Федеральное государственное автономное образовательное учреждение высшего образования
«Нижегородский государственный университет им. Н.И. Лобачевского»
Программа повышение конкурентоспособности ННГУ им. Н.И. Лобачевского
Стратегическая инициатива 7 «Достижение лидирующих позиций в области суперкомпьютерных технологий и высокопроизводительных вычислений»

«Основы помехоустойчивого кодирования»

Пособие предназначено для студентов V курса радиофизического факультета ННГУ им. Н.И. Лобачевского, обучающихся по специальности 090302 «Информационная безопасность телекоммуникационных систем», специализация «Системы подвижной цифровой помехозащищенной связи»

Раздел: Современные помехоустойчивые коды и высокопроизводительные алгоритмы их декодирования

Лекция №1. Сверточные коды

Введение

В последние годы в связи с бурным развитием высокоскоростного Интернета и высокопроизводительной электроники возрастает потребность в построении надежных систем передачи и хранения данных. Одной из основных задач в таких системах является эффективная и безошибочная передача и хранение информации. Долгое время считалось, что наличие шума в канале связи препятствует безошибочной передачи данных. Однако в 1948 году Шеннон опубликовал основополагающую работу теории информации, показывающую возможность передачи данных без ошибок в зашумленных каналах связи. Шеннон показал, что любой канал связи имеет некоторую пропускную способность C. При этом, если скорость передачи данных R не превышает пропускную способность C, то возможно добиться безошибочной передачи информации. Исправление ошибок, возникающих в канале, достигается с помощью схем помехоустойчивого (или канального) кодирования. Стоит отметить, что теорема Шеннона доказывает лишь существование таких кодов, однако не объясняет, как такие коды строить и как декодировать. Теорема Шеннона послужила отправной точкой к построению теории помехоустойчивого кодирования рассматривающей широкий круг задач разработки и анализа схем исправления ошибок.

Не смотря на большие усилия, долгое время существенного прогресса в построении эффективных кодов, позволяющих приблизится к границе Шеннона, не было. В 1970-х годах были найдены некоторые классы кодов с эффективными методами декодирования. Однако помехоустойчивость таких кодов была достаточно далека от границы Шеннона. Несмотря на это, в 1980-х годах в связи с развитием микроэлектроники схемы помехоустойчивого кодирования стали активно внедряться в различные системы передачи данных. Сейчас помехоустойчивое кодирование является неотъемлемой частью любого современного устройства передачи или хранения данных. Стоит отметить, что существенным прорывом в теории кодирования является изобретение кодов с малой плотностью проверки на четность и турбо кодов, позволяющие передавать информацию со скоростью близкой к пропускной способности канала.

Глава 1. Сверточные коды

1.1. Понятие сверточного кодирования

Питер Элиас впервые предложил использовать сверточные коды в 1955 году как альтернативу блоковым кодам. В 1963 году Мессей предложил практическую пороговую схему декодирования сверточных кодов. Однако предложенный Мессей алгоритм декодирования не обеспечивал достаточную помехоустойчивость. В 1967 году Витерби предложил оптимальный алгоритм декодирования сверточных кодов по критерию максимума правдоподобия. В силу относительно невысокой вычислительной сложности алгоритм Витерби и сверточное кодирование получили широкое применение в практических задачах передачи информации. В настоящий момент сверточное кодирование является неотъемлемой частью современных стандартов радиосвязи.

Основным отличием сверточных от блоковых кодов является то, что n выходных кодовых символов зависят не только от текущего информационного блока длины k бит, но и от m предыдущих информационных блоков. Общая структура (n,k,m) сверточного кода показана на Рис. 1. Сверточное кодирование может быть реализовано с помощью линейной цепи имеющей k входов, n выходов и памяти размерности m. Как правило, n и k являются небольшими целыми числами, удовлетворяющие неравенству $n > k$. В то же время, для обеспечения необходимой помехоустойчивости размерность памяти m должна быть существенно больше параметров n и k.

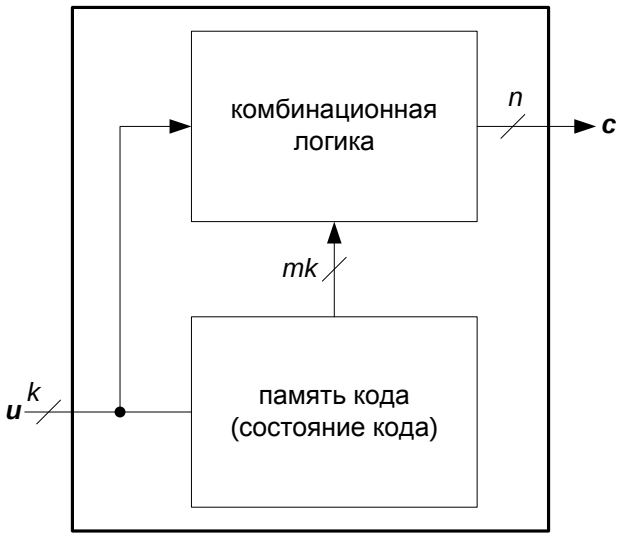

Рис. 1 Структура сверточного кода

Пример сверточного (2,1,2) кода приведен на Рис. 2

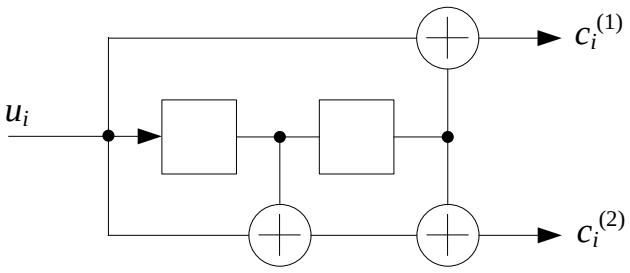

Рис. 2. Пример (2,1,2) сверточного кода

1.2. Методы представления сверточных кодов

Сверточные коды могут быть реализованы с помощью линейных цепей, использующих сумматоры и регистры памяти. В этом случае код может быть описан с помощью, так называемой, *диаграммы состояний*. Узлы в диаграмме представляют собой состояния кода, задаваемые содержимым регистров сдвига, а ребра описывают переход кода из одного состояния в другое. Пусть для (n,k,m) сверточного кода параметр K_i определяет число элементов памяти для входа i, тогда $K = \sum_{i=1}^{k} K_i$ задает суммарную память сверточного кодера. Тогда полное состояние кода в момент времени l задается вектором

$$\left(u_{l-1}^{(1)} \quad u_{l-2}^{(1)} \quad \cdots \quad u_{l-K_1}^{(1)} \quad u_{l-1}^{(2)} \quad u_{l-2}^{(2)} \quad \cdots \quad u_{l-K_2}^{(1)} \quad \cdots \quad u_{l-1}^{(k)} \quad u_{l-2}^{(k)} \quad \cdots \quad u_{l-K_k}^{(k)}\right),$$

(**Ошибка! Текст указанного стиля в документе отсутствует.**.1)

а общее число возможных состояний кода будет равно 2^K. Памятью сверточного кода называется величина, $m = \max_{1 \le i \le k} (K_i)$, а длиной *кодового ограничения*[1]

$$\nu = n \cdot \left(\max_{1 \le i \le k} K_i + 1 \right)$$ (**Ошибка! Текст указанного стиля в документе отсутствует.**.2)

Каждый информационный блок длины k бит приводит к переходу кода из одного стояния в другое. В этом случае общее число ребер выходящих из каждого состояния равно 2^k. Для удобства описания каждое ребро в диаграмме состояний помечается k входными информационными битами $\left(u_l^{(1)} \quad u_l^{(2)} \quad \cdots \quad u_l^{(k)}\right)$ и n выходными кодовыми битами $\left(c_l^{(1)} \quad c_l^{(2)} \quad \cdots \quad c_l^{(n)}\right)$. Состояния кода, как правило, обозначаются, как $S_0, S_1, ..., S_{2^K-1}$, где индекс i состояния S_i соответствует десятичному представлению содержимого регистров сдвига кода. Для каждой входной информационной или кодовой последовательностей, в диаграмме состояний может быть проложен некоторый путь, проходящий через ее узлы.

Пример диаграммы состояний для рассматриваемого (2,1,2) сверточного кода показан на Рис. 3.

[1] На практике используется несколько определений длины кодового ограничения. Другим способом определения длины кодового ограничения является суммарная память кода *K*.

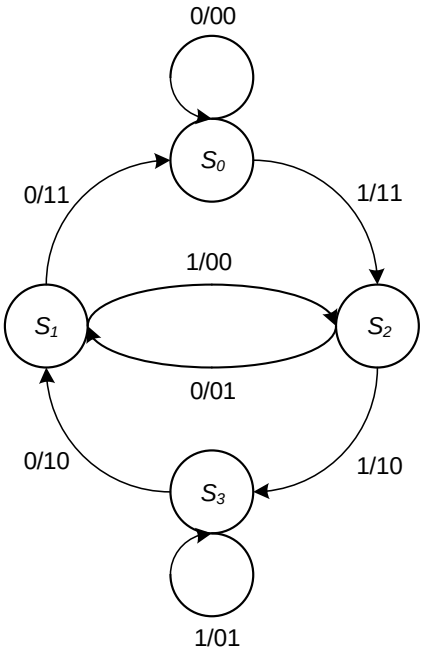

Рис. 3. Диаграмма состояния (2,1,2) сверточного кода

Помимо описания сверточного кода диаграмма состояний может быть также использована для анализа его структуры. В частности, с помощью диаграммы состояний можно определить число и вес всех ненулевых кодовых последовательностей (т.н. распределение веса кода). Для этого диаграмма состояний модифицируется следующим образом: удаляется переход из нулевого стояния в нулевое, а нулевое состояние разделяется на начальное и конечное состояния. Каждое ребро помечается X^i, где i число ненулевых бит на выходе кодера см. Рис. 4.

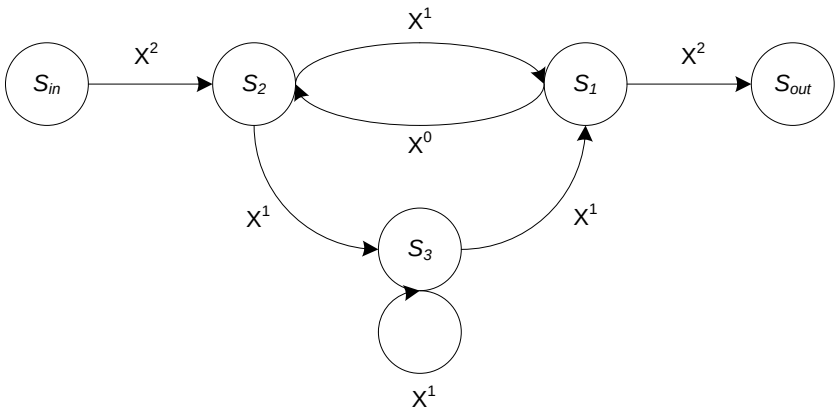

Рис. 4. Модифицированная диаграмма состояний (2,1,2) кода

Тогда путь, соединяющий начальное и конечные состояния, представляет собой ненулевую кодовую последовательность выходящую и входящую в нулевое состояние кода. При этом кодовые последовательности, проходящие через нулевое состояние более

одного раза, могут быть рассмотрены как несколько более коротких последовательностей, начинающихся и заканчивающихся в нулевом состоянии.

Уравнения, определяющие переходы в модифицированной диаграмме состояний, определяются следующим образом

$$\begin{aligned} S_2 &= S_1 + X^2 S_{in} \\ S_1 &= XS_2 + XS_3 \\ S_3 &= XS_2 + XS_3 \\ S_{out} &= X^2 S_1 \end{aligned}, \qquad \textbf{(Ошибка! Текст указанного стиля в документе отсутствует.}.3)$$

Распределение веса кода может быть получено из порождающей функции модифицированной диаграммы состояний следующим образом

$$T(X) = \frac{S_{out}}{S_{in}} = \sum_i A_i X^i = \frac{X^5}{1-2X} = X^5 + 2X^6 + 4X^7 + 8X^8 + \ldots . \qquad \textbf{(Ошибка! Текст указанного стиля в документе отсутствует.}.4)$$

Множество весов ненулевых кодовых последовательностей определяется множеством степеней полинома $T(X)$, а число путей веса i коэффициентом A_i. Например, для рассматриваемого (2,1,2) сверточного минимальный вес ненулевой кодовой последовательности равен 5 и число таких путей равно 1.

Распределение веса (вес кодовых последовательностей их число) определяет помехоустойчивость сверточного кода. Так из распределения веса кода можно вычислить *свободное расстояние кода*, определяемое выражением

$$d_{free} = \min_{\mathbf{c}_i, \mathbf{c}_j, \mathbf{c}_i \neq \mathbf{c}_j} \left(d_H(\mathbf{c}_i, \mathbf{c}_j) \right). \qquad \textbf{(Ошибка! Текст указанного стиля в документе отсутствует.}.5)$$

В силу линейности сверточных кодов свободное расстояние кода может быть также получено как минимальный вес ненулевой кодовой последовательности

$$d_{free} = \min_{\mathbf{c}_i, \mathbf{c}_i \neq \mathbf{0}} \left(w_H(\mathbf{c}_i) \right). \qquad \textbf{(Ошибка! Текст указанного стиля в документе отсутствует.}.6)$$

Тогда для последовательностей большой длины свободное расстояние кода d_{free} равно минимальной степени слагаемого порождающей функции (1.2.4).

Другим способом представления сверточных кодов является *решетчатая диаграмма*. Решетчатая диаграмма может быть получена путем разворота диаграммы состояний кода во времени. Пример решетчатой диаграммы для рассматриваемого (2,1,2) сверточного

кода показан на Рис. 5. Начальное состояние кода в решетчатой диаграмме соответствует нулевому состоянию кода.

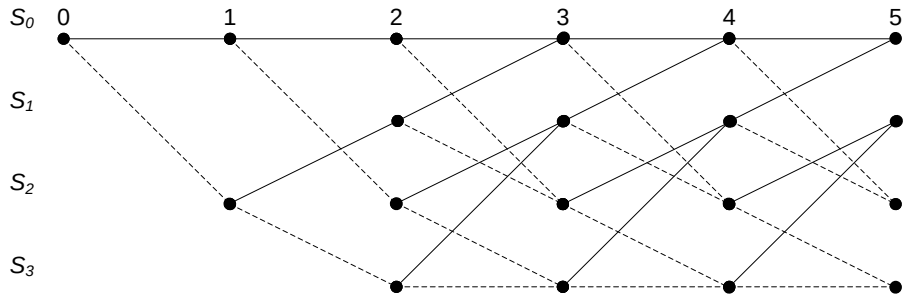

Рис. 5. Решетчатая диаграмма (2,1,2) сверточного кода

Сплошной линией в решетчатой диаграмме показаны ребра, соответствующие нулевому информационному биту, а пунктирной линией ребра соответствующие единичному информационному биту. Отметим, что число разрешенных состояний в решетке удваивается каждый момент времени и достигает максимального значения $2^K = 4$ в момент времени $l = 2$. Далее для $l > 2$ структура решетки начинает повторяться. Процедура кодирования может быть представлена как прокладывание пути в решетчатой диаграмме. Рис. 6 иллюстрирует процедуру кодирования информационной последовательности $\mathbf{u} = (1,0,1,1,1)$.

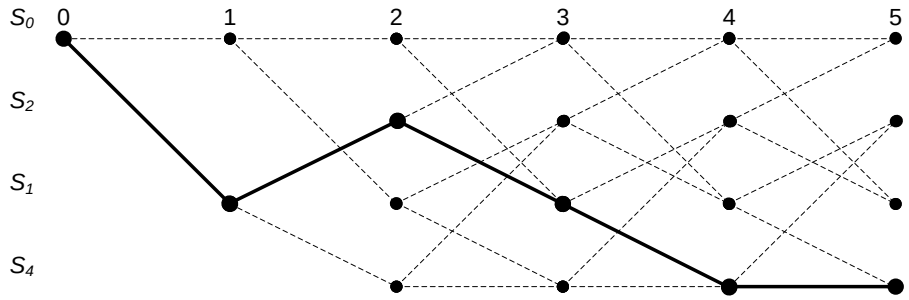

Рис. 6. Пример кодирования для последовательности $\mathbf{u} = (1,0,1,1,1)$

Сплошной линией показан путь в решетке соответствующей информационной последовательности, а пунктирной оставшиеся пути решетки.

1.3. Матричное описание процедуры сверточного кодирования

Процедура кодирования сверточных кодов может быть также описана с помощью матричных операций. В силу линейности цепи кодирования, последовательность на каждом выходе j сверточного кода может быть представлена как свертка информационной последовательности $\mathbf{u}$ и импульсной характеристики линейной цепи $\mathbf{g}^{(j)}$ для $(n,1,m)$ кода. Импульсная характеристика $\mathbf{g}^{(j)}$ называется *порождающей последовательностью* кода. Заметим, что множество $\left\{\mathbf{g}^{(j)}\right\}$ полностью определяет сверточный код. Тогда для случая $(n,1.m)$ кода процедуру кодирования можно представить следующим образом

$$c_l^{(j)} = \sum_{i=0}^{m} u_{l-i} g_i^{(j)} \qquad \textbf{(Ошибка! Текст указанного стиля в документе отсутствует.}.7)$$

Для каждого момента времени выходы сверточного кода, как правило, объединяются в общий поток, т.е.

$$\left(c_0^{(1)} \quad c_0^{(2)} \quad \cdots \quad c_0^{(n)} \quad c_1^{(1)} \quad \cdots \quad c_1^{(n)} \quad \cdots\right), \qquad \textbf{(Ошибка! Текст указанного стиля в документе отсутствует.}.8)$$

тогда порождающая матрица сверточного кода будет иметь следующую структуру

$$\mathbf{G} = \begin{pmatrix} g_0^{(1)} \ldots g_0^{(n)} & \cdots & g_m^{(1)} \ldots g_m^{(n)} & & & \\ & g_0^{(1)} \ldots g_0^{(n)} & \cdots & g_m^{(1)} \ldots g_m^{(n)} & & \\ & & g_0^{(1)} \ldots g_0^{(n)} & \cdots & g_m^{(1)} \ldots g_m^{(n)} & \\ & & & \ddots & & \ddots \end{pmatrix}, \qquad \textbf{(Ошибка! Текст указанного стиля в документе отсутствует.}.9)$$

где пустые области матрицы $\mathbf{G}$ равны 0. Для кодовой последовательности длины L, матрица $\mathbf{G}$ имеет L строк и $n \cdot (L+m)$ столбцов. Процедура сверточного кодирования при этом записывается следующим образом

$$\mathbf{c} = \mathbf{uG} \qquad \textbf{(Ошибка! Текст указанного стиля в документе отсутствует.}.10)$$

Рассмотрим пример сверточного кодирования для (2,1,2) кода и информационной последовательности $\mathbf{u} = (1,0,1,1,1,1)$. Порождающие последовательности кода равны (1,0,1) и (1,1,1). Поражающая матрица этого кода равна

$$\mathbf{G}=\begin{pmatrix} 11 & 01 & 11 & & & & & \\ & 11 & 01 & 11 & & & & \\ & & 11 & 01 & 11 & & & \\ & & & 11 & 01 & 11 & & \\ & & & & 11 & 01 & 11 & \\ & & & & & 11 & 01 & 11 \end{pmatrix}.$$

Для информационной последовательности $\mathbf{u}=(1,0,1,1,1,1)$ с помощью выражения (1.3.4) получим кодовую последовательность $c=(11,01,00,10,01,01,10,11)$.

