ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ
Государственное образовательное учреждение
высшего профессионального образования
Санкт-Петербургский государственный университет
аэрокосмического приборостроения

С. В. Федоренко

Методы быстрого декодирования линейных блоковых кодов

Монография

Санкт-Петербург
2008

УДК 519.725
ББК 32.811.4
Ф33

Рецензенты:
кафедра информатики и информационной безопасности
Санкт-Петербургского государственного университета
путей сообщения;
доцент Санкт-Петербургского государственного политехнического
университета, кандидат технических наук *П. В. Трифонов*

Утверждено
редакционно-издательским советом университета
в качестве научного издания

Федоренко С. В.

Ф33 Методы быстрого декодирования линейных блоковых кодов: монография / С. В. Федоренко. – СПб.: ГУАП, 2008. – 199 с.: ил.
ISBN 978-5-8088-0316-9

В монографии рассмотрены и исследованы методы кодирования и декодирования линейных блоковых кодов, а также связанные с ними алгоритмы вычисления дискретного преобразования Фурье. Предложены новые методы комбинаторного и алгебраического декодирования, принципиально новые алгоритмы вычисления дискретного преобразования Фурье над конечным полем, введен новый тип кодовых решеток.

Монография может быть полезна студентам, аспирантам, инженерам и исследователям в области теории кодов, исправляющих ошибки.

УДК 519.725
ББК 32.811.4

ISBN 978-5-8088-0316-9

Содержание

Предисловие

Теория кодов, исправляющих ошибки, возникла в середине XX века. Дальнейшим развитием теории было ее разделение на два направления, рассматривающих блоковые коды и сверточные коды. Теория блоковых кодов, в свою очередь, содержит комбинаторную и алгебраическую теории кодирования. В 60-е годы прошлого века были введены комбинаторные и алгебраические алгоритмы декодирования. Комбинаторные алгоритмы декодирования применимы ко всем линейным блоковым кодам и обеспечивают исправление ошибок до корректирующей способности кода, однако имеют относительно большую сложность. Алгебраические алгоритмы декодирования проще комбинаторных алгоритмов и также обеспечивают исправление ошибок до конструктивной корректирующей способности кода, но они применимы только для кодов, имеющих алгебраическую структуру. Приложение теории сверточных кодов к блоковым кодам привело к появлению новых методов декодирования блоковых кодов по кодовым решеткам.

Корректирующие коды получили широкое применение в задачах передачи, записи, хранения и защиты информации. В настоящее время блоковые коды представлены в многочисленных технических приложениях, например, в стандартах CCSDS 101.0-B-6 (Consultative Committee for Space Data Systems) и IEEE 802.16 (The Institute of Electrical and Electronics Engineers).

Применение корректирующих кодов приводит к необходимости организации эффективных алгоритмов декодирования. Согласно монографии Р. Блейхута, "под быстрыми алгоритмами мы понимаем детальное описание вычислительной процедуры, которая не является очевидным способом вычисления выхода по данному входу" [8]. Первые быстрые алгоритмы были опубликованы в 60-е годы прошлого века. Заметим, что если предложенные в монографии алгоритмы для алгебраического декодирования включают такие традиционные быстрые процедуры, как вычисление дискретного преобразования Фурье и вычисление наибольшего

общего делителя, то методы комбинаторного декодирования могут быть названы быстрыми только в смысле вышеупомянутого определения Р. Блейхута.

Предлагаемая монография организована следующим образом.

В первой главе рассмотрены общие методы комбинаторного декодирования линейных блоковых кодов, которые не опираются на алгебраические свойства кода и применимы ко всем линейным кодам. Введены декодирование по обобщенным информационным совокупностям и табличные методы декодирования, а также проводится асимптотический анализ сложности декодирования.

Во второй главе рассмотрены методы декодирования, основанные на алгебраических свойствах кодов и состоящие в решении уравнений и/или систем уравнений с полиномиальной сложностью. В данной главе описаны алгебраическое декодирование по укороченным кодам (по обобщенным информационным совокупностям), "генетическая" связь различных алгебраических методов декодирования и декодирование в надкодах.

Третья глава посвящена быстрому вычислению дискретного преобразования Фурье над конечным полем, являющемуся самостоятельной задачей, приложения которой тесно связаны с алгоритмами кодирования и декодирования алгебраических кодов. Рассмотрены также быстрые алгоритмы вычисления корней многочлена и вычисление синдрома для алгебраических кодов.

В четвертой главе, посвященной декодированию блоковых кодов по кодовым решеткам, рассмотрены описание кодов с заданной группой симметрии, построение хороших циклических замкнутых решеток для квадратично-вычетных кодов, введен новый тип звездных решеток, подходящих для описания и декодирования кода Голея и кодов Рида – Соломона.

1. Комбинаторное декодирование линейных блоковых кодов

Под комбинаторным декодированием линейных блоковых кодов следует понимать алгоритмы декодирования, состоящие в отборе по некоторому правилу множества кодовых слов, в котором затем производится поиск декодированного варианта принятого вектора [25]. Комбинаторные алгоритмы не опираются на алгебраические свойства кода и применимы ко всем линейным кодам. В главе рассматривается декодирование по обобщенным информационным совокупностям, табличные методы декодирования и асимптотический анализ сложности декодирования.

1.1. Декодирование по обобщенным информационным совокупностям

Декодирование по информационным совокупностям, предложенное в работе [89], применимо ко всем линейным кодам и имеет различные модификации, исследованные в работах [86, 91, 83], а именно перестановочное декодирование и декодирование с помощью покрывающих полиномов. Для ряда лучших линейных кодов, в частности, для квадратично-вычетных и квазициклических кодов декодирование по информационным совокупностям является самым простым из известных методов декодирования [19].

В работах [22, 24, 84, 35, 59], по сути дела, рассмотрены обобщения декодирования по информационным совокупностям.

В настоящем разделе вводится метод декодирования, предложенный Круком и Федоренко [28] и основанный на понятии обобщенной информационной совокупности. Метод, сводящий декодирование исходного кода к нескольким декодированиям его укорочений, был предложен Думером [16], который также оценил асимптотику сложности декодирования по минимуму расстояния с использованием укорочений кодов [59]. В разделе рассмотрено

декодирование при исправлении ошибок кратности до $\lfloor (d-1)/2 \rfloor$, где d — минимальное расстояние кода. В параграфе 1.1.1 описывается понятие обобщенной информационной совокупности и предлагается основанный на этом понятии алгоритм. В параграфе 1.1.2 изучается связь задачи построения декодера по обобщенным информационным совокупностям и задачи построения покрытий из кодовых слов, в частности, построения покрытия Турана, а также даны примеры построения покрытий из кодовых слов. В параграфе 1.1.3 приведены таблицы декодеров по обобщенным информационным совокупностям.

1.1.1. Понятие обобщенной информационной совокупности. Алгоритм декодирования

Для введения понятия обобщенной информационной совокупности рассмотрим необходимые определения и обозначения.

Пусть $\mathcal{G}$ — линейный (n,k,d)-код над полем $\mathrm{GF}(q)$ (конечное поле порядка q) в метрике Хэмминга. Пронумеруем позиции кодового слова числами из множества $N = \{1,2,...,n\}$. Множество чисел $\gamma = \{j_1,\ldots,j_k\}$, $1 \le j_1 < \ldots < j_k \le n$, называется информационной совокупностью кода, если задание компонент кодового слова с номерами из γ однозначно определяет это слово. Для произвольного вектора $\mathbf{f} = (f_1, f_2, \ldots, f_n)$ длины n и множества $J = \{j_1, j_2, \ldots, j_s\}$, $1 \le j_1 < \ldots < j_s \le n$, определим вектор $\mathbf{f}(J) = (f_{j_1}, f_{j_2}, \ldots, f_{j_s})$ как подвектор длины s вектора $\mathbf{f}$. Аналогично для произвольной матрицы $M = [m_1|\cdots|m_n]$ со столбцами m_i, $i \in [1,n]$, и множества J определим матрицу $M(J) = [m_{j_1}|\cdots|m_{j_s}]$ со столбцами m_j, $j \in J$, как подматрицу матрицы M, составленную из столбцов этой матрицы с номерами из J. Пусть $\mathbf{b}$ — принятое из канала слово: $\mathbf{b} = \mathbf{c} + \mathbf{e}$, $\mathbf{c}$ — переданный кодовый вектор, а $\mathbf{e}$ — вектор ошибок. Информационная совокупность γ называется свободной от ошибок для вектора ошибок $\mathbf{e}$, если $\mathbf{e}(\gamma) = 0$. Если во множестве информационных совокупностей $\Gamma = \{\gamma\}$ для любого вектора ошибок $\mathbf{e}$, вес Хэм-

минга которого $W(\mathbf{e}) \leq t$, найдется свободная от ошибок информационная совокупность, то множество Γ будет достаточным для исправления ошибок кратности до t. Алгоритм декодирования по информационным совокупностям при этом состоит в переборе и кодировании по информационным совокупностям из множества Γ и выборе кодового вектора, находящегося на расстоянии t или менее от принятого вектора $\mathbf{b}$.

Сложность декодирования по информационным совокупностям определяется мощностью множества Γ, достаточного для декодирования ошибок из множества E_t — ошибок кратности до $\lfloor (d-1)/2 \rfloor$.

Для каждой информационной совокупности γ обозначим через $\overline{\gamma}$ множество позиций, дополнительных к γ: $\overline{\gamma} = N \backslash \gamma$ (мощность множества $\overline{\gamma}$ $|\overline{\gamma}| = n - k = r$). Тогда, чтобы множество $\Gamma = \{\gamma\}$ было достаточным для исправления ошибок из E_t, необходимо, чтобы множество $\overline{\Gamma} = \{\overline{\gamma}\}$ было $M(n, r, t)$-покрытием (т. е. таким множеством r-подмножеств множества N, что любое t-подмножество N содержится хотя бы в одном r-подмножестве $\overline{\gamma}$).

Введем понятие обобщенной информационной совокупности.

Обобщенной (m, Δ)-информационной совокупностью J назовем множество номеров позиций кодового слова

$$J = \{j_1, \ldots, j_m\}, \quad 1 \leq m \leq n, \quad 0 \leq \Delta \leq k$$

такое, что $\operatorname{rank} G(J) = k - \Delta$. Очевидно, что информационная совокупность $\gamma = \{j_1, \ldots, j_k\}$ является обобщенной $(k, 0)$-информационной совокупностью. Если J есть (m, Δ)-информационная совокупность, то любому подвектору $\mathbf{c}(J)$, $\mathbf{c} \in \mathcal{G}$, соответствует список из не более чем q^{Δ} кодовых слов, совпадающих с вектором $\mathbf{c}(J)$ на позициях из множества J. Через $L[\mathbf{b}(J)]$ обозначим список из кодовых слов, совпадающих с вектором $\mathbf{b}$ на множестве J.

Отыскание (m, Δ)-информационной совокупности, свободной от ошибок, не дает при $\Delta > 0$ декодированного варианта $\widehat{\mathbf{c}}$

принятого слова $\mathbf{b}$, но дает список $L[\mathbf{b}(J)]$ слов, среди которых этот вариант присутствует. Нахождение декодированного варианта в списке может быть осуществлено, например, перебором по словам из списка, результатом которого является такой вектор $\widehat{\mathbf{c}} \in L[\mathbf{b}(J)]$, что расстояние Хэмминга $d(\mathbf{b}, \widehat{\mathbf{c}}) \leq t$. В противном случае производится отказ от декодирования.

Матрицу $G(J)$ можно рассматривать как порождающую матрицу кода $\mathcal{G}(J)$, полученного выкалыванием позиций множества $\overline{J} = N \backslash J$. Пусть d_J — расстояние кода $\mathcal{G}(J)$. Если число ошибок в подвекторе $\mathbf{b}(J)$ не превышает $t_J = \lfloor (d_J - 1)/2 \rfloor$, то, декодируя вектор $\mathbf{b}(J)$ в коде $\mathcal{G}(J)$, можно получить подвектор $\mathbf{c}(J)$, свободный от ошибок. В этом случае среди векторов списка $L[\mathbf{c}(J)]$ имеется переданный вектор $\mathbf{c}$.

Множество обобщенных информационных совокупностей $\Gamma = \{J\}$ является достаточным для декодирования ошибок из E_t, если для любого вектора $\mathbf{b} = \mathbf{c} + \mathbf{e} \quad (\mathbf{e} \in E_t)$ во множестве Γ найдется обобщенная информационная совокупность J, декодирование которой дает вектор $\mathbf{c}(J)$.

Алгоритм декодирования по множеству обобщенных информационных совокупностей $\Gamma = \{J_i\}, \quad i \in [1, s]$, можно записать в виде последовательности шагов.

Шаг 1. Для каждой $J_i \in \Gamma$ выполняем следующие действия.

1.1. Производим декодирование вектора $\mathbf{b}(J_i)$ в вектор $\widehat{\mathbf{c}}(J_i)$.

1.2. Образуем список $L[\widehat{\mathbf{c}}(J_i)]$ векторов кода $\mathcal{G}$.

Шаг 2. Образуем объединение списков

$$L_\Gamma[\mathbf{b}] = \bigcup_{i=1}^{s} L[\widehat{\mathbf{c}}(J_i)].$$

Шаг 3. Выбираем в $L_\Gamma[\mathbf{b}]$ такой вектор $\widehat{\mathbf{c}}$, что $d(\mathbf{b}, \widehat{\mathbf{c}}) \leq t$; иначе считаем, что произошла неисправляемая ошибка.

Задача построения множества $\Gamma = \{J\}$, достаточного для декодирования ошибок из E_t в коде $\mathcal{G}$, затрудняется необходимостью

оценивать расстояние и число информационных символов укороченных кодов $\mathcal{G}(J)$. В дальнейшем рассмотрим декодирование по обобщенным информационным совокупностям, основанное на методе укорочения кода, предложенном Хелгертом и Стинаффом [79].

Пусть $\mathbf{a}$ — кодовое слово линейного (n, k, d)-кода $\mathcal{G}$, а $W(\mathbf{a})$ и $I_{\mathbf{a}}$ — его вес и множество нулевых позиций соответственно. Тогда [79, 15] укороченный код $\mathcal{G}(I_{\mathbf{a}})$ является $(n - W(\mathbf{a}), k - 1)$-кодом с расстоянием

$$d_{I_{\mathbf{a}}} \geq d - \left\lfloor \frac{q-1}{q} W(\mathbf{a}) \right\rfloor, \qquad (1)$$

если $W(\mathbf{a}) < \dfrac{q}{q-1} d$.

Множество $I_{\mathbf{a}}$ задает таким образом $(n - W(\mathbf{a}), 1)$-обобщенную информационную совокупность. Если вес вектора ошибок $\mathbf{e}(I_{\mathbf{a}})$ не превышает $t_{I_{\mathbf{a}}} = \lfloor (d_{I_{\mathbf{a}}} - 1)/2 \rfloor$, то соответствующий список $L[\widehat{\mathbf{c}}(I_{\mathbf{a}})]$, получаемый в алгоритме декодирования по обобщенным информационным совокупностям, содержит переданный вектор и состоит из q векторов.

Поэтому для построения множества обобщенных информационных совокупностей, обеспечивающего декодирование ошибок из E_t, достаточно выбрать множество $A = \left\{ \mathbf{a} \mid W(\mathbf{a}) < \dfrac{q}{q-1} d \right\}$ кодовых слов кода $\mathcal{G}$, удовлетворяющее следующему свойству:

Если для любого вектора $\mathbf{e} \quad (\mathbf{e} \in E_t)$ найдется слово $\mathbf{a} \in A$ такое, что вес вектора $\mathbf{e}(\overline{I}_{\mathbf{a}})$

$$W(\mathbf{e}(\overline{I}_{\mathbf{a}})) \geq \tau_{\mathbf{a}} = t - t_{I_{\mathbf{a}}},$$

где $\overline{I}_{\mathbf{a}} = N \backslash I_{\mathbf{a}}$, то множество $\Gamma = \{ I_{\mathbf{a}} \mid \mathbf{a} \in A \}$ будет достаточным для декодирования ошибок из E_t. Множество кодовых слов A, удовлетворяющее вышеуказанному свойству, будем называть в дальнейшем τ-покрытием.

1.1.2. Построение τ-покрытий из кодовых слов

Код можно рассматривать как покрытие в следующем смысле. Каждому кодовому слову $\mathbf{a}$ соответствует множество ненулевых позиций $\overline{I}_{\mathbf{a}} = N \backslash I_{\mathbf{a}}$. Набор этих подмножеств при соблюдении некоторых условий будет образовывать M-покрытие.

Для построения τ-покрытия используем покрытие Турана. Покрытием Турана $T(n, t, \tau)$ называется такое семейство τ-подмножеств множества N, что всякое t-подмножество множества N содержит по крайней мере одно из этих τ-подмножеств [42].

Кодовое слово $\mathbf{a} \in \mathcal{G}$ назовем вложенным в множество чисел $V \subset N$, если $\overline{I}_{\mathbf{a}} \subset V$. Произвольное множество кодовых слов $\{\mathbf{f}\} \in \mathcal{G}$ называется вложенным в множество чисел $V \subset N$, если каждое слово $\mathbf{f} \in \{\mathbf{f}\}$ вложено в множество $V \subset N$. Все слова (n, k, d)-кода $\mathcal{G}$ с проверочной матрицей H, вложенные в некоторое множество V, образуют линейный $(|V|, |V| - \operatorname{rank} H(V), d_V)$-код $\mathcal{G}_V$ с проверочной матрицей $H(V)$, где $d_V \geq d$.

Сформулируем несколько вспомогательных утверждений.

Лемма 1.1. *Пусть* $\mathbf{a}$ *— слово двоичного кода* $\mathcal{G}$ *веса* $2d$ *и* $\operatorname{rank} H(\overline{I}_{\mathbf{a}}) = 2d - \Delta$, *тогда все слова веса* d *кода* $\mathcal{G}_V$ *с проверочной матрицей* $H(\overline{I}_{\mathbf{a}})$ *образуют* $M(2d, d, \Delta - 1)$*-покрытие, вложенное в множество* $\overline{I}_{\mathbf{a}}$.

Доказательство. Список векторов кода $\mathcal{G}_V$ состоит из нулевого слова, слова $\mathbf{a}$ (веса $2d$) и $2^{\Delta} - 2$ слов веса d. Поскольку ранг любой $\Delta \times (\Delta - 1)$ подматрицы порождающей матрицы кода $\mathcal{G}_V$ не превышает $\Delta - 1$, то для любых $\Delta - 1$ позиций кода $\mathcal{G}_V$ найдется слово $\mathbf{c} \in \mathcal{G}_V, \quad W(\mathbf{c}) = d$, которое имеет нули на этих позициях. Следовательно, слово $\mathbf{a} + \mathbf{c}$ веса d имеет единицы на этих позициях. Итак, множество из $2^{\Delta} - 2$ слов веса d кода $\mathcal{G}_V$ образует $M(2d, d, \Delta - 1)$-покрытие. Ч.т.д.

Лемма 1.2. *Для любого* q*-ичного* (n, k)*-кода* $\mathcal{G}$ *и множества* $V \subset N$

$$|V| - \operatorname{rank} G(V) = r - \operatorname{rank} H(\overline{V}),$$

где $\overline{V} = N \backslash V$; $r = n - k$; G *и* H — *порождающая и проверочная матрицы кода* $\mathcal{G}$ *соответственно.*

Доказательство. Пусть H_V — проверочная матрица кода $\mathcal{G}(V)$. Очевидно, что $|V| - \text{rank } G(V) = \text{rank } H_V$. Кроме того, строки матрицы H_V, дополненные нулями на множестве $\overline{V}$, ортогональны коду $\mathcal{G}$. Остальные $r - \text{rank } H_V$ строк матрицы H линейно независимы на множестве $\overline{V}$. Поэтому $r - \text{rank } H_V = \text{rank } H(\overline{V})$. Ч.т.д.

Следствие. Для любого самодуального $(n, n/2)$*-кода* $\mathcal{G}$ *с порождающей матрицей* G *и произвольного множества* $V \subset N$, $|V| = n/2$:

$$\text{rank } G(V) = \text{rank } G(\overline{V}).$$

Для построения декодера по обобщенным информационным совокупностям (n, k)-кода $\mathcal{G}$, исправляющего t ошибок, необходимо построить τ-покрытие кодовыми словами веса w, $\tau = t - \lfloor (d_{I_{\mathbf{a}}} - 1)/2 \rfloor$, где $d_{I_{\mathbf{a}}}$ определяется неравенством (1).

Разобьем множество $N = \{1, 2, \ldots, n\}$ на l (возможно, пересекающихся) подмножеств $V_1, \ldots, V_l$: $N = \bigcup_{i=1}^{l} V_i$ так, чтобы объединение всех τ-подмножеств множеств V_i для $i \in [1, l]$ являлось $T(n, t, \tau)$-покрытием (большинство известных покрытий Турана построено таким образом). Затем для каждого V_i, $i \in [1, l]$, строим $M(|V_i|, W(\mathbf{a}_i), \tau)$-покрытие из кодовых слов. Объединяя эти M-покрытия, образуем искомое τ-покрытие.

Используя доказанные выше леммы, можно сократить перебор при построении $M(|V_i|, W(\mathbf{a}_i), \tau)$-покрытий.

Для этого в качестве множеств $V_1, \ldots, V_l$ должны быть выбраны множества $\overline{I}_{\mathbf{a}_1}, \ldots, \overline{I}_{\mathbf{a}_l}$, где $\overline{I}_{\mathbf{a}_j}$ — множество ненулевых позиций вектора $\mathbf{a}_j \in \mathcal{G}$ и $W(\mathbf{a}_j) = 2d$, $j \in [1, l]$. Тогда в силу леммы 1.1 векторы веса d кода $\mathcal{G}_{V_j}$, $j \in [1, l]$, образуют $M(2d, d, \Delta_j - 1)$-покрытие из кодовых слов, вложенное во множество V_j, где $\Delta_j = 2d - \text{rank } H(V_j)$.

Множество покрытий $M(2d, d, \Delta_j - 1)$ задает декодер кода $\mathcal{G}$ по обобщенным информационным совокупностям, исправляющий t ошибок, если

$$\min_j \quad (\Delta_j - 1) \geq \tau.$$

Согласно следствию, для самодуальных кодов достаточно построить покрытие из кодовых слов на множестве V, $|V| = n/2$, так как требуемое покрытие на множестве $\overline{V}$ совпадает с покрытием для V с точностью до перенумерации позиций.

Для примера построим покрытие из кодовых слов для (24, 12, 8)-кода Голея $\mathcal{G}$.

Пример. Покрытие для кода Голея. Для любого вектора $\mathbf{a} \in \mathcal{G}$, $W(\mathbf{a}) = 8$, существует код $\mathcal{G}(I_{\mathbf{a}})$ с параметрами (16, 11, 4), эквивалентный коду Хэмминга. Пусть V_1 и V_2 — такие множества, что $|V_1| = |V_2| = 12$ и $V_1 \bigcup V_2 = \{1, 2, ..., 24\}$. Строим покрытие Турана $T(24, 3, 2)$. Известно [42], что это покрытие является оптимальным и состоит из всех 2-подмножеств множества V_1 и всех 2-подмножеств множества V_2. Затем построим $M(12, 8, 2)$-покрытие из кодовых слов. Пусть $\mathbf{a}_1$ — произвольное кодовое слово минимального веса. Найдем другое кодовое слово $\mathbf{a}_2$, для которого $W(\mathbf{a}_2) = 8$ и $W(\mathbf{a}_1 + a_2) = 8$. Три слова $\{\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_1 + \mathbf{a}_2\}$ вложены в множество $V_1 = \overline{I}_{\mathbf{a}_1} \bigcup \overline{I}_{\mathbf{a}_2}$ и образуют оптимальное $M(12, 8, 2)$-покрытие, представляющее собой все ненулевые слова $(12, 2, 8)$-кода. Из следствия леммы 1.2 следует, что все ненулевые слова, вложенные в множество V_2, также образуют $M(12, 8, 2)$-покрытие. Таким образом, мощность оптимального τ-покрытия для кода Голея равна $s = 6$. Выпишем это покрытие.

Пусть порождающая матрица кода Голея представлена в виде [33, рис. 16.4] $G = [I_{12}|C]$, где I_{12} — единичная матрица с размерами 12×12; C — циркулянтная матрица [33, (16.50)] с циркулянтом $C(X) = 1 + x + x^3 + x^4 + x^5 + x^6 + x^8$ [33, (16.51)]. Тогда оптимальное τ-покрытие из кодовых слов $\{\mathbf{a}_1, \ldots, \mathbf{a}_6\}$ имеет вид:
$\overline{I}_1 = \{1, 13, 14, 16, 17, 18, 19, 21\}$, $\overline{I}_2 = \{1, 9, 10, 11, 13, 15, 16, 17\}$,
$\overline{I}_3 = \{9, 10, 11, 14, 15, 18, 19, 21\}$, $\overline{I}_4 = \{2, 3, 4, 5, 7, 8, 12, 20\}$,

$\overline{I}_5 = \{4,5,6,8,20,22,23,24\}$, $\overline{I}_6 = \{2,3,6,7,12,22,23,24\}$,
где вектор $\mathbf{a}_i$ имеет единицы на позициях из множества $\overline{I}_i$ и нули на остальных позициях.

Алгоритм декодирования, основанный на этом покрытии, имеет сложность порядка 150 операций над векторами длины $r = 12$. Кроме того, необходимо запомнить 6 проверочных матриц кода. Заметим, что для реализации оптимального алгоритма декодирования (24, 12, 8)-кода Голея по информационным совокупностям, использующего множество из 14 информационных совокупностей [101], требуется около 350 таких операций.

Рассмотрим другой пример.

