношения сигнал/шум на информационный символ) и построим зависимость между максимально достижимой скоростью ОЛО-кода и требуемой выходной вероятностью ошибки.

Как видно из рисунков 3.10–3.12, чем больше входная вероятность ошибки, тем большим преимуществом в плане максимально достижимой скорости обладают коды с мягким внутренним декодированием. Более того, чем ниже требуемая выходная вероятность при заданной входной, тем больше разница в скорости. Таким образом, ОЛО-коды с мягким декодированием внутренних кодов целесообразно использовать как в случае, когда требуется получить очень низкую выходную вероятность ошибки, так и в случае, когда входная вероятность ошибки достаточно велика.

С помощью предложенного метода выбора избыточностей внешних компонентных кодов били построены коды длины $n = 4096$ и $n = 6144$.

Результаты декодирования ОЛО-кода длины 4096 представлены на рис. 3.13. На нем можно видеть сравнение кривой эффективности декодирования этого кода предложенным мягким декодером с верхней и нижней оценками на вероятность неправильного декодирования этого же кода при жестком декодировании [10]. Верхняя граница обозначена “точечной” линией, а нижняя — пунктирной. Легко увидеть, что численные результаты предложенного в данной статье метода декодирования очень близки к нижней границе. Однако не доказано, что полученная нижняя

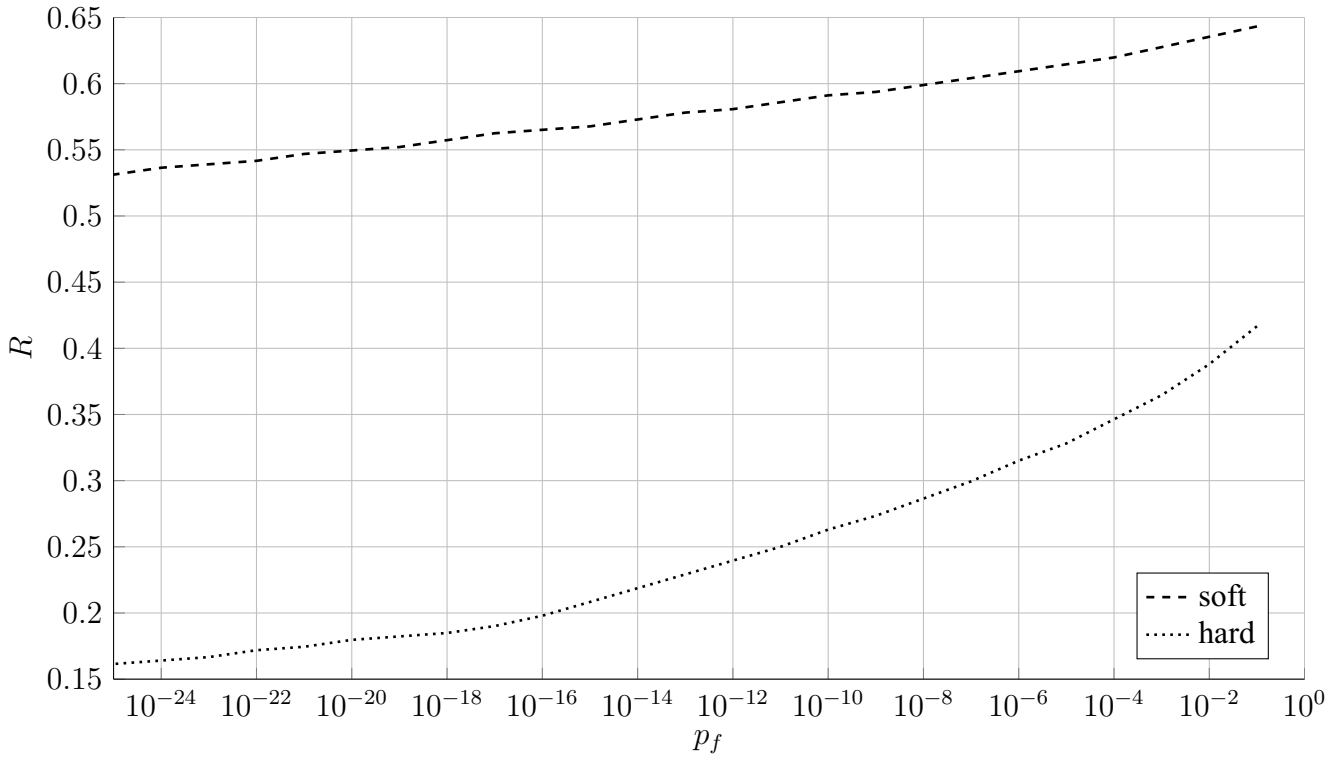


Рисунок 3.10. Зависимость между максимально достижимой скоростью кода и требуемой выходной вероятностью ошибки на блок для входной вероятности $E_s/N_0 = 11$ дБ ($p_s = 0.1617$) для ОЛО-кодов с жестким декодированием и кодов с мягким декодированием внутренних кодов