Обобщим порождающую матрицу для случая произвольного $(n,k.m)$ сверточного кода

$$\mathbf{G}=\begin{pmatrix} \mathbf{G}_0 & \cdots & \mathbf{G}_m & & & \\ & \mathbf{G}_0 & \cdots & \mathbf{G}_m & & \\ & & \mathbf{G}_0 & \cdots & \mathbf{G}_m & \\ & & & \ddots & & \ddots \end{pmatrix}, \qquad \textbf{(Ошибка! Текст указанного стиля в документе отсутствует.}.11)$$

где матрица $\mathbf{G}_l$ определяется выражением

$$\mathbf{G}_l=\begin{pmatrix} g_{1,l}^{(1)} & g_{1,l}^{(2)} & \cdots & g_{1,l}^{(n)} \\ g_{2,l}^{(1)} & g_{2,l}^{(2)} & \cdots & g_{2,l}^{(n)} \\ \vdots & & & \vdots \\ g_{k,l}^{(1)} & g_{k,l}^{(2)} & \cdots & g_{k,l}^{(n)} \end{pmatrix}. \qquad \textbf{(Ошибка! Текст указанного стиля в документе отсутствует.}.12)$$

При этом информационная последовательность на выходе кода имеет следующую структуру

$$\mathbf{u}=(\mathbf{u}_0 \quad \mathbf{u}_1 \quad \cdots)=\left(u_0^{(1)}u_0^{(2)}\cdots u_0^{(k)} \quad u_1^{(1)}u_1^{(2)}\cdots u_1^{(k)} \quad \cdots\right) \quad . \qquad \textbf{(Ошибка! Текст указанного стиля в документе отсутствует.}.13)$$

1.4. Полиномиальное описание сверточного кодирования

В линейных системах каждую последовательность можно заменить некоторым полиномом, а операцию свертки на более удобную операцию умножения полиномов. Например, для (2,1,m) кода процедура кодирования запишется следующим образом

$$\begin{aligned} c^{(1)}(D) &= u(D)\cdot g_1(D) \\ c^{(2)}(D) &= u(D)\cdot g_2(D) \end{aligned}, \qquad \textbf{(Ошибка! Текст указанного стиля в документе отсутствует.}.14)$$

где $u(D) = u_0 + u_1 D + u_2 D^2 + \cdots$ информационный полином, $g^{(1)}(D) = g_0^{(1)} + g_1^{(1)} D + g_2^{(1)} D^2 + \cdots + g_m^{(1)} D^m$ и $g^{(2)}(D) = g_0^{(2)} + g_1^{(2)} D + g_2^{(2)} D^2 + \cdots + g_m^{(2)} D^m$ порождающие полиномы, $c^{(1)}(D) = c_0^{(1)} + c_1^{(1)} D + c_2^{(1)} D^2 + \ldots$ и $c^{(2)}(D) = c_0^{(2)} + c_1^{(2)} D + c_2^{(2)} D^2 + \ldots$ кодовые полиномы. Общая кодовая последовательность на выходе

$$c(D) = c^{(1)}\left(D^2\right) + D\cdot c^{(2)}\left(D^2\right). \qquad \textbf{(Ошибка! Текст указанного стиля в документе отсутствует.}.15)$$

Обобщим полиномиальное описание процедуры кодирования на случай $(n, k.m)$ сверточного кода. Пусть $U(D) = \left(u^{(1)}(D) \;\; u^{(2)}(D) \;\; \cdots \;\; u^{(k)}(D)\right)$ набор полиномов соответствующих k входным информационным последовательностям, $C(D) = \left(c^{(1)}(D) \;\; c^{(2)}(D) \;\; \cdots \;\; c^{(k)}(D)\right)$ набор полиномов соответствующих n выходным кодовым последовательностям, а $g_i^{(j)}(D)$ порождающий полином соответствующий входу i и входу j сверточного кода. Тогда процедуру кодирования можно записать следующим образом

$$C(D) = U(D)G(D), \qquad \textbf{(Ошибка! Текст указанного стиля в документе отсутствует.}.16)$$

где

$$G(D) = \begin{pmatrix} g_1^{(1)}(D) & g_1^{(2)}(D) & \cdots & g_1^{(n)}(D) \\ g_2^{(1)}(D) & g_2^{(2)}(D) & \cdots & g_2^{(n)}(D) \\ \vdots & & & \vdots \\ g_k^{(1)}(D) & g_k^{(2)}(D) & \cdots & g_k^{(n)}(D) \end{pmatrix}. \qquad \textbf{(Ошибка! Текст указанного стиля в документе отсутствует.}.17)$$

Например, для (2,1,7) сверточного кода, представленного на Рис. 7, поражающий полином запишется следующим образом

$$G(D) = \left(1 + D^2 + D^3 + D^5 + D^6 \quad 1 + D + D^2 + D^3 + D^6\right)$$

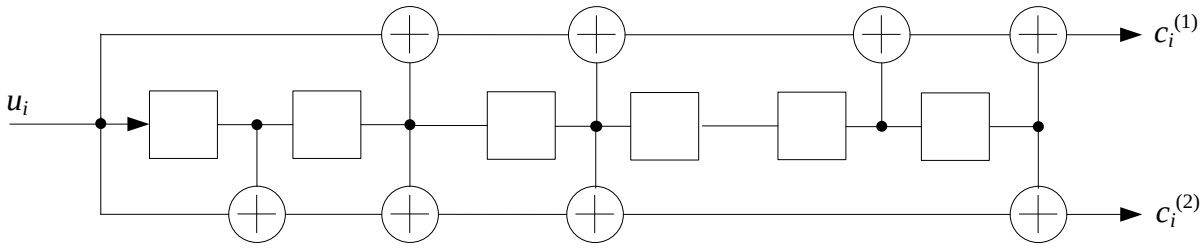

Рис. 7. Схема (2,1,7) сверточного кода

Код, представленный на Рис. 7 получил широкое применение в различных стандартах связи, таких как IEEE 802.11a и IEEE 802.16e.

1.5. Систематические сверточные коды

Сверточный код называют систематическим, если входная информационная последовательность непосредственно поступает на один из выходов кодера без изменения. Систематические коды имеют некоторое преимущество при декодировании, т.к. информационная последовательность может быть получена непосредственно из принятой последовательности. Заметим, что несистематический сверточный код может быть приведен к систематическому виду следующим образом

$$c(D) = g^{(1)}\left(D^2\right) \cdot u\left(D^2\right) \cdot \left(1 + D \cdot \frac{g^{(2)}\left(D^2\right)}{g^{(1)}\left(D^2\right)}\right) = \tilde{u}\left(D^2\right) \cdot \left(1 + D \cdot \frac{g^{(2)}\left(D^2\right)}{g^{(1)}\left(D^2\right)}\right) \qquad \textbf{(Ошибка!}$$

Текст указанного стиля в документе отсутствует..18)

Из выражения (1.5.1), легко видеть, что множество кодовых последовательностей кода с порождающими полиномами $\left\{g^{(1)}(D), g^{(2)}(D)\right\}$ и полиномами $\left\{1, \frac{g^{(2)}(D)}{g^{(1)}(D)}\right\}$ идентичны, т.е. сверточные коды $\left\{g^{(1)}(D), g^{(2)}(D)\right\}$ и $\left\{1, \frac{g^{(2)}(D)}{g^{(1)}(D)}\right\}$ эквивалентны. При этом изменяется отображение информационных последовательностей в кодовые, а код становится рекурсивным.

Пример рекурсивного систематического сверточного (2,1,2) кода показан на Рис. 8

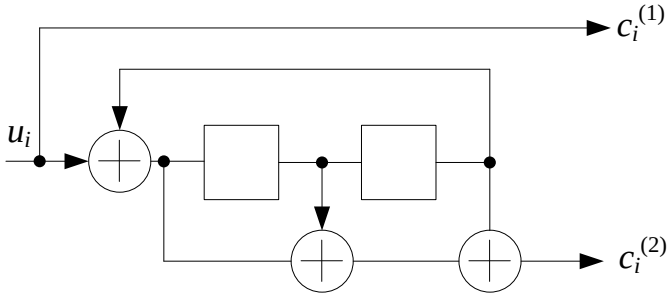

Рис. 8. Рекурсивный систематический (2,1,2) сверточный код

Сверточный код называется *катастрофическим*, если он отображает последовательность с бесконечным весом в кодовую последовательность конечного веса. Очевидно, что использовать в качестве помехоустойчивого кода такие схемы нельзя, т.к. конечное число ошибок в кодовом слове приведет к бесконечному числу ошибок в информационном слове. Оказывается, что необходимым и достаточным условием того, чтобы сверточный код $G(D)$ был некатастрофическим, является существование некоторого преобразования $G^{-1}(D)$, удовлетворяющего условию $G^{-1}(D)G(D)=1$.

1.6. Процедура выкалывания сверточных кодов

Для практических приложений для контроля скорости передачи и помехоустойчивости очень важным является использование различных скоростей кода. При этом для упрощения процедуры кодирования и декодирования требуется сохранение структуры кода. Процедура выкалывания является эффективным способом достижения этих двух целей и заключается в исключении (выкалывании) определенных кодовых бит из передаваемой последовательности. Правило исключения кодовых бит задается с помощью, так называемого, *шаблона выкалывания*. Шаблон выкалывания представляет собой матрицу с числом строк равным числу выходов кода n и числом столбцов равным периоду выкалывания. Ноль в шаблоне выкалывания обозначает, что кодовый бит исключается из передачи. Пример процедуры выкалывания показан на Рис. 9 для шаблона выкалывания $\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$.

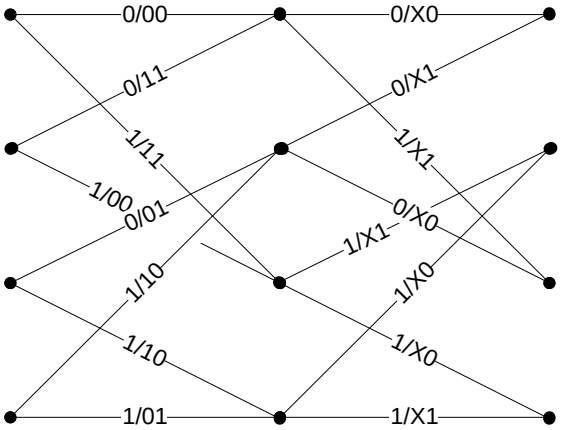

Рис. 9. Процедура выкалывания (2,1,2) сверточного кода

Исключенные из передачи биты, помечены на ребрах решетки символом X. При этом правило выкалывания кодовых бит повторяется каждые два сегмента решетки (входных информационных бит). Для рассматриваемого примера для каждых двух информационных бит на выходе кода получается три кодовых. В этом случае скорость кода равна 2/3. Заметим, что структура решетки кода (число состояний, переходы между состояний), полученная с помощью процедуры выкалывания, остается такой же, как и для исходного кода без выкалывания. Сохранение структуры кода позволяет существенно упростить процедуру декодирования особенно для высоких скоростей кода, т.к. решетка сверточного кода для $k > 1$, как правило, является более сложной.

В качестве примера рассмотрим структуру решетки для (3,2,2) кода с аналогичной скоростью кодирования и порождающей матрицей кода вида

$$G(D) = \begin{pmatrix} 1+D & 1+D & 1 \\ 0 & D & 1+D \end{pmatrix}.$$

Решетчатая диаграмма для такого кода показана на Рис. 10.

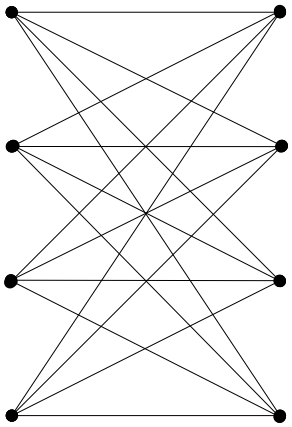

Рис. 10 Сегмент решетки (3,2,2) сверточного кода

Легко видеть, что структура кода (3,2,2) существенно сложнее структуры кода (2,1,2) с выкалыванием. Так, число ребер выходящих из каждого состояния для (3,2,2) кода в два раза больше чем для (2,1,2) кода с выкалыванием. Усложнение структуры, как правило, приводит к усложнению процедуры декодирования.

Другие примеры шаблонов выкалывания для сверточного кода

$$G(D) = \left(1 + D^2 + D^3 + D^5 + D^6 \quad 1 + D + D^2 + D^3 + D^6\right)$$

приведены в Таблице 8.

Таблица 1. Шаблоны выкалывания

Скорость кодирования	Шаблон выкалывания
2/3	$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$
3/4	$\begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$
4/5	$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$
5/6	$\begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix}$
7/8	$\begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$

Заметим, что процедура выкалывания сверточных кодов очень часто применяется в стандартах связи, поддерживающих сверточное кодирование для исправления ошибок. Так с помощью шаблонов выкалывания приведенных в Таблице 8 достигают скорости кода 1/2, 2/3 и 3/4 в стандартах связи IEEE 802.11a (Wi-Fi), IEEE 802.16e (WiMAX)

1.7. Завершение кодирования для сверточных кодов

В силу конечности информационной последовательности на практике применятся процедура завершения кодирования, возвращающая код в определенное состояние. Существуют два метода завершения кодирования сверточных кодов – метод с нулевым

конечным состоянием и метод с циклической структурой. Для метода с нулевым конечным состоянием к концу информационной последовательности добавляются биты возвращающие код в нулевое состояние. Данный метод приводит к небольшому снижению скорости кода, связанной с передачей неинформационных добавочных бит. Решетка кода с процедурой завершения кодирования в нулевое состояние показана на Рис. 11.

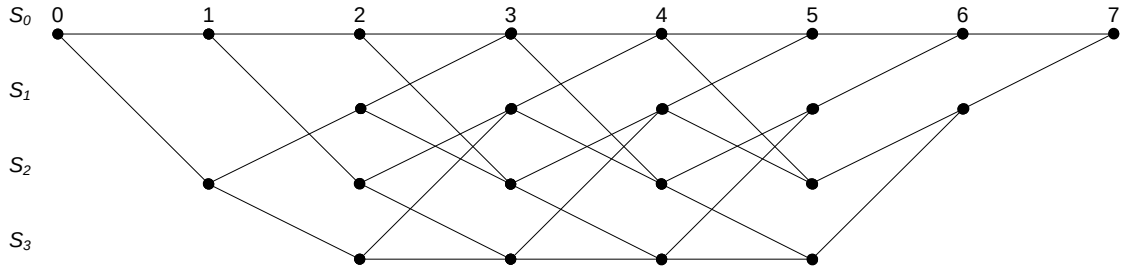

Рис. 11. Решетка кода с завершением кодирования в нулевое состояние

Другой метод с циклической структурой сверточного обеспечивает совпадение начального и конечного состояния кода в решетке при помощи специальной инициализации начального состояния в схеме кодирования. Поскольку инициализационные биты для такого метода зависят от информационной последовательности, начальное и конечное состояния кода заранее не известны на приемнике. Это приводит к некоторому усложнению процедуры декодирования сверточных кодов с циклической структурой, связанное с поиском начального и конечного состояний кода. Стоит отметить, что метод с циклической структурой кода не приводит к снижению скорости кода, что особенно важно для передачи информационных последовательностей малой длины.

Лекция № 2. Сверточные коды (продолжение)

1.8. Декодирование сверточных кодов

В общем случае процедура декодирования сверточных кодов состоит из двух этапов, показанных показана на Рис. 12. На первом этапе, на основании принятой последовательности ***r*** происходит выбор кодовой последовательности. На втором по выбранной кодовой происходит восстановление информационной последовательности.

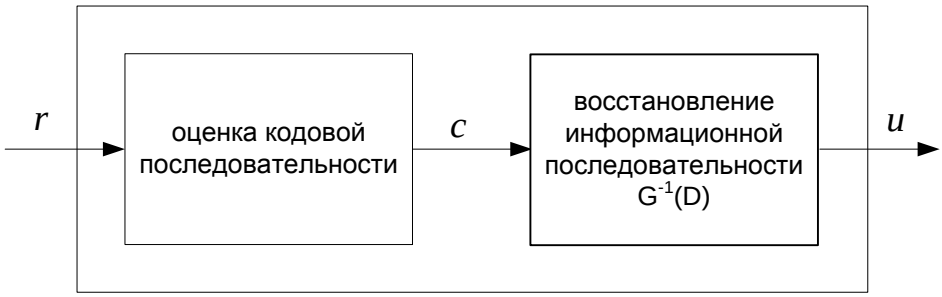

Рис. 12 Процедура декодирования сверточных кодов

Так для поиска кодовой последовательности может применяться алгоритм Витерби, осуществляющий оценку переданной последовательности по критерию максимума правдоподобия. Процедура декодирования включает поиск наиболее правдоподобной последовательности в решетке на основании принятой последовательности $\mathbf{r}$. В зависимости от используемого для поиска расстояния декодирование бывает с мягкими или жесткими решениями.

Предположим, что имеется информационная последовательность $\mathbf{u}=(\mathbf{u}_0,\mathbf{u}_1,\cdots,\mathbf{u}_{L-1})$, длины $k\cdot L$ информационных бит. Каждый информационный символ $\mathbf{u}_l$ состоит из k бит $\mathbf{u}_l=\left(u_l^{(0)},u_l^{(1)},\cdots,u_l^{(k-1)}\right)$. Предположим, что для кодирования используется (n,k,m) сверточный код. Тогда соответствующая кодовая последовательность $\mathbf{c}=(\mathbf{c}_0,\mathbf{c}_1,\cdots,\mathbf{c}_{L+m-1})$ состоит из $L+m$ символов и имеет длину $N=n\cdot(L+m)$ бит. Декодер, осуществляющий поиск кодовой последовательности по критерию максимума правдоподобия, должен найти путь в решетке кода обеспечивающий максимум вероятности

$$\hat{\mathbf{c}} = \arg\max_{\mathbf{c}} \{\mathrm{P}(\mathbf{r} \mid \mathbf{c})\},$$ (**Ошибка! Текст указанного стиля в документе отсутствует.**.19)

где $\mathbf{r} = (\mathbf{r}_0, \mathbf{r}_1, \cdots, \mathbf{r}_{L+m-1})$ принятая последовательность. Для канала без памяти данная вероятность записывается следующим выражением

$$\mathrm{P}(\mathbf{r} \mid \mathbf{c}) = \prod_{i=0}^{L+m-1} \mathrm{P}(\mathbf{r}_i \mid \mathbf{c}_i).$$ (**Ошибка! Текст указанного стиля в документе отсутствует.**.20)

В силу монотонности логарифмической функции процедуру максимизации выражения (1.8.2) можно проводить в логарифмической области

$$\log\{\mathrm{P}(\mathbf{r} \mid \mathbf{c})\} = \sum_{i=0}^{L+m-1} \log\{\mathrm{P}(\mathbf{r}_i \mid \mathbf{c}_i)\},$$ (**Ошибка! Текст указанного стиля в документе отсутствует.**.21)

где $\mathrm{B}(\mathbf{r}_i \mid \mathbf{c}_i) = \log\{\mathrm{P}(\mathbf{r}_i \mid \mathbf{c}_i)\}$ называется *метрикой ребра*, определяемой суммой метрик бит.

$$\mathrm{B}(\mathbf{r}_i \mid \mathbf{c}_i) = \log\{\mathrm{P}(\mathbf{r}_i \mid \mathbf{c}_i)\} = \sum_{j=0}^{n-1} \log\left\{\mathrm{P}\left(r_i^{(j)} \mid c_i^{(j)}\right)\right\}$$ (**Ошибка! Текст указанного стиля в документе отсутствует.**.22)

Тогда *метрика пути* $\mathbf{c}$ определяется как сумма метрик ребер

$$\mathrm{M}(\mathbf{r} \mid \mathbf{c}) = \sum_{i=0}^{L+m-1} \log\{\mathrm{P}(\mathbf{r}_i \mid \mathbf{c}_i)\} = \sum_{i=0}^{N-1} \sum_{j=0}^{n-1} \log\left\{\mathrm{P}\left(r_i^{(j)} \mid c_i^{(j)}\right)\right\},$$ (**Ошибка! Текст указанного стиля в документе отсутствует.**.23)

а *частичная метрика пути* $\mathbf{c}$ определяется как частичная сумма метрик ребер до момента времени j

$$\mathrm{M}_j(\mathbf{r} \mid \mathbf{c}) = \sum_{i=0}^{j} \log\{\mathrm{P}(\mathbf{r}_i \mid \mathbf{c}_i)\}.$$ (**Ошибка! Текст указанного стиля в документе отсутствует.**.24)

1.9. Алгоритм декодирования Витерби

Алгоритм Витерби осуществляет оценку кодовой последовательности по критерию максимума правдоподобия. Для этого на каждом этапе декодирования для каждого состояния кода алгоритм производит поиск кодовой последовательности имеющей максимальную частичную метрику пути. При этом путь, имеющий максимальную метрику, сохраняется.