Пример. Покрытие для (48, 24, 12)-кода. При построении τ-покрытия для квадратично-вычетного (48, 24, 12)-кода $\mathcal{G}$ кодовыми словами минимального веса приходится проводить машинный перебор, хотя аналитические рассуждения позволяют сократить его сложность до величины, кратной числу кодовых слов минимального веса. Все укороченные коды $\mathcal{G}(I_{\mathbf{a}})$, $W(\mathbf{a}) = 12$, имеют параметры $(36, 23, 6)$. Покрытие Турана $T(48, 5, 3)$ состоит из всех 3-подмножеств множества V_1 и всех 3-подмножеств множества V_2, для которых $|V_1| = |V_2| = 24$ и $V_1 \bigcup V_2 = \{1, 2, ..., 48\}$. Построим $M(24, 8, 3)$-покрытие из кодовых слов. Для этого в соответствие с леммой 1.1 выберем два таких кодовых слова минимального веса $\mathbf{a}_1$ и $\mathbf{a}_2$, что

$$\begin{cases} W(\mathbf{a}_1 + \mathbf{a}_2) = 24 \\ \text{rank } H(V_1) = 20 \end{cases}, \qquad (2)$$

где V_1 — множество ненулевых позиций вектора $\mathbf{a}_1 + \mathbf{a}_2$; $G = H$ — порождающая (она же проверочная) матрица кода $\mathcal{G}$.
Все слова веса 12, вложенные в множество V_1, образуют требуемое M-покрытие. Аналогично из следствия леммы 1.2 получим покрытие на множестве V_2, так как rank $H(V_1)$ = rank $H(V_2)$ и вектор из всех единиц лежит в коде $\mathcal{G}$. Итак, для построения покрытия необходимо выбрать слова $\mathbf{a}_1$ и $\mathbf{a}_2$, для которых выполняется условие (2). Для этого зафиксируем произвольный вектор

$\mathbf{a}_1$ и будем порождать слова минимального веса $\mathbf{a}_2$ с помощью дробно-линейной группы подстановок [33, (16.30)] до тех пор, пока условие (2) не выполнится. Сложность перебора не превышает утроенного числа слов минимального веса кода $\mathcal{G}$ (17296).

Отметим, что построенное τ-покрытие состоит из $s = 28$ кодовых слов и будет оптимальным для алгоритма построения τ-покрытия, если справедлива гипотеза Турана о минимальной мощности $T(2p, 5, 3)$-покрытия. Декодирование укороченных (36, 23, 6)-кодов упрощается в связи с тем, что все они разбиваются на три группы эквивалентности [46]. Таким образом, достаточно запомнить таблицы соответствия между синдромами и лидерами смежных классов только для трех укороченных кодов, а для декодирования остальных использовать перестановки. Каждая таблица состоит из 667 слов длины 36. Итак, сложность декодирования (48, 24, 12)-кода определяется s-кратно выполняемой процедурой, состоящей из: перестановки, вычисления синдрома для укороченного кода, обращения к таблице и порождения списка. Реализация алгоритма декодирования по обобщенным информационным совокупностям имеет сложность порядка 1500 операций над векторами длины $r = 24$ вместо 4500 операций, требуемых для реализации алгоритма декодирования по информационным совокупностям.

1.1.3. Таблица для декодирования по обобщенным информационным совокупностям

Параметры декодеров для ряда двоичных кодов при декодировании по обобщенным информационным совокупностям приведены в табл. 1. Через w обозначен вес кодовых слов (n, k, d)-кода (или число позиций, на которые производится укорочение), задающих укороченные $(n_{\mathbf{a}}, k_{\mathbf{a}}, d_{\mathbf{a}})$-коды, а через s — число укороченных кодов, используемых для декодирования. Если для декодирования основного кода используются укороченные коды с разными параметрами, то в таблице их параметры приведены в разных

строках, но под одним номером. Декодеры квадратично-вычетных кодов 1–6 и кодов Боуза – Чоудхури – Хоквингема (БЧХ) 7–9 [44, 45, 28] построены с использованием алгоритма построения τ-покрытий, а декодеры квазициклических кодов 10–13 используют произвольные укорочения.

Таблица 1

Параметры декодеров по обобщенным информационным совокупностям

N	n	k	d	w	$n_{\mathbf{a}}$	$k_{\mathbf{a}}$	$d_{\mathbf{a}}$	s
1	18	9	6	6	12	8	3	3
2	24	12	8	8	16	11	4	6
3	32	16	8	8	24	15	4	12
4	42	21	10	10	32	20	5	15
				12	30	20	4	1
5	48	24	12	12	36	23	6	28
6	48	24	12	16	32	23	4	87
7	31	11	11	11	20	10	6	3
				12	19	10	5	6
				16	15	10	3	3
8	63	39	9	9	54	38	5	21
9	63	24	15	15	48	23	8	15
				16	47	23	7	45
10	30	10	10	10	20	10	6	1
				10	20	10	5	2
11	33	11	11	4	29	11	8	1
				6	27	11	7	10
				11	22	11	5	1
12	39	13	11	7	31	13	7	6
				8	32	13	7	2
				13	26	13	5	1
13	54	36	7	9	45	36	3	17
				9	45	35	3	7

Поясним содержание таблицы. Например, из четвертой строки следует, что для декодирования кода (42, 21, 10) достаточно декодировать 16 укороченных кодов: 15 (32, 20, 5)-кодов и 1 (30, 20, 4)-код. Декодеры для кодов 1 и 2 построены на оптимальных покрытиях. Декодеры для кодов 3, 5, 8, 9 построены на покрытиях, оптимальных для алгоритма, предложенного в параграфе 1.1.2.

Сложность алгоритма декодирования по обобщенным информационным совокупностям определяется не только числом обобщенных информационных совокупностей, но и сложностью декодирования соответствующих им укороченных кодов. Это затрудняет сравнение различных декодеров по обобщенным информационным совокупностям в общем виде. Такое сравнение приходится проводить отдельно для каждого конкретного кода с учетом практических требований, предъявляемых к его реализации. Однако декодирование большинства укороченных кодов, указанных в табл. 1, может быть реализовано на основе таблиц синдромов умеренного объема, порядка 2–3 тыс. слов длины $n - w$. Отметим также, что большинство кодов из таблицы 1 не являются БЧХ-кодами, поэтому к ним не применимы эффективные алгебраические процедуры декодирования типа алгоритма Берлекэмпа. Для БЧХ-кодов (в частности, (63, 39, 9) и (63, 24, 15)) целесообразно, как правило, использовать декодер Берлекэмпа. Представляется, однако, что приведенные покрытия из кодовых слов для БЧХ-кодов имеют самостоятельный интерес.

1.2. Табличное декодирование

Общие методы декодирования линейных кодов, применимые ко всех кодам, будут рассмотрены в следующем разделе. Метод с наименьшей сложностью декодирования в асимптотике рассмотрен в работе [48]. Однако наименьшая сложность декодирования для кодов с ограниченной длиной наблюдается при применении табличного декодирования.

Два метода декодирования линейных кодов имеют наименьшую сложность: декодирование по укороченным кодам (обобщенным информационным совокупностям) [28] и декодирование в надкодах [69, 48, 82, 75]. При рассмотрении декодирования по информационным совокупностям [24] различные совокупности символов, имеющие разные надежности, используются для восстановления кодового слова и дальнейшего сравнения его с принятым вектором в соответствующей метрике. При декодировании в надкодах разные методы применяются при декодировании принятого вектора в различных надкодах. Достоинство этого метода состоит в том, что сложность декодирования в надкоде много меньше сложности декодирования в основном коде. Результатом декодирования в надкоде может быть список кодовых слов этого надкода. Пересечение всех списков дает в результате искомое решение.

Объединим оба метода в табличный алгоритм, сложность которого меньше сложности исходных методов, и рассмотрим три основных типа табличных алгоритмов с жестким принятием решений, а также их различные комбинации.

Основная идея табличного декодирования предложена в работе автора [27] и состоит в том, чтобы организовать информацию об ошибках в таблицы, причем каждая таблица состоит из множества строк. Каждая строка имеет адрес и содержит вектор или список подвекторов, соответствующих вектору ошибок. Адрес, в свою очередь, соответствует синдрому или его подвектору.

Введем некоторые обозначения и определения.

Обозначения и определения. Пусть $\mathcal{C}$ — двоичный (n, k, d)-код, где n — длина кода; k — размерность и число информационных символов; d — минимальное расстояние кода. Число проверочных символов кода обозначим через $r = n - k$, корректирующую способность кода — через $t = \left\lfloor \dfrac{d-1}{2} \right\rfloor$, а скорость кода — через $R = \dfrac{k}{n}$. Порождающую матрицу кода $\mathcal{C}$ обозначим через G, а проверочную матрицу — через H. Для простоты изложения в

этом параграфе рассмотрим только двоичные коды, тогда принятый вектор есть $\mathbf{r} = \mathbf{c} + \mathbf{e} \bmod 2$, где $\mathbf{c} \in \mathcal{C}$ — кодовое слово, а $\mathbf{e} \in F_2^n$ — вектор ошибок. Декодированный вариант принятого вектора обозначим как $\widehat{\mathbf{c}}$, а соответствующий вектор ошибок — как $\widehat{\mathbf{e}}$.

Порождающая матрица $(G(\gamma))^{-1} G$ для информационной совокупности γ должна иметь единичную подматрицу на позициях из множества γ, т. е. $\operatorname{rank} G(\gamma) = k$.

Синдром для принятого вектора $\mathbf{r}$ определяется как $\mathbf{s} = \mathbf{r}H^T = \mathbf{e}H^T$. Вес Хэмминга вектора $\mathbf{f}$ обозначается как $wt(\mathbf{f})$, а расстояние Хэмминга между векторами $\mathbf{f}_1$ и $\mathbf{f}_2$ — как $d(\mathbf{f}_1, \mathbf{f}_2)$. Взаимосвязь между этими определениями записывается как $d(\mathbf{f}_1, \mathbf{f}_2) = wt(\mathbf{f}_1 + \mathbf{f}_2)$.

1.2.1. Синдромное декодирование

Для описания табличного декодирования линейных кодов вначале опишем синдромное декодирование. Построим таблицу, где в адресе таблицы записан синдром, а в строке — вектор ошибок. Код определяется его проверочной матрицей H.

Таблица есть множество строк $\mathcal{T} = \{T(\mathbf{s})\}$, где строка $T(\mathbf{s})$ с адресом $\mathbf{s}$ есть

$$T(\mathbf{s}) = \{\mathbf{f} \in F_2^n \mid \mathbf{f}H^T = \mathbf{s}, wt(\mathbf{f}) \leq t\}.$$

Определим мощность строки $\|T(\mathbf{s})\| \in \{0, 1\}$ как число векторов, содержащихся в этой строке.

Таблица вычисляется заранее и запоминается.

Тогда алгоритм декодирования имеет следующий вид.

Алгоритм синдромного декодирования

Шаг 1. Вычисляется синдром $\mathbf{s} = \mathbf{r}H^T$.

Шаг 2. По адресу $\mathbf{s}$ в таблице $\mathcal{T} = \{T(\mathbf{s})\}$ находится список кандидатов декодирования

$$T(\mathbf{s}) = \{\widehat{\mathbf{e}}\}.$$

Шаг 3. Декодированный вариант переданного слова есть

$$\widehat{\mathbf{c}} = \mathbf{r} + \widehat{\mathbf{e}},$$

если строка $T(\mathbf{s})$ существует; иначе объявляется отказ от декодирования.

Очевидно, что размер таблицы есть $\sum_{i=0}^{t} \binom{n}{i}$, а алгоритм синдромного декодирования является алгоритмом декодирования с ограниченным расстоянием до корректирующей способности кода t. Заметим, что если в таблицу записать все лидеры смежных классов стандартной расстановки [33, 49], то таблица будет иметь объем 2^{n-k} и декодирование будет декодированием по максимуму правдоподобия. Конечно, синдромное декодирование имеем слишком большую сложность для длинных кодов.

1.2.2. Декодирование по информационным совокупностям

Метод декодирования по информационным совокупностям будет рассмотрен на примере кода со скоростью $R = \dfrac{1}{2}$, причем длина кода n — четное число и $k = \dfrac{n}{2}$. Предположим, что $\gamma_A = \{1, 2, \ldots, k\}$ и $\gamma_B = \{k+1, k+2, \ldots, n\}$ являются информационными совокупностями.

Запишем проверочные матрицы для информационных совокупностей γ_A и γ_B как

$$H_A = \left[\begin{array}{c|c} A & I \end{array}\right] \text{ и } H_B = \left[\begin{array}{c|c} I & B \end{array}\right]. \tag{3}$$

Теперь определяем две таблицы $\mathcal{T}_A$ и $\mathcal{T}_B$ следующим образом:

$$T_A(\mathbf{s}_A) = \left\{ \mathbf{f}_1 \in F_2^k \mid wt(\mathbf{f}_1) = t_1 \in \left[0, 1, 2, \ldots, \left\lfloor \frac{t}{2} \right\rfloor\right], \right.$$

$$d(\mathbf{s}_A, \mathbf{f}_1 A^T) = 0, 1, 2, \ldots, t - t_1 \Big\},$$

где $\mathbf{s}_A = \mathbf{e}H_A^T = (\mathbf{e}_1 \mid \mathbf{e}_2) \left[\frac{A^T}{I}\right] = \mathbf{e}_1 A^T + \mathbf{e}_2$;

$$T_B(\mathbf{s}_B) = \left\{ \mathbf{f}_2 \in F_2^k \mid wt(\mathbf{f}_2) = t_2 \in \left[0, 1, 2, \ldots, \left\lfloor \frac{t}{2} \right\rfloor\right], \right.$$

$$d(\mathbf{s}_B, \mathbf{f}_2 B^T) = 0, 1, 2, \ldots, t - t_2 \Big\},$$

где $\mathbf{s}_B = \mathbf{e}H_B^T = (\mathbf{e}_1 \mid \mathbf{e}_2) \left[\frac{I}{B^T}\right] = \mathbf{e}_1 + \mathbf{e}_2 B^T$.

Заметим, что для $\mathbf{e} = (\mathbf{e}_1 \mid \mathbf{e}_2)$ имеем $\mathbf{e}(\gamma_A) = \mathbf{e}_1$ и $\mathbf{e}(\gamma_B) = \mathbf{e}_2$.

Поэтому можем различать два случая: $wt(\mathbf{e}_1) \leq wt(\mathbf{e}_2)$ и $wt(\mathbf{e}_1) > wt(\mathbf{e}_2)$. В первом случае будем использовать для декодирования таблицу $\mathcal{T}_A$, а во втором — таблицу $\mathcal{T}_B$. Так же, как и для синдромного декодирования, две таблицы вычисляются заранее и запоминаются.

Алгоритм декодирования по информационным совокупностям для табличного декодирования

Пусть принятый вектор есть

$$\mathbf{r} = \mathbf{c} + \mathbf{e} = (\mathbf{r}_1 \mid \mathbf{r}_2) = (\mathbf{c}_1 + \mathbf{e}_1 \mid \mathbf{c}_2 + \mathbf{e}_2).$$

Проверочные матрицы H_A и H_B определяются формулой (3).

Шаг 1. Вычисляются синдромы $\mathbf{s}_A$ и $\mathbf{s}_B$:

$$\mathbf{s}_A = \mathbf{r}H_A^T; \quad \mathbf{s}_B = \mathbf{r}H_B^T.$$

Шаг 2. Выбираются два списка $\{\mathbf{f}_1\}$ и $\{\mathbf{f}_2\}$ из таблиц $\mathcal{T}_A$ и $\mathcal{T}_B$:

$$T_A(\mathbf{s}_A) = \{\mathbf{f}_1\}; \quad T_B(\mathbf{s}_B) = \{\mathbf{f}_2\}.$$

Шаг 3. Формируются два списка декодированных вариантов принятого вектора:

$$\{\widehat{\mathbf{c}}_A\} = \{(\mathbf{r}_1 + \mathbf{f}_1 \mid (\mathbf{r}_1 + \mathbf{f}_1)A^T)\}; \quad \{\widehat{\mathbf{c}}_B\} = \{((\mathbf{r}_2 + \mathbf{f}_2)B^T \mid \mathbf{r}_2 + \mathbf{f}_2)\}.$$

Шаг 4. Декодированный вариант принятого вектора есть

$$\widehat{\mathbf{c}} = \arg \min_{\mathbf{a} \in \{\widehat{\mathbf{c}}_A\} \cup \{\widehat{\mathbf{c}}_B\}} d(\mathbf{a}, \mathbf{r}),$$

ближайший к принятому вектору вектор в метрике Хэмминга $d(\widehat{\mathbf{c}}, \mathbf{r})$. Если объединение списков пусто: $\|\{\widehat{\mathbf{c}}_A\} \cup \{\widehat{\mathbf{c}}_B\}\| = 0$, то объявляется отказ от декодирования.

Утверждение 1.1. *Предложенный алгоритм есть алгоритм декодирования с ограниченным расстоянием до корректирующей способности кода.*

1.2.3. Декодирование в надкодах

Надкод $\mathcal{C}_s$ кода $\mathcal{C}$ содержит все кодовые слова кода $\mathcal{C}$, т. е. $\mathcal{C} \subset \mathcal{C}_s$.

Существует много надкодов для каждого кода $\mathcal{C}$, и любое подмножество строк проверочной матрицы H основного кода является проверочной матрицей H_i надкода. Для упрощения изложения будем рассматривать в этом разделе только две проверочные матрицы надкодов

$$H = \begin{bmatrix} H_A \\ H_B \end{bmatrix}, \tag{4}$$

где H_A — $(r_A \times n)$ матрица и H_B — $(r_B \times n)$ матрица, причем $r_A + r_B = n - k$. Очевидно, что H_A — проверочная матрица надкода $\mathcal{C}_A$ и H_B — надкода $\mathcal{C}_B$. Заметим, что минимальное расстояние надкода не превышает минимального расстояния основного кода.

Для принятого вектора $\mathbf{r} = \mathbf{c} + \mathbf{e}$ синдром $\mathbf{s}$ состоит из двух подвекторов $\mathbf{s}_A$ и $\mathbf{s}_B$:

$$\mathbf{s} = (\mathbf{s}_A \mid \mathbf{s}_B) = (\mathbf{r}H_A^T \mid \mathbf{r}H_B^T).$$

Таблицы $\mathcal{T}_A$ и $\mathcal{T}_B$ для надкодов определяются как

$$T_A(\mathbf{s}_A) = \{\mathbf{f} \in F_2^n \mid \mathbf{f}H_A^T = \mathbf{s}_A, wt(\mathbf{f}) \le t\};$$

$$T_B(\mathbf{s}_B) = \{\mathbf{f} \in F_2^n \mid \mathbf{f}H_B^T = \mathbf{s}_B, wt(\mathbf{f}) \le t\}.$$

Теперь сформулируем алгоритм декодирования в надкодах следующим образом.

Алгоритм декодирования в надкодах

Проверочные матрицы H_A и H_B определяются формулой (4).

Шаг 1. Вычисляются подвектора $\mathbf{s}_A$ и $\mathbf{s}_B$ синдрома

$$\mathbf{s} = (\mathbf{s}_A \mid \mathbf{s}_B) = \mathbf{r}H^T.$$

Шаг 2. Выбираются два списка $\{\mathbf{f}_A\}$ и $\{\mathbf{f}_B\}$ из таблиц $\mathcal{T}_A$ и $\mathcal{T}_B$:

$$T_A(\mathbf{s}_A) = \{\mathbf{f}_A\}; \quad T_B(\mathbf{s}_B) = \{\mathbf{f}_B\}.$$

Шаг 3. Вычисляется пересечение списков

$$\{\widehat{\mathbf{e}}\} = \{\mathbf{f}_A\} \cap \{\mathbf{f}_B\}.$$

Шаг 4. Декодированный вариант принятого вектора есть

$$\widehat{\mathbf{c}} = \mathbf{r} + \widehat{\mathbf{e}}_{\min},$$

где $\widehat{\mathbf{e}}_{\min}$ — ближайший к принятому вектору вектор в метрике Хэмминга из списка $\{\widehat{\mathbf{e}}\}$. Если пересечение списков $\{\widehat{\mathbf{e}}\}$ пусто, то объявляется отказ от декодирования.

Утверждение 1.2. *Предложенный алгоритм есть алгоритм декодирования с ограниченным расстоянием до корректирующей способности кода.*

1.2.4. Комбинирование алгоритмов декодирования

Пусть скорость кода $R = \frac{1}{2}$, длина кода n и его размерность k — четные числа, а $\gamma_1 = \{1, 2, \ldots, k\}$, $\gamma_2 = \{k+1, k+2, \ldots, n\}$ — информационные совокупности.

Запишем следующие формулы, необходимые для организации декодирования:

$$H_1 = \left[\begin{array}{cc|cc} & A & I & 0 \\ \hline & B & 0 & I \end{array}\right];$$

$$H_2 = \left[\begin{array}{cc|c} I & 0 & C \\ \hline 0 & I & D \end{array}\right];$$

$$\mathbf{e} = (\mathbf{e}_{11} \mid \mathbf{e}_{12} \mid \mathbf{e}_{21} \mid \mathbf{e}_{22});$$

$$\mathbf{e}_1 = \mathbf{e}(\gamma_1) = (\mathbf{e}_{11} \mid \mathbf{e}_{12}); \quad \mathbf{e}_2 = \mathbf{e}(\gamma_2) = (\mathbf{e}_{21} \mid \mathbf{e}_{22});$$

$$\mathbf{s}_1 = (\mathbf{e}_{11}\mathbf{e}_{12} \mid \mathbf{e}_{21}\mathbf{e}_{22})H_1^T =$$
$$= \left((\mathbf{e}_{11}\mathbf{e}_{12})A^T + \mathbf{e}_{21} \mid (\mathbf{e}_{11}\mathbf{e}_{12})B^T + \mathbf{e}_{22}\right) = \left(\mathbf{s}_{11} \mid \mathbf{s}_{12}\right);$$

$$\mathbf{s}_2 = (\mathbf{e}_{11}\mathbf{e}_{12} \mid \mathbf{e}_{21}\mathbf{e}_{22})H_2^T =$$
$$= \left((\mathbf{e}_{21}\mathbf{e}_{22})C^T + \mathbf{e}_{11} \mid (\mathbf{e}_{21}\mathbf{e}_{22})D^T + \mathbf{e}_{12}\right) = \left(\mathbf{s}_{21} \mid \mathbf{s}_{22}\right);$$

$$T_{11}(\mathbf{s}_{11}) = \Bigg\{(\mathbf{f}_{11}\mathbf{f}_{12}) \in F_2^k \mid d(\mathbf{s}_{11}, (\mathbf{f}_{11}\mathbf{f}_{12})A^T) \le t_{21},$$
$$wt(\mathbf{f}_{11}\mathbf{f}_{12}) = t_1 \in \left[0, 1, 2, \ldots, \left\lfloor \frac{t}{2} \right\rfloor\right]\Bigg\};$$

$$T_{12}(\mathbf{s}_{12}) = \Bigg\{(\mathbf{f}_{11}\mathbf{f}_{12}) \in F_2^k \mid d(\mathbf{s}_{12}, (\mathbf{f}_{11}\mathbf{f}_{12})B^T) \le t_{22},$$
$$wt(\mathbf{f}_{11}\mathbf{f}_{12}) = t_1 \in \left[0, 1, 2, \ldots, \left\lfloor \frac{t}{2} \right\rfloor\right]\Bigg\};$$

$$T_{21}(\mathbf{s}_{21}) = \Bigg\{ (\mathbf{f}_{21}\mathbf{f}_{22}) \in F_2^k \mid d(\mathbf{s}_{21}, (\mathbf{f}_{21}\mathbf{f}_{22})C^T) \le t_{11},$$

$$wt(\mathbf{f}_{21}\mathbf{f}_{22}) = t_2 \in \left[0, 1, 2, \ldots, \left\lfloor \frac{t}{2} \right\rfloor\right] \Bigg\};$$

$$T_{22}(\mathbf{s}_{22}) = \Bigg\{ (\mathbf{f}_{21}\mathbf{f}_{22}) \in F_2^k \mid d(\mathbf{s}_{22}, (\mathbf{f}_{21}\mathbf{f}_{22})D^T) \le t_{12},$$

$$wt(\mathbf{f}_{21}\mathbf{f}_{22}) = t_2 \in \left[0, 1, 2, \ldots, \left\lfloor \frac{t}{2} \right\rfloor\right] \Bigg\}.$$

Переменные $t_{11}, t_{12}, t_{21}, t_{22}$ являются параметрами алгоритма декодирования.

Возможны два варианта расположения ошибок.

1. Если $wt(\mathbf{e}_1) \le wt(\mathbf{e}_2)$, то для декодирования выберем информационную совокупность γ_1.

2. Если $wt(\mathbf{e}_1) > wt(\mathbf{e}_2)$, то выберем информационную совокупность γ_2.

Декодирование в надкодах по информационным совокупностям

1. Для информационной совокупности γ_1:
$\left[\ A \mid I\ \right]$ — проверочная матрица для надкода $\mathcal{C}_{11}$;
$\left[\ B \mid I\ \right]$ — проверочная матрица для надкода $\mathcal{C}_{12}$.
Параметрами декодирования являются $t_{21} = t - t_1$, $t_{22} = t - t_1$.

2. Для информационной совокупности γ_2:
$\left[\ I \mid C\ \right]$ — проверочная матрица для надкода $\mathcal{C}_{21}$;
$\left[\ I \mid D\ \right]$ — проверочная матрица для надкода $\mathcal{C}_{22}$.
Параметрами декодирования являются $t_{11} = t - t_2$, $t_{12} = t - t_2$.

Таблицы записываются в следующем формате: имя_таблицы(адрес)[параметры].

Можно формально записать алгоритм декодирования следующим образом.

Пусть принятый вектор есть

$$\mathbf{r} = \mathbf{c} + \mathbf{e} = (\mathbf{r}_1 \mid \mathbf{r}_2) = (\mathbf{c}_1 + \mathbf{e}_1 \mid \mathbf{c}_2 + \mathbf{e}_2).$$

Шаг 1. $(\mathbf{s}_{11} \mid \mathbf{s}_{12}) = \mathbf{r}H_1^T$ и $(\mathbf{s}_{21} \mid \mathbf{s}_{22}) = \mathbf{r}H_2^T$.