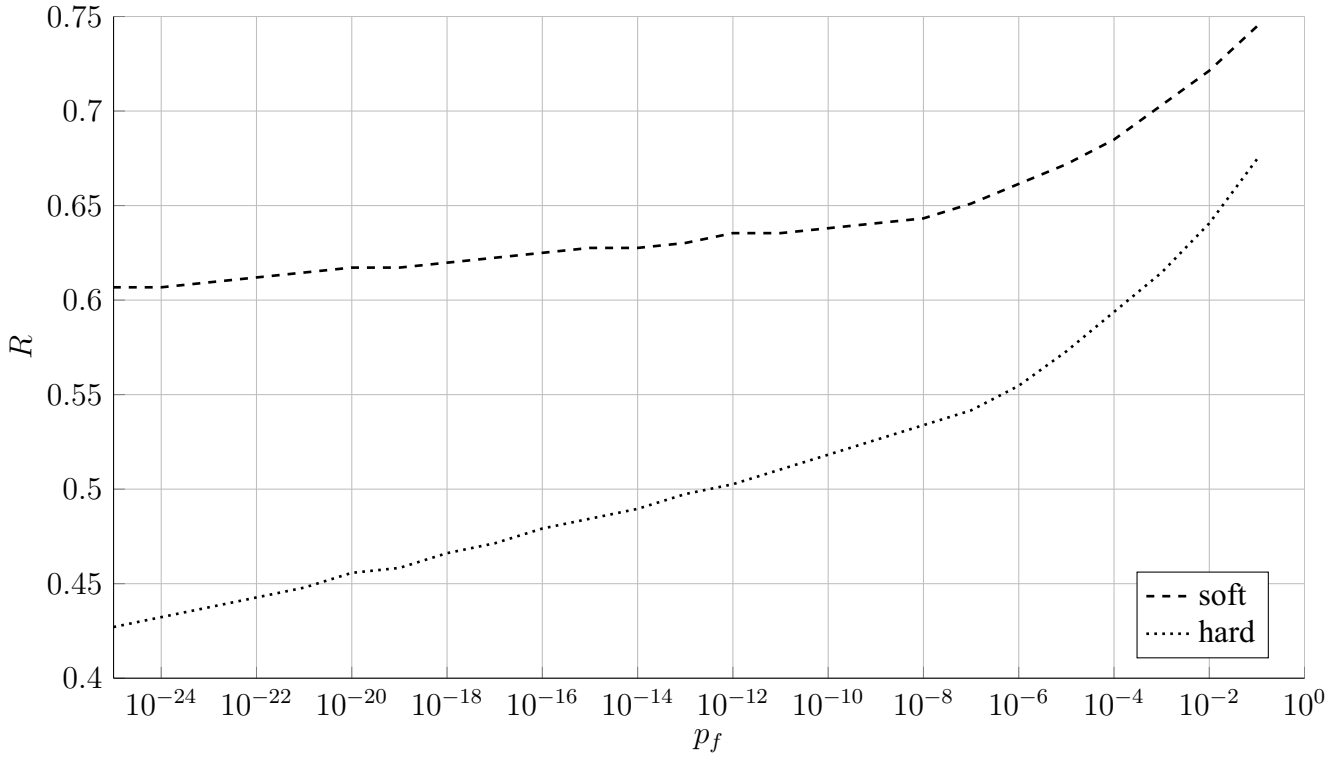


Рисунок 3.11. Зависимость между максимально достижимой скоростью кода и требуемой выходной вероятностью ошибки на блок для входной вероятности $E_s/N_0 = 13$ дБ ($p_s = 0.0675$) для ОЛО-кодов с жестким декодированием и кодов с мягким декодированием внутренних кодов