Алгоритм Витерби можно формально разбить на следующие этапы:

Инициализация

Предполагается, что в момент времени $i=0$, состояние кода известно. Тогда частичная метрика пути для каждого состояния инициализируется следующим образом

$$\mathrm{M}_0(s)=\begin{cases}0, & s=0\\ -\infty, & s\neq 0\end{cases}, \qquad \text{(Ошибка! Текст указанного стиля в документе отсутствует..25)}$$

где s состояние кода. Переходим к моменту времени $i=1$.

Суммирование

В момент времени i продолжаем все возможные пути, выходящие из всех состояний кода момента времени $i-1$. Вычисляем метрики продолженных путей путем суммирования частичной метрики пути состояния в предшествующий момент времени $\mathrm{M}_{i-1}(s)$ и метрики выходящего из этого состояния ребра.

Сравнение

Для каждого состояния s находим путь с наибольшей частичной метрикой. Этот путь называется *выжившим* для данного состояния в момент i.

Выбор

Определяем метрику частичного пути $\mathrm{M}_i(s)$ для каждого состояния равной метрике выжившего пути. Удаляем из решетчатой диаграммы все остальные пути. Увеличиваем i на единицу. Если $i<L+m$, то перейти к шагу *Суммирование*.

Восстановление

Двигаясь, по выжившему пути из состояния $s=0$ в состояние соответствующее моменту $i=L+m-1$ восстанавливаем исходное сообщение.

В силу принципа математической индукции, путь, найденный алгоритмом Витерби является максимально правдоподобным.

Вычислительная сложность алгоритма Витерби

Для двоичного сверточного кода, с полным числом регистров K, общее число состояний равно 2^K. Число ребер выходящих и входящих в каждое состояние равно 2^k. Тогда для каждого сегмента решетки имеется 2^{K+k} различных путей и 2^K выживших путей. Для каждого состояния на каждом этапе декодирования необходимо вычислить метрики всех продолженных 2^{K+k} путей и найти наилучшую метрику для каждого состояния, путем сравнения 2^k метрик. Легко видеть, что сложность вычислений декодирования экспоненциально возрастает с ростом числа сдвиговых регистров сверточного кода.

Алгоритм декодирования Витерби с жесткими решениями

Декодирование для двоичного симметричного канала называется декодированием с жесткими решениями, т.к. декодер получает на вход непосредственные оценки переданных бит без информации о надежности этих оценок. Для двоичного симметричного канала с вероятностью ошибки $p \le 1/2$ принятая последовательность является двоичной, тогда

$$\log\{\mathrm{P}(\mathbf{r}\,|\,\mathbf{c})\} = d(\mathbf{r},\mathbf{c})\cdot\log\left(\frac{p}{1-p}\right) + N\cdot\log(1-p), \qquad \textbf{(Ошибка! Текст}$$

указанного стиля в документе отсутствует..26)

где $d(\mathbf{r},\mathbf{c})$ расстояние Хэмминга между принятой и кодовой последовательностями. Поскольку $\log(p/(1-p)) < 0$, а второе слагаемое выражения (1.9.2) не зависит от последовательности $\mathbf{c}$, максимально правдоподобная оценка обеспечивается при минимуме расстояния Хэмминга

$$d(\mathbf{r},\mathbf{c}) = \sum_{i=0}^{L+m-1} d(\mathbf{r}_i,\mathbf{c}_i) = \sum_{i=0}^{N-1}\sum_{j=0}^{n-1} d\left(r_i^{(j)}, c_i^{(j)}\right). \qquad \textbf{(Ошибка! Текст}$$

указанного стиля в документе отсутствует..27)

Таким образом, метрика ребра и пути для двоично-симметричного канала вычисляется по расстоянию Хэмминга.

Пример декодирования алгоритма Витерби с жесткими решениями

Рассмотрим пример декодирования с помощью алгоритма Витерби. Пусть в кодовой последовательности (00, 11, 10, 10, 11, 00, 00) произошла ошибка во второй момент времени (на четвертом бите). Тогда принятая последовательность, поступающая на

устройство декодирования, равна (00, 10, 10, 10, 11, 00, 00). Декодирование Витерби на каждом этапе показано на Рис. 13. С правой стороны показаны частичные метрики пути для каждого состояния. На итерациях 1-2 алгоритм Витерби не производит выбор пути, т.к. число ребер входящих в каждое состояние равно 1. В силу завершения кодирования в нулевое состояние число разрешенных состояний в момент времени 6 и 7 равно 2 и 1 соответственно. На последней итерации декодер получает единственный путь, имеющий минимальное расстояние Хэмминга от принятой двоичной последовательности. Это расстояние равно 1, что соответствует числу ошибок произошедших на всей длине переданной последовательности.

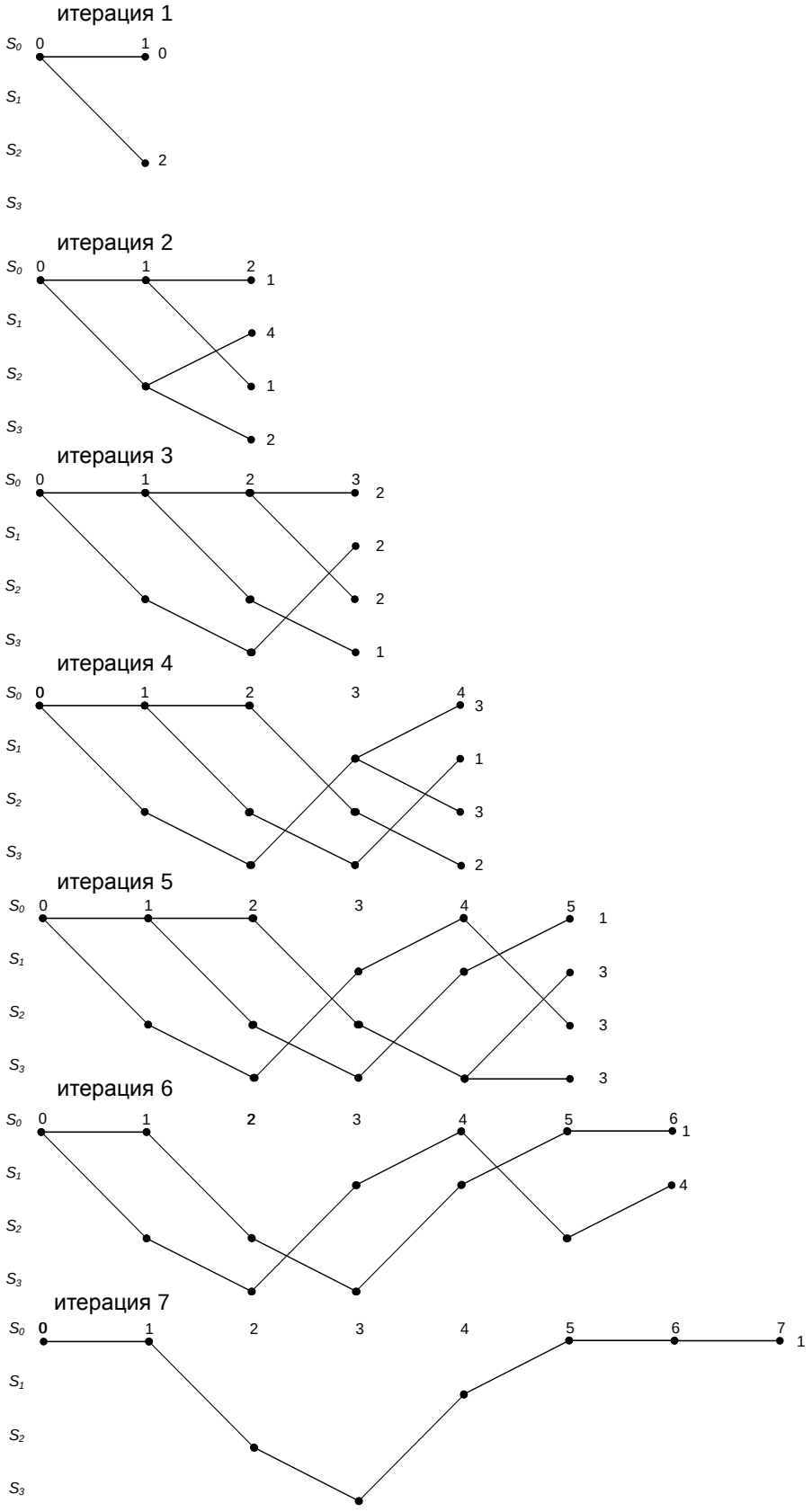

Рис. 13. Пример декодирования с помощью алгоритма Витерби

Алгоритм декодирования Витерби с мягкими решениями

Декодирование, где помимо оценки бит, передается информация о надежности, называется декодированием с мягкими решениями. Для примера декодирования с мягкими решениями рассмотрим двоичный канал с аддитивным белым гауссовским шумом **n**

$$\mathbf{r} = \mathbf{x} + \mathbf{n}, \qquad \text{(Ошибка! Текст указанного стиля в документе отсутствует..28)}$$

где **r** последовательность, полученная на приемнике. Для случая двоичной модуляции запишем соответствие между кодовым битом $c_i^{(j)}$ и выходом двоичного фазового модулятора $x_i^{(j)}$

$$x_i^{(j)} = \begin{cases} +1, & c_i^{(j)} = 1 \\ -1, & c_i^{(j)} = 0 \end{cases}. \qquad \text{(Ошибка! Текст указанного стиля в документе отсутствует..29)}$$

Несложно записать выражение для условной плотности вероятности $r_i^{(j)}$ при заданном символе $x_i^{(j)}$

$$p\left(r_i^{(j)} \mid x_i^{(j)}\right) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{ - \frac{\left(r_i^{(j)} - x_i^{(j)}\right)^2}{2\sigma^2} \right\}. \qquad \text{(Ошибка! Текст указанного стиля в документе отсутствует..30)}$$

Тогда метрика ребра определяется как квадрат расстояния между принятыми и кодовыми символами

$$B(\mathbf{r}_i, \mathbf{x}_i) = \sum_{j=0}^{n-1} \frac{\left(r_i^{(j)} - x_i^{(j)}\right)^2}{2\sigma^2}, \qquad \text{(Ошибка! Текст указанного стиля в документе отсутствует..31)}$$

Алгоритм декодирования на основе Евклидова расстояния находит кодовую последовательность ближайшую к принятой последовательности. Не сложно показать, что произведение $r_i^{(j)} x_i^{(j)}$ можно также использовать в качестве метрики.

1.10. Интерливинг

Интерливинг является одним из часто используемых способов повышения помехоустойчивости практических систем связи. Как правило, реальные каналы связи не являются каналами без памяти, т.е. возникающие ошибки не являются случайными, а

группируются в пакеты ошибок. Если число ошибок на длине кодового слова превышает исправляющую способность кода, то декодер не сможет правильно восстановить переданную последовательность. Процедура интерливинга позволяет решить проблему группировки ошибок путем перестановки кодовых символов на длине превышающей длину кодового слова. В результате на приемнике, после обратной перестановки (де-интерливинга), ошибки приходящие на схему декодирования становятся случайными.

Схемы интерливинга можно условно поделить на две группы: случайные и структурированные. Рассмотрим два примера структурированных перестановок. Первым способом перестановки является блоковый интерливинг. Для блокового интерливера кодовые биты перед модулятором записываются в матрицу размером $N_r \times N_c$ по столбцам, и считываются по строкам. Процедура блокового интерливинга показана на Рис. 14.

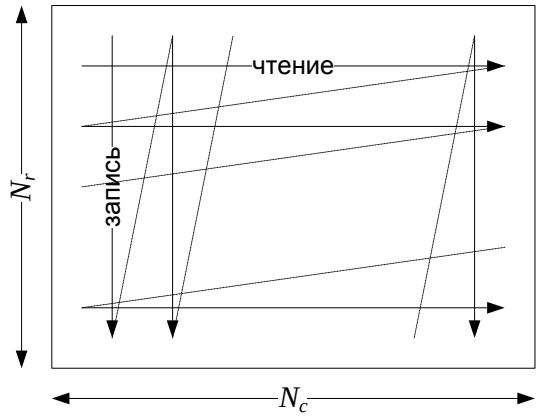

Рис. 14. Блоковый интерливер

Математически индексы перестановки для блокового интерливера могут быть получены с помощью выражения

$$i = N_c \cdot \mathrm{mod}(k, N_r) + \left\lfloor \frac{k}{N_r} \right\rfloor, \qquad \text{(Ошибка! Текст указанного стиля в документе отсутствует..32)}$$

где i индекс бита на выходе интерливинга, k индекс бита на входе интерливинга, $\lfloor \cdot \rfloor$ операция округления к меньшему целому числу. Процедура де-интерливинга на приемнике может быть осуществлена в обратном порядке – записью принятых бит по строкам и считыванием по столбцам.

Вторым способом перестановки является сверточный интерливинг. Структура такого интерливинга показана на Рис. 15

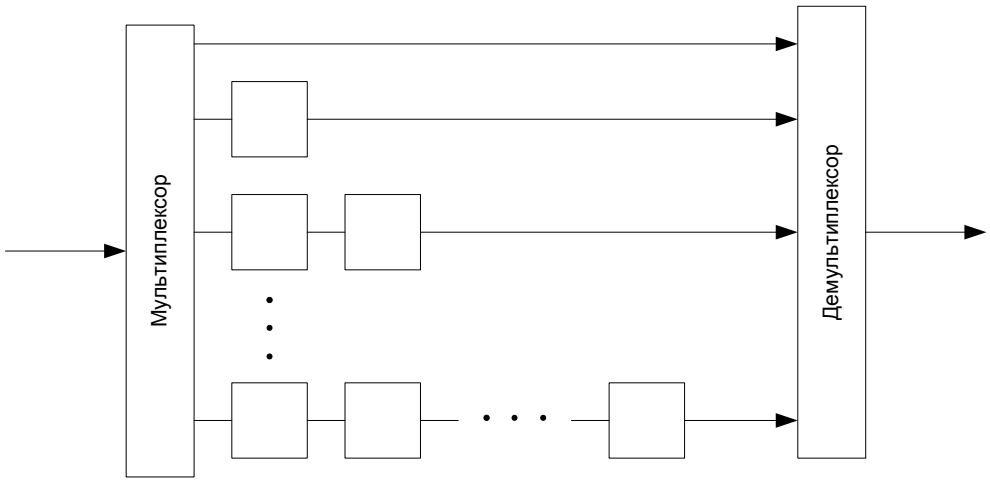

Рис. 15. Сверточный интерливер

Кодовая последовательность поступает на вход мультиплексора, разделяющего входную последовательность на B параллельных потока, которые, в свою очередь, поступают на линии задержки. При этом каждая линия задержки i содержит $i-1$ регистров сдвига. Общее число элементов памяти сверточного интерливера равно $\frac{B\cdot(B-1)}{2}$. Процедура сверточного де-интерливинга может быть выполнена с помощью аналогичной схемы, имеющей $B-i-1$ регистра памяти для линии i.

Использование битового интерливинга и алгоритма декодирования с мягкими решениями не позволяет применять стандартные подходы для вычисления метрик. В качестве метрики декодирования в таких схемах необходимо использовать метрику на бит. В качестве такой метрики, применяется логарифм отношения вероятностей, задаваемый следующим равенством

$$\mathrm{LLR}(c_i)=\log\left\{\frac{P(c_i=1\,|\,r_i)}{P(c_i=0\,|\,r_i)}\right\}=\log\left\{\frac{P(r_i\,|\,c_i=1)}{P(r_i\,|\,c_i=0)}\right\}. \qquad \textbf{(Ошибка! Текст указанного стиля в документе отсутствует.}.33)$$

Знак выражения (1.10.2) определяет «жесткое» решение о принятом бите, а модуль надежность такого решения. Можно показать, что оптимальный путь найденный с помощью метрики (1.10.2) совпадает с путем, найденным с помощью метрики (1.8.5). Для доказательства предположим, что это не так. Пусть $\mathbf{c}'$ и $\mathbf{c}''$ является путями, обеспечивающими максимум суммы метрик (1.8.5) $M^{(1)}(\cdot)$ и (1.10.2) $M^{(2)}(\cdot)$ соответственно. Заметим, что разница метрик для бита 1 и 0 одинакова для способа (1.9.7) и (1.10.2) с точностью до постоянного множителя, которым можно пренебречь, тогда

$$M^{(1)}(\mathbf{c}'')-M^{(1)}(\mathbf{c}')=M^{(2)}(\mathbf{c}'')-M^{(2)}(\mathbf{c}'). \qquad \textbf{(Ошибка! Текст}$$

указанного стиля в документе отсутствует..34)

Поскольку путь $\mathbf{c}'$ является оптимальным для метрики (1.8.5), то $M^{(1)}(\mathbf{c}'')-M^{(1)}(\mathbf{c}')<0$. Используя равенство (1.10.3) следует $M^{(2)}(\mathbf{c}'')-M^{(2)}(\mathbf{c}')<0$, что противоречит предположению, что $\mathbf{c}''$ является оптимальным для метрики $M^{(2)}(\cdot)$.