Шаг 2.

$$T_{11}(\mathbf{s}_{11})[t_{21} = t - t_1] = \{\mathbf{f}_1\},$$

$$T_{12}(\mathbf{s}_{12})[t_{22} = t - t_1] = \{\mathbf{f}_2\},$$

$$T_{21}(\mathbf{s}_{21})[t_{11} = t - t_2] = \{\mathbf{f}_3\},$$

$$T_{22}(\mathbf{s}_{22})[t_{12} = t - t_2] = \{\mathbf{f}_4\}.$$

Шаг 3.

$$\{\mathbf{f}_5\} = \{\mathbf{f}_1\} \cap \{\mathbf{f}_2\},$$

$$\{\mathbf{f}_6\} = \{\mathbf{f}_3\} \cap \{\mathbf{f}_4\}.$$

Шаг 4.

$$\{\widehat{\mathbf{c}}_A\} = \{(\mathbf{r}_1 + \mathbf{f}_5 \mid (\mathbf{r}_1 + \mathbf{f}_5)(A^T \mid B^T))\},$$

$$\{\widehat{\mathbf{c}}_B\} = \{((\mathbf{r}_2 + \mathbf{f}_6)(C^T \mid D^T) \mid \mathbf{r}_2 + \mathbf{f}_6)\}.$$

Шаг 5. Декодированный вариант принятого вектора есть

$$\widehat{\mathbf{c}} = \arg \min_{\mathbf{a} \in \{\widehat{\mathbf{c}}_A\} \cup \{\widehat{\mathbf{c}}_B\}} d(\mathbf{a}, \mathbf{r}).$$

Если объединение списков пусто: $\|\{\widehat{\mathbf{c}}_A\} \cup \{\widehat{\mathbf{c}}_B\}\| = 0$, то объявляется отказ от декодирования.

Декодирование в подкодах по информационным совокупностям

1. Для информационной совокупности γ_1:
$\left[\begin{array}{c|c} A & I \end{array}\right]$ — проверочная матрица для подкода $\mathcal{C}_1$;
$\left[\begin{array}{c|c} B & I \end{array}\right]$ — проверочная матрица для подкода $\mathcal{C}_2$.
Параметрами декодирования являются $t_{21} = \left\lfloor \dfrac{t - t_1}{2} \right\rfloor$, $t_{22} = \left\lfloor \dfrac{t - t_1}{2} \right\rfloor$.

2. Для информационной совокупности γ_2:
$\left[\begin{array}{c|c} I & C \end{array}\right]$ — проверочная матрица для подкода $\mathcal{C}_3$;
$\left[\begin{array}{c|c} I & D \end{array}\right]$ — проверочная матрица для подкода $\mathcal{C}_4$.

Параметрами декодирования являются $t_{11} = \left\lfloor \frac{t - t_2}{2} \right\rfloor$, $t_{12} = \left\lfloor \frac{t - t_2}{2} \right\rfloor$.

Можно формально записать алгоритм декодирования следующим образом.

Пусть принятый вектор есть

$$\mathbf{r} = \mathbf{c} + \mathbf{e} = (\mathbf{r}_1 \mid \mathbf{r}_2) = (\mathbf{c}_1 + \mathbf{e}_1 \mid \mathbf{c}_2 + \mathbf{e}_2).$$

Шаг 1. $(\mathbf{s}_{11} \mid \mathbf{s}_{12}) = \mathbf{r}H_1^T$ и $(\mathbf{s}_{21} \mid \mathbf{s}_{22}) = \mathbf{r}H_2^T$.

Шаг 2.

$$T_{11}(\mathbf{s}_{11})[t_{21} = \lfloor (t - t_1)/2 \rfloor] = \{\mathbf{f}_1\},$$
$$T_{12}(\mathbf{s}_{12})[t_{22} = \lfloor (t - t_1)/2 \rfloor] = \{\mathbf{f}_2\},$$
$$T_{21}(\mathbf{s}_{21})[t_{11} = \lfloor (t - t_2)/2 \rfloor] = \{\mathbf{f}_3\},$$
$$T_{22}(\mathbf{s}_{22})[t_{12} = \lfloor (t - t_2)/2 \rfloor] = \{\mathbf{f}_4\}.$$

Шаг 3.

$$\{\mathbf{f}_5\} = \{\mathbf{f}_1\} \cup \{\mathbf{f}_2\},$$
$$\{\mathbf{f}_6\} = \{\mathbf{f}_3\} \cup \{\mathbf{f}_4\}.$$

Шаг 4.

$$\{\widehat{\mathbf{c}}_A\} = \{(\mathbf{r}_1 + \mathbf{f}_5 \mid (\mathbf{r}_1 + \mathbf{f}_5)(A^T \mid B^T))\},$$
$$\{\widehat{\mathbf{c}}_B\} = \{((\mathbf{r}_2 + \mathbf{f}_6)(C^T \mid D^T) \mid \mathbf{r}_2 + \mathbf{f}_6)\}.$$

Шаг 5. Декодированный вариант принятого вектора есть

$$\widehat{\mathbf{c}} = \arg \min_{\mathbf{a} \in \{\widehat{\mathbf{c}}_A\} \cup \{\widehat{\mathbf{c}}_B\}} d(\mathbf{a}, \mathbf{r}).$$

Если объединение списков пусто: $\|\{\widehat{\mathbf{c}}_A\} \cup \{\widehat{\mathbf{c}}_B\}\| = 0$, то объявляется отказ от декодирования.

Сравнение двух методов

Два метода декодирования — в надкодах и подкодах — по существу используют одни и те же коды, однако параметры декодирования в этих методах различные.

Для первой информационной совокупности γ_1 справедливо неравенство

$$\|T_i(s_i)[t_j = t - t_1]\| \geq \|T_i(s_i)[t_j = \lfloor (t - t_1)/2 \rfloor]\|,$$

аналогично для второй информационной совокупности γ_2

$$\|T_i(s_i)[t_j = t - t_2]\| \geq \|T_i(s_i)[t_j = \lfloor (t - t_2)/2 \rfloor]\|.$$

Для объемов результирующих списков кандидатов декодирования выполняется следующее свойство: объем результирующего списка при декодировании в надкодах не превышает объема результирующего списка при декодировании в подкодах.

Пример. Код Голея (1). $(24, 12, 8)$-код Голея $\mathcal{C}$ имеет следующее представление порождающей G и проверочной H матриц:

$$G = H = \left[\begin{array}{c|c} I & A \end{array} \right] =$$

$$= \left[\begin{array}{c|c}
1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 & 1\,1\,1\,0\,1\,1\,0\,0\,0\,1\,0\,1 \\
0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 & 1\,1\,0\,1\,1\,0\,0\,0\,1\,0\,1\,1 \\
0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0 & 1\,0\,1\,1\,0\,0\,0\,1\,0\,1\,1\,1 \\
0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0 & 0\,1\,1\,0\,0\,0\,1\,0\,1\,1\,1\,1 \\
0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0 & 1\,1\,0\,0\,0\,1\,0\,1\,1\,1\,1\,0 \\
0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0 & 1\,0\,0\,0\,1\,0\,1\,1\,1\,1\,0\,1 \\
0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0 & 0\,0\,0\,1\,0\,1\,1\,1\,1\,0\,1\,1 \\
0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0 & 0\,0\,1\,0\,1\,1\,1\,1\,0\,1\,1\,0 \\
0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0 & 0\,1\,0\,1\,1\,1\,1\,0\,1\,1\,0\,0 \\
0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0 & 1\,0\,1\,1\,1\,1\,0\,1\,1\,0\,0\,0 \\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0 & 0\,1\,1\,1\,1\,0\,1\,1\,0\,0\,0\,1 \\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1 & 1\,1\,1\,1\,0\,1\,1\,0\,0\,0\,1\,0
\end{array} \right].$$

Для этого представления выполняется свойство $A = A^T$, а таблицы для обеих информационных совокупностей совпадают: $T_{11}(\mathbf{s}_{11}) = T_{21}(\mathbf{s}_{21})$ и $T_{12}(\mathbf{s}_{12}) = T_{22}(\mathbf{s}_{22})$.

Общими параметрами декодирования для обоих методов являются $t = 3$, $t_1 = t_2 = 1$, $t_{11} = t_{12} = t_{21} = t_{22}$.

Параметры для декодирования в подкодах есть $t_{21} = 1$, а для декодирования в надкодах — $t_{21} = 2$.

Максимальное число векторов в строке таблицы для каждого метода есть

$$\max \|T_{11}(\mathbf{s}_{11})[t_{21} = 1]\| \leq 3;$$
$$\max \|T_{11}(\mathbf{s}_{11})[t_{21} = 2]\| \leq 6$$

соответственно.

1.2.5. Декодирование кодов с большой группой симметрии

Примером кодов с большой группой симметрии является класс расширенных квадратично-вычетных кодов. Расширенный квадратично-вычетный код есть циклический код с дополнительной проверкой на четность. Некоторые квадратично-вычетные коды являются самодуальными кодами, т. е. у них порождающая матрица G совпадает с проверочной матрицей H.

Более того, некоторые квадратично-вычетные коды являются квазициклическими кодами и некоторые из них могут быть представлены в виде

$$G = H = \left[\begin{array}{c|c|c} \text{A} & \text{B} & 0 \\ \hline 0 & \text{A} & \text{B} \\ \hline \text{B} & 0 & \text{A} \end{array}\right],$$

где A и B — некоторые матрицы, причем $AB^T = 0$, $AA^T = BB^T$ [66].

Табличный метод декодирования для квадратично-вычетных кодов с таким видом проверочной матрицы был введен в работе автора [67].

Пусть $3|n$, $3|k$, $3|t$,

$$\mathbf{e} = (\mathbf{e}_1 \mid \mathbf{e}_2 \mid \mathbf{e}_3), \quad t = 3l, \quad \frac{2t}{3} = 2l.$$

Тогда синдром принятого вектора $\mathbf{r}$ есть

$$\mathbf{s} = (\mathbf{r}_1 \mid \mathbf{r}_2 \mid \mathbf{r}_3)H^T =$$

$$= \left(\mathbf{e}_1 A^T + \mathbf{e}_2 B^T \mid \mathbf{e}_2 A^T + \mathbf{e}_3 B^T \mid \mathbf{e}_3 A^T + \mathbf{e}_1 B^T\right) = (\mathbf{s}_1 \mid \mathbf{s}_2 \mid \mathbf{s}_3).$$

Пусть $\mathbf{f}_i \in F_2^{k/3}$ для всех подвекторов $\mathbf{f}_i$. Тогда таблицы для декодирования можно составить следующим образом:

$$T_1(\mathbf{s}_1) = \{(\mathbf{f}_1 \mid \mathbf{f}_2) \mid \mathbf{f}_1 A^T + \mathbf{f}_2 B^T = \mathbf{s}_1, \quad wt(\mathbf{f}_1 \mid \mathbf{f}_2) < 2l\};$$

$$T_2(\mathbf{s}_2) = \{(\mathbf{f}_2 \mid \mathbf{f}_3) \mid \mathbf{f}_2 A^T + \mathbf{f}_3 B^T = \mathbf{s}_2, \quad wt(\mathbf{f}_2 \mid \mathbf{f}_3) < 2l\};$$

$$T_3(\mathbf{s}_3) = \{(\mathbf{f}_3 \mid \mathbf{f}_1) \mid \mathbf{f}_3 A^T + \mathbf{f}_1 B^T = \mathbf{s}_3, \quad wt(\mathbf{f}_1 \mid \mathbf{f}_3) < 2l\};$$

$$T_4(\mathbf{s}_1) = \{(\mathbf{f}_1 \mid \mathbf{f}_2) \mid \mathbf{f}_1 A^T + \mathbf{f}_2 B^T = \mathbf{s}_1, \quad wt(\mathbf{f}_1) = wt(\mathbf{f}_2) = l\};$$

$$T_5(\mathbf{s}_2) = \{(\mathbf{f}_2 \mid \mathbf{f}_3) \mid \mathbf{f}_2 A^T + \mathbf{f}_3 B^T = \mathbf{s}_2, \quad wt(\mathbf{f}_2) = wt(\mathbf{f}_3) = l\};$$

$$T_6(\mathbf{s}_3) = \{(\mathbf{f}_3 \mid \mathbf{f}_1) \mid \mathbf{f}_3 A^T + \mathbf{f}_1 B^T = \mathbf{s}_3, \quad wt(\mathbf{f}_1) = wt(\mathbf{f}_3) = l\}.$$

Заметим, что некоторые таблицы совпадают: $T_1(\mathbf{s}_1) = T_2(\mathbf{s}_2) = T_3(\mathbf{s}_3)$ и $T_4(\mathbf{s}_1) = T_5(\mathbf{s}_2) = T_6(\mathbf{s}_3)$.

Для любого вектора $(\mathbf{c}_i \mid \mathbf{c}_j \mid ?)$ с одним неопределенным подвектором существует не более одного кодового слова $(\mathbf{c}_i \mid \mathbf{c}_j \mid ?) \in \mathcal{C}$.

Теперь можем формально записать алгоритм декодирования следующим образом.

Пусть принятый вектор есть

$$\mathbf{r} = \mathbf{c} + \mathbf{e} = (\mathbf{r}_1 \mid \mathbf{r}_2 \mid \mathbf{r}_3) = (\mathbf{c}_1 + \mathbf{e}_1 \mid \mathbf{c}_2 + \mathbf{e}_2 \mid \mathbf{c}_3 + \mathbf{e}_3).$$

Шаг 1. $(\mathbf{s}_1 \mid \mathbf{s}_2 \mid \mathbf{s}_3) = \mathbf{r}H^T$.

Шаг 2.

$$T_1(\mathbf{s}_1) = \{(\mathbf{f}_{11} \mid \mathbf{f}_{12})\},$$
$$T_2(\mathbf{s}_2) = \{(\mathbf{f}_{22} \mid \mathbf{f}_{23})\},$$
$$T_3(\mathbf{s}_3) = \{(\mathbf{f}_{33} \mid \mathbf{f}_{31})\},$$
$$T_4(\mathbf{s}_1) = \{(\mathbf{f}_{41} \mid \mathbf{f}_{42})\},$$
$$T_5(\mathbf{s}_2) = \{(\mathbf{f}_{52} \mid \mathbf{f}_{53})\},$$
$$T_6(\mathbf{s}_3) = \{(\mathbf{f}_{63} \mid \mathbf{f}_{61})\}.$$

Шаг 3.

$$\text{If } \begin{cases} (\mathbf{f}_1 \mid \mathbf{f}_2) \in \{(\mathbf{f}_{41} \mid \mathbf{f}_{42})\} \\ (\mathbf{f}_2 \mid \mathbf{f}_3) \in \{(\mathbf{f}_{52} \mid \mathbf{f}_{53})\} \\ (\mathbf{f}_3 \mid \mathbf{f}_1) \in \{(\mathbf{f}_{63} \mid \mathbf{f}_{61})\} \end{cases} \quad \text{then } \{\mathbf{f}_A\} = \{(\mathbf{f}_1 \mid \mathbf{f}_2 \mid \mathbf{f}_3)\}.$$

$$\text{If } \begin{cases} (\mathbf{f}_1 \mid \mathbf{f}_2) \in \{(\mathbf{f}_{11} \mid \mathbf{f}_{12})\} \\ (\mathbf{f}_2 \mid \mathbf{f}_3) \in \{(\mathbf{f}_{22} \mid \mathbf{f}_{23})\} \\ (\mathbf{f}_3 \mid \mathbf{f}_1) \in \{(\mathbf{f}_{33} \mid \mathbf{f}_{31})\} \\ (\mathbf{f}_1 \mid \mathbf{f}_2 \mid \mathbf{f}_3) \in \{\mathbf{f}_A\} \end{cases} \text{then } \{\mathbf{f}_B\} = \{(\mathbf{f}_1 \mid \mathbf{f}_2 \mid \mathbf{f}_3)\}.$$

Шаг 4.

$$\{\widehat{\mathbf{c}}\} = \begin{cases} (\mathbf{r}_1 + \mathbf{f}_1 \mid \mathbf{r}_2 + \mathbf{f}_2 \mid \mathbf{r}_3 + \mathbf{f}_3) & \text{if } \{(\mathbf{f}_1 \mid \mathbf{f}_2 \mid \mathbf{f}_3)\} \in \{\mathbf{f}_B\} \\ (\mathbf{r}_1 + \mathbf{f}_1 \mid \mathbf{r}_2 + \mathbf{f}_2 \mid \mathbf{r}_3 + \mathbf{f}_3) & \text{if } \{(\mathbf{f}_i \mid \mathbf{f}_j \mid ?)\} \in \{\mathbf{f}_B\} \end{cases}.$$

Шаг 5. Декодированный вариант принятого вектора есть

$$\widehat{\mathbf{c}} = \arg \min_{\mathbf{a} \in \{\widehat{\mathbf{c}}\}} d(\mathbf{a}, \mathbf{r}).$$

Если список пуст: $\|\{\widehat{\mathbf{c}}\}\| = 0$, то объявляется отказ от декодирования.

Заметим, что множество $\{\mathbf{f}_B\}$ может включать векторы

$$(\mathbf{f}_i \mid \mathbf{f}_j \mid ?)$$

с одним неопределенным подвектором.

Пример. Код Голея (2). (24, 12, 8)-код Голея $\mathcal{C}$ имеет следующее представление порождающей G и проверочной H матриц:

$$G = H = \left[\begin{array}{c|c|c} \mathrm{A} & \mathrm{B} & 0 \\ \hline 0 & \mathrm{A} & \mathrm{B} \\ \hline \mathrm{B} & 0 & \mathrm{A} \end{array}\right] =$$

$$= \left[\begin{array}{c|c|c} 1\,0\,0\,0\,1\,1\,0\,1 & 0\,0\,1\,0\,0\,1\,1\,1 & 0\,0\,0\,0\,0\,0\,0\,0 \\ 0\,1\,0\,0\,0\,1\,1\,0 & 1\,0\,1\,1\,0\,1\,1\,0 & 0\,0\,0\,0\,0\,0\,0\,0 \\ 0\,0\,1\,0\,1\,1\,0\,0 & 1\,1\,1\,0\,1\,0\,1\,0 & 0\,0\,0\,0\,0\,0\,0\,0 \\ 0\,0\,0\,1\,1\,0\,1\,1 & 1\,1\,1\,0\,0\,1\,0\,0 & 0\,0\,0\,0\,0\,0\,0\,0 \\ \hline 0\,0\,0\,0\,0\,0\,0\,0 & 1\,0\,0\,0\,1\,1\,0\,1 & 0\,0\,1\,0\,0\,1\,1\,1 \\ 0\,0\,0\,0\,0\,0\,0\,0 & 0\,1\,0\,0\,0\,1\,1\,0 & 1\,0\,1\,1\,0\,1\,1\,0 \\ 0\,0\,0\,0\,0\,0\,0\,0 & 0\,0\,1\,0\,1\,1\,0\,0 & 1\,1\,1\,0\,1\,0\,1\,0 \\ 0\,0\,0\,0\,0\,0\,0\,0 & 0\,0\,0\,1\,1\,0\,1\,1 & 1\,1\,1\,0\,0\,1\,0\,0 \\ \hline 0\,0\,1\,0\,0\,1\,1\,1 & 0\,0\,0\,0\,0\,0\,0\,0 & 1\,0\,0\,0\,1\,1\,0\,1 \\ 1\,0\,1\,1\,0\,1\,1\,0 & 0\,0\,0\,0\,0\,0\,0\,0 & 0\,1\,0\,0\,0\,1\,1\,0 \\ 1\,1\,1\,0\,1\,0\,1\,0 & 0\,0\,0\,0\,0\,0\,0\,0 & 0\,0\,1\,0\,1\,1\,0\,0 \\ 1\,1\,1\,0\,0\,1\,0\,0 & 0\,0\,0\,0\,0\,0\,0\,0 & 0\,0\,0\,1\,1\,0\,1\,1 \end{array}\right].$$

Общие параметры для декодирования есть $t = 3$, $l = 1$.

Максимальное число векторов в строке для каждой таблицы есть

$$\max \|T_1(\mathbf{s}_1)\| \leq 2;$$
$$\max \|T_4(\mathbf{s}_1)\| \leq 8$$

соответственно.

1.2.6. Декодирование квадратично-вычетных кодов

В работе автора [66] введена конструкция, описывающая порождающие (проверочные) матрицы квадратично-вычетных кодов в виде

$$G = H = \begin{bmatrix} H_1 & H_2 & 0 \\ 0 & H_1 & H_2 \\ H_2 & 0 & H_1 \end{bmatrix},$$

где H_1 и H_2 — подматрицы матрицы H. Детальное описание свойств этой конструкции приведено в параграфе 4.4.1.

Введенная конструкция позволяет исследовать свойства квадратично-вычетных кодов, разрабатывать алгоритмы декодирования, а также строить по матрице G кодовые решетки [62].

Рассмотренную конструкцию можно использовать для организации табличного декодирования. Табличное декодирование, введенное в работе [27], является модификацией декодирования по информационным совокупностям и заключается в уменьшении времени выполнения вычислений за счет увеличения объема памяти, организованной в таблицы. Асимптотика этого метода рассмотрена в работе [48].

Пусть n — длина кода $\mathcal{G}$, k — число информационных символов кода, d — минимальное расстояние Хэмминга кода, $\mathbf{b} = \mathbf{a} + \mathbf{e} \pmod 2$ — принятый вектор, где $\mathbf{a} \in \mathcal{G}$, $\mathbf{e} = (\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3)$ — вектор ошибок с весом Хэмминга $W(\mathbf{e}) \leq [(d-1)/2]$; $G = H = \begin{bmatrix} H_1 & H_2 & 0 \\ 0 & H_1 & H_2 \\ H_2 & 0 & H_1 \end{bmatrix}$ — порождающая и проверочная матрица кода $\mathcal{G}$, H_1 и H_2 — проверочные матрицы подкодов $\mathcal{G}_1$ и $\mathcal{G}_2$ соответственно, $\mathbf{S} = (\mathbf{S}_1, \mathbf{S}_2, \mathbf{S}_3) = (\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3)H^T$ — синдром от принятого вектора $\mathbf{b} = (\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3)$.

Так как код $\mathcal{G}$ квазициклический, то сдвигу на $\dfrac{n}{3}$ символа принятого вектора $(\mathbf{b}_2, \mathbf{b}_3, \mathbf{b}_1)$ соответствует сдвиг синдрома $(\mathbf{S}_2, \mathbf{S}_3, \mathbf{S}_1)$.

Каждая таблица состоит из $2^{\frac{k}{3}}$ строк, в которых содержится список векторов. Декодированный вариант принятого вектора вычисляется как результат действий (например, объединение и пересечение) со строками таблиц.

Обозначим через σ произвольный вектор длины $\dfrac{k}{3}$, через ε — длины $\dfrac{n}{3}$, тогда таблица $T_{A,L} = \{T_{A,L}(\sigma)\}$ есть совокупность строк для всех возможных векторов σ, где $T_{A,L}(\sigma) = \{\varepsilon \,|\, \varepsilon A^T = \sigma, W(\varepsilon) \in L\}$ — строка с номером σ; A — $\left(\dfrac{k}{3} \times \dfrac{n}{3}\right)$ матрица, L —

множество неотрицательных чисел.

Аналогично определим еще два типа таблиц: $T_{A,B,M} = \{T_{A,B,M}(\sigma)\}$ как совокупность строк для всех возможных векторов σ, где $T_{A,B,M}(\sigma) = \{(\varepsilon_1, \varepsilon_2) \,|\, \varepsilon_1 A^T + \varepsilon_2 B^T = \sigma,\ (W(\varepsilon_1), W(\varepsilon_2)) \in M\}$ — строка с номером σ; A, B — $\left(\dfrac{k}{3} \times \dfrac{n}{3}\right)$ матрицы; M — множество неотрицательных пар чисел, и $T_{A,B,L} = \{T_{A,B,L}(\sigma)\}$ как совокупность строк для всех возможных векторов σ, где $T_{A,B,L}(\sigma) = \{\varepsilon \,|\, \varepsilon A^T = \sigma,\ \varepsilon B^T = 0,\ W(\varepsilon) = L\}$ — строка с номером σ; L — неотрицательное число.

Кроме того, через $\widehat{\mathbf{S}} = (\widehat{\mathbf{S}}_1, \widehat{\mathbf{S}}_2, \widehat{\mathbf{S}}_3) = ((\varepsilon_1, \varepsilon_2, \varepsilon_3) + \mathbf{b})H^T$ обозначим синдром после наложения на принятый вектор $\mathbf{b}$ вектора ошибок $(\varepsilon_1, \varepsilon_2, \varepsilon_3)$ (одна или две компоненты которого могут быть нулевыми векторами).

Пример. Табличное декодирование (48, 24, 12)-кода. Пусть подматрицы проверочной матрицы (48, 24, 12)-кода имеют вид

$$H_1 = \begin{bmatrix} 1000001001010101 \\ 0100001000101010 \\ 0010001001100001 \\ 0001001001100110 \\ 0000101000110111 \\ 0000010001110011 \\ 0000000100101011 \\ 0000000011111101 \end{bmatrix}, \quad H_2 = \begin{bmatrix} 0011110110011011 \\ 1000101101000110 \\ 1110001101000001 \\ 1000011110100000 \\ 1100110010101011 \\ 0110000010011001 \\ 0110010101010010 \\ 1010101111100100 \end{bmatrix}.$$

Таким образом, подкоды $\mathcal{G}_1$ и $\mathcal{G}_2$ эквивалентны и имеют наибольшее возможное расстояние — 4. При весе вектора ошибок $W(\mathbf{e}) = 5$ (случай $W(\mathbf{e}) \leq 5$ поглощается) возможные варианты распределения ошибок сведены в табл. 2. Так как при декодировании предполагается использовать квазициклические сдвиги принятого вектора, то в таблицу занесены варианты распределения ошибок с точностью до квазициклического сдвига на $\dfrac{n}{3}$ символа.