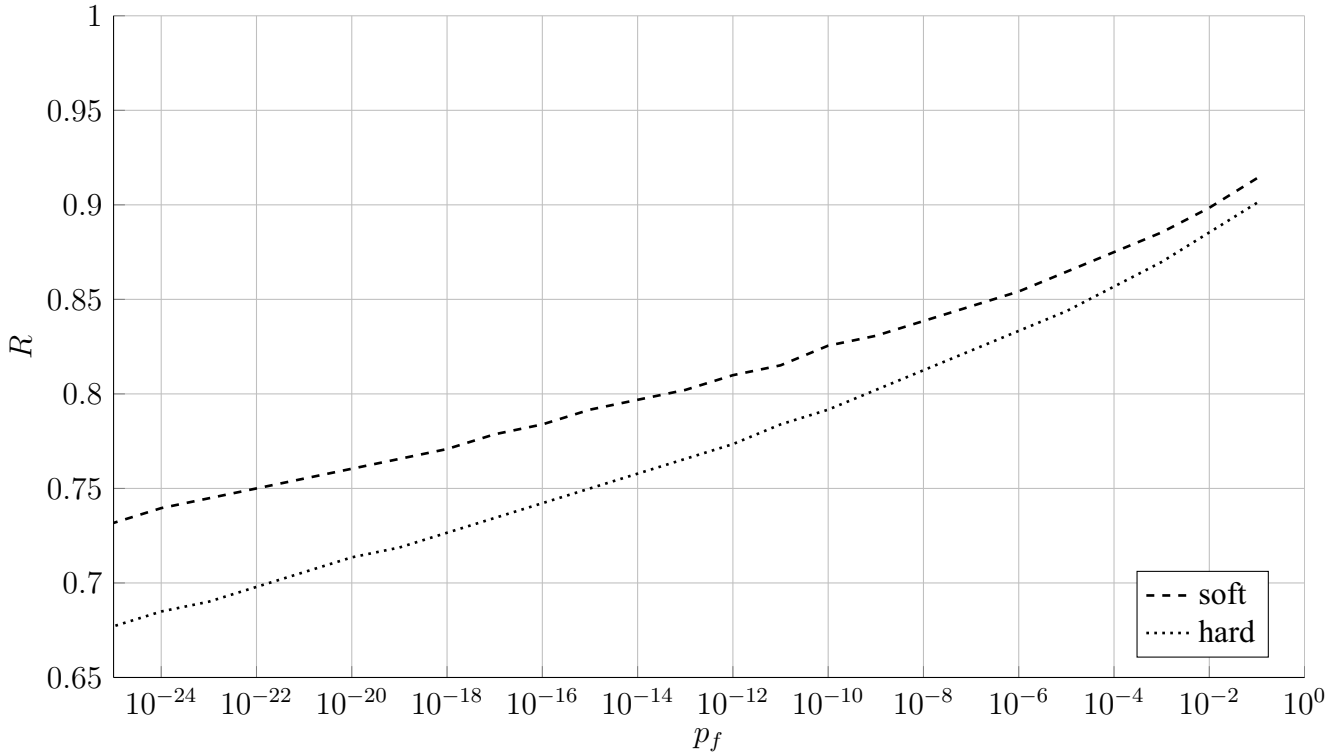


Рисунок 3.12. Зависимость между максимально достижимой скоростью кода и требуемой выходной вероятностью ошибки на блок для входной вероятности $E_s/N_0 = 15$ дБ ($p_s = 0.0178$) для ОЛО-кодов с жестким декодированием и кодов с мягким декодированием внутренних кодов

граница точна. Верхняя граница на вероятность неправильного декодирования кода, имеющего такую же скорость, но с конструкцией, оптимизированной для жесткого декодирования, показана штриховой линией. Видно, что она хуже, чем у кода с мягким декодированием, но ведет себя значительно лучше, чем верхняя граница на вероятность неправильного декодирования первого кода жестким алгоритмом. Также представлены результаты моделирования двух МПП-кодов Галлагера [2]. Оба кода являются регулярными: вес каждого столбца для первого кода равен $l = 3$, а для второго — $l = 4$. Скорости кодов равны $R = 0.8$.

Аналогичные результаты для ОЛО-кода длины 6144 представлены на рис. 3.14.

Полученные результаты позволяют сделать вывод, что предложенная конструкция ОЛО-кода вместе с методом ее декодирования может быть эффективно использована для достижения очень малой выходной вероятности ошибки, то есть в каналах с достаточно низкой вероятностью ошибки.