Вернемся к вычислению выражения (1.10.2). Используя правило Байеса, логарифм отношения вероятностей (1.10.2) может быть записан в следующем виде

$$\mathrm{LLR}(c_i)=\log\left\{\frac{\sum\limits_{x_i\in X^{(1)}}P(r_i\mid x_i)}{\sum\limits_{x_i\in X^{(0)}}P(r_i\mid x_i)}\right\}, \qquad \textbf{(Ошибка! Текст указанного}$$

стиля в документе отсутствует..35)

где суммирование в числителе и знаменателе ведется по всем сигнальным точкам созвездия соответствующим биту $c_i=1$ (множество $X^{(1)}$) или $c_i=0$ (множество $X^{(0)}$) соответственно. Пример разделения точек 16-QAM сигнального созвездия на два множества $X^{(1)}$ и $X^{(0)}$ для первого бита, приведен на Рис. 16.

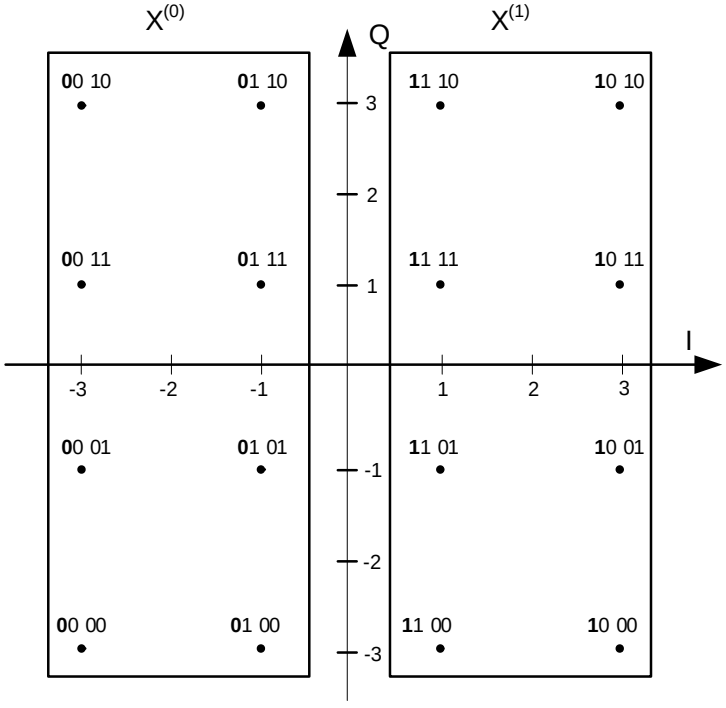

Рис. 16. Разделение точек 16-QAM сигнального созвездия на множества $X^{(1)}$ и $X^{(0)}$

Используя аппроксимацию $\log\left\{\sum\limits_i a_i\right\}\approx\max\left(\log(a_i)\right)$, получим

$$\mathrm{LLR}(c_i) = \max_{X^{(1)}} \{\log P(r_i \mid x_i)\} - \max_{X^{(0)}} \{\log P(r_i \mid x_i)\}. \quad \textbf{(Ошибка! Текст указанного стиля в документе отсутствует.}.36)$$

Для канала с аддитивным белым гауссовским шумом

$$\mathrm{LLR}(c_i) = \frac{1}{2\sigma^2}\left\{\min_{X^{(0)}} |r_i - x_i|^2 - \min_{X^{(1)}} |r_i - x_i|^2\right\}. \quad \textbf{(Ошибка! Текст указанного стиля в документе отсутствует.}.37)$$

Выражение (1.10.6) задает общий подход к расчету метрик. Дальнейшие упрощения возможны при рассмотрении конкретных примеров сигнальных созвездий. Так для созвездия 16-QAM, логарифм отношения правдоподобия может быть вычислен с помощью

$$\mathrm{LLR}(c_i) = \begin{cases} \mathrm{Re}(r_i), & |\mathrm{Re}(r_i)| \le 2 \\ 2\cdot(\mathrm{Re}(r_i) - 1), & \mathrm{Re}(r_i) > 2 \\ 2\cdot(\mathrm{Re}(r_i) + 1), & \mathrm{Re}(r_i) < -2 \end{cases}, \quad \textbf{(Ошибка! Текст указанного стиля в документе отсутствует.}.38)$$

для первого бита и с помощью

$$\mathrm{LLR}(c_i) = -2|\mathrm{Re}(r_i)| + 1, \quad \textbf{(Ошибка! Текст указанного стиля в документе отсутствует.}.39)$$

для второго. Заменяя операцию взятия действительной части $\mathrm{Re}(\cdot)$ на операцию взятия мнимой части $\mathrm{Im}(\cdot)$, аналогичным способом могут быть вычислены метрики для третьего и четвертого бита. График отображения принятого символа в метрику бита $\mathrm{LLR}(c_i)$ для 16-QAM модуляции показана на Рис. 17.

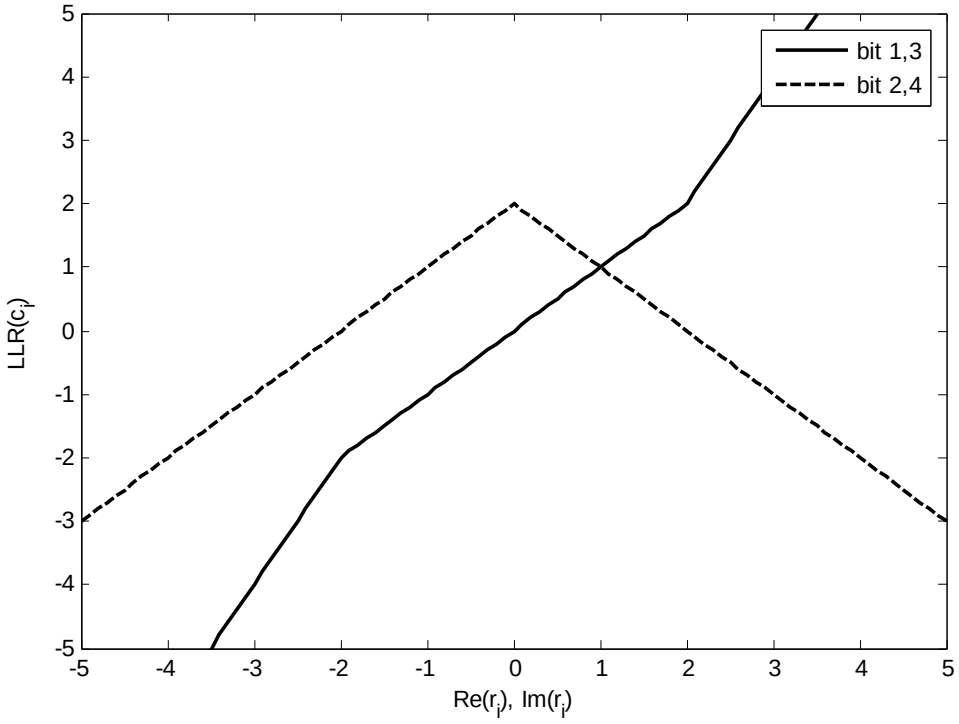

Рис. 17. Функция отображения $\mathrm{LLR}(c_i)$ для 16-QAM модуляции

Лекция № 3. Турбо коды

Глава 2. Турбо коды

2.1. Алгоритм декодирования по критерию максимума апостериорной вероятности

Изучение турбо кодов мы начнем с рассмотрения алгоритма декодирования сверточных кодов по критерию максимума апостериорной вероятности (MAB). Этот алгоритм был предложен в 1974 году. В литературе алгоритм MAB также получил названия BCJR (по первым буквам фамилий авторов Bahl, Cocke, Jelinek и Raviv) или «прямой-обратный» алгоритм. Стоит отметить, что в силу высокой вычислительной сложности, алгоритм MAB долгое время не использовался в практических приложениях. Однако, ситуация существенно изменилась после изобретения в 1993 году группой математиков К. Берроу, А. Главьё и П. Ситимашимой нового класса кодов – турбо кодов. Для декодирования предложенного ими кода был использован модифицированный BCJR алгоритм. После этого различные модификации MAB алгоритма стали активно внедряться в различные практические приложения декодирования турбо кодов.

Пусть $\mathbf{x} = \mathbf{x}_0, \mathbf{x}_2, \ldots, \mathbf{x}_{N-1}$ кодовая последовательность, состоящая из N символов длины n бит, полученная с помощью сверточного кодирования. Символ $\mathbf{x}_k$ соответствует кодовой последовательности, полученной сверточным кодом в момент времени k. Для

простоты рассмотрения будем считать что -1 и +1 соответствуют 0 и 1 соответственно. Тогда информационная последовательность u_k может принимать значения -1 или +1 и имеет априорную вероятность $P(u_k)$. Предположим, что кодовая последовательность **x** была передана в канале с аддитивным белым гауссовским шумом. Тогда принятая последовательность $\mathbf{r} = \mathbf{r}_0, \mathbf{r}_2, \ldots, \mathbf{r}_{N-1}$ может быть представленная в следующем виде

$$\mathbf{r} = \mathbf{x} + \mathbf{n}, \qquad \text{(Ошибка! Текст указанного стиля в документе отсутствует..40)}$$

где **n** реализация гауссовского шума. Для декодирования по критерию MAB принятая последовательность **r** используется для вычисления логарифма отношения вероятностей

$$L(u_k \mid \mathbf{r}) = \log \frac{P(u_k = +1 \mid \mathbf{r})}{P(u_k = -1 \mid \mathbf{r})}, \qquad \text{(Ошибка! Текст указанного стиля в документе отсутствует..41)}$$

где числитель и знаменатель в логарифме выражения (2.1.2) представляет собой апостериорную вероятность при условии принятой последовательности **r**. Информация, содержащаяся в $L(u_k \mid \mathbf{r})$, может быть использована как для принятия решения о преданном информационного бите, так и предана на следующий блок декодирования в качестве априорной информации.

Для удобства рассмотрим решетку сверточного кода со скоростью $R = 1/2$, $n = 2$ и двумя регистрами сдвига. Число состояний кода M в этом случае равно 4. Пример сегмента решетки кода представлен на Рис. 18

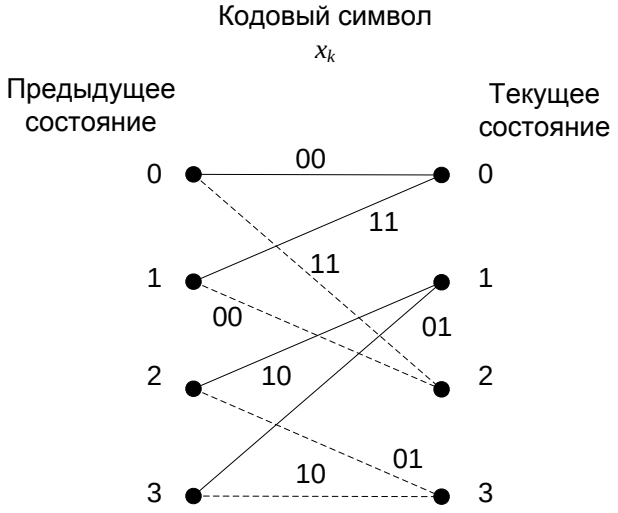

Рис. 18. Сегмент решетки сверточного кода

Напомним, что узлы обозначают возможные состояния кода, сплошные линии показывают ребра решетки соответствующие информационному биту -1, а пунктирные, ребра соответствующие информационному биту +1. Рассмотрим сегмент решетки в момент времени k. Состояние в текущий момент времени k обозначим как $S_k = s$, а в предыдущий $k-1$ как $S_{k-1} = s'$. Поскольку при заданном состоянии информационный бит однозначно определяет переход кода из одного состояния решетки в другое выражение (2.1.2) может быть записано в следующем виде

$$L(u_k \mid \mathbf{r}) = \log \frac{P(u_k = +1 \mid \mathbf{r})}{P(u_k = -1 \mid \mathbf{r})} = \log \frac{\sum\limits_{R_1} P(s', s \mid \mathbf{r})}{\sum\limits_{R_0} P(s', s \mid \mathbf{r})}, \quad \textbf{(Ошибка! Текст}$$

указанного стиля в документе отсутствует..42)

где R_1 и R_0 задают множество переходов (ребер в решетке) с начальным состоянием s' и конечным s, соответствующее информационным битам $u_k = +1$ и $u_k = -1$. Выражение (2.1.3) можно переписать в следующем виде

$$L(u_k \mid \mathbf{r}) = \log \frac{\sum\limits_{R_1} P(s', s \mid \mathbf{r}) P(\mathbf{r})}{\sum\limits_{R_0} P(s', s \mid \mathbf{r}) P(\mathbf{r})} = \log \frac{\sum\limits_{R_1} P(s', s, \mathbf{r})}{\sum\limits_{R_0} P(s', s, \mathbf{r})} \quad \textbf{(Ошибка! Текст}$$

указанного стиля в документе отсутствует..43)

Для дальнейшего упрощения выражения (2.1.4) принятую последовательность $\mathbf{r}$ можно условно разделить на три сегмента

$$\mathbf{r} = \underbrace{\mathbf{r}_0, \mathbf{r}_2, ..., \mathbf{r}_{k-1}}_{\mathbf{r}_0^{k-1}}, \mathbf{r}_k, \underbrace{\mathbf{r}_{k+1}, ..., \mathbf{r}_{N-1}}_{\mathbf{r}_{k+1}^{N-1}} = \mathbf{r}_0^{k-1} \mathbf{r}_k \mathbf{r}_{k+1}^{N-1}. \quad \textbf{(Ошибка! Текст}$$

указанного стиля в документе отсутствует..44)

Тогда используя правило Байеса можно получить

$$P(s', s, \mathbf{r}) = P(s', s, \mathbf{r}_0^{k-1} \mathbf{r}_k \mathbf{r}_{k+1}^{N-1}) = P(\mathbf{r}_{k+1}^{N-1} \mid s', s, \mathbf{r}_0^{k-1} \mathbf{r}_k) P(s', s, \mathbf{r}_0^{k-1} \mathbf{r}_k) \quad \textbf{(Ошибка!}$$

Текст указанного стиля в документе отсутствует..45)

Поскольку состояние решетки $S_k = s$ в момент времени k задано, то последовательность $\mathbf{r}_{k+1}^{N-1}$ не зависит от состояния $S_{k-1} = s'$ и последовательностей $\mathbf{r}_0^{k-1} \mathbf{r}_k$. В этом случае выражение (2.1.6) можно записать

$$P(s', s, \mathbf{r}) = P(\mathbf{r}_{k+1}^{N-1} \mid s) P(s', s, \mathbf{r}_0^{k-1} \mathbf{r}_k). \quad \textbf{(Ошибка! Текст}$$

указанного стиля в документе отсутствует..46)

Упрощая второй множитель выражения (2.1.7) получим

$$P(s', s, \mathbf{r}) = P(\mathbf{r}_{k+1}^{N-1} \mid s) P(s, \mathbf{r}_k \mid s', \mathbf{r}_0^{k-1}) P(s', \mathbf{r}_0^{k-1}) = P(\mathbf{r}_{k+1}^{N-1} \mid s) P(s, \mathbf{r}_k \mid s') P(s', \mathbf{r}_0^{k-1})$$

(**Ошибка! Текст указанного стиля в документе отсутствует.**.47)

Вероятность $P(\mathbf{r}_{k+1}^{N-1} \mid s) = \beta_k(s)$ определяет условную вероятность последовательности $\mathbf{r}_{k+1}^{N}$ при заданном состоянии $S_k = s$. $P(s, \mathbf{r}_k \mid s') = \gamma_k(s', s)$ задает вероятность последовательности $\mathbf{r}_k$ и состояния $S_k = s$ при заданном состоянии $S_{k-1} = s'$. Вероятность $P(s', \mathbf{r}_0^{k-1}) = \alpha_{k-1}(s')$ задает совместную вероятность принятой последовательности $\mathbf{r}_0^{k-1}$ и состояния $S_{k-1} = s'$. Подставляя полученные выражения в (2.1.3)

$$L(u_k \mid \mathbf{r}) = \log \frac{\sum_{R_1} \alpha_{k-1}(s') \gamma_k(s', s) \beta_k(s)}{\sum_{R_0} \alpha_{k-1}(s') \gamma_k(s', s) \beta_k(s)}.$$ (**Ошибка! Текст**

указанного стиля в документе отсутствует..48)

Рассмотрим вычисление множителя $P(s, r_k \mid s') = \gamma_k(s', s)$. Используя правило Байеса

$$\gamma_k(s', s) = P(s, \mathbf{r}_k \mid s') = P(\mathbf{r}_k \mid s', s) P(s \mid s')$$ (**Ошибка! Текст**

указанного стиля в документе отсутствует..49)

Рассмотрим второй множитель $P(s \mid s')$. Поскольку состояние s' задано, то вероятность перерода в одно из двух возможных состояний будет полностью определятся вероятностью соответствующего информационного бита, т.е. $P(u_k)$. Например, если информационные биты равновероятны $P(u_k = \pm 1) = 1/2$, то вероятность перехода из текущего состояния s' в одно из двух возможных состояний s будет также равновероятным. Аналогично вероятность $P(\mathbf{r}_k \mid s', s)$ будет равна вероятности $P(\mathbf{r}_k \mid u_k) = P(\mathbf{r}_k \mid \mathbf{x}_k)$. Возвращаясь к выражению $P(u_k)$ и вводя $L(u_k) = \log \frac{P(u_k = +1)}{P(u_k = -1)}$ легко получить

$$P(u_k = \pm 1) = c_{1k} \exp(u_k L(u_k)/2),$$ (**Ошибка! Текст указанного**

стиля в документе отсутствует..50)

где коэффициент $c_{1k} = \frac{\exp(L(u_k)/2)}{1 + \exp(L(u_k))}$ не зависит от u_k и равен 1/2 при $P(u_k) = 1/2$.

Вероятность $P(\mathbf{r}_k \mid \mathbf{x}_k)$, что n элементов $r_k^{(1)}, r_k^{(2)}, ..., r_k^{(n)}$ получено при заданной кодовой последовательности $x_k^{(1)}, x_k^{(2)}, ..., x_k^{(n)}$, равно произведению вероятностей $P\left(r_k^{(l)} \mid x_k^{(l)}\right)$, т.е.

$$P(\mathbf{r}_k \mid \mathbf{x}_k) = \prod_{l=1}^{n} P\left(r_k^{(l)} \mid x_k^{(l)}\right) \quad \textbf{(Ошибка! Текст указанного стиля в документе отсутствует.}.51)$$

Для рассматриваемого примера сверточного кода

$$P(\mathbf{r}_k \mid \mathbf{x}_k) = P\left(r_k^{(1)} \mid x_k^{(1)}\right) P\left(r_k^{(2)} \mid x_k^{(2)}\right) \quad \textbf{(Ошибка! Текст указанного стиля в документе отсутствует.}.52)$$

Отметим, что вероятности $P\left(r_k^{(l)} \mid x_k^{(l)}\right)$ зависят от модели канала и используемой модуляции. Для двоичной фазовой модуляции (BPSK)

$$P\left(r_k^{(l)} \mid x_k^{(l)}\right) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\left(r_k^{(l)} - x_k^{(l)}\right)^2}{2\sigma^2}\right) \quad \textbf{(Ошибка! Текст указанного стиля в документе отсутствует.}.53)$$

Подставляя полученное выражение (2.1.14) в (2.1.12) получим

$$P(\mathbf{r}_k \mid \mathbf{x}_k) = \exp\left(-\sum_{l=1}^{n} \frac{r_k^{(l)} x_k^{(l)}}{\sigma^2}\right) \cdot \underbrace{\left\{\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^n \exp\left(-\sum_{l=1}^{n} \frac{r_k^{(l)2}}{2\sigma^2}\right) \exp\left(-\sum_{l=1}^{n} \frac{x_k^{(l)2}}{2\sigma^2}\right)\right\}}_{c_{2k}} =$$

$$= c_{2k} \exp\left(\sum_{l=1}^{n} \frac{r_k^{(l)} x_k^{(l)}}{\sigma^2}\right) \quad \textbf{(Ошибка! Текст указанного стиля в документе отсутствует.}.54)$$

Множитель c_{2k} не зависит от информационной последовательности и является одинаковой величиной для числителя и знаменателя выражения (2.1.9). Напротив, значение второго множителя выражения (2.1.15) зависит от кодовой последовательности.