Таблица 2

Варианты распределения ошибок

#	$W(e_1)$	$W(e_2)$	$W(e_3)$
1	0	2	3
2	2	0	3
3	1	1	3
4	0	1	4
5	1	0	4
6	1	2	2
7	0	0	5

Таблицы организованы следующим образом:

$$\begin{array}{ll} T_1 = T_{H_1,\{0,1,2\}}; & T_2 = T_{H_2,\{0,1,2\}}; \\ T_3 = T_{H_1,H_2,\{(1,1)\}}; & T_4 = T_{H_1,H_2,\{(1,2)\}}; \\ T_5 = T_{H_1,\{0,1,2,3\}}; & T_6 = T_{H_2,\{0,1,2,3\}}; \\ T_7 = T_{H_1,\{0,1\}}; & T_8 = T_{H_2,\{0,1\}}; \\ T_9 = T_{H_1,\{4\}}; & T_{10} = T_{H_2,\{4\}}; \\ T_{11} = T_{H_1,H_2,\{5\}}; & T_{12} = T_{H_2,H_1,\{5\}}. \end{array}$$

Данные о максимальной величине списка векторов в строке для каждой таблицы приведены ниже:

$$\begin{array}{ll} \max|T_1(\sigma)| \le 2; & \max|T_2(\sigma)| \le 2; \\ \max|T_3(\sigma)| \le 4; & \max|T_4(\sigma)| \le 15; \\ \max|T_5(\sigma)| \le 5; & \max|T_6(\sigma)| \le 5; \\ \max|T_7(\sigma)| \le 1; & \max|T_8(\sigma)| \le 1; \\ \max|T_9(\sigma)| \le 14; & \max|T_{10}(\sigma)| \le 14; \\ \max|T_{11}(\sigma)| \le 1; & \max|T_{12}(\sigma)| \le 1. \end{array}$$

Алгоритм декодирования (48, 24, 12)-кода состоит из последовательного выполнения семи шагов. Результатом каждого шага является либо единственный вариант вектора ошибок $\{\widehat{\mathbf{e}}_i\}$,

$i \in [1,7]$, либо пустое множество. Алгоритм завершается на произвольном шаге при получении такого варианта вектора ошибок, что $W(\widehat{\mathbf{e}}_i) \le 5$.

Шаг 1. По принятому вектору $\mathbf{b} = (\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3)$ вычисляется синдром $\mathbf{S} = (\mathbf{S}_1, \mathbf{S}_2, \mathbf{S}_3) = (\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3)H^T$. По значению синдрома $\mathbf{S}_1$ в таблице T_2 находится список векторов $T_2(\mathbf{S}_1)$. Для каждого вектора $\varepsilon \in T_2(\mathbf{S}_1)$ вычисляется значение синдрома $\widehat{\mathbf{S}} = (\widehat{\mathbf{S}}_1, \widehat{\mathbf{S}}_2, \widehat{\mathbf{S}}_3) = ((0, \varepsilon, 0) + \mathbf{b})H^T$ и пересечение списков $T_6(\widehat{\mathbf{S}}_2) \cap T_5(\widehat{\mathbf{S}}_3))$, содержащее не более одного вектора. Результатом будет вектор

$$\widehat{\mathbf{e}}_{10} = (0, T_2(\mathbf{S}_1), T_6(\widehat{\mathbf{S}}_2) \cap T_5(\widehat{\mathbf{S}}_3)).$$

Шаг повторяется еще два раза для квазициклических сдвигов принятого вектора $(\mathbf{b}_2, \mathbf{b}_3, \mathbf{b}_1)$ и $(\mathbf{b}_3, \mathbf{b}_1, \mathbf{b}_2)$ с результатом $\widehat{\mathbf{e}}_{11}$ и $\widehat{\mathbf{e}}_{12}$ соответственно. Результатом выполнения всего шага будет список вариантов вектора ошибок $\{\widehat{\mathbf{e}}_1\} = \{\widehat{\mathbf{e}}_{10}, \widehat{\mathbf{e}}_{11}, \widehat{\mathbf{e}}_{12}\}$, содержащий не более одного вектора.

Шаги 2–6 аналогичны шагу 1. Результатом их выполнения будут векторы

$$\begin{aligned}
&\widehat{\mathbf{e}}_{2i} = (T_1(\mathbf{S}_1), 0, T_6(\widehat{\mathbf{S}}_2) \cap T_5(\widehat{\mathbf{S}}_3));\\
&\widehat{\mathbf{e}}_{3i} = (T_3(\mathbf{S}_1), T_6(\widehat{\mathbf{S}}_2) \cap T_5(\widehat{\mathbf{S}}_3));\\
&\widehat{\mathbf{e}}_{4i} = (0, T_8(\mathbf{S}_1), T_{10}(\widehat{\mathbf{S}}_2) \cap T_9(\widehat{\mathbf{S}}_3));\\
&\widehat{\mathbf{e}}_{5i} = (T_7(\mathbf{S}_1), 0, T_{10}(\widehat{\mathbf{S}}_2) \cap T_9(\widehat{\mathbf{S}}_3));\\
&\widehat{\mathbf{e}}_{6i} = (T_4(\mathbf{S}_1), T_2(\widehat{\mathbf{S}}_2) \cap T_1(\widehat{\mathbf{S}}_3)), \qquad i \in [0,2].
\end{aligned}$$

Шаг 7. В зависимости от значения синдрома выполняется одно из трех действий:

$$\widehat{\mathbf{e}}_{7i} = \begin{cases} (0, 0, T_{11}(\mathbf{S}_3)), & \text{если } \mathbf{S}_1 = \mathbf{S}_2 = 0;\\ (0, 0, T_{12}(\mathbf{S}_2)), & \text{если } \mathbf{S}_1 = \mathbf{S}_3 = 0;\\ \mathbf{b} + \mathbf{b}(\gamma)[G(\gamma)]^{-1}G, & \text{если } \mathbf{S}_1 = 0,\ \mathbf{S}_2 \ne 0,\ \mathbf{S}_3 \ne 0. \end{cases}$$

Запись $\mathbf{b}(\gamma)[G(\gamma)]^{-1}G$ означает восстановление кодового слова по информационной совокупности $\gamma \subset \left[1, \frac{2n}{3}\right]$, свободной от ошибок. Шаг выполняется три раза для всех квазициклических сдвигов принятого вектора.

Максимальная сложность алгоритма определяется в основном вычислением синдрома, 24 пересечениями списков мощности до 5 и 45 пересечениями списков мощности до 2, а также 6 пересечениями списков мощности до 14 и умножением вектора на порождающую матрицу.

Максимальная сложность алгоритма составляет примерно 510 операций над векторами длины 8 по сравнению с 1500 операциями над векторами длины 24 [28].

1.3. Асимптотика

1.3.1. Обзор алгоритмов декодирования линейных блоковых кодов для декодеров с жесткими решениями

Асимптотический анализ сложности декодирования линейных блоковых кодов по максимуму правдоподобия основан на утверждении, носящем название леммы Евсеева [18]. Эта лемма сводит задачу декодирования произвольного линейного кода по максимуму правдоподобия к комбинаторной задаче исправления ошибок заданной кратности. Из нее, в частности, следует, что в канале с независимыми ошибками для того, чтобы обеспечить вероятность ошибочного декодирования, не более чем в два раза худшую по сравнению с максимумом правдоподобия (почти по максимуму правдоподобия), не нужно исправлять все ошибки, декодируемые кодом, но достаточно исправлять все ошибки кратности до d_0, которые может исправить код, где d_0 — расстояние (n, k)-кода, лежащего на границе Варшамова – Гилберта [29].

Пусть $\mathcal{G}$ — (n, k, d) q-ичный линейный код, где n — длина кода, k — число информационных символов кода, d — минималь-

ное расстояние кода в метрике Хэмминга. Пусть E_q^n — линейное пространство q-ичных векторов длины n над конечным полем F_q, $d(\mathbf{a}, \mathbf{b})$ — расстояние Хэмминга между векторами $\mathbf{a}$ и $\mathbf{b}$.

Сформулируем принцип декодирования почти по максимуму правдоподобия.

Декодирование почти по максимуму правдоподобия: для данного вектора $\mathbf{b} \in E_q^n$, причем $d(\mathbf{b}, \mathcal{G}) \leq d_0$, найти кодовое слово $\mathbf{a} \in \mathcal{G}$, ближайшее к $\mathbf{b}$, где $d(\mathbf{b}, \mathcal{G}) = \min\limits_{\mathbf{a} \in \mathcal{G}} d(\mathbf{b}, \mathbf{a})$, d_0 — расстояние Варшамова – Гилберта кода $d_0 = d_0(n, k)$.

Введем необходимые обозначения [47]. Пусть G — порождающая матрица размерности $k \times n$ линейного кода $\mathcal{G}$. Пусть $J \subset \{1, \ldots, n\}$ — произвольное подмножество множества индексов $\{1, \ldots, n\}$. Через $G(J)$ обозначим подматрицу матрицы G, образованную столбцами матрицы G с номерами из множества индексов J, а через $\mathbf{a}(J)$ — подвектор вектора $\mathbf{a}$, образованный компонентами вектора $\mathbf{a}$ с номерами из множества индексов J. Информационной совокупностью называется такое k-подмножество $\gamma \subset \{1, \ldots, n\}$, что соответствующая $k \times k$ подматрица $G(\gamma)$ является невырожденной. Обобщенной информационной совокупностью называется произвольное подмножество $J \subset \{1, \ldots, n\}$. Каждая обобщенная информационная совокупность определяет подкод. Обозначим через $\mathcal{G}_J$ линейный код с порождающей матрицей $G(J)$. Этот подкод имеет параметры $(|J|, \operatorname{rank} G(J), d_J)$, причем $d_J \leq d$.

Обозначим принятый из канала вектор как $\mathbf{b} = \mathbf{a} + \mathbf{e} \pmod q$, где $\mathbf{a}$ — кодовое слово кода $\mathcal{G}$ и $\mathbf{e}$ — вектор ошибок. Пусть $W(\mathbf{c})$ — вес Хэмминга вектора $\mathbf{c}$ и $R = \dfrac{k}{n}$ — скорость кода.

Списочное декодирование подкода $\mathcal{G}_J$ приводит к созданию списка кодовых слов

$$L_J(\mathbf{b}, t) = \{\mathbf{c} \in \mathcal{G} \mid d(\mathbf{b}(J), \mathbf{c}(J)) \leq t\}.$$

Этот список построен таким образом, что для любого подвектора вектора ошибок $\mathbf{e}(J)$ с весом Хэмминга до t, т. е. при весе подвектора ошибок $W(\mathbf{e}(J)) \leq t$, список L содержит переданный

вектор.

Запишем алгоритм декодирования по обобщенным информационным совокупностям в виде последовательности шагов.

Шаг 1. Выбираем такую обобщенную информационную совокупность J_i, что $W(\mathbf{e}(J_i)) \leq t$, где t — некоторый параметр.

Шаг 2. Образуем список кодовых слов $L_{J_i}(\mathbf{b}, t)$, ближайший к $\mathbf{b}$ вектор из которого считаем декодированным вариантом принятого вектора. Иначе считаем, что произошла неисправляемая ошибка.

Классификация алгоритмов декодирования по типам

Классификация алгоритмов декодирования была введена Федоренко и Круком в обзоре [68].

Рассмотрим алгоритмы декодирования в асимптотике. Имеются три класса алгоритмов декодирования для почти всех кодов.

1. Алгоритм "соседних нулей" [85].

2. Алгоритм декодирования с разделением синдромов [17].

3. Декодирование по обобщенным информационным совокупностям [18, 24, 56, 59, 48, 40].

Показатели экспоненты сложности декодеров с жесткими решениями для двоичных кодов показаны на рис. 1. Формальное определение показателя экспоненты сложности дано в параграфе 1.3.2.

Кратко опишем следующие алгоритмы декодирования как алгоритм декодирования по обобщенным информационным совокупностям [28], который состоит в многократном декодировании принятого вектора с использованием подкодов.

Декодирование с покрывающими векторами [18]

Шаг 1. Выбираем такую информационную совокупность γ, что $W(\mathbf{e}(\gamma)) \leq Rd_0$.

Шаг 2. Вычисляем $L_\gamma(\mathbf{b}, Rd_0)$.

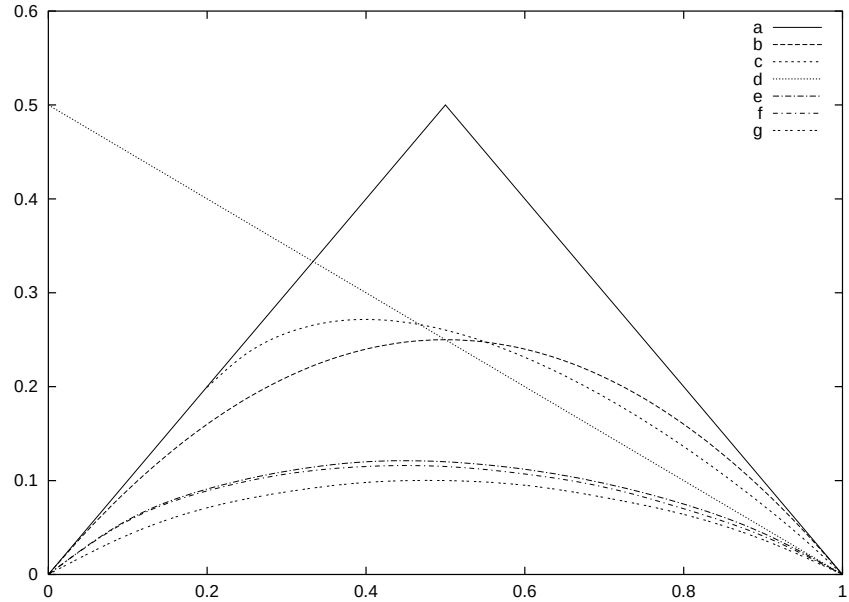

***Рис.* 1.** *Зависимость сложности алгоритмов декодирования двоичных кодов от скорости кода (ось x — скорость кода, ось y — показатель экспоненты сложности):*
a — алгоритм декодирования по максимуму правдоподобия $(\min(R, 1-R))$*; b — алгоритм декодирования с покрывающими векторами* [18]*; c — алгоритм "соседних нулей"* [85]*; d — алгоритм декодирования с разделением синдромов* [17]*; e — алгоритм декодирования с построением покрытий* [24, 56, 40]*; f — алгоритм декодирования по укороченным кодам* [59, 40]*; g — алгоритм декодирования с надкодами* [48]

Декодирование с построением покрытий [24, 56, 40]

Шаг 1. Выбираем такую информационную совокупность γ, что $\mathbf{e}(\gamma) = 0$.

Шаг 2. Вычисляем $L_\gamma(\mathbf{b}, 0)$.

Декодирование по укорочениям кодов [59, 40]

Шаг 1. Выбираем такую обобщенную информационную совокупность J, что $W(\mathbf{e}(J)) \leq t$.

Шаг 2. Подкод $\mathcal{G}_J$ будем декодировать, применяя декодирование с разделением синдромов $L_J(\mathbf{b}, t)$.

Декодирование с надкодами [48]

Шаг 1. Выбираем такую обобщенную информационную совокупность γ и набор множеств $\{T_i\}$: $T_i \cap T_j = \emptyset$, $T_i \cap \gamma = \emptyset$, $|T_i| = \text{const}$, что $\begin{cases} W(\mathbf{e}(\gamma)) \leq t_1 \\ W(\mathbf{e}(T_i)) \leq t_2 \end{cases}$.

Шаг 2. Множество $J_i = \gamma \cup T_i$ есть обобщенная информационная совокупность, а $\mathcal{G}_{J_i}$ является подкодом и надкодом одновременно. Декодируем каждый код $\mathcal{G}_{J_i}$, применяя декодирование с разделением синдромов, и вычисляем список $L = \bigcup\limits_i L_{J_i}(\mathbf{b}, t_1 + t_2)$.

1.3.2. Сложность декодирования линейных блоковых кодов

В последнее двадцатипятилетие появился ряд работ, посвященных проблеме сложности декодирования линейных блоковых кодов [18, 24, 17, 59, 40, 48]. Однако предлагаемые в них алгоритмы представляют собой решение по существу разных задач, что вызывает необходимость разработки классификации этих алгоритмов. В параграфе рассматривается декодирование почти всех q-ичных кодов в q-ичном симметричном канале без памяти с вероятностью ошибки, асимптотически совпадающей с вероятностью

ошибки при декодировании по максимуму правдоподобия, предложенное автором [40].

Сравнительный анализ алгоритмов декодирования

Пусть $H_q(x) = -x\log_q(x) - (1-x)\log_q(1-x) + x\log_q(q-1)$, $x \in [0, (q-1)/q]$ — q-ичная энтропия, а $H_q^{-1}(y)$ — функция, обратная к q-ичной энтропии.

Для описания алгоритмов декодирования обозначим через $\mathcal{G}$ линейный (n, Rn)-код над полем $\mathrm{GF}(q)$ длины n, скорости R с расстоянием Варшамова – Гилберта d_0 в метрике Хэмминга, т. е. d_0 — минимальное целое число, удовлетворяющее неравенству

$$\sum_{i=0}^{d_0} \binom{n}{i} (q-1)^i \geq q^{n(1-R)}.$$

Выражая расстояние Варшамова – Гилберта кода из этой формулы, имеем $d_0 = (H_q^{-1}(1-R) + o(1))n$.

Относительное расстояние Варшамова – Гилберта обозначим через $\delta = \lim\limits_{n\to\infty} \dfrac{d}{n}$. Евсеев показал [18], что вероятность ошибки при декодировании до расстояния Варшамова – Гилберта не превосходит удвоенной вероятности ошибки декодирования по максимуму правдоподобия, а Блиновский доказал [9], что при $n \to \infty$ почти все линейные коды имеют радиус покрытия $d_0(1 + o(1))$. Эти результаты доказывают асимптотическое равенство и эквивалентность вероятностей ошибки соответственно при декодировании до расстояния Варшамова – Гилберта и по максимуму правдоподобия. Сложность алгоритма декодирования будем характеризовать асимптотическим показателем экспоненты сложности

$$\chi = \lim_{n\to\infty} \frac{1}{n} \log_q N,$$

где N — минимально возможное число операций над q-ичными символами при реализации алгоритма декодирования. Аналогично определим асимптотический показатель экспоненты объема памяти

$$\chi = \lim_{n\to\infty} \frac{1}{n} \log_q \Pi,$$

где П — объем памяти, используемый при реализации алгоритма декодирования.

Обозначим через E_1 алгоритм из работы [18], который можно рассматривать как обобщение алгоритмов декодирования по информационным совокупностям и с покрывающими полиномами. Для реализации этого алгоритма требуется неэкспоненциальная память (под неэкспоненциальной памятью понимаем память такого объема, асимптотический показатель экспоненты объема которого равен нулю). Назовем все алгоритмы, для которых требуется неэкспоненциальная память, T-алгоритмами. Отметим, что асимптотический показатель экспоненты сложности алгоритма E_1 есть $\chi \leq R(1-R)$.

Крук предложил алгоритмы [24], которые обозначим через K_1 и K_2, с асимптотическим показателем экспоненты сложности $\chi \leq \chi_K = (1-R)(1-H_q(d_0/(1-R)))$. Алгоритм K_1 является алгоритмом со случайным выбором информационных совокупностей, образующих покрытие, а алгоритм K_2 является алгоритмом с неслучайным покрытием, хранящимся в памяти. Алгоритм K_1 можно назвать случайным T-алгоритмом. Для организации декодирования в соответствии с алгоритмом K_2 требуется память с показателем экспоненты объема памяти $\chi_{\text{П}}$ и время с показателем экспоненты сложности χ_T такие ,что $\begin{cases} \chi_T \leq \chi_K \\ \chi_{\text{П}} \leq \chi_K \end{cases}$.

Думер ввел алгоритм [17], обозначаемый через D_1, реализованный на двухленточной машине Тьюринга с асимптотическим показателем экспоненты сложности $\chi \leq (1-R)/2$ и применимый ко всем линейным кодам. Им описан алгоритм для двоичных кодов [59], обозначаемый через D_2 и являющийся комбинацией алгоритмов K_2 и D_1. Асимптотический показатель экспоненты сложности этого алгоритма в более простом виде был независимо опубликован автором в работе [40]:

$$\chi \leq \min_{\lambda}(1-R-\lambda)\left(1-H_2\left(\frac{1-\alpha(\lambda)}{1-R-\lambda}\delta\right)\right),$$

где $\alpha(\lambda) = \dfrac{\lambda + R}{\delta} H_2^{-1}\left(\dfrac{2\lambda}{\lambda + R}\right), \quad 0 \le \lambda \le \min(R, 1-R).$

В работе [48] введен алгоритм декодирования в надкодах, обозначаемый через K_3, реализованный на двухленточной машине Тьюринга с асимптотическим показателем экспоненты сложности $\zeta(R)$ и применимый почти ко всем линейным кодам.

Теорема 1.1. [48]. *В q-ичном симметричном канале с независимыми ошибками существует алгоритм декодирования с использованием надкодов для почти всех линейных (n,k)-кодов со скоростью $R = k/n$, который обеспечивает вероятность ошибочного декодирования $P_\epsilon \le P_{\epsilon 0}(1 + o(1))$ при сложности декодирования χ*

$$\frac{\log_q \chi}{n} \le \zeta(R)(1 + o(1)),$$

где $P_{\epsilon 0}$ — вероятность ошибки, обеспечиваемая этим кодом при декодировании по максимуму правдоподобия; $\delta_0 = \dfrac{d_0}{n}$; $H_q(x)$ — q-ичная энтропия;

$$\zeta(R) = \min_{v,\alpha,\ell}\left\{\epsilon_1(R,\alpha) + \max\left[\frac{1}{2}\epsilon_2(R,\alpha,v,\ell) + v, \epsilon_2(R,\alpha,v,\ell) - \ell v\right]\right\};$$

$$\epsilon_1(R,\alpha) = H_q(\delta_0) - RH_q\left(\frac{\alpha}{R}\right) - (1-R)H_q\left(\frac{\delta_0 - \alpha}{1-R}\right);$$

$$\epsilon_2(R,v,\alpha,\ell) = RH_q\left(\frac{\alpha}{R}\right) + vH_q\left(\frac{\delta_0 - \alpha}{1 - R - (\ell-1)v}\right),$$

а параметры оптимизации в выражении для $\zeta(R)$ ограничены условиями

$$\max(0, \delta_0 + R - 1) < \alpha < \min(\delta_0, R), \quad \ell v \le 1 - R.$$

Сложность декодирования с использованием надкодов, приведенная в теореме 1.1, меньше сложности декодирования всех известных общих алгоритмов декодирования во всем диапазоне кодовых скоростей.

Очевидно, что не каждый из приведенных алгоритмов может быть реализован на уровне требований, соответствующих реализации другого алгоритма. Так, алгоритмы D_1, D_2 и K_3 никак

нельзя реализовать при требованиях, предъявляемых к реализации T-алгоритмов (невозможно эффективно осуществить пересечение двух списков без использования памяти, так как сложность реализации при использовании одноленточной машины Тьюринга существенно возрастет); алгоритм K_1 — при требованиях, предъявляемых к неслучайным алгоритмам (в алгоритме используется генератор случайного покрытия), а алгоритмы E_1, K_1, K_2, K_3 и D_2 неприменимы ко всем линейным кодам (существуют коды, для которых эти алгоритмы применять нельзя).

Классификация алгоритмов декодирования по уровню требований

Алгоритмы из работ [18, 24, 17, 59, 40, 48] можно дифференцировать по уровню требований, предъявляемых к реализации этих алгоритмов:

C_1 — используется неэкспоненциальная память (алгоритм E_1);

C_2 — используется генератор случайного покрытия и неэкспоненциальная память (алгоритм K_1);

C_3 — используется экспоненциальная память (асимптотический показатель экспоненты объема памяти больше нуля) (алгоритм K_2);

C_4 — используется двухленточная машина Тьюринга, и применяется для произвольного кода (алгоритм D_1);

C_5 — используется двухленточная машина Тьюринга, но применяется для почти всех кодов (алгоритмы D_2 и K_3).

Введенные требования к реализации алгоритмов можно отнести ко всем теоретически возможным алгоритмам декодирования, хотя предложенный список требований не является исчерпывающим.

Обозначим через $A(C_i)$ реализацию алгоритма A при требованиях C_i. Если для любого алгоритма A значения всех видов сложности (как по числу операций, так и по объему памяти) реализации $A(C_i)$ не превышают значений соответствующих видов сложности $A(C_j)$, то будем говорить, что требование C_j строже требования C_i, и обозначать это через $C_j \prec C_i$. Очевидно, что для указанных выше требований, предъявляемых к реализации

алгоритмов, справедливы утверждения: $C_1 \prec C_2, C_1 \prec C_3, C_1 \prec C_4, C_1 \prec C_5, C_4 \prec C_5$.

Будем классифицировать алгоритмы декодирования по следующим признакам:

a)

— декодирование по максимуму правдоподобия,

— декодирование почти по максимуму правдоподобия;

b)

— декодирование для всех линейных кодов,

— декодирование для почти всех линейных кодов,

— декодирование для специальных классов линейных кодов;

c)

— неслучайные алгоритмы декодирования,

— алгоритмы, использующие случайные генераторы;

d)

— декодирование для специальных устройств.

Однако, ограничиваясь алгоритмами декодирования для почти всех линейных блоковых кодов почти по максимуму правдоподобия, а также учитывая сформулированные выше условия строгости требований к реализации алгоритмов, можно выделить два основных класса алгоритмов:

1) алгоритмы декодирования, использующие неэкспоненциальную память;

2) алгоритмы декодирования для машины Тьюринга, не принадлежащие предыдущему классу.

Заметим, что, кроме рассмотренных алгоритмов, известны другие алгоритмы декодирования для почти всех линейных блоковых кодов почти по максимуму правдоподобия. Они, однако, уступают по сложности соответствующим алгоритмам из работ [18, 24, 17, 59, 40, 48].

Метод построения неслучайного покрытия со сложностью, асимптотически совпадающей с мощностью этого покрытия, был предложен автором в работе [40].