3.4 Выводы к главе

– В разделе 3.2 предложено несколько конструкций двоичных кодов с обобщённой локализацией ошибок, в которых в качестве внешних кодов используются коды с малой плотностью проверок.Полученные ОЛО-коды могут рассматриваться как коды с малой плотностью проверок, проверочная матрица которых может быть получена из матриц внутреннего

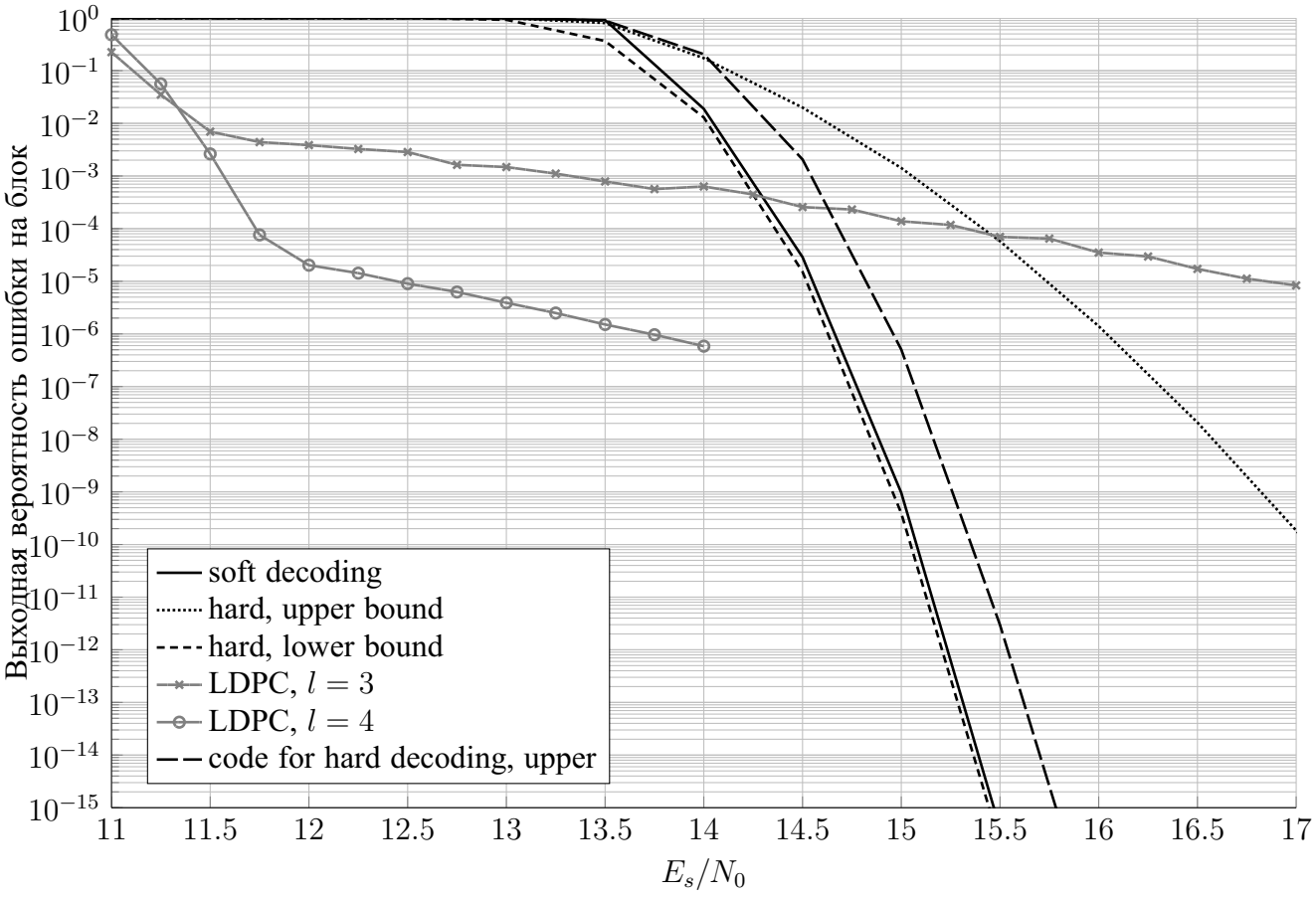


Рисунок 3.13. Зависимость между вероятностью ошибки на блок (FER) от входного отношения сигнал/шум E_s/N_0 для предложенных кодов $n = 4096$ с мягким внутренним декодированием (сплошная линия) $R = 0.8$, верхняя и нижняя оценки вероятности ошибки ОЛО-кодов, предложенных в [10], результаты моделирования для МПП-кодов Галлагера при числе единиц в столбце $l \in \{3, 4\}$, $R = 0.8$

и внешнего кодов ОЛО-кода, что позволяет их декодировать стандартными алгоритмами, разработанными для МПП-кодов.