Вводя $L_c = \frac{2}{\sigma^2}$ и $r_k \cdot x_k = \sum_{l=1}^{n} r_k^{(l)} x_k^{(l)}$, получим

$$\gamma_k(s', s) = C_k \exp\left(u_k \frac{L(u_k)}{2}\right) \exp\left(\frac{L_c}{2}(r_k \cdot x_k)\right), \quad \textbf{(Ошибка! Текст указанного стиля в документе отсутствует.}.55)$$

где множитель $c_k = c_{1k} \cdot c_{2k}$ является одинаковым для числителя и знаменателя выражения (2.1.9).

Рассмотрим вычисление множителя $\alpha_{k-1}(s') = P\left(s', \mathbf{r}_1^{k-1}\right)$. Рассмотрим момент времени k

$$\alpha_k(s) = P\left(s, \mathbf{r}_k \mathbf{r}_0^{k-1}\right) = \sum_{s'} P\left(s, s', \mathbf{r}_k \mathbf{r}_0^{k-1}\right) \quad \textbf{(Ошибка! Текст указанного стиля в документе отсутствует.}.56)$$

Используя правило Байеса

$$\sum_{s'} P\left(s, s', \mathbf{r}_k \mathbf{r}_0^{k-1}\right) = \sum_{s'} P\left(s, \mathbf{r}_k \mid s', r_0^{k-1}\right) P\left(s', \mathbf{r}_0^{k-1}\right) = \sum_{s'} P\left(s, \mathbf{r}_k \mid s'\right) P\left(s', \mathbf{r}_0^{k-1}\right).$$

(Ошибка! Текст указанного стиля в документе отсутствует..57)

Легко видеть, что

$$\alpha_k(s) = \sum_{s'} \alpha_{k-1}(s') \gamma_k(s', s). \quad \textbf{(Ошибка! Текст указанного стиля в документе отсутствует.}.58)$$

Таким образом, если нам известно $\alpha_{k-1}(s')$, значение $\alpha_k(s)$ может быть получено путем суммирования произведений $\alpha_{k-1}(s')$ и $\gamma_k(s', s)$ соответствующих ребрам, входящим в состояние s в момент времени k. На Рис. 19 приведен пример вычисления $\alpha_k(s)$ для второго состояния.

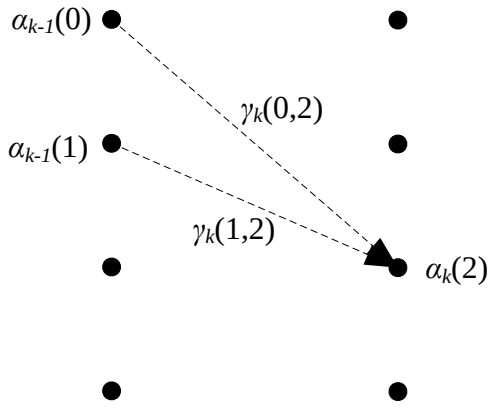

Рис. 19. Пример вычисления $\alpha_k(s)$

Таким образом, значения $\alpha_k(s)$ могут быть вычислены рекурсивно (прямая рекурсия) при заданном начальном состоянии $\alpha_0(s)$. Например, для решеток с фиксированным начальным (нулевым) состоянием

$$\alpha_0(s)=\begin{cases}1, s=0\\0, s\neq 0\end{cases}.$$ (**Ошибка! Текст указанного стиля в документе отсутствует.**.59)

Рекурсивное вычисление $\beta_k(s)=P\left(\mathbf{r}_{k+1}^{N-1}\mid s\right)$ может быть получено аналогичным способом. Для этого рассмотрим момент времени $k-1$

$$\beta_{k-1}(s')=P\left(\mathbf{r}_k^{N-1}\mid s'\right)=\sum_s P\left(s,\mathbf{r}_{k+1}^{N-1}\mathbf{r}_k\mid s'\right).$$ (**Ошибка! Текст указанного стиля в документе отсутствует.**.60)

Применяя правило Байеса, получим

$$\beta_{k-1}(s')=\sum_s P\left(\mathbf{r}_{k+1}^{N-1}\mid s',s,\mathbf{r}_k\right)P\left(s,\mathbf{r}_k\mid s'\right)=\sum_s P\left(\mathbf{r}_{k+1}^{N-1}\mid s\right)P\left(s,\mathbf{r}_k\mid s'\right)=\sum_s \beta_k(s)\gamma_k(s',s)$$

. (**Ошибка! Текст указанного стиля в документе отсутствует.**.61)

При завершении кодирования конечное состояние кода считается известным. Тогда начальные значения для рекурсивного вычисления $\beta_k(s)$ (обратная) определяются следующим образом

$$\beta_N(s)=\begin{cases}1, s=0\\0, s\neq 0\end{cases}.$$ (**Ошибка! Текст указанного стиля в документе отсутствует.**.62)

Пример вычисления $\beta_k(s)$ показан на Рис. 20.

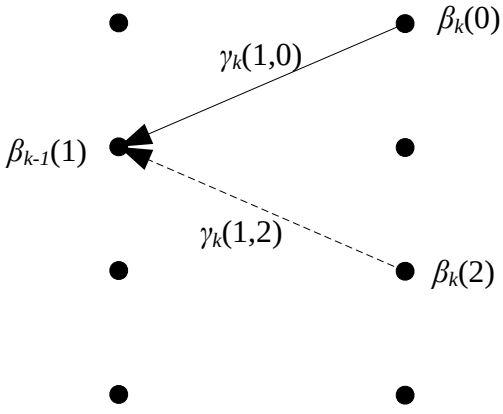

Рис. 20. Пример вычисления $\beta_k(s)$

После вычисления $\gamma_k(s',s),\alpha_{k-1}(s')$ и $\beta_k(s)$ для каждого сегмента решетки кода можно вычислить значение $P(s',s,\mathbf{r})$ для каждого момента времени и пары состояний s' и s.

$$P(s', s, \mathbf{r}) = \alpha_{k-1}(s')\gamma_k(s', s)\beta_k(s)$$ (**Ошибка! Текст указанного стиля в документе отсутствует.**.63)

Подставляя вычисленные значения $P(s', s, \mathbf{y})$ в выражение (2.1.9) получим искомые значения $L(u_k \mid \mathbf{r})$. Для рассматриваемого примера кода $L(u_k \mid \mathbf{r})$ равно

$$L(u_k \mid \mathbf{r}) = \log \frac{P(0,2,\mathbf{r}) + P(1,2,\mathbf{r}) + P(2,3,\mathbf{r}) + P(3,3,\mathbf{r})}{P(0,0,\mathbf{r}) + P(1,0,\mathbf{r}) + P(2,1,\mathbf{r}) + P(3,1,\mathbf{r})}$$ (**Ошибка! Текст указанного стиля в документе отсутствует.**.64)

Лекция № 4. Турбо коды (продолжение)

2.2. Алгоритм декодирования турбо кодов

Далее рассмотрим систематический сверточный код, в котором кодовый бит $x_k^{(1)} = u_k$ равен информационному биту. Чуть позже мы покажем, что $L(u_k \mid \mathbf{r})$ может быть представлена как сумма трех слагаемых

$$L(u_k \mid \mathbf{r}) = L(u_k) + L_c r_k^{(1)} + L_e(u_k) \qquad \textbf{(Ошибка! Текст указанного стиля в документе отсутствует.}.65)$$

Первые два слагаемых относятся к информационному символу u_k. Напротив, третий множитель $L_e(u_k)$ зависит только от проверочных бит. Слагаемое $L_e(u_k)$ принято называть *внешней информацией*. Эта информация является оценкой априорной информации, поскольку при заданных входных значениях $L(u_k)$ и $L_c r_{k1}$, декодер МАВ получает $L(u_k \mid \mathbf{r})$. Тогда дополнительная информация, получаемая из процедуры декодирования, может быть получена

$$L_e(u_k) = L(u_k \mid \mathbf{r}) - L(u_k) - L_c r_k^{(1)} \qquad \textbf{(Ошибка! Текст указанного стиля в документе отсутствует.}.66)$$

Полученная оценка априорной информации может быть использована как входная информация на следующем устройстве декодирования, на выходе которого мы ожидаем получить более точную оценку $L(u_k \mid \mathbf{r})$. Эта идея используется при декодировании турбо кодов, которые мы рассмотрим далее. Общая структура турбо кода с параллельным соединением кодов показана на Рис. 21.

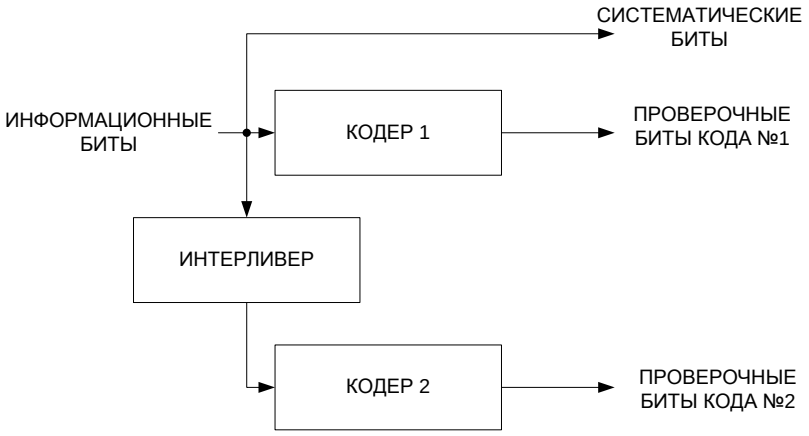

Рис. 21. Структура турбо кода с параллельным соединением кодов

Соответствующая структура турбо декодера показана на Рис. 22.

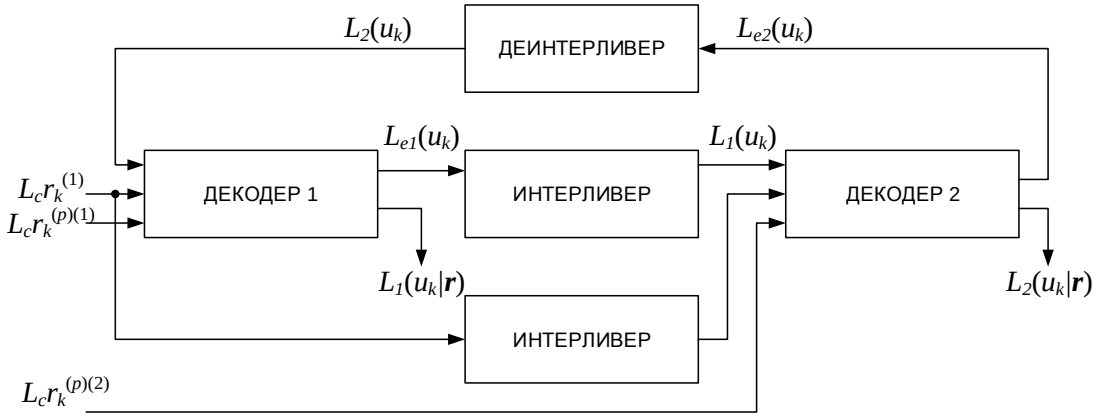

Рис. 22 Структура турбо декодера

Рассмотрим принцип работы итеративного турбо декодирования:

1. На первой итерации происходит инициализация параметров. Поскольку приемник не имеет априорной информации о переданных информационных битах $L(u_k)=0$.

2. На основе принятой кодовой последовательности (систематической $L_c r_{kl}$ и проверочных $L_c r_k^{(p)(1)}$ частей) и априорной информации $L(u_k)$ декодер МАВ первого кода вычисляет внешнюю информацию $L_{e1}(u_k \mid \mathbf{r})$

3. После соответствующей перестановки в блоке интерливинга внешняя информация $L_{e1}(u_k \mid \mathbf{r})$ поступает на вход декодера второго кода в качестве априорной информации $L_1(u_k)$ о систематических битах. На основании априорной информации и принятой кодовой последовательности (систематической $L_c r_k^{(l)}$ и проверочных $L_c r_k^{(p)(2)}$ частей) декодер МАВ второго кода вычисляет внешнюю информацию $L_{e2}(u_k \mid \mathbf{r})$, которая подается на декодер первого кода.

4. После определенного числа итераций с выхода декодера первого или второго кода значения $L_1(u_k \mid \mathbf{r})$ или $L_2(u_k \mid \mathbf{r})$ подаются на блок оценки переданной информационной последовательности.

5. Если $L(u_k \mid \mathbf{r})>0$, то принимается решение $u_k=+1$. Если $L(u_k \mid \mathbf{r})\le 0$, то принимается решение $u_k=-1$

Вернемся к выражению (2.2.2) и покажем его справедливость. Напомним, что рассматриваемый сверточный код является систематическим и $x_{1k} = u_k$. Тогда выражение (2.1.16) можно записать в следующем виде

$$\gamma_k(s', s) = C_k \exp\left(u_k \left[\frac{L(u_k)}{2} + \frac{L_c r_k^{(1)}}{2} \right] \right) \exp\left(\frac{L_c}{2} \sum_{l=2}^{n} x_k^{(l)} r_k^{(l)} \right) \quad \textbf{(Ошибка! Текст указанного стиля в документе отсутствует.}.67)$$

Введем дополнительное обозначение второго множителя выражения (2.2.3)

$$\delta_k(s', s) = \exp\left(\frac{L_c}{2} \sum_{l=2}^{n} x_k^{(l)} r_k^{(l)} \right). \quad \textbf{(Ошибка! Текст указанного стиля в документе отсутствует.}.68)$$

Подставляя (2.2.4) в выражение (2.2.3) а затем в (2.1.9) получим

$$L(u_k \mid \mathbf{r}) = \log \frac{\sum_{R_1} C_k \exp\left(u_k \left[\frac{L(u_k)}{2} + \frac{L_c r_k^{(1)}}{2} \right] \right) \alpha_{k-1}(s') \delta_k(s', s) \beta_k(s)}{\sum_{R_0} C_k \exp\left(u_k \left[\frac{L(u_k)}{2} + \frac{L_c r_k^{(1)}}{2} \right] \right) \alpha_{k-1}(s') \delta_k(s', s) \beta_k(s)} \quad \textbf{(Ошибка! Текст указанного стиля в документе отсутствует.}.69)$$

Поскольку множества R_0 и R_1 соответствуют информационным битам $u_k = -1$ и $u_k = +1$, прямая подстановка u_k в (2.2.5) дает

$$L(u_k \mid \mathbf{r}) = \log\left\{ \exp(L(u_k)) \cdot \exp\left(L_c r_k^{(1)}\right) \cdot \frac{\sum_{R_1} \alpha_{k-1}(s') \delta_k(s', s) \beta_k(s)}{\sum_{R_0} \alpha_{k-1}(s') \delta_k(s', s) \beta_k(s)} \right\}$$

(Ошибка! Текст указанного стиля в документе отсутствует..70)

Легко видеть, что выражение (2.2.6) может быть представлено как сумма трех слагаемых

$$L(u_k \mid \mathbf{r}) = L(u_k) + L_c r_k^{(1)} + L_e(u_k), \quad \textbf{(Ошибка! Текст указанного стиля в документе отсутствует.}.71)$$

где

$$L_e(u_k) = \log\left\{\frac{\sum_{R_1} \alpha_{k-1}(s')\delta_k(s',s)\beta_k(s)}{\sum_{R_0} \alpha_{k-1}(s')\delta_k(s',s)\beta_k(s)}\right\} \quad \text{(Ошибка! Текст}$$

указанного стиля в документе отсутствует..72)

Заметим, что для вычисления $L_e(u_k)$ можно использовать как выражение (2.2.8) так и выражение (2.2.2).

2.3. Модификация алгоритмов декодирования турбо кодов

Как уже отмечалось ранее, основным недостатком алгоритма декодирования заключается в относительно высокой вычислительной сложности. Далее рассмотрим модификации алгоритма MAB, позволяющие существенно упростить процедуру декодирования с приемлемой потерей помехоустойчивости. Для этого введем следующие переменные

$$\Gamma_k(s',s) = \log(\gamma_k(s',s)) = \log(C_k) + u_k \frac{L(u_k)}{2} + \frac{L_c}{2}(r_k \cdot x_k), \quad \text{(Ошибка! Текст}$$

указанного стиля в документе отсутствует..73)

$$\mathrm{A}_k(s) = \log(\alpha_k(s)) = \max_{s'}{}^*(\mathrm{A}_{k-1}(s') + \Gamma_k(s',s)), \quad \text{(Ошибка! Текст}$$

указанного стиля в документе отсутствует..74)

$$\mathrm{B}_{k-1}(s') = \log(\beta_{k-1}(s')) = \max_{s}{}^*(\mathrm{B}_k(s) + \Gamma_k(s',s)), \quad \text{(Ошибка! Текст}$$

указанного стиля в документе отсутствует..75)

где

$$\max_{x,y}{}^*(x+y) = \begin{cases} \max(x,y) + \log(1+\exp(-|x-y|)) & \log \text{ MAB алгоритм} \\ \max(x,y) & \max - \log \text{ MAB алгоритм} \end{cases} \quad \text{(Ошибка!}$$

Текст указанного стиля в документе отсутствует..76)

Тогда выражение для вычисления апостериорной вероятности информационного бита можно упростить как

$$L(u_k \mid \mathbf{r}) = \max_{R_1}{}^{*}\left(\mathrm{A}_{k-1}(s') + \Gamma_k(s', s) + \mathrm{B}_k(s)\right) - \max_{R_0}{}^{*}\left(\mathrm{A}_{k-1}(s') + \Gamma_k(s', s) + \mathrm{B}_k(s)\right)$$

(**Ошибка! Текст указанного стиля в документе отсутствует.**.77)

Вычисление $\max^{*}(\cdot)$ от трех аргументов в выражении 7.3.5 может быть выполнено рекурсивно с использованием следующего равенства

$$\max^{*}(x_1 + x_2 + x_3) = \max^{*}\left(\max^{*}(x_1 + x_2) + x_3\right).$$ (**Ошибка! Текст указанного стиля в документе отсутствует.**.78)

2.4. Основные принципы построения турбо кодов

После рассмотрения процедуры декодирования обсудим структуру турбо кодов, показанную на Рис. 21. Напомним, что в классической теореме Шеннона утверждается, что с помощью случайного кода можно обеспечить передачу данных со сколь угодно низкой вероятностью ошибки. При этом безошибочная передача обеспечивается при больших длинах кодового слова. Как мы уже видели ранее, проблема использования случайного кода с большой длиной кодового слова состоит в сложности его декодирования. Случайность или отсутствие структуры кода приводит к существенным вычислительным затратам при декодировании. Идея построения случайного кода с приемлемой сложностью декодирования реализуется в турбо кодах. Как видно из Рис. 21, турбо код состоит из двух параллельно соединенных компонентных кодов, разделенных внутренним интерливером. Задача интерливера является наиболее важной для обеспечения случайности кода и состоит в перестановке информационной последовательности для «декорреляции» проверочных бит, полученных компонентными кодами. Заметим, что для получения случайного кода, применение псевдослучайного интерливера является предпочтительным. Стоит также отметить, что высокая помехоустойчивость всего кода достигается даже при использовании относительно простых компонентных кодов (например, сверточных кодов с малой длиной кодового ограничения). Не смотря на простоту каждого компонентного кода, структура всего турбо кода является достаточно сложной для использования методов прямого декодирования. Использование итеративного метода декодирования, рассмотренного выше, позволяет получить достаточно высокую эффективность кодирования. При этом систематичность компонентных кодов требуется для применения таких алгоритмов декодирования. Стоит

отметить, что для уменьшения вероятности появления ненулевой кодовой последовательности минимального веса в качестве компонентных кодов, как правило, используют рекурсивные систематические сверточные коды. Использование рекурсивных сверточных кодов, разделенных псевдослучайным интерливером, позволяет существенно уменьшить вероятность одновременного появления кодовых последовательностей минимального веса на выходе компонентных кодов.