Метод построения неслучайного покрытия

Для описания метода построения неслучайного покрытия

введем необходимые определения. Семейство p-подмножеств n-множества, содержащее все l-подмножества ($n \geq p \geq l$), называется $M(n,p,l)$-покрытием [42]. Обозначим через $(P_1;P_2)$ такую пару, состоящую из двух множеств чисел P_1 и P_2, для которой $|P_1| = p_1, |P_2| = p_2, P_1 \cap P_2 = \emptyset, P_1 \cup P_2 = \{1,2,\ldots,n\}$. Если для каждого множества L ($|L| = l = l_1 + l_2$) найдется пара $(P_1;P_2)$, допускающая представление множества L в виде $L = L_1 \cup L_2$, причем

$$\begin{cases} L_1 \subset P_1 \\ L_2 \subset P_2 \end{cases},$$

где $|L_1| = l_1 \leq p_1, |L_2| = l_2 \leq p_2, L_1 \cap L_2 = \emptyset$, то семейство пар $(P_1;P_2)$ будем называть $P(n,p_1,p_2,l_1,l_2)$-покрытием. Известно [42], что существует $M(n,p,l)$-покрытие с мощностью

$$M(n,p,l) \leq \frac{\binom{n}{l}}{\binom{p}{l}} \left(1 + \ln \binom{p}{l}\right) \leq o(n) \frac{\binom{n}{l}}{\binom{p}{l}}.$$

Так как в дальнейшем будут рассматриваться $P(n,p_1,p_2,l_1,l_2)$-покрытия с наложенными на их параметры ограничениями, то выпишем такие покрытия. Семейство, состоящее из n множеств $K_i = \{k_{i,1}, k_{i,2}, \ldots, k_{i,p_1}\}$, где

$$k_{i,j} = \begin{cases} i+j-1, & \text{если } i+j-1 \leq n, \\ (i+j-1)-n, & \text{если } i+j-1 > n, \end{cases} \quad i \in [1,n],\ j \in [1,p_1],$$

является $P(n,p_1,p_2,l_1,l_2)$-покрытием, если на него наложены ограничения $l/n = l_1/n_1 = l_2/n_2$. Это покрытие представляет собой n циклических сдвигов множества $K_1 = \{1,2,\ldots,p_1\}$.

Рассмотрим алгоритм построения неслучайного $M(n,p,l)$-покрытия, состоящий из двух этапов.

Первый этап. Построим семейство разбиений множества $N = \{1,\ 2,\ldots,n\}$ на взаимно непересекающиеся части такое, что на каждой из частей разместится одинаковое число элементов из любого подмножества с мощностью l. То есть для любого множества E ($|E| = l$) в семействе найдется такой набор из мно-

жеств $\{J_0, J_1, \ldots, J_{s-1}\}$ ($J_i \bigcap_{i \neq j} J_j = \emptyset$, $\bigcup_{i=0}^{s-1} J_i = N$, $|J_i| = n/s$), что будут верны равенства $|E \bigcap J_i| = l/s, \quad i \in [0, s-1]$. Заметим, что без ограничения общности можно считать числа n/s и l/s целыми. Построим $s-1$ покрытий $P_i = P(n - in/s, n/s, n - (i+1)n/s, l/s, l - (i+1)l/s)$ на множествах $N_i = \{1, 2, ..., n - in/s\}$, где $i \in [0, s-1]$. Каждому покрытию P_i соответствует семейство пар множеств $\{(I_{ij}, N_i \backslash I_{ij})\}$, причем $|I_{ij}| = n/s$ и $I_{ij} \subset N_i$. Так как второе множество в каждой паре является дополнением первого в N_i, то покрытие P_i задается семейством множеств $\{I_i\} = \{(I_{ij})\}$. Будем считать элементы покрытия $P_i = \{(I_{ij})\}$ перенумерованными числами $j \in [1, n - in/s]$.

Итоговое покрытие состоит из семейства наборов множеств $\{J_0, J_1, \ldots, J_{s-1}\}$. При построении набора выбор очередного множества J_i ($i \in [0, s-2]$) определяется номером j в покрытии $P_i = \{(I_{ij})\}$ и уже построенными множествами $J_0, J_1, \ldots, J_{i-1}$.

Опишем рекуррентную процедуру $Rec(i, j)$, ($i \in [0, s-2], j \in [1, n - in/s]$), построения множества J_i. Пусть множества $J_0, J_1, \ldots, J_{i-1}$ уже построены, а число j задано. Обозначим через $U_{i-1} = N \backslash (\bigcup_{t=0}^{i-1} J_t)$ разность между множествами. Построим взаимно-однозначное соответствие между множествами N_i и U_{i-1} (эти множества имеют одинаковую мощность). Теперь очевидна процедура построения множества J_i как отображение элемента покрытия $I_{ij} \in P_i$ на множество U_{i-1} по взаимно-однозначному соответствию между множествами N_i и U_{i-1}.

Введем еще одну рекуррентную процедуру $Recm(i)$:

FOR $j = 1$ TO $n - in/s$ DO
BEGIN
$Rec(i, j)$
$Recm(i+1)$
END

при $i \in [0, s-2]$, которая строит множество J_{s-1} как разность

между множествами $J_{s-1} = N \backslash (\bigcup_{t=0}^{s-2} J_t)$ и выдает набор множеств $\{J_0, J_1, \ldots, J_{s-1}\}$ при $i = s-1$.

Итак, итоговое покрытие можно строить путем вызова рекуррентной процедуры $Recm(0)$.

Общее число наборов из множеств $\{J_0, J_1, \ldots, J_{s-1}\}$ не превышает величины

$$Q = \bigcap_{i=0}^{s-2} P(n - in/s, n/s, n - (i+1)n/s, l/s, l - (i+1)l/s) < n^{s-1}.$$

Выбирая $s = \sqrt{n}$, имеем неэкспоненциальную сложность получения требуемого семейства, так как

$$\chi_Q = \lim_{n\to\infty} \frac{s \log_q n}{n} = \lim_{n\to\infty} \frac{\sqrt{n} \log_q n}{n} = 0.$$

Итак, время и объем памяти выполнения первого этапа имеют неэкспоненциальный порядок.

Второй этап. Зафиксируем набор из множеств $\{J_0, J_1, \ldots, J_{s-1}\}$, принадлежащий построенному на первом этапе семейству. Построим и запомним $M(n/s, p/s, l/s)$-покрытие, состоящее из семейства множеств $\{N\}$. Мощность этого покрытия не превышает величины

$$M(n/s, p/s, l/s) \le o(n) \frac{\binom{n/s}{l/s}}{\binom{p/s}{l/s}} \le o(n) \exp_q(n/s),$$

где через $\exp_y(x)$ обозначена функция $\exp_y(x) = y^x$. Подставляя в оценку $s = \sqrt{n}$, получаем неэкспоненциальный объем памяти, которая необходима для хранения покрытия:

$$\chi_M = \lim_{n\to\infty} \left(\frac{\log_q o(n)}{n} + \frac{1}{s} \right) = \lim_{n\to\infty} \frac{1}{\sqrt{n}} = 0.$$

Каждому множеству чисел $\{v_0, v_1, \ldots, v_{s-1}\}$, $v_i \in [1, M(n/s, p/s, l/s)]$, поставим в соответствие множество

$$x = \bigcup_{i=0}^{s-1} x_i,$$

где x_i — результат отображения множества $N_{v_i} \to J_i$. Множество x будет элементом искомого $M(n,p,l)$-покрытия. Время выполнения второго этапа для фиксированного набора $\{J_0, J_1, \ldots, J_{s-1}\}$ не превышает величины

$$[M(n/s, p/s, l/s)]^s \le [o(n/s)]^s \exp_q(nH_q(n/l) - pH_q(p/l)) = n^s \frac{\binom{n}{l}}{\binom{p}{l}},$$

что обеспечивает сложность, асимптотически совпадающую с мощностью $M(n,p,l)$-покрытия.

Отметим, что второй этап будет выполняться для каждого набора, входящего в построенное на первом этапе семейство.

Итак, для любого множества E, $|E| = l$, на первом этапе производится разбиение множества $\{1, 2, \ldots, n\}$ на такие s частей, что на каждой из них в ходе выполнения второго этапа становится возможным независимо выбрать по p/s позиций, в совокупности образующих требуемое покрытие. Таким образом, справедлива следующая теорема.

Теорема 1.2. *Сложность построения неслучайного $M(n,p,l)$-покрытия асимптотически совпадает с мощностью этого покрытия.*

Неслучайный алгоритм декодирования по информационным совокупностям при использовании неэкспоненциальной памяти предложен автором в работе [40].

Реализация неслучайного алгоритма декодирования по информационным совокупностям при использовании неэкспоненциальной памяти

Для описания T-алгоритма декодирования введем необходимые обозначения. Принятый вектор обозначим через $\mathbf{b}=\mathbf{a}+\mathbf{e}$, где $\mathbf{e}$ — вектор ошибки с весом Хэмминга $W(e) \le d_0$; $\mathbf{a}$ — слово из (n, Rn)-кода $\mathcal{G}$.

Опишем алгоритм декодирования K_1 кода $\mathcal{G}$ со случайным выбором информационных совокупностей, образующих покрытие [24].

Алгоритм K_1

Для каждого $i \in [1, Q]$, $\chi_Q = \chi_K$, выполняем следующие действия.

Шаг 1. Случайным образом выбираем множество J_i, $|\overline{J_i}| = Rn$.

Шаг 2. По множеству $\overline{J_i}$ выделяем Rn информационных позиций и формируем список кодовых слов $\{\mathbf{a}_i\}$, совпадающих с принятым вектором $\mathbf{b}$ на выделенных позициях.

Шаг 3. В качестве декодированного варианта принятого вектора $\mathbf{b}$ выбираем ближайший по расстоянию Хэмминга к $\mathbf{b}$ вектор из объединения всех списков $\{\mathbf{a}_i\}$.

Неслучайный алгоритм K_2 отличается от алгоритма K_1 изменением шага 1, который будет выполняться предварительно, т. е. только один раз до начала декодирования принятых векторов.

Алгоритм K_2

Шаг 1. Строим $M(n, (1-R)n, d_0)$-покрытие, состоящее из семейства множеств $\{J_1, J_2, \ldots, J_Q\}$, $|\overline{J_i}| = Rn$, $\chi_Q = \chi_K$, и храним его в памяти.

Шаги 2, 3. Для каждого $i \in [1, Q]$ выполняем шаги 2 и 3 алгоритма K_1.

Однако, если порождать $M(n, (1-R)n, d_0)$-покрытие в соответствии с методом построения неслучайного покрытия, описанным выше, то алгоритм декодирования не будет требовать экспоненциальной памяти и случайного генератора покрытия, а сложность такого алгоритма будет асимптотически совпадать со сложностью алгоритмов K_1 и K_2.

Сформулируем T-алгоритм декодирования [40].

T-алгоритм декодирования

Для каждого $i \in [1, Q]$, $\chi_Q = \chi_K$, выполняем следующие действия.

Шаг 1. Методом построения неслучайного $M(n, (1-R)n, d_0)$-покрытия выбираем множество J_i, $|\overline{J_i}| = Rn$.

Шаги 2, 3. Выполняем шаги 2 и 3 алгоритма K_1.

Итак, применяя алгоритм построения неслучайного покрытия, получаем T-алгоритм, который имеет асимптотически равную с алгоритмами K_1 и K_2 сложность и более строгие требования к реализации.

Замечания к главе

Материал главы представляет собой результаты автора (совместно с Е. А. Круком и другими) в области комбинаторного декодирования линейных блоковых кодов. Декодирование по обобщенным информационным совокупностям введено в работах [28, 26], табличное декодирование — в работах [27, 66, 67], классификация алгоритмов декодирования в асимптотике — в работах [68, 40], метод построения неслучайного покрытия — в работе [40].

2. Алгебраическое декодирование

Под алгебраическим декодированием будем понимать методы декодирования, основанные на алгебраических свойствах кодов и состоящие в решении уравнений и/или систем уравнений с полиномиальной сложностью. При рассмотрении алгебраических кодов ограничимся классами БЧХ, Рида – Соломона, Гоппы, обобщенных Гоппы и альтернантных кодов. В главе рассматривается алгебраическое декодирование по укорочениям кодов (по обобщенным информационным совокупностям), "генетическая" связь различных алгебраических методов декодирования и декодирование в надкодах.

2.1. Алгебраическое декодирование по обобщенным информационным совокупностям

Декодирование по обобщенным информационным совокупностям для произвольных линейных блоковых кодов было введено в работах [84, 28, 26]. Каждая обобщенная информационная совокупность задает линейный код, называемый укорочением основного кода. Алгоритм декодирования по обобщенным информационным совокупностям состоит в декодировании основного кода путем многократных декодирований укороченных кодов. В настоящем разделе рассматривается основанное на работе Мирончикова и Федоренко [35] применение декодирования по обобщенным информационным совокупностям к классу обобщенных кодов Гоппы [13, 6], который также называется классом (L, g)-кодов. В этом случае укорочения основного кода будут принадлежать этому же классу кодов, а их декодирование можно осуществить известными алгебраическими методами. В качестве примера использования декодирования (L, g)-кодов по обобщенным информационным совокупностям приводится алгоритм декодирования в стирающем канале [34].

2.1.1. Основные понятия и определения

Пусть $\mathcal{G}$ — линейный (n, k, d)-код над полем $\mathrm{GF}(q)$ с порождающей матрицей G, где n — длина кода, k — число информационных символов, а d — расстояние в метрике Хэмминга. Пронумеруем позиции кодового слова числами от 1 до n. Множество чисел $J = \{j_1, \ldots, j_s\}$, $1 \le j_1 < \ldots < j_s \le n$, $s \ge n - d + 1$, называется обобщенной информационной совокупностью кода [84, 28], если задание компонент кодового слова с номерами из J однозначно задает это слово. Символом $\overline{J}$ обозначим дополнение множества J в множестве $\{1, 2, \ldots, n\}$. Для произвольного вектора $\mathbf{f} = (f_1, f_2, \ldots, f_n)$ длины n и множества J определим подвектор длины s как вектор $\mathbf{f}(J) = (f_{j_1}, f_{j_2}, \ldots, f_{j_n})$. Аналогично для матрицы $G = [m_1|\cdots|m_n]$, где через m_i, $i \in [1, n]$, обозначены ее столбцы, и множества J определим подматрицу $G(J) = [m_{j_1}|\cdots|m_{j_s}]$. Код с порождающей матрицей $G(J)$ назовем укороченным кодом и будем обозначать его через $\mathcal{G}(J)$. Укороченный код $\mathcal{G}(J)$ является линейным (s, k, d_J)-кодом, где $d_J \ge d - (n - s)$.

В основе алгоритма декодирования по обобщенным информационным совокупностям лежит идея поиска подвектора принятого вектора, число ошибок в котором не превышает корректирующей способности соответствующего укороченного кода.

Напомним определение (L, g)-кодов. Пусть L — множество линейно-независимых над $\mathrm{GF}(q)$ рациональных функций:

$$L = \left\{ \frac{h_1}{z - \alpha_1}, \frac{h_2}{z - \alpha_2}, \ldots, \frac{h_n}{z - \alpha_n} \right\},$$

$\alpha_i, h_i \in \mathrm{GF}(q^m)$, $n \le q^m$, а генераторный многочлен $g(z)$ есть приведенный многочлен с коэффициентами из поля $\mathrm{GF}(q^m)$, причем $g(\alpha_i) \ne 0$ для всех $i \in [1, n]$. С каждым вектором $\mathbf{c} = (c_1, c_2, \ldots, c_n)$ над $\mathrm{GF}(q^m)$ свяжем рациональную функцию

$$\sum_{i=1}^{n} c_i \frac{h_i}{z - \alpha_i}. \tag{5}$$

Тогда (L, g)-код длины n (или обобщенный код Гоппы [6]) состоит из всех таких векторов $\mathbf{c}$, что

$$\sum_{i=1}^{n} c_i \frac{h_i}{z - \alpha_i} \equiv 0 \bmod g(z).$$

Известно [13], что расстояние (L, g)-кода не меньше, чем $\deg g(z)+1$, где через $\deg g(z)$ обозначена степень многочлена $g(z)$, а алгебраические алгоритмы декодирования (L, g)-кода позволяют исправлять до $t = \left[\dfrac{\deg g(z)}{2}\right]$ ошибок, где через $[x]$ обозначена целая часть числа x.

2.1.2. Укорочения (L, g)-кодов

Пусть $\mathcal{G}$ есть (L, g)-код. Каждое множество чисел $J \subset \{1, 2, \ldots, n\}$, мощность которого $|J| = s \geq n - d + 1$, является обобщенной информационной совокупностью и задает укороченный $\mathcal{G}(J)$-код. Для декодирования $\mathcal{G}(J)$-кода алгебраическими методами необходимо описать его как обобщенный код Гоппы [6].

Построим (L_s, g_s)-код, содержащий все кодовые слова $\mathbf{c}(J)$ укороченного кода $\mathcal{G}(J)$. Приведенный многочлен $g_s(z)$ с коэффициентами из поля $\mathrm{GF}(q^m)$, степень которого равна $\deg g_s(z) = \deg g(z) - (n - s)$, является генераторным многочленом (L_s, g_s)-кода, если $g_s(\alpha_j) \neq 0$ для всех $j \in J$. Множество

$$L_s = \left\{ \frac{\beta_j}{z - \alpha_j} \right\}$$

для всех $j \in J$, где коэффициенты β_j — некоторые элементы поля $\mathrm{GF}(q^m)$, будет множеством рациональных функций для кода (L_s, g_s). Для всех слов (L_s, g_s)-кода при фиксированных коэффициентах β_i, $i \in J$, справедливо тождество

$$\sum_{i \in J} c_i \frac{\beta_i}{z - \alpha_i} \equiv 0 \bmod g_s(z). \qquad (6)$$

Найдем такие коэффициенты $\beta_i, \quad i \in J$, что для любого кодового слова $\mathbf{c} \in \mathcal{G}$ его укорочение $\mathbf{c}(J)$ будет принадлежать (L_s, g_s)-коду. Для этого рациональную функцию (5) умножим на дробь, в которой числитель и знаменатель имеют одинаковые степени, причем знаменатель — генераторный многочлен $g(z)$, а числитель — произведение генераторного многочлена (L_s, g_s)-кода $g_s(z)$ и многочлена $\prod\limits_{j \in \overline{J}}(z - \alpha_j)$:

$$\left[\sum_{i=1}^{n} c_i \frac{h_i}{z - \alpha_i}\right] \frac{\left[\prod\limits_{j \in \overline{J}}(z - \alpha_j)\right] g_s(x)}{g(x)} =$$

$$= \frac{g(x)p(x)}{\left[\prod\limits_{c_i \neq 0}(z - \alpha_j)\right]} \frac{\left[\prod\limits_{j \in \overline{J}}(z - \alpha_j)\right] g_s(x)}{g(x)} =$$

$$= \frac{g_s(x)p(x)\left[\prod\limits_{j \in \overline{J}}(z - \alpha_j)\right]}{\left[\prod\limits_{c_i \neq 0}(z - \alpha_j)\right]}. \tag{7}$$

Выражение (7) сравнимо с нулем по $\bmod g_s(z)$ и, следовательно, является рациональной функцией, связанной с вектором $\mathbf{c}(J)$ (L_s, g_s)-кода, в соответствии с формулой (6). Приравняем левые части (7) и (6)

$$\left[\sum_{i=1}^{n} c_i \frac{h_i}{z - \alpha_i}\right] \frac{\left[\prod\limits_{j \in \overline{J}}(z - \alpha_j)\right] g_s(x)}{g(x)} = \sum_{i \in J} c_i \frac{\beta_i}{z - \alpha_i}$$

и разложим на простейшие дроби левую часть полученного равенства. Для этого умножим обе части этого равенства на $(z-\alpha_v)$ для каждого числа $v \in J$:

$$\left[c_v h_v + (z-\alpha_v)\sum_{\substack{i=1\\ i\neq v}}^{n} c_i \frac{h_i}{z-\alpha_i}\right] \frac{\left[\prod_{j\in\overline{J}}(z-\alpha_j)\right] g_s(x)}{g(x)} =$$

$$= c_v\beta_v + (z-\alpha_v)\sum_{\substack{i\in J\\ i\neq v}} c_i \frac{\beta_i}{z-\alpha_i}.$$

Подставим в полученную формулу значение $z=\alpha_v$ и опустим нулевые слагаемые:

$$c_v h_v \frac{\left[\prod_{j\in\overline{J}}(\alpha_v-\alpha_j)\right] g_s(\alpha_v)}{g(\alpha_v)} = c_v\beta_v.$$

Последнее равенство справедливо для любой компоненты c_v произвольного кодового слова $\mathbf{c}(J) \in \mathcal{G}(J)$. Отсюда выразим коэффициенты β_v через известные величины:

$$\beta_v = \frac{h_v \left[\prod_{j\in\overline{J}}(\alpha_v-\alpha_j)\right] g_s(\alpha_v)}{g(\alpha_v)}. \tag{8}$$

Итак, если коэффициенты $\beta_v, \quad v \in J$, в определении множества L_s вычислять по формуле (8), то все слова $\mathcal{G}(J)$-кода содержатся в (L_s, g_s)-коде.

2.1.3. Применение декодирования укороченных (L_s, g_s)-кодов

Пусть принятый из канала вектор $\mathbf{b} = \mathbf{c} + \mathbf{e}$, $\mathbf{c}$ — кодовый вектор, а $\mathbf{e}$ — вектор ошибки. Рассмотрим задачу исправления до t ошибок (L, g)-кодом, если его укороченные (L_s, g_s)-коды допускают исправление алгебраическими методами до $\tau = \left[\dfrac{\deg g_s(z)}{2}\right]$ ошибок. Если вес Хэмминга вектора ошибки $W(\mathbf{e}) \leq t$, то для декодирования достаточно задать семейство обобщенных информационных совокупностей $\{J\}$, в котором для некоторой обобщенной информационной совокупности $J \in \{J\}$ выполняется неравенство $W(\mathbf{e}(J)) \leq \tau$. Каждой обобщенной информационной совокупности $J \in \{J\}$ соответствует свой укороченный (L_s, g_s)-код, а алгоритм декодирования основного кода состоит в декодировании всех (L_s, g_s)-кодов, причем после каждого декодирования укороченного кода вычисляется декодированный вариант кодового вектора $\widehat{\mathbf{c}}$ и проверяется выполнение условия $d(\mathbf{b}, \widehat{\mathbf{c}}) \leq t$. Сложность данного алгоритма определяется мощностью семейства $\{J\}$ и сложностью декодирования укороченного (L_s, g_s)-кода. Задача построения минимального семейства обобщенных информационных совокупностей является комбинаторной и может быть сведена к комбинированию классических задач построения покрытий одних множеств другими. К сожалению, мощность семейства обобщенных информационных совокупностей велика и непосредственное применение алгоритма декодирования по обобщенным информационным совокупностям не всегда будет оправдано, поэтому следует применять этот алгоритм в комбинации с другими известными алгоритмами или при других постановках задачи декодирования.

Рассмотрим задачу исправления ошибок и стираний. Алгебраические методы для ее решения, связанные с использованием многочлена стираний, описаны, например, в монографии [7]. Введем иной алгебраический алгоритм декодирования, основанный на декодировании укороченных (L_s, g_s)-кодов. Известно, что код $\mathcal{G}$ исправляет p ошибок и v стираний, если $2p + v \leq \deg g(z)$. Выбе-

рем обобщенную информационную совокупность как множество всех нестертых позиций, тогда (L_s, g_s)-код, соответствующий этой обобщенной информационной совокупности, способен исправлять до $\tau = \left[\dfrac{\deg g_s(z) - v}{2}\right] \geq p$ ошибок. Итак, декодирование основного кода при наличии стираний сводится к обычному декодированию одного укороченного кода [34]. Приведем в качестве примера алгоритм декодирования двоичного кода.

Пример. Пусть $\mathcal{G}$ — двоичный (15, 5, 7)-код БЧХ с

$$L = \left\{ \frac{1}{z - \alpha^{15}}, \frac{1}{z - \alpha^{14}}, \ldots, \frac{1}{z - \alpha^{1}} \right\},$$

$\alpha \in \mathrm{GF}(2^4)$ и $g(z) = z^6$, а принятый вектор

$$\mathbf{b} = (0000110\chi100\chi0\chi\chi),$$

где χ — символ стирания. Тогда алгоритм исправления одной ошибки и четырех стираний записывается следующим образом.

Шаг 1. Выберем обобщенную информационную совокупность как множество всех нестертых позиций

$$J = \{1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 13\}, \text{ а } \overline{J} = \{8, 12, 14, 15\}.$$

Шаг 2. Построим (L_s, g_s)-код для J при $g_s(z) = z^2$. Для этого вычислим коэффициенты $\beta_i \in J$ по формуле (8):

$$\beta_i = \frac{\prod\limits_{j \in \overline{J}} (\alpha_i - \alpha_j)}{\alpha_i^4}$$

(напомним, что $\alpha_i = \alpha^{16-i}$), и получим

$$\beta_1 = \alpha^0, \beta_2 = \alpha^9, \beta_3 = \alpha^3, \beta_4 = \alpha^2, \beta_5 = \alpha^6, \beta_6 = \alpha^5,$$

$$\beta_7 = \alpha^4, \beta_9 = \alpha^{12}, \beta_{10} = \alpha^1, \beta_{11} = \alpha^{10}, \beta_{13} = \alpha^8.$$

Шаг 3. Применяя декодирование (L_s, g_s)-кода к вектору $\mathbf{b}(J)$, получим множество локаторов ошибок $E = \{\alpha^{12}\} = \{\alpha_4\}$ [13].

Шаг 4. Строим вектор $\widehat{\mathbf{b}}$, подвектор которого $\widehat{\mathbf{b}}(J)$ свободен от ошибок:

$$\widehat{\mathbf{b}} = (0001110\chi100\chi0\chi\chi).$$

Шаг 5. Вычислим вектор $\widehat{\mathbf{c}}$, совпадающий с вектором $\widehat{\mathbf{b}}$ на множестве J, путем линейных преобразований над строками порождающей матрицы кода $\mathcal{G}$

$$\widehat{\mathbf{c}} = (000111011001010).$$

Шаг 6. Так как расстояние по нестертым позициям между векторами $\mathbf{b}$ и $\widehat{\mathbf{c}}$ не превышает $p = 1$, то вектор $\widehat{\mathbf{c}}$ есть декодированный вариант кодового вектора.

Заметим, что введенный алгоритм исправления ошибок и стираний будет более эффективным при декодировании кодов с алфавитом из больших полей.

2.2. "Генетическая" связь алгебраических методов декодирования

В работе Гао [76] предложен алгоритм декодирования, который представляется самым простым и естественным в классе алгоритмов декодирования алгебраических кодов до их корректирующей способности. Асимптотическая сложность алгоритма совпадает со сложностью лучших алгоритмов декодирования, а описание является самым простым из описаний известных алгоритмов. В разделе рассматривается интерпретация этого алгоритма через цепочку родственных связей от алгоритмов Велча – Берлекэмпа [99] и Евклида [94, 33], предложенная автором в работах [63, 64].