– Для этих конструкций построены нижние границы на кодовое расстояние. Стоит отметить, что кодовое расстояние всех этих конструкций растёт пропорционально длине кода.

– Предложен мягкий алгоритм декодирования этих кодов, основанный на классическом жёстком алгоритме декодирования ОЛО-кодов, но использующий в вычислениях мягкие решения. Кроме этого, предложен гибридный алгоритм, совмещающий мягкий каскадный декодер и алгоритм “распространения доверия”. Оба этих алгоритма выигрывают у алгоритма “распространения доверия” при низких требуемых выходных вероятностях ошибки.

– В разделе 3.3 предложен новый метод декодирования ОЛО-кодов, в котором короткие внутренние коды декодируются мягко по максимуму правдоподобия, а внешние — с использованием декодера по минимальному расстоянию. В главе приведен метод выбора минимально-достаточных избыточностей внешних кодов так, чтобы гарантировать заданную вероятность ошибки для заданного отношения сигнал/шум.

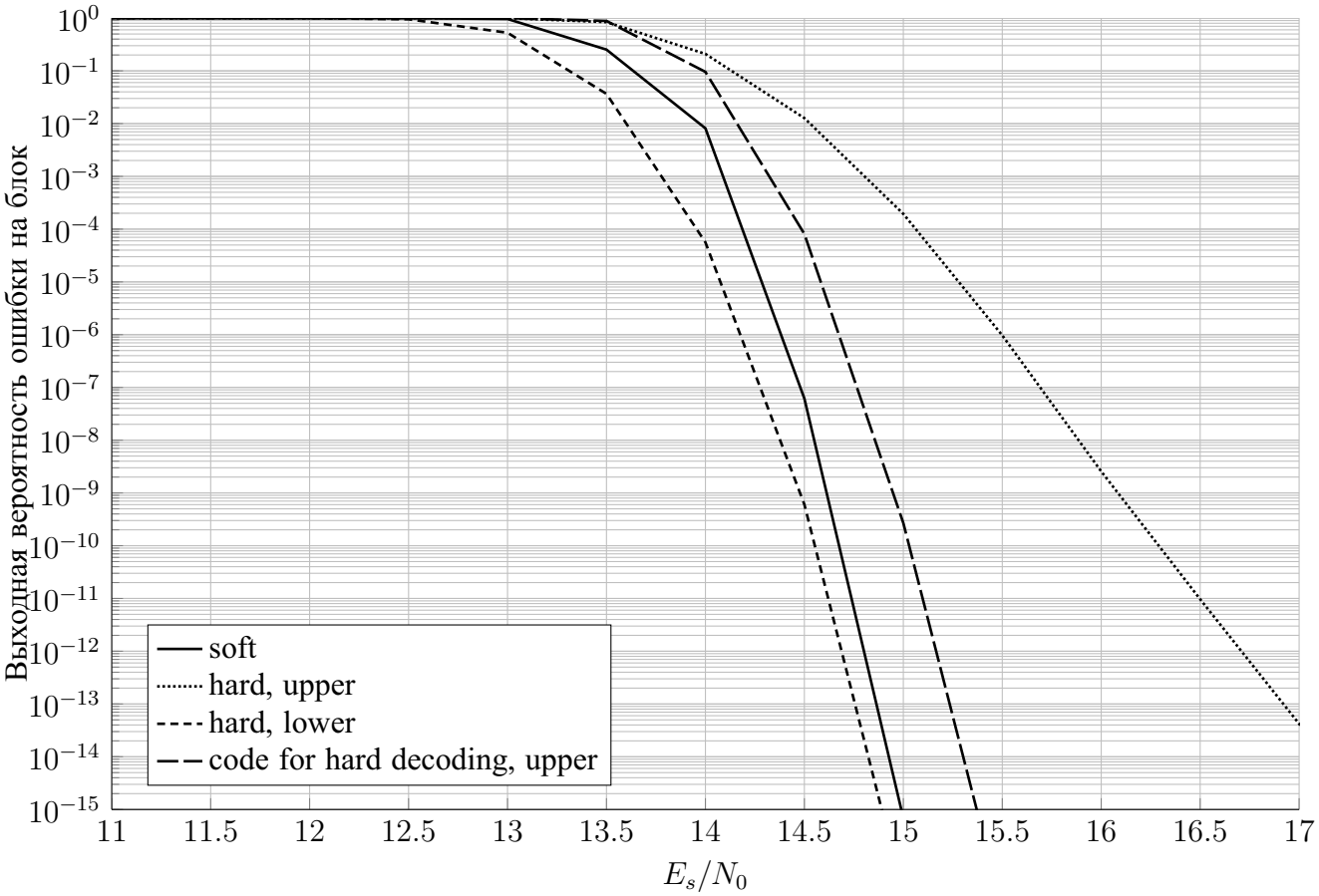


Рисунок 3.14. Зависимость между вероятностью ошибки на блок (FER) от входного отношения сигнал/шум E_s/N_0 для предложенных кодов $n = 6144$ с мягким внутренним декодированием (сплошная линия) $R = 0.8$, верхняя и нижняя оценки вероятности ошибки ОЛО-кодов, предложенных в [10]

– Полученные численные результаты позволяют сделать вывод о том, что коды, построенные предложенным методом, имеют меньшую избыточность (большую скорость) чем коды, оптимизированные для жесткого декодирования. В частности, эти коды позволяют работать при таких соотношениях сигнал/шум, где код для жёсткого декодирования не может быть построен.

– Возможность оценить сверху вероятность неправильного декодирования рассматриваемых кодов позволяет избежать их компьютерного моделирования, что существенно увеличивает скорость разработки современных систем связи.

– Можно отметить, что рассмотренные в разделе 3.3 конструкции имеют одни и те же алгоритмы кодирования и декодирования, независимо от их скорости, а изменение скорости кодовой конструкции производится благодаря перестройке параметров кодеров и декодеров внешних кодов.

Заключение

В настоящей работе:

– Предложен способ векторизации алгоритма “распространения доверия” для q-ичных МПП-кодов.