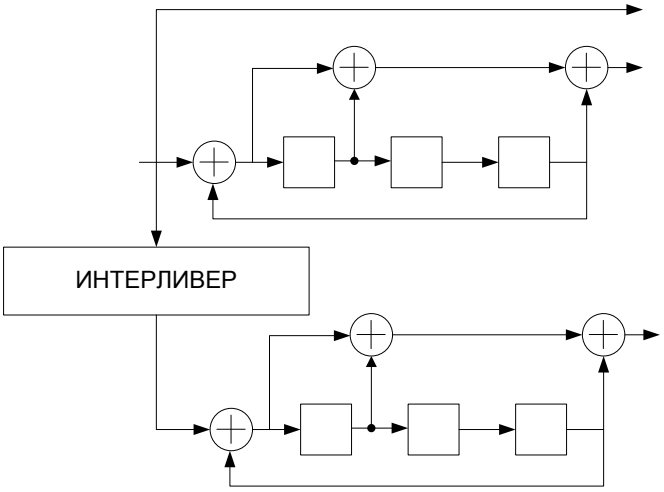

Рис. 23. Пример турбо кода стандарта 3GPP Release 8

Пример турбо кода используемого в 3GPP Release 8 стандарте сотовой связи показан на Рис. 23. В коде используются два систематических рекурсивных сверточных кода со скоростью кода 1/2. В качестве интерливинга используется квадратичный интерливер с индексами перестановки задаваемыми выражением

$$j = \left(f_1 \cdot i + f_2 \cdot i^2\right) \bmod N, \qquad \text{(Ошибка! Текст указанного стиля в документе отсутствует..79)}$$

где N длина информационного блока, а f_1 и f_2 параметры интерливера.

2.5. Метод параллельного декодирования турбо кодов

Последовательное вычисление внешней информации для систематических бит компонентного кода с помощью алгоритма MAB, как правило, приводит к существенной задержке при итеративном декодировании турбо кодов. Поэтому для высокоскоростных систем связи одной из наиболее важных задач является возможность реализации параллельного декодирования принятой последовательности.

Для турбо кодов параллельное декодирование реализуется с помощью разбиения решетки кода на сегменты длины W и параллельного вычисления внешней информации для систематических бит $L_e(u_k)$ в каждом сегменте решетки (см. Рис. 24).

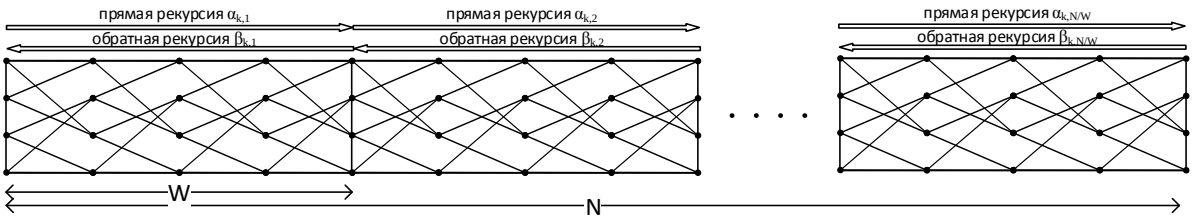

Рис. 24. Разбиение решетки кода на ссегменты для параллельного декодирования принятой последовательности

Эффективность такого подхода параллельного вычисления внешней информации определяется, главным образом, возможностью параллельного доступа в память для записи вычисленных значений внешней информации каждого сегмента решетки. Для обеспечения такой записи в архитектуре декодера используется несколько блоков памяти, а интерливинг строится, таким образом, чтобы избежать одновременного доступа к одному блоку памяти более чем с одного сегмента решетки при записи вычисленных значений $L_e(u_k)$.

Пример процедуры интерливинга не имеющего конфликта параллельного доступа к памяти показан на Рис. 25 для четырех сегментов решетки длины $W = 4$. Из рисунка видно, что для фиксированного индекса бита i внутри каждого сегмента решётки запись производится в память с адресом $f(i+tW)$ (где $f(\cdot)$ функция интерливера). При этом индекс блока памяти, определяемый значением $\lfloor f(i+tW)/W \rfloor$, для каждого сегмента решетки *t* всегда оказывается разным.

В общем случае условие отсутствия конфликта одновременного доступа к памяти для интерливера $f(\cdot)$ можно записать следующим образом

$$\left\lfloor \frac{f(i+vW)}{W} \right\rfloor \neq \left\lfloor \frac{f(i+uW)}{W} \right\rfloor, 0 \le v,u \le N-1, \qquad \text{(Ошибка! Текст}$$

указанного стиля в документе отсутствует..80)

где $i \in 0,...,W-1$ индекс систематического бита внутри сегмента решетки.

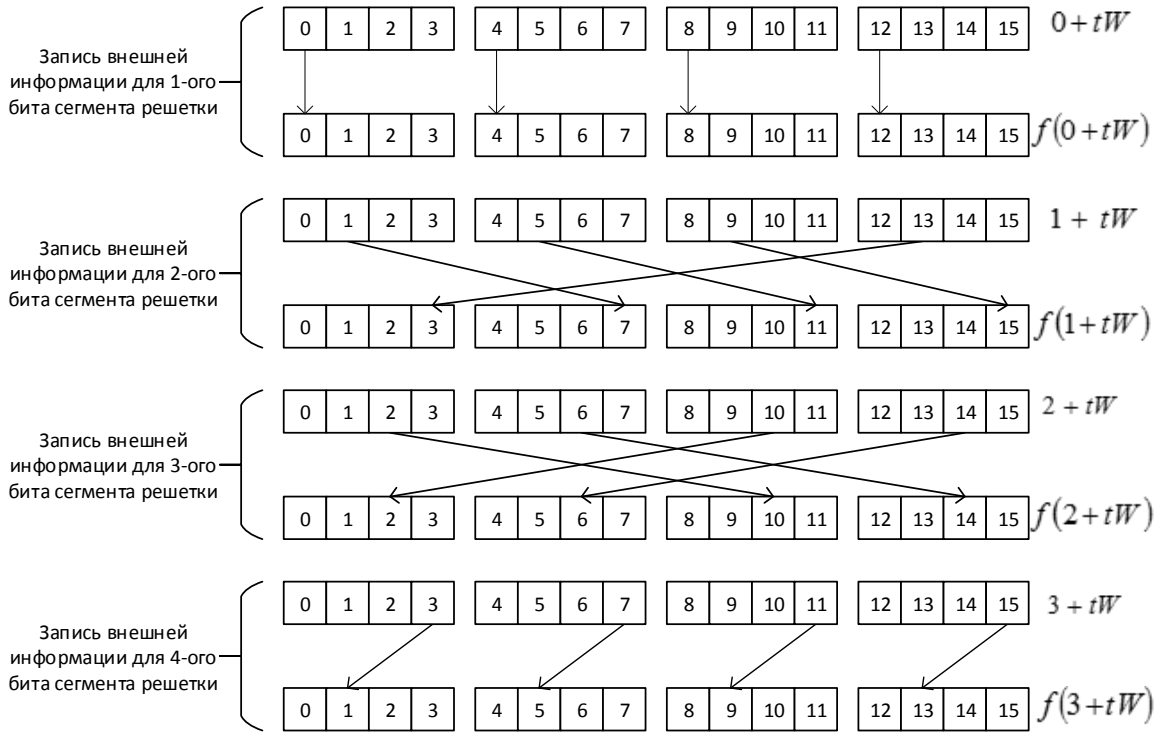

Рис. 25. Пример процедуры интерливинга для параллельной архитектуры кода

Определение: Интерливер обеспечивающий отсутствие конфликта одновременного доступа к памяти для всех возможных значений длин сегментов решетки W (являющегося делителем N) называется интерливером с *максимальным* отсутствием конфликта одновременного доступа к памяти.

Можно показать, что квадратичный интерливер $j=\left(f_1\cdot i+f_2\cdot i^2\right)\bmod N$ удовлетворяет условию максимально отсутствия конфликта одновременного доступа к памяти.

Доказательство:

Пусть

$$Q_v=\left\lfloor\frac{f\left(i+vW\right)}{W}\right\rfloor,\quad Q_u=\left\lfloor\frac{f\left(i+uW\right)}{W}\right\rfloor, \qquad \textbf{(Ошибка! Текст указанного стиля в документе отсутствует.}.81)$$

тогда

$$\begin{aligned} f\left(i+vW\right)&=Q_vW+\bmod\left(f\left(i+vW\right),W\right)\\ f\left(i+uW\right)&=Q_uW+\bmod\left(f\left(i+uW\right),W\right)\end{aligned}. \qquad \textbf{(Ошибка! Текст указанного стиля в документе отсутствует.}.82)$$

Необходимо показать, что для любого $i \in 0,...,W-1$, $Q_v \neq Q_u$ для любого $u \neq v$. Действительно, предположим что $Q_v = Q_u$. Тогда используя выражение (2.5.3) получим

$$Q_v - Q_u = \frac{f(i+vW) - f(i+uW) + \mathrm{mod}(f(i+vW),W) - \mathrm{mod}(f(i+uW),W)}{W} = 0. \quad (\textbf{Ошибка! Текст указанного стиля в документе отсутствует.}.83)$$

Поскольку для квадратичного интерливера выполняется равенство $\mathrm{mod}(f(i+vW),W) = \mathrm{mod}(f(i+uW),W)$, то $f(i+vW) = f(i+uW)$. Данное равенство противоречит уникальности функции интерливера $f(\cdot)$. Отсюда следует, что $Q_v \neq Q_u$ для любого $u \neq v$ и справедливо для всех индексов $i \in 0,...,W-1$.

Лекция № 5. Коды с малой плотностью проверки на четность

Глава 3. Коды с малой плотностью проверки на четность

3.1. Определение кодов с малой плотностью проверки на четность

Коды с малой плотностью проверки на четность являются подклассом линейных блоковых кодов, рассмотренных в Главе 2. Напомним, что (n,k) линейный блоковый код можно рассматривать как отображение (с помощью порождающей матрицы $\mathbf{G}$) k-мерного подпространства в пространство размерности n. При этом процедура кодирования может быть представлена в матричном виде

$$\mathbf{c} = \mathbf{u} \cdot \mathbf{G}. \qquad \text{(Ошибка! Текст указанного стиля в документе отсутствует..84)}$$

Строки порождающей матрицы $\mathbf{G}$ задают базис кодового подпространства размерности k, а базисные вектора ортогональные кодовому подпространству задают проверочную матрицу $\mathbf{H}$ размерности $(n-k,n)$. В силу ортогональности подпространств, для любого кодового слова $\mathbf{c}$ выполняется равенство

$$\mathbf{c} \cdot \mathbf{H}^T = \mathbf{0}, \qquad \text{(Ошибка! Текст указанного стиля в документе отсутствует..85)}$$

Отличительной особенностью кодов с малой плотностью проверки на четность является разреженность проверочной матрицы $\mathbf{H}$, т.е. относительное число ненулевых элементов является незначительным по отношению к общему числу элементов матрицы $\mathbf{H}$. Данное ограничение позволяет существенно упростить процедуру декодирования кодов, которую мы рассмотрим ниже.

Несмотря на то, что эти коды были изобретены в 1963 году Робертом Галлагером, их практическое применение началось относительно недавно.

Далее мы ограничимся рассмотрением двоичных кодов с малой плотностью проверки на четность, заданных над полем $GF(2)$. Введем некоторые определения. Код с малой плотностью проверки на четность является кодом регулярности t, если вес (число единиц) каждого столбца проверочной матрицы $\mathbf{H}$ является постоянной величиной. В этом случае среднее число единиц в каждой строке проверочной матрицы $\mathbf{H}$ равно $nt/(n-k)$. Если вес каждой строки проверочной матрицы также является постоянной величиной равной $s = nt/(n-k)$, то код называется (s,t) регулярным.

Пример кода с малой плотностью проверки на плотность регулярности $t = 2$ приведен ниже

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

Напомним, что для любого кодового слова $\mathbf{c}$ должно выполняться равенство

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_8 \\ c_9 \end{bmatrix} = 0 .$$

Выражение (3.1.2) определяет набор уравнений проверок на четность, которые для рассматриваемого примера можно записать следующим образом

$$\begin{aligned} p_0 &= c_0 + c_1 + c_2 + c_3 \\ p_1 &= c_0 + c_4 + c_5 + c_6 \\ p_2 &= c_1 + c_4 + c_7 + c_8 \\ p_3 &= c_2 + c_5 + c_7 + c_9 \\ p_4 &= c_3 + c_6 + c_8 + c_9 \end{aligned}$$

Коды с малой плотностью проверки на четность описываются с помощью двухстороннего (двудольного) графа, где нижние (битовые) узлы соответствуют кодовым битам c_i, а верхние (проверочные) узлы соответствуют проверочным уравнениям p_j. Битовый узел соединяется с проверочным узлом ребром, если кодовый бит присутствует в уравнении проверки на четность (3.1.2). Пример двустороннего графа для рассматриваемого кода приведен на Рис. 26.

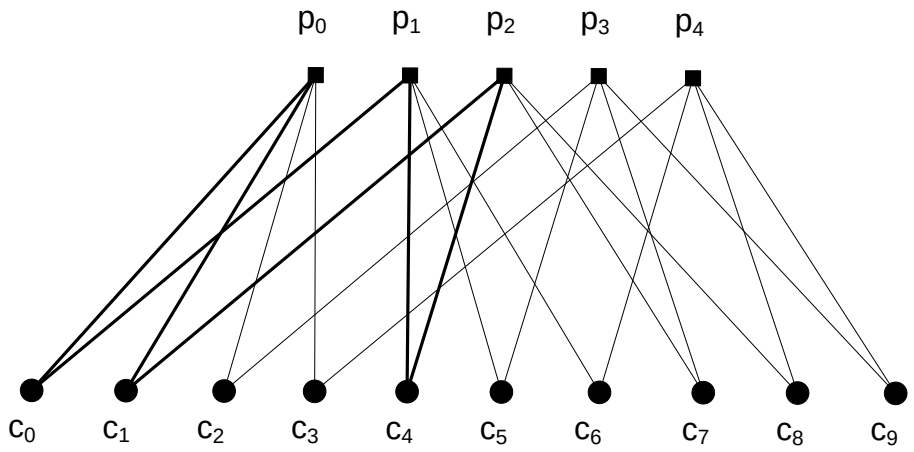

Рис. 26. Двусторонний граф кода с малой плотностью проверки на четность

Циклом длины ν в двустороннем графе будем называть замкнутый путь, состоящий из ν ребер. На Рис. 26 жирной линией выделен цикл длины 6. Цикл минимальной длины называется *обхватом кода*.

3.2. Примеры построения кодов с малой плотностью проверки на четность

Коды Галлагера

Коды Галлагера является регулярными кодами с малой плотностью проверки на четность. Проверочная матрица таких кодов может быть представлена в виде

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_1 \\ \mathbf{H}_2 \\ \vdots \\ \mathbf{H}_t \end{bmatrix}, \qquad \text{(Ошибка! Текст указанного стиля в документе отсутствует.}.86)$$

где подматрицы $\mathbf{H}_d$ имеют следующую структуру. Для любого целого μ и s больше единицы, подматрица $\mathbf{H}_d$ имеет размерность $\mu \times \mu \cdot s$, а вес каждой строки и столбца равен s и 1 соответственно. Первая подматрица $\mathbf{H}_1$ имеет специальный вид, такой, что строка i для $i = 1,..., \mu$ содержит все s единиц в столбцах с индексами от $(i-1)\cdot s+1$ до $i \cdot s$. Остальные подматрицы $\mathbf{H}_d$ где $d = 2,..., t$ могут быть получены путем перестановки столбцов матрицы $\mathbf{H}_1$. Очевидно, что код Галлагера является регулярным, а матрица $\mathbf{H}$ имеет размерность $\mu \cdot t \times \mu \cdot s$. Отсутствие циклов длины 4 не гарантируется. Однако, их можно избежать путем соответствующего выбора матрицы $\mathbf{H}$.

Коды МакКея

Построение кодов МакКея основывается на случайном поиске матриц. Ниже приведены основные способы построения кода, предложенные МакКеем:

1. Матрица $\mathbf{H}$ создается путем набора векторов весом s
2. Матрица $\mathbf{H}$ создается из набора векторов весом s, гарантирующего вес t для каждой строки матрицы. При этом любая пара векторов имеет расстояние Хэмминга не более чем 1.
3. Из множества матриц $\mathbf{H}$, удовлетворяющих построению 2, исключаются матрицы, имеющие короткие циклы в двустороннем графе.
4. Матрица $\mathbf{H}$ удовлетворяет построению 3. При этом $\mathbf{H}$ можно представить как $\mathbf{H}=[\mathbf{H}_1 \quad \mathbf{H}_2]$, где матрица $\mathbf{H}_2$ обратима.