2.2.1. Основные понятия и определения

Рассмотрим код Рида – Соломона (n, k, d) над конечным полем $\mathrm{GF}(q)$ с длиной $n = q - 1$, числом информационных символов k и конструктивным расстоянием $d = n - k + 1$, где q — степень простого числа. Корректирующая способность кода равна $\left[\frac{d-1}{2}\right]$, где $[a]$ — целая часть числа a.

Порождающий многочлен кода Рида – Соломона обозначим через

$$g(x) = \prod_{i=b}^{b+d-2} (x - \alpha^i),$$

где α — примитивный элемент $\mathrm{GF}(q)$; b — произвольное натуральное число.

Принятый вектор представлен многочленом

$$R(x) = \sum_{i=0}^{n-1} r_i x^i = C(x) + E(x) = \sum_{i=0}^{n-1} c_i x^i + \sum_{i=0}^{n-1} e_i x^i,$$

где $C(x)$ — кодовое слово; $E(x)$ — вектор ошибок.

Пусть $E(x)$ содержит $t \le \frac{d-1}{2}$ ошибок, которые имеют координаты $i_1, i_2, \ldots, i_t$, причем $0 \le i_1 < i_2 < \ldots < i_t \le n - 1$, и значения $e_{i_1}, e_{i_2}, \ldots, e_{i_t}$. Назовем величины $Z_1 = \alpha^{i_1}, Z_2 = \alpha^{i_2}, \ldots, Z_t = \alpha^{i_t}$ локаторами ошибок, а $Y_1 = e_{i_1}, Y_2 = e_{i_2}, \ldots, Y_t = e_{i_t}$ — значениями ошибок.

Многочлен локаторов ошибок обозначим через

$$W(x) = \prod_{i=1}^{t} (x - Z_i),$$

где $t \le \frac{d-1}{2}$, — число ошибок в векторе ошибок $E(x)$; Z_i — локатор ошибки в векторе ошибок $E(x)$.
Положим по определению $W(x) = 1$, если ошибок нет.

Известно несколько методов кодирования для кодов Рида – Соломона. В настоящем разделе будут рассмотрены два из них: систематическое кодирование во временной области и спектральное кодирование в частотной области для несистематического кодирования. Заметим, что алгоритм декодирования не зависит от метода кодирования.

В параграфах 2.2.3 и 2.2.4 применяется систематическое кодирование. Кодовое слово состоит из двух частей — информационной и проверочной. Многочлен локаторов ошибок, входящих в информационную часть, обозначим как

$$W_m(x) = \prod_{i=1}^{t_m} (x - Z_i),$$

где $t_m \leq \dfrac{d-1}{2}$ — число ошибок в информационной части вектора ошибок $E(x)$; $Z_i \in \{\alpha^j \mid j \in [d-1, n-1]\}$ — локатор ошибки в информационной части вектора ошибок $E(x)$.

В параграфах 2.2.2 и 2.2.5 применяется спектральное кодирование. Для упрощения изложения будем рассматривать только случай $b = 1$. Информационный многочлен кода Рида – Соломона обозначим как

$$M(x) = \sum_{i=0}^{k-1} m_i x^i.$$

При спектральном кодировании компонента c_i кодового слова $C(x)$ вычисляется как

$$c_i = M(\alpha^i), \quad i \in [0, n-1].$$

2.2.2. Алгоритм Гао

В настоящем параграфе описывается алгоритм Гао [76], который ранее был введен Шиозаки [92]. Для упрощения изложения будем рассматривать только классические коды Рида – Соломона с параметрами $n = q - 1$ и $b = 1$.

Шаг 1. *Интерполяция.* Построим такой интерполяционный многочлен $T(x)$, что

$$T(\alpha^i) = r_i, \quad i \in [0, n-1],$$

где $\deg T(x) < n$.

Шаг 2. *Незаконченное вычисление наибольшего общего делителя.* Решаем сравнение

$$\begin{cases} W(x)T(x) \equiv P(x) \bmod x^n - 1 \\ \deg P(x) < \dfrac{n+k}{2} \\ \text{maximize} \deg P(x) \end{cases}$$

применением расширенного алгоритма Евклида к многочленам $x^n - 1$ и $T(x)$, получая единственную пару многочленов $P(x)$ и $W(x)$.

Шаг 3. *Деление.* Информационный многочлен есть

$$M(x) = \frac{P(x)}{W(x)}.$$

Асимптотическая сложность алгоритма $O(n(\log n)^2)$ совпадает со сложностью лучших классических алгоритмов декодирования кодов Рида – Соломона [33, 81].

Первый шаг алгоритма Гао может быть выполнен любым быстрым алгоритмом вычисления дискретного преобразования Фурье (ДПФ) над конечным полем (например, [98, 43]), со сложностью $O(n(\log n)^2)$ операций. В случае, когда необходимо минимизировать число умножений, лучший алгоритм для малых длин ($n \leq 511$) предложен в работе [39].

Одна из лучших реализаций второго шага есть алгоритм Мёнка [87] со сложностью $O(n(\log n)^2)$ операций, который также воспроизведен в монографиях [2, 8]. Заметим, что при этом второй шаг полностью совпадает с алгоритмом решения ключевого уравнения Сугиямы и других [94].

Деление на третьем шаге алгоритма выполняется за $O(n \log n \log\log n)$ операций.

При приложении алгоритма Гао к другим классам алгебраических кодов, таких как БЧХ, коды Гоппы или альтернантные коды, необходимо добавить дополнительный шаг, восстанавливающий кодовое слово по информационному многочлену.

2.2.3. Оригинальный алгоритм Велча – Берлекэмпа

Алгоритм декодирования Велча – Берлекэмпа лежит во временной области. В изложении алгоритма будем следовать патенту [99] и статье [88], где описание алгоритма упрощено.

Алгоритм декодирования Велча – Берлекэмпа состоит из следующих шагов.

Шаг 1. Вычисляем синдром для циклического кода

$$S(x) = \sum_{i=0}^{d-2} s_i x^i \equiv R(x) \bmod g(x).$$

Шаг 2. Решаем ключевое уравнение

$$\begin{cases} p_j \alpha^j N(\alpha^j) = s_j W_m(\alpha^j), \quad j \in [0, d-2] \\ \deg N(x) < \deg W_m(x) \leq (d-1)/2 \end{cases}, \tag{9}$$

где

$$p(x) = \frac{g(x)}{x - \alpha^b} = \sum_{i=0}^{d-2} p_i x^i,$$

и получаем многочлены $N(x)$ и $W_m(x)$.

Шаг 3. Определяем локаторы ошибок в информационной части принятого вектора из $W_m(x)$ и получаем множество $\{Z_i\} \in \{\alpha^j \mid j \in [d-1, n-1]\}$.

Шаг 4. Определяем значения ошибок в информационной части принятого вектора

$$Y_i = f(Z_i) \frac{N(Z_i)}{W'_m(Z_i)},$$

где $W'_m(x)$ — формальная производная для $W_m(x)$ и

$$f(Z) = Z^{-b} \sum_{i=0}^{d-2} \frac{p_i \alpha^{i(b+1)}}{\alpha^i - Z}$$

для $Z \in \alpha^j$, $j \in [d-1, n-1]$.

Шаг 5. Вычисляем проверочную часть кодового слова.

2.2.4. Интерпретация Чамберса

Чамберс описал интерпретацию второго шага алгоритма декодирования Велча – Берлекэмпа [51].

Для кода Рида – Соломона всегда $p_j \neq 0$, $j \in [0, d-2]$, и из ключевого уравнения (9) имеем

$$\begin{cases} N(\alpha^j) = \dfrac{s_j}{p_j \alpha^j} W_m(\alpha^j), \quad j \in [0, d-2] \\ \deg N(x) < \deg W_m(x) \leq (d-1)/2 \end{cases}.$$

Построим такой интерполяционный многочлен $L(x)$, что

$$L(\alpha^j) = \frac{s_j}{p_j \alpha^j}, \quad j \in [0, d-2],$$

где $\deg L(x) < d-1$.

Из

$$N(\alpha^j) = L(\alpha^j) W_m(\alpha^j), \quad j \in [0, d-2],$$

имеем

$$\begin{cases} N(x) \equiv L(x) W_m(x) \bmod G(x) \\ \deg N(x) < \deg W_m(x) \leq (d-1)/2 \\ \text{minimize} \deg W_m(x) \end{cases},$$

где $G(x) = \prod_{j=0}^{d-2} (x - \alpha^j)$.

Решаем сравнение применением расширенного алгоритма Евклида к многочленам $G(x)$ и $L(x)$, получая многочлены $N(x)$ и $W_m(x)$.

2.2.5. Алгоритм Велча – Берлекэмпа в частотной области

Идея модификации Геммеля и Судана [77] состоит в переносе алгоритма Велча – Берлекэмпа в частотную область. Очевидно, что описание модификации значительно проще описания оригинального алгоритма Велча – Берлекэмпа, однако ее прямая реализация неэффективна.

Если $r_i = c_i$, то $r_i = M(\alpha^i)$. Если $r_i \neq c_i$, то $W(\alpha^i) = 0$. Следовательно:

$$W(\alpha^i) r_i = W(\alpha^i) M(\alpha^i), \quad i \in [0, n-1].$$

Пусть $P(x) = W(x)M(x)$. Тогда ключевое уравнение имеет вид

$$W(\alpha^i) r_i = P(\alpha^i), \quad i \in [0, n-1]. \tag{10}$$

Решаем ключевое уравнение и получаем многочлены $P(x)$ и $W(x)$. Информационный многочлен получается делением

$$M(x) = \frac{P(x)}{W(x)}.$$

2.2.6. Вывод алгоритма Гао

Приложим метод Чамберса к ключевому уравнению (10).

Построим такой интерполяционный многочлен $T(x)$, что

$$T(\alpha^i) = r_i, \quad i \in [0, n-1],$$

где $\deg T(x) < n$. Это соответствует первому шагу алгоритма Гао.

Далее, из

$$W(\alpha^i) T(\alpha^i) = P(\alpha^i), \quad i \in [0, n-1],$$

имеем сравнение

$$\begin{cases} W(x)T(x) \equiv P(x) \bmod x^n - 1 \\ \deg W(x) \leq \dfrac{d-1}{2} \\ \text{maximize} \deg W(x) \end{cases}.$$

Учитывая, что $\deg P(x) = \deg M(x) + \deg W(x) \leq (k-1) + \dfrac{d-1}{2} \leq \dfrac{n+k}{2} - 1 < \dfrac{n+k}{2}$, переписываем условие решения сравнения на эквивалентное

$$\begin{cases} W(x)T(x) \equiv P(x) \bmod x^n - 1 \\ \deg P(x) < \dfrac{n+k}{2} \\ \text{maximize} \deg P(x) \end{cases}.$$

Решение этого сравнения соответствует второму шагу алгоритма Гао.

Очевидно, что комбинируя методы из параграфов 2.2.4 и 2.2.5, получаем алгоритм Гао.

2.2.7. Расширенный алгоритм Евклида

Напомним описание и некоторые свойства расширенного алгоритма Евклида. Расширенный алгоритм Евклида применяется к многочленам $r_{-1}(z)$ и $r_0(z)$, причем $\deg r_{-1}(z) \geq \deg r_0(z)$. Два начальных шага $i = -1$ и $i = 0$ являются фиктивными, и для них принимаются начальные условия $x_{-1}(z) = y_0(z) = 1$ и $x_0(z) = y_{-1}(z) = 0$. На i-м шаге ($i \geq 1$) выполняется деление с остатком $r_{i-2}(z) = r_{i-1}(z)q_{i-1}(z) + r_i(x)$, где $\deg r_{i-1}(z) > \deg r_i(z)$, и вычисляются линейные коэффициенты $x_i(z)$ и $y_i(z)$. Запишем i-й шаг расширенного алгоритма Евклида

$$r_{-1}(z)x_i(z) + r_0(z)y_i(z) = r_i(z), \quad i \geq 1,$$

где

$$\begin{cases} r_i(z) = & r_{i-2}(z) - r_{i-1}(z)q_{i-1}(z) \\ x_i(z) = & x_{i-2}(z) - x_{i-1}(z)q_{i-1}(z) \\ y_i(z) = & y_{i-2}(z) - y_{i-1}(z)q_{i-1}(z) \end{cases}.$$

Завершается расширенный алгоритм Евклида шагом с нулевым остатком $r_i(z) = 0$ (дополнительным шагом). Однако незаконченный расширенный алгоритм Евклида может быть завершен на любом шаге при выполнении некоторого ограничения на степень остатка или линейного коэффициента.

Как было отмечено выше, для последовательности остатков выполняется неравенство

$$\deg r_{i-1}(z) > \deg r_i(z). \tag{11}$$

Для степени линейного коэффициента $y_i(z)$ справедливо равенство [33, 12.8]

$$\deg y_i(z) = \deg r_{-1}(z) - \deg r_{i-1}(z). \tag{12}$$

2.2.8. Корректность алгоритма Гао

Для завершения вывода алгоритма Гао необходимо доказать существование и единственность полученного решения.

Теорема 2.1. *Алгоритм Гао приводит к единственному решению при декодировании до корректирующей способности кода Рида – Соломона.*

Доказательство. Пусть имеются два решения сравнения

$$\begin{cases} W(x)T(x) \equiv P(x) \bmod x^n - 1 \\ \deg P(x) < \dfrac{n+k}{2} \\ \text{maximize} \deg P(x) \end{cases}. \tag{13}$$

Первое решение, полученное применением расширенного алгоритма Евклида к многочленам $x^n - 1$ и $T(x)$ с правилом остановки

$\deg P(x) < \dfrac{n+k}{2}$, дает пару многочленов $P(x)$ и $W(x)$. Если $P(x)$ делится на $W(x)$ без остатка, то многочлен $M(x) = \dfrac{P(x)}{W(x)}$ будет информационным многочленом кода Рида – Соломона.

Заметим, что расширенный алгоритм Евклида надо проводить с нулевого (фиктивного) шага, а не с первого, как обычно. Заканчивать расширенный алгоритм Евклида надо дополнительным (с нулевым остатком) шагом.

Другое решение (истинное) удовлетворяет сравнению

$$\widetilde{W}(x)T(x) \equiv \widetilde{P}(x) \bmod x^n - 1$$

при декодировании до корректирующей способности кода Рида – Соломона
$\deg \widetilde{W}(x) \leq \dfrac{d-1}{2}$ и приводит к другому информационному многочлену кода Рида – Соломона

$$\widetilde{M}(x) = \frac{\widetilde{P}(x)}{\widetilde{W}(x)}.$$

Вначале рассмотрим вырожденный случай, когда решение сравнения (13) заканчивается на нулевом шаге расширенного алгоритма Евклида, и докажем, что информационные многочлены, полученные из обоих решений, совпадают: $M(x) = \widetilde{M}(x)$.

Пусть $\widetilde{W}(x) = 1$. Тогда $\widetilde{M}(x) = M(x) = P(x) = T(x)$, так как из нулевого шага расширенного алгоритма Евклида следует, что $\deg T(x) \leq k - 1 < \dfrac{n+k}{2}$.

Далее будем полагать, что $\widetilde{W}(x) \neq 1$.

Докажем, что $\deg W(x) \leq \dfrac{d-1}{2}$, используя свойства расширенного алгоритма Евклида.

Лемма 2.1. $\deg W(x) \leq \dfrac{d-1}{2}$.

Доказательство. На i-м шаге расширенного алгоритма Евклида из (11) и $\deg P(x) < \dfrac{n+k}{2}$ следует, что

$$\deg r_{i-1}(z) \geq \frac{n+k}{2}.$$

Тогда из (12) и $\deg r_{-1}(z) = n$ следует

$$\deg y_i(z) = \deg r_{-1}(z) - \deg r_{i-1}(z) \leq n - \frac{n+k}{2} = \frac{d-1}{2}.$$

Таким образом, $\deg W(x) \leq \frac{d-1}{2}$. Ч.т.д.

Выполним деление с остатком

$$P(x) = W(x)M(x) + U(x), \quad \text{причем} \quad \deg U(x) < \deg W(x).$$

Если остаток от деления равен нулю: $U(x) = 0$, то многочлен $M(x)$ будет информационным многочленом кода Рида – Соломона.

Очевидно, что $\deg \widetilde{P}(x) = \deg \widetilde{M}(x) + \deg \widetilde{W}(x) \leq (k-1) + \frac{d-1}{2} \leq \frac{n+k}{2} - 1 < \frac{n+k}{2}$.

Покажем, что информационные многочлены, полученные из обоих решений, совпадают, т. е.

$$\begin{cases} M(x) & = \widetilde{M}(x) \\ U(x) & = 0 \end{cases}.$$

После преобразований

$$P(x)\widetilde{W}(x) \equiv \Big(W(x)T(x)\Big)\widetilde{W}(x) = W(x)\Big(T(x)\widetilde{W}(x)\Big) \equiv$$

$$\equiv W(x)\widetilde{P}(x) \bmod x^n - 1$$

получаем сравнение

$$P(x)\widetilde{W}(x) \equiv W(x)\widetilde{P}(x) \bmod x^n - 1.$$

Оценим степени произведения многочленов в каждой части сравнения. Из

$$\begin{cases} \deg W(x), \widetilde{W}(x) & \leq \dfrac{d-1}{2} \\ \deg P(x), \widetilde{P}(x) & < \dfrac{n+k}{2} \end{cases}$$

следует

$$\begin{cases} \deg\left(P(x)\widetilde{W}(x)\right) & < \dfrac{n+k}{2} + \dfrac{d-1}{2} = n \\ \deg\left(W(x)\widetilde{P}(x)\right) & < \dfrac{d-1}{2} + \dfrac{n+k}{2} = n \end{cases},$$

т. е.

$$\begin{cases} \deg\left(P(x)\widetilde{W}(x)\right) & < n \\ \deg\left(W(x)\widetilde{P}(x)\right) & < n \end{cases}.$$

Видно, что степени произведения многочленов в каждой части сравнения не превышают степень модуля $x^n - 1$, т. е. сравнение справедливо как равенство для многочленов

$$P(x)\widetilde{W}(x) = W(x)\widetilde{P}(x).$$

Отсюда после деления имеем

$$P(x) = W(x)\frac{\widetilde{P}(x)}{\widetilde{W}(x)} = W(x)\widetilde{M}(x).$$

Из последнего равенства видно, что $P(x)$ делится на $W(x)$ без остатка, т. е. $U(x) = 0$ и $M(x) = \dfrac{P(x)}{W(x)}$.

После еще одного деления имеем

$$\frac{P(x)}{W(x)} = \widetilde{M}(x)$$

и

$$M(x) = \widetilde{M}(x).$$

Показано, что решение сравнения (13), полученное применением расширенного алгоритма Евклида, всегда существует и совпадает с истинным решением. Ч.т.д.

2.2.9. Пример

Рассмотрим процедуры кодирования и декодирования для кода Рида – Соломона (4, 2, 3). Коэффициенты многочлена будем записывать как вектор-строку и как вектор-столбец (без обозначения операции транспонирования).

Конечное поле. Вначале введем конечное поле $\mathrm{GF}(5) = \{0, 1, 2, 3, 4\} \mod 5$. Или в другом представлении: $\mathrm{GF}(5) = \{0, \alpha^0, \alpha^1, \alpha^2, \alpha^3\}$, где $\alpha = 2$ — примитивный элемент.

α^0	α^1	α^2	α^3	$\alpha^4 = 1$
1	2	4	3	1

Код Рида – Соломона. Введем код Рида – Соломона (4, 2, 3) над GF(5). Его порождающий многочлен имеет вид

$$g(x) = (x - \alpha)(x - \alpha^2) = (x - 2)(x - 4) = x^2 + 4x + 3,$$

порождающая матрица

$$G = \begin{bmatrix} g_0 & g_1 & g_2 & 0 \\ 0 & g_0 & g_1 & g_2 \end{bmatrix} = \begin{bmatrix} 3 & 4 & 1 & 0 \\ 0 & 3 & 4 & 1 \end{bmatrix}$$

и проверочная матрица

$$H = \begin{bmatrix} \alpha^0 & \alpha^1 & \alpha^2 & \alpha^3 \\ \alpha^0 & \alpha^2 & \alpha^0 & \alpha^2 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 4 & 3 \\ 1 & 4 & 1 & 4 \end{bmatrix}.$$

Спектральное кодирование. Кодовое слово вычисляется по формуле

$$\mathbf{C} = V\mathbf{M},$$

где матрица Вандермонда есть

$$V = \begin{bmatrix} \alpha^0 & \alpha^0 & \alpha^0 & \alpha^0 \\ \alpha^0 & \alpha^1 & \alpha^2 & \alpha^3 \\ \alpha^0 & \alpha^2 & \alpha^0 & \alpha^2 \\ \alpha^0 & \alpha^3 & \alpha^2 & \alpha^1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 3 \\ 1 & 4 & 1 & 4 \\ 1 & 3 & 4 & 2 \end{bmatrix}.$$

Пусть информационный многочлен есть

$$M(x) = \begin{bmatrix} m_0 & m_1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 0 & 0 \end{bmatrix} = 2 + 3x.$$

Тогда процедура кодирования имеет вид

$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} \alpha^0 & \alpha^0 & \alpha^0 & \alpha^0 \\ \alpha^0 & \alpha^1 & \alpha^2 & \alpha^3 \\ \alpha^0 & \alpha^2 & \alpha^0 & \alpha^2 \\ \alpha^0 & \alpha^3 & \alpha^2 & \alpha^1 \end{bmatrix} \begin{bmatrix} m_0 \\ m_1 \\ 0 \\ 0 \end{bmatrix};$$

$$\begin{bmatrix} 0 \\ 3 \\ 4 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 3 \\ 1 & 4 & 1 & 4 \\ 1 & 3 & 4 & 2 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \\ 0 \\ 0 \end{bmatrix};$$

$$C(x) = \begin{bmatrix} c_0 & c_1 & c_2 & c_3 \end{bmatrix} = \begin{bmatrix} 0 & 3 & 4 & 1 \end{bmatrix} = 3x + 4x^2 + x^3.$$

Обратное преобразование. Запишем матрицу, обратную к матрице Вандермонда:

$$V^{-1} = -\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 3 & 4 & 2 \\ 1 & 4 & 1 & 4 \\ 1 & 2 & 4 & 3 \end{bmatrix}.$$

Тогда обратное ДПФ есть

$$\mathbf{M} = V^{-1}\mathbf{C};$$

$$C(x) = \begin{bmatrix} c_0 & c_1 & c_2 & c_3 \end{bmatrix} = \begin{bmatrix} 0 & 3 & 4 & 1 \end{bmatrix} = 3x + 4x^2 + x^3;$$

$$\begin{bmatrix} 2 \\ 3 \\ 0 \\ 0 \end{bmatrix} = - \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 3 & 4 & 2 \\ 1 & 4 & 1 & 4 \\ 1 & 2 & 4 & 3 \end{bmatrix} \begin{bmatrix} 0 \\ 3 \\ 4 \\ 1 \end{bmatrix};$$

$$M(x) = \begin{bmatrix} m_0 & m_1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 0 & 0 \end{bmatrix} = 2 + 3x.$$

Декодирование. Пусть из канала принят вектор $R(x) = C(x) + E(x)$:

$$\begin{array}{lcl} C(x) & = & (0 \quad 3 \quad 4 \quad 1) \\ E(x) & = & (0 \quad 0 \quad 2 \quad 0) \\ R(x) & = & (0 \quad 3 \quad 1 \quad 1) \end{array};$$

Z	α^0	α^1	α^2	α^3
Z	1	2	4	3

$$R(x) = \begin{bmatrix} r_0 & r_1 & r_2 & r_3 \end{bmatrix} = \begin{bmatrix} 0 & 3 & 1 & 1 \end{bmatrix} = 3x + x^2 + x^3.$$

Шаг 1. Интерполяция.

$$\begin{bmatrix} 0 \\ 0 \\ 3 \\ 2 \end{bmatrix} = - \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 3 & 4 & 2 \\ 1 & 4 & 1 & 4 \\ 1 & 2 & 4 & 3 \end{bmatrix} \begin{bmatrix} 0 \\ 3 \\ 1 \\ 1 \end{bmatrix};$$

$$\mathbf{T} = V^{-1}\mathbf{R} = \begin{bmatrix} T_0 & T_1 & T_2 & T_3 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 3 & 2 \end{bmatrix} = 3x^2 + 2x^3.$$

Шаг 2. Незаконченное вычисление наибольшего общего делителя.

$$\begin{cases} W(x)T(x) \equiv P(x) \bmod x^n - 1 \\ \deg P(x) < \dfrac{n+k}{2} \\ \text{maximize} \deg P(x) \end{cases};$$

$$\begin{cases} W(x)(3x^2 + 2x^3) \equiv P(x) \bmod x^4 - 1 \\ \deg P(x) < 3 \end{cases};$$

$$x^4 - 1 = (3x^2 + 2x^3)(3x + 3) + (x^2 - 1);$$

$$(3x + 3)(3x^2 + 2x^3) = -(x^2 - 1) + x^4 - 1;$$

$$(3x + 3)(3x^2 + 2x^3) \equiv 4x^2 + 1 \bmod x^4 - 1;$$

$$\begin{cases} W(x) = 3x + 3 = 3(x - 4) = 3(x - \alpha^2) \\ P(x) = 4x^2 + 1 \end{cases}.$$

Шаг 3. Деление.

$$M(x) = \frac{P(x)}{W(x)} = \frac{4x^2 + 1}{3x + 3} = 3x + 2.$$

Информационный многочлен есть

$$M(x) = \begin{bmatrix} m_0 & m_1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 0 & 0 \end{bmatrix} = 2 + 3x.$$

2.2.10. Замечания к разделу

Установленная связь между известными алгоритмами декодирования имеет важное методическое значение с точки зрения теории кодирования. Из раздела также видно, как перенос алгоритмов из временной области в частотную область приводит к упрощению их описания. На текущий момент алгоритм Гао и его модификации представляются автору самыми простыми для кодов с ограниченной длиной при любых реализациях. Доказательства корректности всех рассмотренных в разделе алгоритмов являются простыми и могут быть найдены в соответствующих

источниках и в популярных монографиях по теории кодирования [33, 7].