– Разработан метод применения алгоритма декодирования “распространения доверия” с мягким входом для каналов с жёстким решением.

– Произведено сравнение различных алгоритмов декодирования МПП-кодов с единичной памятью с циклическим замыканием.

– Предложен метод выбора структуры ОЛО-кода и оптимизации скоростей внешних кодов, обеспечивающий наибольшую возможною скорость ОЛО-кода при условии, что вероятность неправильного декодирования на кодовое слово не превышает заданную при заданной входной вероятности ошибки на символ.

– Предложена нижняя граница на вероятность неправильного декодирования ОЛО-кода.

– Предложена конструкция ОЛО-кодов, обеспечивающая энергетический выигрыш больше, чем применяемый в существующих ВОЛС код БЧХ и пр этом обладающая меньшей сложностью реализации.

– Построены ОЛО-коды с использованием МПП-кодов в качестве внешних. Показано, что эти коды в свою очередь являются МПП-кодами. Для них разработан алгоритм мягкого декодирования, аналогичный алгоритму декодирования ОЛО-кодов.

– Предложен алгоритм мягкого декодирования ОЛО-кодов, основанных на кодах Рида-Соломона. Для него предложена верхняя граница на вероятность неправильного декодирования, для которой применим метод из п. 3.4.

Список литературы

1. *Polyanskiy Yury, Poor H Vincent, Verdú Sergio*. Channel coding rate in the finite blocklength regime // *Information Theory, IEEE Transactions on*. — 2010. — Vol. 56, no. 5. — Pp. 2307–2359.

2. *Gallager R. G.* Low-density parity-check codes: Ph.D. thesis. — 1963. — 90 pp. http://web.mit.edu/gallager/www/pages/ldpc.pdf.

3. *Wolf J, Elspas B*. Error-locating codes — A new concept in error control // *Information Theory, IEEE Transactions on*. — 1963. — Vol. 9, no. 2. — Pp. 113–117.

4. *Wolf Jack K.* On an extended class of error-locating codes // *Information and Control*. — 1965. — Vol. 8, no. 2. — Pp. 163–169.

5. *Зяблов В. В.* Новая трактовка кодов для локализации ошибок, их корректирующие свойства и алгоритмы декодирования // *Сб. Передача дискретных сообщений по каналам с группирующимися ошибками*. — 1972. — С. 8–18.

6. *Блох Э. Л., Зяблов В. В.* Обобщенные каскадные коды: Алгебраическая теория и сложность реализации. — М.: Связь, 1976. — 240 с.

7. *Блох Э. Л., Зяблов В. В.* Линейные каскадные коды. — М.: Наука, 1982. — 229 с.