3.3. Методы декодирования кодов с малой плотностью проверки на четность

Пусть вектор $\mathbf{r}=(r_0,r_1,\ldots,r_{n-1})$ длины n задает принятую последовательность, а вектор $\mathbf{c}=(c_0,c_1,\ldots,c_{n-1})$ кодовую. Для простоты рассмотрения будем читать что -1 и +1 соответствуют 0 и 1 соответственно. Тогда кодовая последовательность c_k может принимать значения -1 или +1 при передаче в канале. Задача декодирования ставится как оценка элементов кодовой последовательности c_k на основе наблюдения $\mathbf{r}$. Декодирование кодовых бит c_k будем осуществлять путем вычисления логарифма отношения апостериорной вероятности на основе наблюдения $\mathbf{r}$

$$L(c_k \mid \mathbf{r}) = \log\left(\frac{P(c_k=0 \mid \mathbf{r})}{P(c_k=1 \mid \mathbf{r})}\right). \qquad \textbf{(Ошибка! Текст указанного стиля в документе отсутствует.}.87)$$

Если логарифм отношения апостериорной вероятности $L(c_k \mid \mathbf{r})>0$, то $\hat{c}_k=0$. В противном случае $\hat{c}_k=1$. Напомним, что все n кодовых бит при декодировании должны удовлетворять равенствам проверки на четность. Данное структурное ограничение кода должно также учитывается при декодировании. Тогда выражение для апостериорной вероятности модифицируется следующим образом

$$P(c_k \mid S_k, \mathbf{r}),$$ (**Ошибка! Текст указанного стиля в документе отсутствует.**.88)

где событие S_k соответствует выполнению всех уравнений проверки на четность для бита c_k. Используя правило Байеса, апостериорная вероятность $P(c_k \mid S_k, \mathbf{r})$ может быть преобразована в следующий вид

$$P(c_k \mid S_k, \mathbf{r}) = K \cdot P(r_k \mid c_k) \cdot P(S_k \mid c_k, \mathbf{r}).$$ (**Ошибка! Текст указанного стиля в документе отсутствует.**.89)

Первый множитель K в равенстве (3.3.3) является константой и в дальнейшем может быть опущен из рассмотрения, т.к. является величиной независящей от c_k. Второй множитель $p(r_k \mid c_k)$ выражения (3.3.3) может быть вычислен из наблюдения r_k с использованием модели канала. Например, для двоичного канала с аддитивным белым гауссовским шумом вероятность $p(r_k \mid c_k)$ определяется как

$$p(r_k \mid c_k) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{(r_k - c_k)^2}{2\sigma^2}\right\}.$$ (**Ошибка! Текст указанного стиля в документе отсутствует.**.90)

Третий множитель в равенстве (3.3.3) задает вероятность выполнения всех проверок на четность с участием кодового бита c_k. Событие S_k может быть представлено как объединение событий $\{S_{1k}, S_{2k}, ..., S_{Lk}\}$, где S_{mk} является событием, соответствующим выполнению проверочного выражения для проверочного узла m, соединенного с битовом узлом k. Пусть $M(k)$ является набором проверочных узлов соответствующих биту c_k, а $N(m)$ множеством битовых узлов соответствующих проверочному узлу m. Предположим, что в двустороннем графе отсутствуют замкнутые циклы, тогда третий множитель может быть представлен в виде произведения

$$P(S_k \mid c_k = b, \mathbf{r}) = p(S_{1k}, S_{2k}, ..., S_{Lk} \mid c_k = b, \mathbf{r}) = \prod_{m \in M(k)} P(S_{mk} \mid c_k, \mathbf{r}),$$ (**Ошибка! Текст указанного стиля в документе отсутствует.**.91)

где $p(S_{mk} \mid c_k, \mathbf{r})$ вероятность, что проверка соответствующая проверочному узлу m, соединенная с битом c_k, выполняется. Если $c_k = 0$, то $p(S_{mk} \mid c_k, \mathbf{r})$ равняется вероятности того, что все битовые узлы, соединяющие проверочный узел m за исключением узла соответствующего биту c_k, имеют четное число единиц. Аналогично, если $c_k = 1$, то для выполнения условия проверки на четность в проверочном узле m, число единиц в оставшихся битовых узлах нечетно.

Далее предположим, что два бита удовлетворяют уравнению проверки на четность (т.е. $c_1 + c_2 = 0$), а вероятности равны $P(c_1 = 0) = p_1 = 1 - q_1$, $P(c_2 = 0) = p_2 = 1 - q_2$. Тогда вероятность выполнения проверки на четность для двух бит

$$P(c_1 + c_2 = 0) = (1 - p_1) \cdot (1 - p_2) + p_1 \cdot p_2 = 2 \cdot p_1 \cdot p_2 + 1 - p_1 - p_2. \quad \textbf{(Ошибка! Текст указанного стиля в документе отсутствует.}.92)$$

Преобразуем выражение (3.3.6) в следующий вид

$$2 \cdot P(c_1 + c_2 = 0) - 1 = (1 - 2 \cdot p_1) \cdot (1 - 2 \cdot p_2) = (q_1 - p_1) \cdot (q_2 - p_2). \quad \textbf{(Ошибка! Текст указанного стиля в документе отсутствует.}.93)$$

Обобщим выражение (3.3.7) на случай L кодовых бит, участвующих в уравнении проверки на четность $z_L = c_1 + c_2 + ... + c_L$

$$P(z_L = 0) = \frac{1}{2}\left(1 + \prod_{i=1}^{L}(q_i - p_i)\right), \quad \textbf{(Ошибка! Текст указанного стиля в документе отсутствует.}.94)$$

где $p_i = 1 - q_i = P(c_i = 0)$. Аналогичным образом можно показать, что

$$P(z_L = 1) = \frac{1}{2}\left(1 - \prod_{i=1}^{L}(q_i - p_i)\right). \quad \textbf{(Ошибка! Текст указанного стиля в документе отсутствует.}.95)$$

Тогда вероятность $P(S_{mk} \mid c_k, \mathbf{r})$ при условии $c_k = 0$ равна

$$P(S_{mk} \mid c_k = 0, \mathbf{r}) = \frac{1}{2}\left(1 + \prod_{n' = N(m) \setminus k}\left(q_{mn'}(0) - q_{mn'}(1)\right)\right), \quad \textbf{(Ошибка! Текст указанного стиля в документе отсутствует.}.96)$$

где $q_{mn'}(0)$ вероятность, что кодовый бит $c_{n'}$ равен 0, $N(m) \setminus k$ множество кодовых узлов за исключением узла k. Заметим, что равенство (3.3.10) включает в себя произведение по всем ребрам, соединяющими проверочный узел m кодовыми, за исключением узла k. Таким образом, при вычислении $P(S_{mk} \mid c_k, \mathbf{r})$ исключается информация о бите c_k, а полученная информация является *внешней*, вычисленной с использованием структуры кода.

Объедения, полученные выше выражения

$$P(c_k = 0 \mid S_k, r) = K \cdot P(r_k \mid c_k = 0) \cdot \prod_{m \in M(k)} \frac{1}{2}\left(1 + \prod_{n' = N(m)\setminus k}^{L}\left(q_{mn'}(0) - q_{mn'}(1)\right)\right).$$ (**Ошибка! Текст указанного стиля в документе отсутствует.**.97)

$$P(c_k = 1 \mid S_k, r) = K \cdot P(r_k \mid c_k = 1) \cdot \prod_{m \in M(k)} \frac{1}{2}\left(1 - \prod_{n' = N(m)\setminus k}^{L}\left(q_{mn'}(0) - q_{mn'}(1)\right)\right).$$ (**Ошибка! Текст указанного стиля в документе отсутствует.**.98)

Равенства (3.3.11) и (3.3.12) можно рассматривать как последовательное вычисление сообщений от проверочных узлов к битовым узлам и наоборот. Например, для $c_k = 1$ вычисление выражения (3.3.12) можно разбить на два этапа

$$\underbrace{\overbrace{P(r_k \mid c_k = 1)}^{\substack{\text{априорная вероятность} \\ \text{на основе наблюдений } r_k}} \cdot \prod_{m \in M(k)} \overbrace{\frac{1}{2}\left(1 - \prod_{n' = N(m)\setminus k}^{L}\left(q_{mn'}(0) - q_{mn'}(1)\right)\right)}^{\text{проверочный узел}}}_{\text{битовый узел}}$$ (**Ошибка! Текст указанного стиля в документе отсутствует.**.99)

Первый этап соответствует вычислению сообщений от проверочных узлов к битовым узлам. Вводя обозначение $\delta q_{mn'} = q_{mn'}(0) - q_{mn'}(1)$, выражения, соответствующие первому этапу, задаются следующим образом

$$r_{mk}(0) = \frac{1}{2}\left(1 + \prod_{n' = N(m)\setminus k} \delta q_{mn'}\right)$$ (**Ошибка! Текст указанного стиля в документе отсутствует.**.100)

$$r_{mk}(1) = \frac{1}{2}\left(1 - \prod_{n' = N(m)\setminus k} \delta q_{mn'}\right).$$ (**Ошибка! Текст указанного стиля в документе отсутствует.**.101)

Графически вычисление $r_{mk}(\cdot)$ проиллюстрировано на Рис. 27.

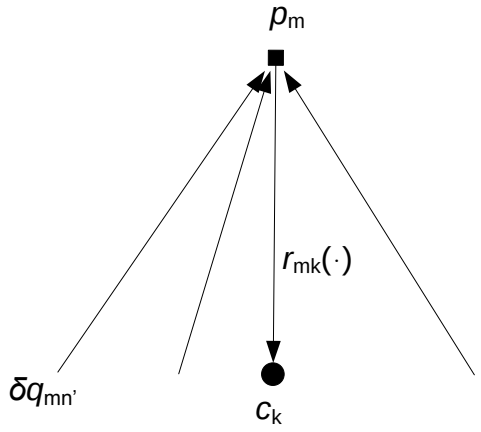

Рис. 27. Вычисление $r_{mk}(\cdot)$

Для вычисления сообщения $r_{mk}(\cdot)$ находится набор ребер $N(m)\setminus k$ двухстороннего графа, входящих в узел p_m, при этом исключается ребро, соединяющее битовый узел c_k с проверочным узлом p_m. Далее согласно выражениям (3.3.14) и (3.13.15) производится вычисление сообщения $r_{mk}(\cdot)$ от проверочного узла p_m к битовому узлу c_k.

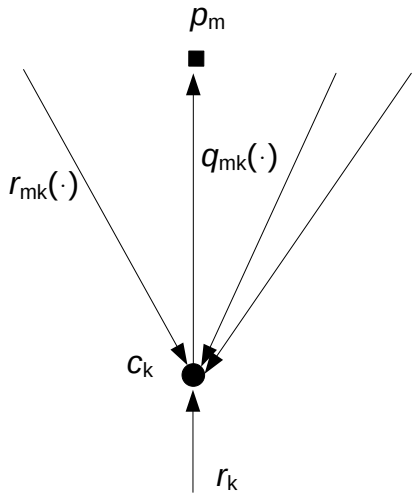

Рис. 28 Вычисление $q_{mk}(\cdot)$

Аналогичным способом выполняется второй этап вычисления выражений (3.3.11) и (3.3.12), соответствующий вычислению сообщений от битовых узлов к проверочным (см. Рис. 28). При этом учитывается априорная информация $P(r_k \mid c_k)$ кодового бита c_k, полученная на основе принятого символа r_k.

Псевдокод алгоритма декодирования кодов с малой плотностью проверки на четность приведен ниже:

Инициализация

Переменная $q_{mn}(\cdot)$ инициализируются следующим образом

$$q_{mk}(\cdot) = f_k(\cdot) = \begin{cases} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(r_k+1)^2}{2\sigma^2}\right), & c_k = 0 \\ \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(r_k-1)^2}{2\sigma^2}\right), & c_k = 1 \end{cases}$$ (**Ошибка! Текст указанного стиля в документе отсутствует.**.102)

Вычисление сообщений проверочных узлов

Для каждого ребра исходящего из проверочного узла с индексом $m \in M(k)$ вычисляется сообщение

$$\delta r_{mk} = \prod_{i' \in N(m) \setminus k} \delta q_{mi'},$$ (**Ошибка! Текст указанного стиля в документе отсутствует.**.103)

где $r_{mk}(0) = \frac{1}{2}\left(1 + \delta r_{mk}\right),\ r_{mk}(1) = \frac{1}{2}\left(1 - \delta r_{mk}\right).$

Вычисление сообщений битовых узлов

Для каждого ребра, исходящего из битового узла с индексом $k = 0,...,n-1$, вычисляется *внешнее* сообщение, исключающее информацию проверочного узла m

$$q_{mk}(0) = \alpha_{mk} f_k(0) \prod_{m' \in M(k) \setminus m} r_{m'k}(0)$$ (**Ошибка! Текст указанного стиля в документе отсутствует.**.104)

$$q_{mk}(1) = \beta_{mk} f_k(1) \prod_{m' \in M(k) \setminus m} r_{m'k}(1)$$ (**Ошибка! Текст указанного стиля в документе отсутствует.**.105)

Постоянные нормировки α_{mk} и β_{mk} выбираются таким образом, чтобы выполнялось условие нормировки вероятности $q_{mk}(0) + q_{mk}(1) = 1$.

Вычисление выражений проверки на четность

Для каждого элемента кодовой последовательности с индексом $k = 0,...,n-1$ вычисляется *полная* вероятность

$$q_k(0) = \alpha_k f_k(0) \prod_{m' \in M(k)} r_{m'k}(0)$$ (**Ошибка! Текст указанного стиля в документе отсутствует.**.106)

$$q_k(1) = \beta_k f_k(1) \prod_{m' \in M(k)} r_{m'k}(1).$$ (**Ошибка! Текст указанного стиля в документе отсутствует.**.107)

Постоянные нормировки α_k и β_k выбираются таким образом, чтобы выполнялось условие нормировки вероятности $q_k(0) + q_k(1) = 1$. На основе вероятностей кодовых узлов вычисляются решения

$$c_k = \begin{cases} 0, & q_k(0) > 1/2 \\ 1, & q_k(1) \le 1/2 \end{cases}$$ (**Ошибка! Текст указанного стиля в документе отсутствует.**.108)

Если проверка на четность $\mathbf{c} \cdot \mathbf{H}^T = \mathbf{0}$ не выполняется, и число итераций не превысило максимального значения, процедура декодирования повторяется.

Заметим, что при вычислении сообщений для ребер исходящих из битовых и проверочных узлов исключается информация соответствующего ребра. Таким образом, на каждой итерации вычисляется только *внешняя информация*, которая передается далее по графу. При этом для вычисления выражений проверки на четность учитывается *полная* информация. Можно показать, что если в графе отсутствуют замкнутые циклы, то приведенный выше алгоритм осуществляет оценку, обеспечивающую максимум апостериорной вероятности для каждого кодового бита. Однако для двухстороннего графа с обхватом γ информация возвращается к началу цикла на итерации γ и алгоритм в общем случае не является оптимальным. Не смотря на это, результаты моделирования практических схем показали, что эффективность алгоритма декодирования кодов остается достаточно высокой при использовании кодов с обхватом 6 или более.

Основным недостатком полученного итеративного алгоритма декодирования является вычислительная неустойчивость, связанная с наличием большого числа умножений. На практике предпочтительным является реализация алгоритма декодирования на основе операций сложения. Для этого декодирование необходимо проводить в области логарифма вероятности. Тогда вычислительно-емкие операции умножения и деления заменяются относительно простыми операциями сложения и деления. Для упрощения процедуры декодирования будем использовать следующее выражение для произведения множителей

$$\prod_i a_i = \left(\prod_i sign(a_i) \right) \cdot \exp\left(\sum_i \log\left(|a_i| \right) \right), \quad \textbf{(Ошибка! Текст указанного стиля в документе отсутствует.}.109)$$

Введем функцию

$$\tanh\left(\frac{x}{2}\right) = \frac{\exp(x)-1}{\exp(x)+1}, \quad \textbf{(Ошибка! Текст указанного стиля в документе отсутствует.}.110)$$

и получим равенство для логарифма отношения апостериорной вероятности

$$L(c_k \mid \mathbf{r}) = \log\left(\frac{p(r_k \mid c_k = 0)}{p(r_k \mid c_k = 1)} \right) + \log \prod_{m \in M(k)} \frac{1 + \prod_{n' = N(m) \setminus k} \left(q_{mn'}(0) - q_{mn'}(1) \right)}{1 - \prod_{n' = N(m) \setminus k} \left(q_{mn'}(0) - q_{mn'}(1) \right)}.$$

(Ошибка! Текст указанного стиля в документе отсутствует..111)

Используя определения $\delta q_{mn'} = q_{mn'}(0) - q_{mn'}(1)$ и $\mathrm{LLR}\left(q_{mn'}\right) = \log\left(\frac{q_{mn'}(0)}{q_{mn'}(1)} \right)$, простая подстановка дает $\delta q_{mn'} = \tanh\left(\frac{\mathrm{LLR}\left(q_{mn'}\right)}{2} \right)$. Тогда равенство (3.3.25) можно представить в следующем виде

$$L(c_k \mid \mathbf{r}) = 2\frac{r_k}{\sigma^2} + \sum_{m \in M(k)} \log \frac{1 + s_{mk} \exp(A_{mk})}{1 - s_{mk} \exp(A_{mk})}, \quad \textbf{(Ошибка! Текст указанного стиля в документе отсутствует.}.112)$$

где коэффициенты s_{mk} и A_{mk} определяются как

$$s_{mk} = \prod_{n' = N(m) \setminus k} sign(\delta q_{mn'}) = \prod_{n' = N(m) \setminus k} sign\left(\mathrm{LLR}\left(q_{mn'}\right) \right), \quad \textbf{(Ошибка! Текст указанного стиля в документе отсутствует.}.113)$$

$$A_{mk} = \sum_{n' = N(m) \setminus k} \log\left(\left| \tanh\left(\frac{\mathrm{LLR}\left(q_{mn'}\right)}{2} \right) \right| \right). \quad \textbf{(Ошибка! Текст указанного стиля в документе отсутствует.}.114)$$

Легко видеть, что коэффициент s_{mk} содержит информацию о знаке сообщения, т.е. «жестком» решении, а коэффициент A_{mk} об абсолютном значении сообщения, т.е. надежности этого решения.

Равенство (3.3.26) можно упростить

$$L(c_k \mid \mathbf{r}) = 2\frac{r_k}{\sigma^2} - \sum_{m \in M(k)} s_{mk} \log\left(-\tanh\left(\frac{A_{mk}}{2}\right)\right). \quad \textbf{(Ошибка! Текст указанного стиля в документе отсутствует.}.115)$$

Вводя обозначение $\psi(x) = \log\left(\left|\tanh\left(\frac{x}{2}\right)\right|\right)$, получим

$$L(c_k \mid \mathbf{r}) = 2\frac{r_k}{\sigma^2} - \sum_{m \in M(k)} s_{mk}\psi(A_{mk}), \quad \textbf{(Ошибка! Текст указанного стиля в документе отсутствует.}.116)$$

где $A_{mk} = \sum_{n' = N(m)\setminus k} \log\left(\left|\tanh\left(\frac{\mathrm{LLR}\left(q_{mn'}\right)}{2}\right)\right|\right) = \sum_{n' = N(m)\setminus k} \psi\left(\mathrm{LLR}\left(q_{mn'}\right)\right)$. График функции $\psi(x)$ представлен на Рис. 29.

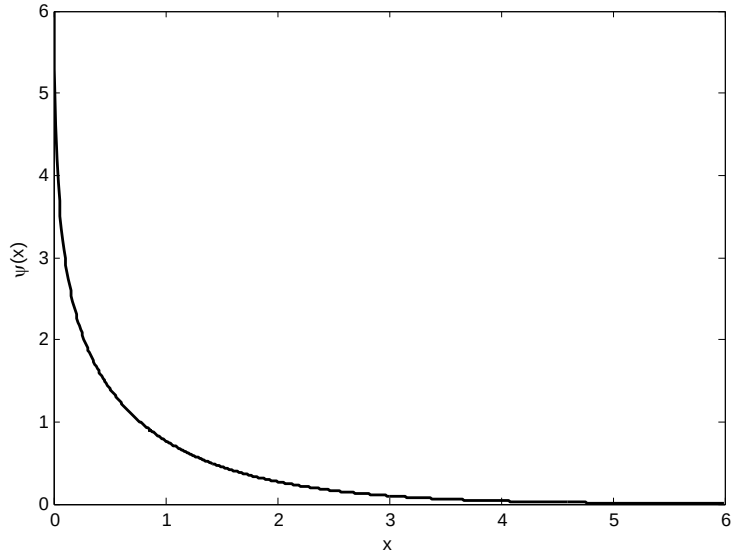

Рис. 29. График функции $\psi(x)$ для $x > 0$

Отметим, что функция $\psi(x)$ является обратной, т.е. $\psi^{-1}(x) = \psi(x)$ для $x > 0$.