2.3. Декодирование в надкодах

Рассмотрим декодирование циклических кодов до истинного расстояния путем введения дополнительных тождеств, связывающих степенные суммы с коэффициентами локаторного многочлена.

Как известно, циклические коды обладают особенной структурой, позволяющей использовать алгебраические соотношения для их декодирования. Наиболее трудным оказалось исправление ошибок вплоть до истинного минимального расстояния при превышении последним конструктивного расстояния. Для многих кодов к настоящему времени эта задача решена [73]. Однако по-прежнему привлекательно явное вычисление коэффициентов локаторного многочлена. Эта задача имеет известную историю, но относительно недавно было найдено остроумное решение для кода Голея [60]. В настоящем разделе предлагается обоснование упомянутого решения, предложенное автором в работах [36, 69] и позволяющее обобщить результат на другие коды.

2.3.1. Основные определения

Пусть $\mathcal{G}$ — двоичный БЧХ (n, k, d)-код с множеством индексов нулей Z, с порождающим многочленом $g(x) = \prod_{i \in Z} (x - \beta^i)$ и с проверочной матрицей $H = \|\beta^{ij}\|$ над полем $\mathrm{GF}(2^m)$, $i \in Z$, $j \in [0, n-1]$, где n — длина кода, k — число информационных символов, d — расстояние кода в метрике Хэмминга, $\beta = \alpha^{(2^m-1)/n}$, α — примитивный элемент поля $\mathrm{GF}(2^m)$. Если для некоторого целого числа $b_0 \geq 0$ множество $\{b_0, b_0 + 1, \ldots, b_0 + \delta - 2\} \in Z$, то число δ называется конструктивным расстоянием кода.

Принятый из канала вектор есть $\mathbf{b} = \mathbf{a} + \mathbf{e} \pmod 2$, где $\mathbf{a}$ — вектор из кода $\mathcal{G}$, а $\mathbf{e} = (e_0, e_1, \ldots, e_{n-1})$ — вектор ошибок.

Многочлен локаторов $\sigma(z)$ для вектора ошибок $\mathbf{e}$ имеет вид

$$\sigma(z) = \prod_{i=1}^{t}(z - X_i) = \sum_{i=0}^{t} \sigma_i z^i,$$

где $X_i = \beta^{j_i}$, $i \in [1, t]$, если $e_{j_i} = 1$; $t = W(\mathbf{e})$ — вес Хэмминга вектора ошибок.

Степенную сумму

$$S_l = \sum_{i=1}^{t} X_i^l \tag{14}$$

назовем l-й компонентой синдрома.

Очевидно, что $S_{l+n} = S_l$, а индексы компонент синдрома следует вычислять по $\bmod\ n$.

Запись обобщенных тождеств Ньютона зависит от характеристики поля, в котором эти тождества используются, от определения многочлена локаторов и от способа нумерации его коэффициентов. В поле характеристики 2 обобщенные тождества Ньютона принимают вид

$$\begin{aligned} \sum_{i=0}^{j-1} S_{j-i}\sigma_{t-i} + j\sigma_{t-j} &= 0 \quad \text{при} \quad j \in [1, t]; \\ \sum_{i=0}^{t} S_{j-i}\sigma_{t-i} &= 0 \quad \text{при} \quad j > t. \end{aligned} \tag{15}$$

Этот вид, хотя и отличается от вида аналогичных тождеств из работ [38, 33], позволяет упростить запись приводимых ниже выражений. Используя тождества (15), можно исправлять до $\lfloor(\delta - 1)/2\rfloor$ ошибок в коде БЧХ [38, 33].

2.3.2. Дополнительные тождества

Из второго тождества в (15) и равенства $S_{l+n} = S_l$ следует

$$\sum_{i=0}^{t} S_{j-i}\sigma_{t-i} = 0 \quad \text{для любого} \quad j,$$

т. е. вектор $(\sigma_0, \sigma_1, \ldots, \sigma_t)$ ортогонален вектору $(S_1, S_2, \ldots, S_{n-1}, S_0)$ и его любому циклическому сдвигу. Следовательно, можно ввести циклический код $\mathcal{C}$ длины n над полем $\mathrm{GF}(2^m)$ с порождающим многочленом $\sigma(z) = \sum_{i=0}^{t} \sigma_i z^i$. Слово этого кода $\mathbf{c}$ соответствует многочлену $c(z) = \sum_{i=0}^{p} c_i z^i = \sigma(z)q(z)$, степень которого меньше n и для которого

$$\sum_{i=0}^{p} S_{j-i}c_{p-i} = 0 \quad \text{при любом} \quad j. \tag{16}$$

Последнее соотношение позволяет получить дополнительные тождества, связывающие известные степенные суммы (14), которые совместно с тождествами (15) дают возможность находить коэффициенты многочлена локаторов $\sigma(z)$ и исправлять в некоторых случаях больше ошибок, чем допускает конструктивное расстояние.

Разделим с остатком многочлен $u(z)$, $\deg u(z) = p > t$, на многочлен $\sigma(z)$:

$$u(z) = \sigma(z)q(z) + r(z), \quad \text{где} \quad \deg r(z) < t. \tag{17}$$

Многочлен

$$u(z) - r(z) = \sigma(z)q(z) = \sum_{i=0}^{p} \mu_i z^i$$

приводит к тождеству

$$\sum_{i=1}^{p+1} S_i \mu_{i-1} = 0. \qquad (18)$$

2.3.3. Алгоритм исправления трех ошибок

Рассмотрим процедуру исправления трех ошибок кодами с конструктивным расстоянием $\delta = 5$ и с истинным расстоянием $d \geq 7$. Алгебраический алгоритм декодирования, использующий обобщенные тождества Ньютона, исправляет всего две ошибки. По принятому вектору $\mathbf{b}$ можно вычислить только такие компоненты синдрома S_i, что $i \in Z$. Остальные компоненты синдрома считаем неопределенными. В частности, всегда не определена компонента S_5. Из обобщенных тождеств Ньютона получаются два линейно независимых уравнения, которые не содержат неопределенные компоненты синдрома:

$$\begin{cases} S_1\sigma_3 + \sigma_2 = 0 \\ S_3\sigma_3 + S_2\sigma_2 + S_1\sigma_1 + \sigma_0 = 0 \end{cases},$$

что приводит к

$$\begin{cases} \sigma_2 = S_1 \\ \sigma_0 = S_3 + S_2\sigma_2 + S_1\sigma_1 \end{cases}. \qquad (19)$$

Напомним, что всегда $\sigma_t = 1$. Этих уравнений, очевидно, недостаточно для определения коэффициентов многочлена локаторов $\sigma(z)$ при $t = 3$.

Из формулы (17) при $u(z) = z^p$ и $p = 6, 8, 10$ имеем

$$z^p = (z^3 + \sigma_2 z^2 + \sigma_1 z + \sigma_0)q(z) + \mu_2 z^2 + \mu_1 z + \mu_0,$$

где

$$\begin{cases} \mu_2 = \sigma_1^2 + \sigma_1\sigma_2^2 + \sigma_2^4 \\ \mu_1 = \sigma_0\sigma_2^2 + \sigma_1\sigma_2^3 \\ \mu_0 = \sigma_0^2 + \sigma_0\sigma_2^3 \end{cases} \quad \text{для } p = 6;$$

$$\begin{cases} \mu_2 = \sigma_0^2 + \sigma_1^3 + \sigma_1\sigma_2^4 + \sigma_2^6 \\ \mu_1 = \sigma_0\sigma_1^2 + \sigma_1^3\sigma_2 + \sigma_0\sigma_2^4 + \sigma_1\sigma_2^5 \\ \mu_0 = \sigma_0\sigma_1^2\sigma_2 + \sigma_0^2\sigma_2^2 + \sigma_0\sigma_2^5 \end{cases} \quad \text{для } p = 8;$$

$$\begin{cases} \mu_2 = \sigma_0^2\sigma_1 + \sigma_1^4 + \sigma_1^2\sigma_2^4 + \sigma_1\sigma_2^6 + \sigma_2^8 \\ \mu_1 = \sigma_0^3 + \sigma_0^2\sigma_1\sigma_2 + \sigma_0\sigma_2^6 + \sigma_1\sigma_2^7 \\ \mu_0 = \sigma_0^2\sigma_1^2 + \sigma_0^3\sigma_2 + \sigma_0^2\sigma_2^4 + \sigma_0\sigma_2^7 \end{cases} \quad \text{для } p = 10.$$

Таким образом, из (18) при $u(z) = z^p$ и при каждом значении p имеем по дополнительному уравнению

$$S_{p+1} + S_3\mu_2 + S_2\mu_1 + S_1\mu_0 = 0. \tag{20}$$

Подставляя (19) в уравнение (20) и используя тривиальное соотношение $S_2 = S_1^2$, получаем следующие уравнения:

$$(S_1^3 + S_3)\sigma_1^2 + (S_1^5 + S_1^2S_3)\sigma_1 + \\ +(S_1^7 + S_1^4S_3 + S_1S_3^2 + S_7) = 0 \quad \text{при } p = 6; \tag{21}$$

$$(S_1^3 + S_3)\sigma_1^3 + (S_1^5 + S_1^2S_3)\sigma_1^2 + (S_1^7 + S_1^4S_3)\sigma_1 + \\ +(S_1^9 + S_1^3S_3^2 + S_3^3 + S_9) = 0 \quad \text{при } p = 8; \tag{22}$$

$$(S_1^3 + S_3)\sigma_1^4 + (S_1^5 + S_1^2S_3)\sigma_1^3 + (S_1^4S_3 + S_1S_3^2)\sigma_1^2 + \\ +(S_1^3S_3^2 + S_3^3)\sigma_1 + (S_1^{11} + S_{11} + S_1^8S_3 + S_1^5S_3^2) = 0 \quad \text{при } p = 10. \tag{23}$$

Запишем формально алгоритм исправления трех ошибок.

Шаг 1. По принятому вектору $\mathbf{b} = (b_0, b_1, \ldots, b_{n-1})$ вычисляем компоненты синдрома $S_i,\ i \in Z$ по формуле

$$S_i = \sum_{j=0}^{n-1} b_j \beta^{ij}.$$

Шаг 2. Если $S_i = 0$ для всех $i \in Z$, то $t = 0$ (ошибок нет).

Шаг 3. Вычисляем $\Delta = S_1^3 + S_3$.

Шаг 4. Если $\Delta = 0$, то $t = 1$ и $X_1 = S_1$ (одна ошибка). Переход к шагу 9.

Шаг 5. Из дополнительных уравнений типа (20) находим σ_1 (для каждого кода используем свои дополнительные уравнения).

Шаг 6. Вычисляем

$$\sigma_2 = S_1; \quad \sigma_0 = \Delta + S_1\sigma_1.$$

Шаг 7. Составляем многочлен локаторов для вектора ошибок

$$\sigma(z) = z^3 + \sigma_2 z^2 + \sigma_1 z + \sigma_0.$$

Шаг 8. Находим корни многочлена локаторов в циклической группе $(\beta^0, \beta^1, \ldots, \beta^{n-1})$. Если $t = 2$, то многочлен локаторов имеет один корень, равный нулю, и два корня X_1 и X_2 в циклической группе $(\beta^0, \beta^1, \ldots, \beta^{n-1})$. Если $t = 3$, то многочлен локаторов имеет три корня X_1, X_2 и X_3 в циклической группе $(\beta^0, \beta^1, \ldots, \beta^{n-1})$.

Шаг 9. По найденному множеству $\{X_i\}$ исправляем ошибки.

Перейдем к рассмотрению алгоритма полного декодирования кода Голея.

Пример. Код Голея. Над полем $\mathrm{GF}(2^{11})$, заданным примитивным многочленом $x^{11} + x^2 + 1$, определим (23, 12, 7)-код Голея с порождающим многочленом $x^{11} + x^9 + x^7 + x^6 + x^5 + x + 1$ и с конструктивным расстоянием $\delta = 5$. Множество индексов нулей кода принадлежит одному циклотомическому классу, а именно {(1, 2, 4, 8, 16, 9, 18, 13, 3, 6, 12)}.

Преобразуем уравнение (22) к виду

$$(S_1^3 + S_3)\left(\left(\sigma_1 + S_1^2\right)^3 + \frac{C}{S_1^3 + S_3}\right) = 0,$$

где $C = S_1^6 S_3 + S_1^3 S_3^2 + S_3^3 + S_9$.

Тогда оно имеет единственное решение

$$\sigma_1 = S_1^2 + \sqrt[3]{\frac{C}{S_1^3 + S_3}}$$

над полем $\mathrm{GF}(2^{11})$.

Теперь подробно запишем шаг 5 алгоритма полного декодирования кода Голея.

Шаг 5. Вычисляем

$$\begin{aligned} C &= S_1^6 S_3 + S_1^3 S_3^2 + S_3^3 + S_9; \\ \sigma_1 &= S_1^2 + \sqrt[3]{\frac{C}{\Delta}} = S_1^2 + \left(\frac{C}{\Delta}\right)^{1365}. \end{aligned}$$

Рассмотрим еще пример.

Пример. (21, 7, 8)-код. Над полем $\mathrm{GF}(2^6)$, заданным примитивным многочленом $x^6 + x + 1$, определим (21, 7, 8)-код с конструктивным расстоянием $\delta = 5$. Множество индексов нулей кода принадлежит четырем циклотомическим классам, а именно $\{(1, 2, 4, 8, 16, 11), (3, 6, 12), (7, 14), (9, 18, 15)\}$.

Если квадратное уравнение (21) имеет два корня в поле $\mathrm{GF}(2^6)$, то один из них является посторонним. Разделим с остатком многочлены из уравнений (22) и (23) на многочлен (21). Если все корни уравнения (21) являются корнями уравнений (22) и (23), то оба многочлена должны делиться на многочлен (21) без остатка, что невозможно при $t > 1$ (доказательство этого факта достаточно громоздко, хотя и очевидно). Итак, посторонний

корень уравнения (21) не может быть одновременно корнем уравнений (22) и (23). Напротив, истинный корень (21) одновременно будет корнем уравнений (22) и (23).

Подробно запишем шаг 5 алгоритма декодирования (21, 7, 8)-кода.

Шаг 5. Находим корни квадратного уравнения (21) в поле $\mathrm{GF}(2^6)$. Если имеется один корень кратности два, то он принимается за σ_1. Если имеются два различных корня, то путем их подстановки в уравнения (22) и (23) выделяется истинный корень, который принимается за σ_1.

2.3.4. Декодирование свыше конструктивного расстояния на основе надкодов

В статье [48] предложено декодирование в надкодах. Надкодом $\mathcal{G}_i$ линейного блокового кода $\mathcal{G}$ называется код, содержащий все кодовые слова исходного кода. Пусть $\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_s$ — набор надкодов для кода $\mathcal{G}$, $\mathcal{G} \subset \mathcal{G}_i$, $i \in [1, s]$. Алгоритм декодирования ошибок кратности до t в надкодах состоит в следующем.

Шаг 1. Для каждого кода $\mathcal{G}_i$ проводится декодирование в список L_i. Список строится таким образом, что для любых ошибок кратности до t список L_i содержит переданное кодовое слово.

Шаг 2. Вычисляется список L как пересечение списков L_i.

Шаг 3. Из L выбирается слово, наиболее близкое к принятому вектору.

Алгоритм основан на том, что итоговый список L оказывается существенно меньшим, чем отдельные списки L_i. Поэтому поиск переданного слова в L имеет небольшую алгоритмическую сложность.

В настоящем разделе рассматривается декодирование циклических кодов свыше конструктивного расстояния с помощью надкодов.

Для описания алгоритма декодирования некоторого кода $\mathcal{G}$ с помощью надкодов требуется задать множество надкодов $\mathcal{G}_i$, $i \in [1, s]$, указать метод их декодирования в списки L_i и способ построения итогового списка-пересечения L.

Пусть Z — множество индексов нулей циклического кода $\mathcal{G}$, исправляющего ошибки кратности до t. Любой код, у которого множество индексов нулей Z_i является подмножеством Z ($Z_i \subset Z$), задает надкод $\mathcal{G}_i$ кода $\mathcal{G}$. При этом код $\mathcal{G}_i$ имеет расстояние, меньшее чем $\mathcal{G}$, что, вообще говоря, не позволяет исправлять ошибки кратности t. Декодируя в нем ошибки кратности до t, получим список L_i, содержащий переданное слово.

Выбирая s подмножеств $Z_i \subset Z$, $i \in [1, s]$, получим s надкодов. Декодируя принятое слово в этих надкодах, получим списки L_i, $i \in [1, s]$, в каждом из которых находится переданное слово. Поэтому список-пересечение L также содержит это слово. Эффективность алгоритма декодирования достигается в том случае, когда итоговый список L становится небольшим, например, состоит из одного (искомого) вектора. При декодировании надкодов $\mathcal{G}_i$ не обязательно получать список L_i в явном виде. Декодирование упрощается, если удается описать пересечение списков L_i без выписывания самих списков.

В статье [36] предложено декодирование циклических кодов свыше конструктивного расстояния с использованием дополнительных тождеств, связывающих степенные суммы с коэффициентами локаторного многочлена.

Используя результаты этой работы, в дальнейшем представим описание списков L_i как множества решений системы уравнений, связывающих степенные суммы и коэффициенты многочленов локаторов ошибок. Это описание позволяет осуществить пересечение списков L_i без их выписывания.

Формулу (16) можно использовать для получения дополнительных тождеств. Выберем слово $\mathbf{c} = (c_0, c_1, \ldots, c_{n-1}) \in \mathcal{C}$ с множеством индексов ненулевых компонент $\{0, 1, \ldots, t-2, t-1; p\}$. Если множество индексов компонент синдрома $\{j-p, j-p+1, \ldots, j-p+t-2, j-p+t-1; j\} \in Z$, то все составляющие формулы (16)

будут определены. Положим $j = p + 1$, тогда при $p + 1 \in Z$, $p > t$, имеем Z_i,

$$Z \supset Z_i \supset \{1, 2, \ldots, t-1, t; p+1\},$$

задающее надкод $\mathcal{G}_i$ и дополнительное тождество (18):

$$\sum_{i=1}^{p+1} S_i \mu_{i-1} = 0,$$

где коэффициенты μ однозначно определяются выбранным кодовым словом $\mathbf{c} \in \mathcal{C}$. Таким образом, для каждого надкода $\mathcal{G}_i$ имеем дополнительное тождество, которое совместно с тождествами (15) позволяет провести декодирование в список L_i.

2.3.5. Пример

Рассмотрим процедуру исправления трех ошибок кодами с конструктивным расстоянием $\delta = 5$ и с истинным расстоянием $d \geq 7$ при тех же условиях, что и в параграфе 2.3.3.

Построим надкоды для $p = 6, 8, 10$. Имеем

$$\begin{array}{lll} Z_1 & \supset & \{1, 2, 3, 7\}; \\ Z_2 & \supset & \{1, 2, 3, 9\}; \\ Z_3 & \supset & \{1, 2, 3, 11\} \end{array} \qquad (24)$$

и дополнительные соотношения для каждого из этих надкодов (21)–(23).

Над полем $\mathrm{GF}(2^6)$, заданным примитивным многочленом $x^6 + x + 1$, определим (21, 7, 8)-код $\mathcal{G}$ с конструктивным расстоянием $\delta = 5$. Множество индексов нулей кода принадлежит четырем циклотомическим классам $Z = C_1 \cup C_3 \cup C_7 \cup C_9 = \{(1, 2, 4, 8, 16, 11), (3, 6, 12), (7, 14), (9, 18, 15)\}$.

Выберем множество индексов нулей надкодов в соответствии с (24), учитывая структуру циклотомических классов кода $\mathcal{G}$: $Z_1 = C_1 \cup C_3 \cup C_7$, $Z_2 = C_1 \cup C_3 \cup C_9$ и $Z_3 = C_1 \cup C_3$, которые

задают надкоды $\mathcal{G}_1$, $\mathcal{G}_2$ и $\mathcal{G}_3$ соответственно. Список L_1 для надкода $\mathcal{G}_1$ определяется решением системы (19) и (21), L_2 — (19) и (22) и L_3 — (19) и (23). Каждому корню уравнений (21)–(23) соответствует единственное кодовое слово в соответствующем списке кодовых слов L_i, а пересечение списков L есть совместное решение системы (19), (21)–(23). Надкоды $\mathcal{G}_1$ и $\mathcal{G}_2$ имеют единственное общее слово в списках L_1 и L_2, соответствующее общему корню уравнений (21) и (22)

$$\sigma_1 = \frac{S_1^9 + S_1^3 S_3^2 + S_3^3 + S_9}{S_1 S_3^2 + S_7}$$

при $S_1 S_3^2 + S_7 \neq 0$. Если $S_1 S_3^2 + S_7 = 0$, то для надкодов $\mathcal{G}_1$ и $\mathcal{G}_3$ единственный общий корень уравнений (21) и (23)

$$\sigma_1 = \frac{S_1^8 (S_1^{24} + S_3)}{S_1^9 + S_3^3}.$$

Теперь запишем формально алгоритм исправления трех ошибок в (21, 7, 8)-коде $\mathcal{G}$.

Шаг 1. По принятому вектору $\mathbf{b} = (b_0, b_1, \ldots, b_{n-1})$ вычисляем компоненты синдрома $S_i,\ i \in Z$ по формуле

$$S_i = \sum_{j=0}^{n-1} b_j \beta^{ij}.$$

Шаг 2. Если $S_i = 0$ для всех $i \in Z$, то $t = 0$ (ошибок нет).

Шаг 3. Вычисляем $\Delta = S_1^3 + S_3$.

Шаг 4. Если $\Delta = 0$, то $t = 1$ и $X_1 = S_1$ (одна ошибка). Переход к шагу 9.

Шаг 5. Если $\Delta \neq 0$, то для надкодов $\mathcal{G}_1$ и $\mathcal{G}_2$ вычисляем общий корень уравнений (21) и (22)

$$\sigma_1 \quad = \quad \frac{S_1^9 + S_1^3 S_3^2 + S_3^3 + S_9}{S_1 S_3^2 + S_7} \qquad \text{при} \qquad S_1 S_3^2 + S_7 \neq 0,$$

или для надкодов $\mathcal{G}_1$ и $\mathcal{G}_3$ — общий корень уравнений (21) и (23)

$$\sigma_1 \quad = \quad \frac{S_1^8(S_1^{24}+S_3)}{S_1^9+S_3^3} \qquad \text{при} \qquad S_1S_3^2+S_7=0.$$

Шаг 6. Вычисляем

$$\sigma_2 = S_1; \quad \sigma_0 = \Delta + S_1\sigma_1.$$

Шаг 7. Составляем многочлен локаторов для вектора ошибок

$$\sigma(z) = z^3 + \sigma_2 z^2 + \sigma_1 z + \sigma_0.$$

Шаг 8. Находим корни многочлена локаторов в циклической группе $(\beta^0, \beta^1, \ldots, \beta^{n-1})$. Если $t=2$, то многочлен локаторов имеет один корень, равный нулю, и два корня X_1 и X_2 в циклической группе $(\beta^0,\ \beta^1,\ \ldots,\ \beta^{n-1})$. Если $t=3$, то многочлен локаторов имеет три корня X_1, X_2 и X_3 в циклической группе $(\beta^0,\ \beta^1,\ \ldots,\ \beta^{n-1})$.

Шаг 9. По найденному множеству $\{X_i\}$ исправляем ошибки.

Замечания к главе

Алгебраическое декодирование по укорочениям кодов представляет собой результат автора (совместно с Е. Т. Мирончиковым) [35]. Основной метод декодирования второго раздела предложен С. Гао [76] и А. Шиозаки [92], однако анализ алгоритма, доказательство его корректности и “генетическая” связь с другими алгоритмами являются оригинальным результатом автора [63, 64]. Метод алгебраического декодирования в надкодах предложен автором (совместно с Е. Т. Мирончиковым и Е. А. Круком) в работах [36, 69].

3. Вычисление дискретного преобразования Фурье над конечным полем

Вычисление ДПФ над конечным полем представляет собой самостоятельную и интересную задачу, приложения которой тесно связаны с алгоритмами кодирования и декодирования алгебраических кодов. Вычисление корней многочлена можно рассматривать как в связи с вычислением ДПФ, так и независимо. Циклотомический и рекуррентный алгоритмы вычисления ДПФ в случае, когда необходимо минимизировать число умножений, являются лучшими алгоритмами для малых длин ($n \leq 511$). Неполное ДПФ применяют при вычислении синдрома для алгебраических кодов.

3.1. Вычисление корней многочлена

В настоящем разделе вводятся новые методы поиска корней многочленов над конечным полем, основанные на работах автора [71, 72, 57]. Это позволит значительно ускорить декодирование алгебраических кодов, включая БЧХ, Рида – Соломона, Гоппы и альтернантных.

Известно, что одним из самых сложных по времени этапов декодирования алгебраических кодов является поиск корней локаторного многочлена. Обычно для этого используют процедуру Ченя [54], которая состоит из подстановки всех элементов конечного поля в многочлен и сравнения полученных значений с нулем. Эта процедура становится довольно сложной при вычислениях в больших конечных полях и для многочленов большой степени.

В работе [55] было показано, что каждый многочлен степени не выше 5 может быть преобразован в каноническую форму с одним или двумя параметрами, так что возможно построить таблицы для поиска корней. Кроме того, если некоторые корни располо-

жены в одном и том же циклотомическом классе, то возможно их вычислить с помощью алгоритма Евклида. Трунг, Дженг и Рид предложили преобразование [96], которое позволяет группировать некоторые слагаемые в многочлене степени не выше 11 в произведение аффинных многочленов. Так как значения аффинных многочленов в элементах конечного поля можно легко вычислить, используя предварительно составленные небольшие таблицы, то возможно ускорение вычислений. Однако алгоритм [96] имеет следующие недостатки.

1. Он применим только для многочленов степени не выше, чем 11.

2. Требуется преобразование многочленов. Предложенное преобразование $y = x + f_6/f_7$ для многочлена $F(x) = \sum_{i=0}^{11} f_i x^i$ не может применяться, если $f_7 = 0$, так что алгоритм поиска корней становится более сложным.

3. После преобразования многочлен содержит слагаемые $f_{10}y^{10} + f_9 y^9$ (и $f_6 x^6$, если преобразование невозможно). Вычисление значений многочленов для этих слагаемых требует применения процедуры Ченя.