8. *Maucher Johannes, Zyablov Victor V, Bossert Martin*. On the equivalence of generalized concatenated codes and generalized error location codes // *Information Theory, IEEE Transactions on*. — 2000. — Vol. 46, no. 2. — Pp. 642–649.

9. *Zhilin I., Zyablov V.* LDPC code construction as a generalized concatenated code // Problems of Redundancy in Information and Control Systems (REDUNDANCY), 2014 XIV International Symposium on / IEEE. — 2014. — June. — Pp. 107–110.

10. *Zhilin I. V., Kreshchuk A. A., Zyablov V. V.* Generalized Error-Locating Codes and Minimization of Redundancy for Specified Input and Output Error Probabilities // *Journal of Communications Technology and Electronics*. — 2015. — Vol. 60, no. 6. — Pp. 695–706.

11. *Жилин И. В., Иванов Ф. И., Зяблов В. В.* Обобщенные коды с локализацией ошибок с мягким декодированием внутренних кодов // *Информационные процессы*. — 2015. — Т. 15, № 2. — С. 111–127.

12. *Жилин И. В., Иванов Ф. И., Зяблов В. В.* Распараллеливание вычислений при декодировании недвоичных кодов с малой плотностью проверок // Сборник трудов 36-й конференции молодых ученых и специалистов “Информационные технологии и системы”. — 2013. — Sep. — С. 121–125.

13. On the Decoding of Tail-Biting UM-LDPC Codes / Igor Zhilin, Pavel Rybin, Fedor Ivanov, Victor Zyablov // Fourteenth International Workshop on Algebraic and Combinatorial Coding Theory (ACCT-XIV). — 2014. — Sep.

14. *Zhilin Igor, Rybin Pavel, Zyablov Victor*. High-Rate Codes for High-Reliability Data Transmission // 2015 IEEE International Symposium on Information Theory (ISIT 2015) / IEEE. — 2015. — Jun.

15. *Tanner R. M.* A Recursive Approach to Low Complexity Codes // *IEEE Trans. Inform. Theory*. — 1981. — Sep. — Vol. 27, no. 5. — Pp. 533–547.

16. *Зяблов Виктор Васильевич, Пинскер Марк Семенович*. Оценка сложности исправления ошибок низкоплотностными кодами Галлагера // *Проблемы передачи информации*. — 1975. — Т. 11, № 1. — С. 23–36.

17. *MacKay David JC*. Good error-correcting codes based on very sparse matrices // *Information Theory, IEEE Transactions on*. — 1999. — Vol. 45, no. 2. — Pp. 399–431.

18. *Davey Hllatthew C, MacKay David JC*. Low density parity check codes over GF (q) // Information Theory Workshop, 1998 / IEEE. — 1998. — Pp. 70–71.

19. О корректирующей способности кодов с малой плотностью проверок на четность / Камиль Шамильевич Зигангиров, Али Емре Пусане, Дмитрий Камильевич Зигангиров, Даниэль Дж Костелло // *Проблемы передачи информации*. — 2008. — Т. 44, № 3. — С. 50–62.

20. *Зяблов Виктор Васильевич, Рыбин Павел Сергеевич*. Исправление стираний кодами с малой плотностью проверок // *Проблемы передачи информации*. — 2009. — Т. 45, № 3. — С. 15–32.

21. *Зяблов Виктор Васильевич, Рыбин Павел Сергеевич*. Анализ связи свойств МПП-кодов и графа Таннера // *Проблемы передачи информации*. — 2012. — Т. 48, № 4. — С. 3–29.

22. О минимальном расстоянии низкоплотностных кодов с проверочными матрицами, составленными из перестановочных матриц / Арвинд Шридхаран, Михаэль Лентмайер, Дмитрий Владимирович Трухачев и др. // *Проблемы передачи информации*. — 2005. — Т. 41, № 1. — С. 39–52.

23. *Rybin P., Zyablov V.* Asymptotic estimation of error fraction corrected by binary LDPC code // Proc. IEEE Int. Symposium on Inform. Theory. — 2011. — aug. — Pp. 351 –355.

24. *Фролов Алексей Андреевич, Зяблов Виктор Васильевич*. Границы минимального кодового расстояния для недвоичных кодов на двудольных графах // *Проблемы передачи информации*. — 2011. — Т. 47, № 4. — С. 27–42.

25. *Declercq David, Fossorier Marc*. Decoding algorithms for nonbinary LDPC codes over GF // *Communications, IEEE Transactions on*. — 2007. — Vol. 55, no. 4. — Pp. 633–643.