Псевдокод алгоритма декодирования кодов с малой плотностью проверки на четность в области логарифма вероятности можно записать следующим образом:

Инициализация

Переменная $\mathrm{LLR}\left(q_{mk}\right)$ инициализируются следующим образом

$$\mathrm{LLR}\left(q_{mk}\right)=\log\left(\frac{q_{mk}(0)}{q_{mk}(1)}\right)=\frac{2r_k}{\sigma^2}$$ (**Ошибка! Текст указанного стиля в документе отсутствует.**.117)

Вычисление сообщений проверочных узлов

Для каждого ребра исходящего из проверочного узла с индексом $m\in M(k)$ вычисляется сообщение $R_{mk}=s_{mk}\psi(A_{mk})$, где $A_{mk}=\sum_{n'=N(m)\setminus k}\psi\left(\mathrm{LLR}\left(q_{mn'}\right)\right)$.

Вычисление сообщений битовых узлов

Для каждого ребра исходящего из битового узла с индексом $k=0,...,n-1$ вычисляется сообщение, $\mathrm{LLR}\left(q_{mk}\right)=\mathrm{LLR}\left(q_k\right)-R_{mk}$, где $\mathrm{LLR}\left(q_k\right)=\sum_{m\in M(k)}R_{mk}$.

Вычисление выражений проверки на четность

Для каждого элемента кодовой последовательности с индексом $k=0,...,n-1$ с помощью $\mathrm{LLR}\left(q_k\right)$ вычисляется «жесткое» решение

$$c_k=\begin{cases}0, \mathrm{LLR}\left(q_k\right)>0\\ 1, \mathrm{LLR}\left(q_k\right)\le 0\end{cases}.$$ (**Ошибка! Текст указанного стиля в документе отсутствует.**.118)

Если проверка на четность $\mathbf{c}\cdot\mathbf{H}^T=\mathbf{0}$ не выполняется, и число итераций не превысило максимального значения, процедура декодирования повторяется.

Заметим, что алгоритм на основе логарифма вероятности является вычислительно устойчивым по сравнению с алгоритмом декодирования в области вероятности. Стоит также отметить, что для декодирования принятой последовательности требуется информация о мощности аддитивного шума σ^2.

Упрощение алгоритма декодирования кодов с малой плотностью проверки на четность возможно при использовании различных аппроксимаций функции $\psi(\cdot)$. Например, используя простейшую аппроксимацию

$$\psi\left(\sum_i \psi(x_i)\right) \approx \min_i x_i \qquad \text{(Ошибка! Текст указанного стиля в документе отсутствует..119)}$$

можно получить следующее упрощение выражения (3.3.30)

$$L(c_k \mid \mathbf{r}) = 2\frac{r_k}{\sigma^2} - \sum_{m \in M(k)} s_{mk} \min_{n' = N(m)\setminus k} \mathrm{LLR}\left(q_{mn'}\right). \qquad \text{(Ошибка! Текст указанного стиля в документе отсутствует..120)}$$

В этом случае сообщение исходящее из проверочного соответствует входящему ребру с минимальным значением $\mathrm{LLR}\left(q_{mn'}\right)$. Поскольку вычисление функции $\min(\cdot)$ можно проводить с точностью до постоянного множителя, постоянный множитель мощности шума σ^2 можно опустить при инициализации $\mathrm{LLR}\left(q_{mk}\right)$. Это дает дополнительное упрощение алгоритма декодирования кодов с малой плотностью проверки на четность, т.к. не требует оценки σ^2 на приемнике.

3.4. Методы эффективного кодирования кодов с малой плотностью проверки на четность

В этом разделе рассматривается вычислительно эффективный алгоритм кодирования кодов с малой плотностью проверки на четность. Сокращение сложности кодирования достигается за счет использования разреженности проверочной матрицы кода. Пусть проверочная матрица кода размерности $m \times n$ задается матрицей **H**. По определению линейного блокового кода, любая кодовая последовательность **c** удовлетворяет следующему равенству

$$\mathbf{H}\mathbf{c}^T = \mathbf{0}. \qquad \text{(Ошибка! Текст указанного стиля в документе отсутствует..121)}$$

На первом этапе с помощью алгоритма последовательного исключения система уравнений приводится к равносильной системе с проверочной матрицей треугольного вида, представленной на Рис. 30.

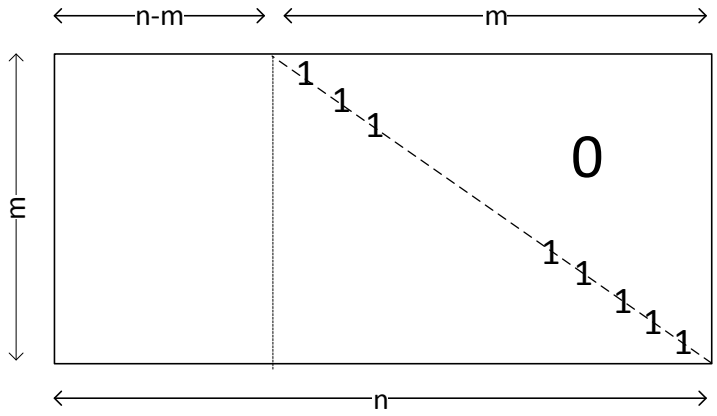

Рис. 30 Эквивалентная проверочная матрица кода треугольного вида

Пусть кодовая последовательность задается объединением систематического $\mathbf{u}=(u_0,u_1,...,u_{k-1})$ и проверочного $\mathbf{p}=(p_0,p_1,...,p_{n-k-1})$ векторов – $\mathbf{c}=[\mathbf{s},\mathbf{p}]$. Тогда проверочная последовательность $\mathbf{p}$ длины $m=n-k$ бит может быть получена из уравнения (3.4.2) последовательно с помощью метода обратной подстановки

$$p_l=\sum_{j=0}^{k-1}H_{l,j}u_j+\sum_{j=0}^{l-1}H_{l,j+k}p_j \qquad \textbf{(Ошибка! Текст указанного стиля в документе отсутствует.}.122)$$

Сложность процедуры кодирования, главным образом определяется преобразованием проверочной матрицы к треугольному виду, что требует порядка $O(n^3)$ операций и обратной подстановкой для получения проверочных бит, что требует порядка $O(n^2)$ операций. Поскольку после преобразования проверочной матрицы к треугольному виду, эквивалентная матрица в общем случае не является разреженной, фактическое число требуемых операций для кодирования может быть значительным.

Рассмотрим другой подход кодирования использующий разреженность проверочной матрицы кода. Предположим, что проверочную матрицу с помощью элементарных операций перестановки столбцов и строк можно привести к следующему виду

$$\mathbf{H}=\begin{bmatrix}\mathbf{A} & \mathbf{B} & \mathbf{T}\\ \mathbf{C} & \mathbf{D} & \mathbf{E}\end{bmatrix}, \qquad \textbf{(Ошибка! Текст указанного стиля в документе отсутствует.}.123)$$

где размеры матриц **A**, **B**, **T**, **C**, **D** и **E** равны $m-g\times n-m$, $m-g\times g$, $m-g\times m-g$, $g\times n-m$, $g\times g$ и $m-g\times g$ соответственно (см. Рис. 31).

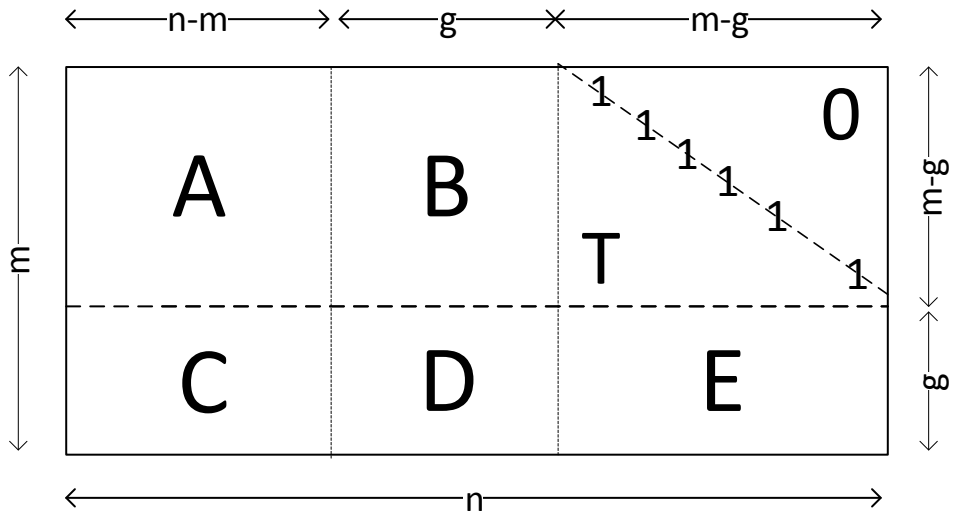

Рис. 31 Эквивалентная проверочная матрица кода приблизительно треугольного вида

Умножая матрицу **H** слева на матрицу

$$\begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{E}\mathbf{T}^{-1} & \mathbf{I} \end{bmatrix}, \qquad \text{(Ошибка! Текст указанного стиля в документе отсутствует..124)}$$

можно получить

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} & \mathbf{T} \\ -\mathbf{E}\mathbf{T}^{-1}\mathbf{A}+\mathbf{C} & -\mathbf{E}\mathbf{T}^{-1}\mathbf{B}+\mathbf{D} & \mathbf{0} \end{bmatrix}. \qquad \text{(Ошибка! Текст указанного стиля в документе отсутствует..125)}$$

Пусть $\mathbf{c}=[\mathbf{s},\mathbf{p}_1,\mathbf{p}_2]$ кодовая последовательность состоящая из систематической части $\mathbf{s}$ и двух проверочных частей $\mathbf{p}_1$ и $\mathbf{p}_2$ длиной g и $m-g$ соответственно. Тогда из равенств (3.4.1) и (3.4.5) можно получить систему из двух уравнений

$$\begin{aligned} &\mathbf{A}\mathbf{s}^T+\mathbf{B}\mathbf{p}_1^T+\mathbf{T}\mathbf{p}_2^T=\mathbf{0} \\ &\left(-\mathbf{E}\mathbf{T}^{-1}\mathbf{A}+\mathbf{C}\right)\mathbf{s}^T+\left(-\mathbf{E}\mathbf{T}^{-1}\mathbf{B}+\mathbf{D}\right)\mathbf{p}_1^T=\mathbf{0} \end{aligned}. \qquad \text{(Ошибка! Текст указанного стиля в документе отсутствует..126)}$$

Пусть $\mathbf{\Phi}=-\mathbf{E}\mathbf{T}^{-1}\mathbf{B}+\mathbf{D}$ является невырожденной, тогда из выражения (3.4.6) можно получить первую проверочную последовательность

$$\mathbf{p}_1^T = \mathbf{\Phi}^{-1}\left(-\mathbf{E}\mathbf{T}^{-1}\mathbf{A} + \mathbf{C}\right)\mathbf{s}^T. \qquad \text{(Ошибка! Текст указанного стиля в документе отсутствует..127)}$$

Поскольку матрица $\mathbf{\Phi}^{-1}\left(-\mathbf{E}\mathbf{T}^{-1}\mathbf{A} + \mathbf{C}\right)$ может быть вычислена заранее, значение вектора $\mathbf{p}_1^T$ может быть получено с помощью $O(g \times k)$ операций.

Сложность вычисления $\mathbf{p}_1^T$ можно сократить с помощью поэтапного вычисления вектора $\mathbf{p}_1^T$. На первом этапе данного метода вычисляется произведение $\mathbf{A}\mathbf{s}^T$, требующее $O(n)$ операций в силу разрежённости матрицы $\mathbf{A}$. На втором этапе, полученный вектор умножают на матрицу $\mathbf{T}^{-1}$. Поскольку $\mathbf{T}$ треугольная матрица и $\mathbf{T}^{-1}\mathbf{A}\mathbf{s}^T = \mathbf{y}^T$ эквивалентно решению системы уравнений $\mathbf{A}\mathbf{s}^T = \mathbf{T}\mathbf{y}^T$, вектор $\mathbf{y}^T$ может быть получен с помощью обратной подстановки с помощью $O(n)$ операций. Произведение $\mathbf{C}\mathbf{s}^T$ может быть также получено за $O(n)$ операций в силу разрежённости матрицы $\mathbf{C}$. На последнем этапе вычисляется произведение $\mathbf{\Phi}^{-1}\left(-\mathbf{E}\mathbf{T}^{-1}\mathbf{A}\mathbf{s}^T + \mathbf{C}\mathbf{s}^T\right)$, что требует порядка $O\left(g^2\right)$ операций из-за общего (не разреженного) вида матрицы $\mathbf{\Phi}^{-1}$. Таким образом. Общее число операций требуемых для вычисления $\mathbf{p}_1^T$ равно $O\left(n + g^2\right)$. Аналогичным образом можно получить значения второго проверочного вектора $\mathbf{p}_2$ с помощью выражения

$$\mathbf{p}_2^T = -\mathbf{T}^{-1}\left(\mathbf{A}\mathbf{s}^T + \mathbf{B}\mathbf{p}_1^T\right)^{-1}, \qquad \text{(Ошибка! Текст указанного стиля в документе отсутствует..128)}$$

требующее порядка $O(n)$ операций.

Контрольные вопросы

1. **Основные критерии декодирования помехоустойчивых кодов.** Декодирование по критерию минимального расстояния для двоично-симметричного канала и двоичного канала с аддитивным белым гауссовским шумом.

2. **Основные задачи теории помехоустойчивого канального кодирования.** Расстояние и вес Хэмминга. Статистическая модель системы связи. Классификация схем помехоустойчивого кодирования. Границы помехоустойчивости и их геометрическая интерпретация.

3. **Введение в алгебру.** Построение поля $GF(p^m)$. Понятие векторного пространства.

4**. Определение линейного блокового кода.** Порождающая и проверочная матрицы. Процедура кодирования систематического кода. Соотношение между порождающей и проверочной матрицами. Процедуры выкалывания, расширения и выбрасывания для модификации линейных блоковых кодов.

5. **Свойства линейных блоковых кодов.** Минимальное расстояние линейного блокового кода. Связь минимального расстояния кода и количества исправляемых ошибок. Геометрическая интерпретация.

6. **Синдромное декодирование линейных блоковых кодов.** Понятие синдрома. Стандартное расположение. Таблица синдромов.

7. **Границы помехоустойчивости линейных блоковых кодов.** Границы Синглтона и Хэмминга.

8. **Примеры построения линейных блоковых кодов.** Коды Хэмминга и их свойства. Минимальное расстояние кодов Хэмминга.

9. **Определение циклического кода.** Алгебраическая связь вектора и его циклического сдвига. Свойства циклических кодов. Порождающая матрица циклического кода. Систематические циклические коды.

10. **Кодирование циклических кодов.** Схемная реализация кодирования циклических кодов. Ускоренные методы кодирования циклических кодов. Практическое применение циклических кодов.

11. **Декодирование циклических кодов.** Вычисление синдрома циклических кодов. Теорема Меггита. Структурная схема декодер Меггита.

12. **Определение кодов Рида-Соломона.** Построение и схемная реализация процедуры кодирования.

13. **Декодирование кодов Рида-Соломона.** Декодер Питерсона– Горенстейна–Цилера.

14. **Определение сверточного кода.** Диаграмма состояний, решетчатая диаграмма, ребро, путь. Процедура кодирования. Завершение кодирования в нулевое состояние и сверхточный код с циклической структурой. Простейшие примеры сверточного кодирования.

15. **Декодирование сверточных кодов.** Метрика ребра, частичного пути и пути. Алгоритм Витерби. Вычислительная сложность алгоритма Витерби. Пример декодирования Витерби для простейших сверточных кодов.

16. **Практические особенности применения сверточных кодов в системах связи.** Процедура блочного и сверточного интерливинга. Повышение скорости кодирования сверточных кодов с помощью процедуры выкалывания.

17. **Декодирование сверточных кодов с мягкими решениями**. Вычисление LLR для 16-КАМ модуляции.

18. **Турбо кодирование.** Структурная схема процедуры кодирования, на примере параллельного соединения двух сверточных кодов. Декодирования по критерию максимума апостериорной информации. BCJR алгоритм декодирования.

19. **Коды с малой плотностью проверки на четность.** Методы построения и описания кодов с помощью двудольного графа. Регулярные коды. Построение кодов Галлагера и Мак-Кея. Декодирование кодов с малой плотностью проверки на четность.

Литература

1. Прокис Дж. Цифровая связь. Пер. с англ. / Под ред. Д. Д. Кловского. – М.: Радио и связь, 2000. – 800 с.

2. Феер К. Беспроводная цифровая связь. Методы модуляции и расширения спектра. Пер. с англ. – М.: Радио и связь, 2000. – 520 с.

3. Блейхут Р. Теория и практика кодов, контролирующих ошибки. – М.: Мир, 1986. – 576 с.

4. Кларк Дж., мл., Кейн Дж. Кодирование с исправлением ошибок в системах цифровой связи: Пер. с англ. – М.: Радио и связь, 1987. – 392 с.

5. Витерби А. Д., Омура Дж. К. Принципы цифровой связи и кодирования. – М.: Радио и связь, 1982. – 536 с.

6. T. K. Moon, Error Correction Coding: Mathematical Methods and Algorithms, Wiley 2005. – p. 756.

7. Мак-Вильямс Ф. Дж., Слоэн Н.Дж.А. Теория кодов, исправляющих ошибки. – М.: Связь, 1979. – 687 с.

8. Sklar, Bernard. Digital Communications: Fundamentals and Applications. –Englewood Cliffs, N.J.: Prentice-Hall, 1988. – p. 1079.

9. Clark, George C. Jr. and J. Bibb Cain. Error-Correction Coding for Digital Communications. – New York: Plenum Press, 1981. – p. 436.

10. Lin, Shu and Daniel J. Costello, Jr. Error Control Coding: Fundamentals and Applications. – Englewood Cliffs, N.J.: Prentice-Hall, 1983. – p. 603.

11. Peterson, W. Wesley and E. J. Weldon, Jr. Error-correcting Codes, 2nd ed. –Cambridge, Mass.: MIT Press, 1972. – p. 572.

12. van Lint, J. H. Introduction to Coding Theory. – New York: Springer-Verlag, 1982. – p. 241.

13. Proakis, John G. Digital Communications, 3rd ed. – New York: McGraw-Hill, 1995. – p. 1024.

14. Blahut, Richard E. Theory and Practice of Error Control Codes. Reading, Mass.: Addison-Wesley, 1983, – p.105.

15. Lang, Serge. Algebra. Third Edition. Reading, Mass.: Addison-Wesley, 1993. – p. 914.