26. Constructions of nonbinary quasi-cyclic ldpc codes: A finite field approach / Lingqi Zeng, Lan Lan, Ying Y Tai et al. // *Communications, IEEE Transactions on*. — 2008. — Vol. 56, no. 4. — Pp. 545–554.

27. Algebraic constructions of nonbinary quasi-cyclic LDPC codes: Array masking and dispersion / Shu Lin, Shumei Song, Bo Zhou et al. // Proc. 9th International Symposium on Communication Theory and Applications (ISCTA) / Citeseer. — 2007.

28. *Carrasco Rolando Antonio, Johnston Martin*. Non-binary error control coding for wireless communication and data storage. — John Wiley & Sons, 2008.

29. *Зяблов В. В. Рыбин П. С. Фролов А. А.* Алгоритм декодирования с вводом стираний для МПП-кодов, построенных над полем GF(q) // *Информационно-управляющие системы*. — 2011. — С. 62–68.

30. *Фролов Алексей Андреевич, Зяблов Виктор Васильевич*. Асимптотическая оценка доли ошибок, исправляемых q-ичными МПП-кодами // *Проблемы передачи информации*. — 2010. — Т. 46, № 2. — С. 47–65.

31. *MacKay David J. C.* Information Theory, Inference, and Learning Algorithms. — Cambridge University Press, 2005. — 640 pp.

32. *nan Lee Lin*. Short unit-memory byte-oriented binary convolutional codes having maximal free distance (Corresp.) // *IEEE Trans. Inform. Theory*. — 1976. — May. — Vol. 22, no. 3. — Pp. 349–352.

33. *Zyablov V., Sidorenko V.* On periodic (partial) unit memory codes with maximum free distance // Error Control, Cryptology, and Speech Compression / Ed. by Andrew Chmora, StephenB. Wicker. — Springer Berlin Heidelberg, 1994. — Vol. 829 of *Lecture Notes in Computer Science*. — Pp. 74–79.

34. *Justesen J.* Bounded distance decoding of unit memory codes // *Information Theory, IEEE Transactions on*. — 1993. — Sep. — Vol. 39, no. 5. — Pp. 1616–1627.

35. *Dettmar U., Sorger U.K.* New optimal partial unit memory codes based on extended BCH codes // *Electronics Letters*. — 1993. — Nov. — Vol. 29, no. 23. — Pp. 2024–2025.

36. *Kondrashov K., Zyablov V.* On the lower bound of the free distance of partial unit memory codes based on LDPC Codes // Proc. IEEE Int. Symposium on Inform. Theory. — 2011. — July. — Pp. 1831–1835.

37. *И. Иванов Ф., В. Жилин И., В. Зяблов В.* Алгоритм декодирования кодов с малой плотностью проверок на четность с большим распараллеливанием // *Информационно-управляющие системы*. — 2012. — № 6 (61). — С. 49–55.

38. Using GEL Codes for Optical Channel. — 2009. — May.

39. *Некучаев А. О., Зяблов В. В.* Проект «Континент»—новый подход для передачи данных по магистральным ВОЛС. — 2008.

40. Low-complexity GEL codes for digital magnetic storage systems / Achim Fahrner, Helmut Grießer, Robert Klarer, Victor V Zyablov // *Magnetics, IEEE Transactions on*. — 2004. — Vol. 40, no. 4. — Pp. 3093–3095.

41. *Lewis David, Corbeil Sacha, Mason Beck*. 400G DMT PMD for 2km SMF. http://www.ieee802.org/3/bs/public/14_05/lewis_3bs_01_0514.pdf.

42. *Frolov A., Zyablov V.* Asymptotic estimation of the fraction of errors correctable by q-ary LDPC codes // *Problems of Information Transmission*. — 2010. — Vol. 46, no. 2. — Pp. 142–159.

43. *Зяблов Виктор Васильевич, Сидоренко Владимир Рэмович*. Границы сложности декодирования линейных блоковых кодов с помощью решеток // *Проблемы передачи информации*. — 1993. — Т. 29, № 3. — С. 3–9.

44. *Зяблов Виктор Васильевич, Потапов Владимир Георгиевич, Сидоренко Владимир Рэмович*. Декодирование в список максимального правдоподобия с помощью кодовой решетки // *Проблемы передачи информации*. — 1993. — Т. 29, № 4. — С. 3–10.

Список рисунков

Список таблиц