В настоящем разделе вводится общий подход к разложению и вычислению значений многочленов. Описание приведено для случая $\mathrm{GF}(2^m)$, но может быть обобщено для произвольного конечного поля. Эта методика может быть использована для ускорения процедуры Ченя.

Проблема нахождения корней многочлена может быть формально определена как нахождение всех различных $x_i : F(x_i) = 0$, $F(x) = \sum_{j=0}^{t} f_j x^j$, $\quad x_i, f_j \in \mathrm{GF}(2^m)$. Процедура Ченя решает эту проблему путем вычисления значений многочлена $F(x)$ во всех элементах конечного поля $x \in \mathrm{GF}(2^m) \setminus \mathbf{0}$ с временной сложностью

$$W = (C_{\text{add}} + C_{\text{mul}})\, t\, (2^m - 1), \tag{25}$$

где C_{add} и C_{mul} — временная сложность вычисления сложения и умножения элементов в конечном поле соответственно.

Вводимый ниже алгоритм [71] уменьшает сложность вычисления значений многочлена в одной точке за счет использования специального упорядочения элементов конечного поля.

3.1.1. Алгоритм быстрого поиска корней многочлена

Перед описанием алгоритма запишем необходимые понятия и определения.

Определение 3.1. *Многочлен $L(y)$ над конечным полем* $\mathrm{GF}(2^m)$ *называется линеаризованным многочленом, если он может быть представлен в виде*

$$L(y) = \sum_i L_i y^{2^i}, \quad L_i \in \mathrm{GF}(2^m).$$

Основное свойство линеаризованных многочленов описывается следующей леммой.

Лемма 3.1 [5]. *Пусть $y \in \mathrm{GF}(2^m)$ и $\alpha^0, \ldots, \alpha^{m-1}$ — стандартный базис конечного поля. Если*

$$y = \sum_{k=0}^{m-1} y_k \alpha^k, \quad y_k \in \mathrm{GF}(2),$$

и $L(y) = \sum_j L_j y^{2^j}$, то

$$L(y) = \sum_{k=0}^{m-1} y_k L(\alpha^k).$$

Определение 3.2. *Многочлен $A(y)$ над $\mathrm{GF}(2^m)$ называется аффинным многочленом, если он может быть представлен в виде*

$$A(y) = L(y) + \beta, \quad \beta \in \mathrm{GF}(2^m),$$

где $L(y)$ — линеаризованный многочлен.

Описанная выше лемма позволяет вычислять значения аффинного многочлена $A(x)$ для каждого элемента конечного поля $x_i \in \mathrm{GF}(2^m)$, затрачивая только одно сложение, если все элементы конечного поля x_i упорядочены в их векторном представлении как код Грея.

Определение 3.3. *Код Грея есть упорядочение всех двоичных векторов длины m таким образом, что два соседних вектора отличаются только в одном бите.*

Если векторы $\mathbf{x}_i \in \mathrm{GF}(2^m)$ упорядочены как код Грея, т. е. $wt(\mathbf{x}_i - \mathbf{x}_{i-1}) = 1$, где $wt(\mathbf{a})$ — вес Хэмминга вектора $\mathbf{a}$, то справедлива следующая цепочка равенств:

$$A(\mathbf{x}_i) = A(\mathbf{x}_{i-1}) + L(\Delta_i), \quad \Delta_i = \mathbf{x}_i - \mathbf{x}_{i-1} = \alpha^{\delta(\mathbf{x}_i, \mathbf{x}_{i-1})},$$

где $\delta(\mathbf{x}_i, \mathbf{x}_{i-1})$ указывает на позицию вектора $\mathbf{x}_i$, отличающуюся от вектора $\mathbf{x}_{i-1}$. Если $\mathbf{x}_0 = 0$, то положим $A(\mathbf{x}_0) = \beta$, и записанное выше правило позволяет описать алгоритм вычисления значений многочлена $A(x)$ во всех точках конечного поля $\mathrm{GF}(2^m)$.

Пример. Рассмотрим конечное поле $\mathrm{GF}(2^3)$, заданное примитивным многочленом $\pi(\alpha) = \alpha^3 + \alpha + 1$ над $\mathrm{GF}(2)$. Одним из возможных вариантов кода Грея является последовательность 000, 001, 011, 010, 110, 111, 101, 100 или $0, 1, \alpha^3, \alpha, \alpha^4, \alpha^5, \alpha^6, \alpha^2$. Для выполнения алгоритма быстрого вычисления корней многочлена необходимо предварительно запомнить таблицу величин $L(\alpha^0), L(\alpha^1), L(\alpha^2)$. Тогда $A(1) = A(0) + L(\alpha^0)$, $A(\alpha^3) = A(1) + L(\alpha^1)$ и так далее.

Этот алгоритм может быть применен для поиска корней любого многочлена, если разложить его в сумму аффинных сомножителей.

Теорема 3.1. *Любой многочлен* $F(x) = \sum_{j=0}^{t} f_j x^j, \quad f_j \in \mathrm{GF}(2^m)$, *может быть представлен как*

$$F(x) = f_3 x^3 + \sum_{i=0}^{\lceil (t-4)/5 \rceil} x^{5i} \left(f_{5i} + \sum_{j=0}^{3} f_{5i+2^j}\, x^{2^j} \right),$$

где $\lceil a \rceil$ *— наименьшее целое число, большее или равное* a.

Доказательство. Пусть k — такое наименьшее целое число, что $5k - 1 \geq t$, и положим для всех $i > t$ коэффициенты многочлена равными нулю: $f_i = 0$. Тогда доказываемое выражение может быть записано как

$$F(x) = F_k(x) = f_3 x^3 + \sum_{i=0}^{k-2} x^{5i} \left(f_{5i} + \sum_{j=0}^{3} f_{5i+2^j} x^{2^j} \right) +$$

$$+ x^{5(k-1)} \left(f_{5(k-1)} + \sum_{j=0}^{2} f_{5(k-1)+2^j} x^{2^j} \right).$$

Докажем истинность последнего выражения методом полной математической индукции. Для $t = 4$ $(k = 1)$ это очевидно.

Предположим, что $F_k(x)$ может быть разложен на слагаемые, как описано выше. Тогда $F_{k+1}(x) = F_k(x) + x^{5k}(f_{5k} + f_{5k+1}x + f_{5k+2}x^2 + f_{5k+4}x^4) + x^{5(k-1)} f_{5(k-1)+8} x^8$. Последнее слагаемое этого выражения может быть сгруппировано с последним слагаемым разложения $F_k(x)$. Ч.т.д.

Линеаризованный многочлен, полученный в этом разложении, имеет только 4 слагаемых. В некоторых случаях введение дополнительных слагаемых позволяет уменьшить общее число аффинных многочленов в конечном разложении.

Запишем алгоритм поиска корней многочлена в виде последовательности шагов.

Шаг 1. Вычисляем $L_i^{(k)} = L_i(\alpha^k)$, $k = [0, m-1]$, $i \in [0, \lceil (t-4)/5 \rceil]$, где $L_i(x)$ — линеаризованный многочлен, представляемый в виде разложения: $L_i(x) = \sum_{j=0}^{3} f_{5i+2^j} x^{2^j}$.

Шаг 2. Инициализация $A_i^{(0)} = f_{5i}$.

Шаг 3. Представляем каждый элемент $\mathbf{x}_j \in \mathrm{GF}(2^m)$, $j \in [0, 2^m - 1]$, в стандартном базисе как элемент кода Грея с $\mathbf{x}_0 = 0$, вычисляем $A_i^{(j)} = A_i^{(j-1)} + L_i^{(\delta(\mathbf{x}_j, \mathbf{x}_{j-1}))}$, $j \in [1, 2^m - 1]$.

Шаг 4. Вычисляем $F(x_j) = f_3 x_j^3 + \sum_{i=0}^{\lceil (t-4)/5 \rceil} x_j^{5i} A_i^{(j)}$, $j \in [1, 2^m - 1]$, и $F(0) = f_0$. Если $F(x_j) = 0$, то x_j — корень многочлена. Заметим, что второе слагаемое в этой сумме может быть вычислено с использованием метода Горнера.

Общая временная сложность этого алгоритма, состоящая из сложности предварительных вычислений (первое слагаемое) и сложности вычисления значений многочленов (второе слагаемое), равна

$$W_{\text{fast}} = m \left\lceil \frac{t+1}{5} \right\rceil (4C_{\text{mul}} + 3C_{\text{add}}) + \\ + \left(\left\lceil \frac{t+1}{5} \right\rceil (2C_{\text{add}} + C_{\text{mul}}) + 2C_{\text{exp}} \right) (2^m - 1), \qquad (26)$$

где через C_{exp} обозначена временная сложность вычисления возведения в степень над конечным полем.

3.1.2. Результаты моделирования

Предложенный алгоритм был реализован на языке программирования C++ с использованием компилятора MS Visual C++ 6.0. Моделирование проводилось на персональном компьютере, оснащенном процессором AMD Athlon 1700 XP. Умножение элементов в конечном поле $\mathrm{GF}(2^8)$ осуществлялось с использованием

таблиц логарифмов и антилогарифмов. Время, затраченное для вычисления значений многочленов в элементах конечного поля $\alpha^0, \ldots, \alpha^{254}$, было усреднено более чем для 100 000 вычислений и показано в табл. 3.

Заметим, что относительное время вычисления для метода Трунга, Дженга и Рида значительно больше, чем показано в работе [96]. Это объясняется различием в реализации выполнения операции умножения, используемого в проведенном моделировании и в работе [96].

Сравнение выражений (25), (26) и соответствующих экспериментальных результатов показывает, что предложенный алгоритм работает до 2,6 раз быстрее, чем процедура Ченя в зависимости от реализации выполнения операций над конечным полем $\mathrm{GF}(2^m)$.

Таким образом, можно сделать вывод, что предложенный в разделе алгоритм для вычисления значений произвольного многочлена в большом множестве элементов конечного поля работает значительно эффективнее, чем процедура Ченя. Кроме того, иногда возможно получение еще большего выигрыша при реализации алгоритма, если использовать другие разложения многочленов.

3.1.3. Специальные разложения многочленов

В настоящем параграфе вводятся два специальных разложения многочленов, которые могут быть полезны при декодировании (255, 239, 17)- и (255, 223, 33)-кодов Рида – Соломона.

Пример. Многочлены степени, не превышающей 8, могут быть разложены как

$$\begin{aligned} F(x) = &\sum_{i=0}^{8} f_i x^i = && A_1(x) + x^3(A_2(x) + f_6 x^3); \\ A_1(x) = &\ f_0 + L_1(x) = && f_0 + f_1 x + f_2 x^2 + f_4 x^4 + f_8 x^8; \\ A_2(x) = &\ f_3 + L_2(x) = && f_3 + f_5 x^2 + f_7 x^4. \end{aligned} \qquad (27)$$

Таблица 3

Время вычисления корней многочленов в микросекундах

Степень многочлена	Процедура Ченя	Метод [96]	Предложенный метод	Выигрыш для предложенного метода в разах
6	17.2	16.7	14.9	1.15
7	19.8	18.2	15.1	1.31
8	22.2	19.6	15.2	1.46
9	24.6	20.3	15.3	1.60
10	27.2	20.9	17.3	1.57
11	29.6	20.6	18.2	1.62
16	42.3	—	21.4	1.97
24	61.8	—	25.8	2.39
32	81.4	—	31.4	2.59

Разложение, использованное в этом примере, позволяет исключить одно возведение в степень перемещением одного слагаемого в аффинный многочлен. При применении этого разложения временная сложность вычисления значений многочлена во всех элементах конечного поля $\mathrm{GF}(2^m)$ составляет

$$W_8 = m(6C_{\mathrm{mul}} + 4C_{\mathrm{add}}) + (4C_{\mathrm{add}} + 2C_{\mathrm{mul}} + C_{\mathrm{exp}})\,(2^m - 1).$$

Из сравнения последнего выражения с формулой (26) видно, что сложность разложения уменьшается как на стадии предварительных вычислений, так и на одно возведение в степень для каждого элемента конечного поля в основной стадии вычислений.

Пример. Многочлены степени, не превышающей 17, могут быть разложены как

$$\begin{aligned} F(x) = \;& \widetilde{A_1}(x) + x^3(A_2(x) + x^3(f_6 + x^3(A_3(x) + \\ & + x^3(A_4(x) + f_{15}x^3)))); \\ A_3(x) = \;& f_9 + f_{10}x + f_{11}x^2 + f_{13}x^4 + f_{17}x^8; \\ A_4(x) = \;& f_{12} + f_{14}x^2, \end{aligned} \tag{28}$$

где $\widetilde{A_1}(x) = A_1(x) + f_{16}x^{16}$. При применении этого разложения временная сложность вычисления значений многочлена во всех элементах конечного поля $\mathrm{GF}(2^m)$ составляет

$$W_{17} = m(12C_{\mathrm{mul}} + 8C_{\mathrm{add}}) + (9C_{\mathrm{add}} + 5C_{\mathrm{mul}} + C_{\mathrm{exp}})\,(2^m - 1).$$

Очевидно, что по сравнению с (26) сложность последнего разложения уменьшается на стадии предварительных вычислений, а на основной стадии вычислений одно возведение в степень заменяется на одно сложение и одно умножение для каждого элемента конечного поля. Следовательно, соотношение сложностей выполнения операций умножения и возведения в степень должно учитываться при реализации вычислений.

3.1.4. Гибридный метод

Известно много методов вычисления корней многочленов небольшой степени с очень низкой сложностью [53, 55, 10, 31]. Поэтому при комбинировании процедуры Ченя с одним из этих методов ожидается значительное уменьшение сложности вычисления корней многочлена по сравнению с полной процедурой Ченя. Такое комбинирование может быть реализовано следующим образом. Многочлен локаторов ошибок $F(x)$ делится на множитель $(x - x_i)$ после того, как корень x_i был обнаружен. Далее применяется аналитический алгоритм для поиска корней многочленов, когда степень многочлена станет достаточно малой.

В этом параграфе предлагается вероятностный алгоритм для ускорения процедуры Ченя. Пусть процедура Ченя применяется к многочлену $F(x)$, имеющему t корней во множестве $X = \{x_i \in \mathrm{GF}(2^m) \setminus \mathbf{0} \mid i \in [1, n]\}$, $n = 2^m - 1$. Вероятность того, что k-й элемент X будет d-м найденным корнем при применении процедуры Ченя, есть

$$p\{F(x_k) = 0_d\} = \frac{\binom{k}{d}\binom{n-k}{t-d}}{\binom{n}{t}}\frac{d}{k}.$$

Это — вероятность того, что первые k значений, выбранных из X, содержат d корней многочлена $F(x)$ и последнее выбранное значение — также корень.

Таким образом, средняя временная сложность нахождения d корней есть

$$W = \sum_{k=1}^{n} C(t) k\, p\{F(x_k) = 0_d\} = C(t)\frac{d}{\binom{n}{t}} \sum_{k=1}^{n} \binom{k}{d}\binom{n-k}{t-d},$$

где $C(t)$ — временная сложность вычисления значения многочлена степени t в одной точке при применении некоторого алгоритма. Используя известное комбинаторное тождество [14, (5.26)]

$$\sum_{k=0}^{r} \binom{r-k}{m} \binom{s+k}{n} = \binom{r+s+1}{m+n+1},$$

получим

$$W = C(t)\, d\, \frac{n+1}{t+1}. \tag{29}$$

Из вышеприведенных рассуждений видно, что для многочленов небольшой степени t применение гибридного метода для поиска корней существенно сокращает временную сложность.

Формально опишем гибридный алгоритм поиска всех различных корней $x_i \in \mathrm{GF}(2^m)$ многочлена $F(x) = \sum_{i=0}^{t} f_i x^i$, $f_i \in \mathrm{GF}(2^m)$.

Пусть ранее были найдены корни $x_j, \quad j \in [1,k], \quad k \le t$. Для нахождения оставшихся корней может быть использован следующий алгоритм.

Шаг 1. Понизить степень многочлена: $F(x) = \dfrac{F(x)}{\prod_{j=1}^{k}(x - x_j)}, \quad t = t - k.$

Шаг 2. Если степень полученного многочлена $t > 4$, то:

2.1. Разложить многочлен на сумму аффинных многочленов по формуле (27) — для многочленов степени не выше 8 [или по формуле (28) — для многочленов степени не выше 17].

2.2. Произвести инициализацию $A_i = A_i(0)$.

2.3. Подготовить таблицы значений линеаризованных многочленов $L_i(x) = A_i(x) - A_i(0)$: $\quad L_{ij} = L_i(\alpha^j), \quad j \in [0, m-1]$.

2.4. Упорядочив векторное представление элементов конечного поля $\mathbf{x}_j$ в соответствии с кодом Грея, последовательно вычислить значения $F(\mathbf{x}_j)$ в соответствии с (27) (или (28)), причем на каждом шаге $A_i(\mathbf{x}_j) = A_i = A_i + L_{i\,\delta(\mathbf{x}_j, \mathbf{x}_{j-1})}$, где $\mathbf{x}_0 = 0$, $\delta(\mathbf{x}_j, \mathbf{x}_{j-1})$ — функция, указывающая позицию, в которой различаются два соседних элемента кода Грея.

2.5. Если $F(x_j) = 0$, то данный элемент является корнем многочлена. Исходный многочлен должен быть разделен на $(x - x_j)$: $F(x) = \dfrac{F(x)}{x - x_j}$, $t = t - 1$.

Шаг 3. Если степень полученного многочлена $t \leq 4$, то применить алгебраические методы, описанные в параграфе 3.1.5.

Заметим, что если в поле $\mathrm{GF}(2^8)$ аналитический алгоритм поиска корней из параграфа 3.1.5 не используется, то среднее число операций для поиска корней многочлена степени $t = 8$ составляет 4080 для процедуры Ченя и 1712 для предложенного метода. При $t = 16$ среднее число операций составляет 8160 и 3822 соответственно.

3.1.5. Аналитические методы вычисления корней многочленов степени до четырех

В настоящем параграфе будут рассмотрены алгоритмы вычисления корней многочленов небольшой степени. Идеи, лежащие в основе этих методов, широко известны [5, 10, 100]. Предлагаем некоторое улучшение этих методов.

Многочлены второй степени. Так как все многочлены второй степени являются аффинными, то их корни можно найти, решая соответствующую систему линейных уравнений по методу Берлекэмпа – Рамсея – Соломона [5]. Кроме того, можно показать, что любой многочлен второй степени с ненулевыми коэффициентами может быть приведен к виду $P(x) = x^2+x+a$. Пусть имеется многочлен второй степени $F(x) = x^2+ax+b$, выполним подстановку $x = ay$. В тех случаях, когда $a = 0$, подстановка невозможна, но многочлен имеет один корень кратности два: $x = \sqrt{b}$. Многочлен преобразуется в $G(y) = y^2 + y + c$, $c = \dfrac{b}{a^2}$. Далее будем полагать, что многочлен уже приведен к виду $P(x) = x^2 + x + a$. Его корни могут быть найдены с помощью только одного матричного

умножения [100]. Уравнение $P(x) = 0$ может быть представлено в векторной форме как

$$\overline{\mathbf{x}}L = \overline{\mathbf{a}},$$

или

$$(x_0, x_1, ..., x_{m-1}) \begin{bmatrix} L(\alpha^0) \\ L(\alpha) \\ \cdots \\ L(\alpha^{m-1}) \end{bmatrix} = (a_0, a_1, \ldots, a_{m-1}),$$

причем $L(y) = y^2 + y$.

Так как $y^2 + y = 0$ тогда и только тогда, когда $y = 0$ или $y = 1$, то $\operatorname{rank} L = m - 1$. Следовательно, существует такая $m \times m$ матрица M, что

$$LM = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 & 0 \\ & & & \cdots & & & \\ 0 & 0 & 0 & \cdots & 1 & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 & 0 \end{bmatrix}$$

(в некоторых полях правая часть выражения может отличаться от вышеприведенной на сомножитель-перестановку).

Решение рассматриваемого уравнения (если оно существует) может быть представлено как

$$\overline{\mathbf{x}} = (c, d_0, d_1, \ldots, d_{m-2}),$$

где $\overline{\mathbf{d}} = \overline{\mathbf{a}}M = (d_0, d_1, \ldots, d_{m-1}), \quad c \in \{0, 1\}$.

Решение существует тогда и только тогда, когда $d_{m-1} = 0$. Так как матрица M может быть вычислена для любого конечного поля предварительно, то корни многочлена могут быть найдены с помощью очень простой логической схемы. Более того, существует много способов выбора матрицы M, и всегда можно выбрать из них матрицу, минимизирующую сложность реализации.

Далее будем использовать этот метод, так как он кажется более эффективным по сравнению с представленным в работе [53], потому что не требует возведения в степень над конечным полем.

Приведем в качестве примера матрицу M над конечным полем $\mathrm{GF}(2^8)$, заданным многочленом $\pi(x) = x^8 + x^7 + x^2 + x + 1$ над $\mathrm{GF}(2)$:

$$M = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}.$$

Многочлены третьей степени. Корни любого многочлена $P(x) = x^3 + ax^2 + bx + c$ могут быть аналитически выражены с помощью резольвент Лагранжа [10]:

$$\begin{aligned} r_0 &= \quad x_0 + x_1 + x_2 \quad = a; \\ r_1 &= x_0 + wx_1 + w^2 x_2; \\ r_2 &= x_0 + w^2 x_1 + wx_2, \end{aligned}$$

где $w^3 = 1$.

Корни могут быть найдены применением обратного преобразования Фурье к вектору резольвент:

$$\begin{aligned} x_0 &= \quad r_0 + r_1 + r_2; \\ x_1 &= r_0 + w^2 r_1 + wr_2; \\ x_2 &= r_0 + wr_1 + w^2 r_2. \end{aligned}$$

Можно показать, что резольвенты Лагранжа удовлетворяют тождеству $g(R_i) = 0$ [100], где $g(x) = x^2 + (ab + c)x + (a^2 + b)^3$,

$R_i = r_i^3, \quad i = 1, 2$. Корни последнего уравнения могут быть вычислены по методике решения квадратного уравнения, описанной выше. Учитывая, что извлечение кубического корня может иметь неоднозначный результат в некоторых конечных полях, для вычисления корректных значений r_i следует применять теорему Виета [10].

С другой стороны, вычисления упростятся, если кубический многочлен $P(x) = x^3 + ax^2 + bx + c$ преобразовать к аффинному многочлену четвертой степени путем умножения на $(x - a)$. Это приводит к модифицированному многочлену $\widehat{P}(x) = x^4 + qx^2 + px + r$, где $q = b - a^2$, $p = -ab + c$, $r = -ac$. Для нахождения корней последнего многочлена может быть использована рассмотренная ниже процедура для случая многочленов четвертой степени (корень $x = a$ должен быть отброшен).

Многочлены четвертой степени. Для нахождения корней x_1, x_2, x_3, x_4 многочлена четвертой степени $P(x) = x^4 + ax^3 + bx^2 + cx + d$ необходимо найти величины

$$\begin{aligned} \theta_1 &= (x_1 + x_2)(x_3 + x_4); \\ \theta_2 &= (x_1 + x_3)(x_2 + x_4); \\ \theta_3 &= (x_1 + x_4)(x_2 + x_3). \end{aligned}$$

Они являются корнями многочлена $q(x) = x^3 + (ac + b^2)x + (abc + a^2d + c^2) = 0$ [10, 100]. Обозначим через ξ_1, ξ_2 корни многочлена $\xi^2 + a\xi + \theta_1$, а через η_1, η_2 — корни многочлена $\eta^2 + (\theta_1 + b)\eta + d$. Тогда корнями исходного многочлена $P(x)$ являются корни многочленов $x^2 + \xi_1 x + \eta_1$ и $x^2 + \xi_2 x + \eta_2$.

Для упрощения вычислений при $a \neq 0$ может быть сделана замена переменных $x = \dfrac{1}{y} + \sqrt{\dfrac{c}{a}}$ (в полях характеристики 2 квадратный корень единственен). Если $a = 0$, то преобразование не нужно. Теперь задача поиска корней исходного многочлена сводится к задаче поиска корней многочлена

$$g(y) = y^4 + E_1 y^2 + E_2 y + E_3,$$

где

$$E_1 = \frac{\sqrt{ac} + b}{d + \frac{bc}{a} + \frac{c^2}{a^2}}; \quad E_2 = \frac{a}{d + \frac{bc}{a} + \frac{c^2}{a^2}}; \quad E_3 = \frac{1}{d + \frac{bc}{a} + \frac{c^2}{a^2}}.$$

Величина $\theta_1 = r_0 + r_1 + r_2$, где r_i, $i \in [1,3]$, — резольвенты Лагранжа уравнения $x^3 + E_1^2 x + E_2^2 = 0$. Значение $r_0 = 0$, а $R_{1,2} = r_{1,2}^3$ есть корни уравнения

$$r^2 + E_2^2 r + E_1^6 = 0.$$

Тогда $r_i^3 = E_2^2 p_i$, где p_i, $i \in [1,2]$, могут быть найдены с помощью вышеописанной процедуры для решения уравнения $p^2 + p + p_0 = 0$, причем

$$p_0 = \frac{E_1^6}{E_2^4} = \frac{(\sqrt{ac} + b)^6}{a^4 (d + \frac{bc}{a} + \frac{c^2}{a^2})^2} = E_2^2 \left(\sqrt{\frac{c}{a}} + \frac{b}{a} \right)^6.$$

Так как значения $r_{1,2}$ находятся неоднозначно, необходимо применить следующую процедуру.

Шаг 1. Найти корень R_1 уравнения $r^2 + E_2^2 r + E_1^6 = 0$.

Шаг 2. Найти одно произвольное значение его кубического корня $r_1' = \sqrt[3]{R_1}$.

Шаг 3. В силу того, что $r_1 r_2 = E_1^2$, значение r_2 может быть вычислено как $r_2 = r_2' \dfrac{E_1^2}{r_1' r_2'} = \dfrac{E_1^2}{r_1'}$, $r_1 = r_1'$.

Можно заметить, что $\xi = \xi_1 = \xi_2 = \sqrt{\theta_1}$. Величины η_i, $i \in [1,2]$, могут быть найдены как корни из уравнения $\eta^2 + (\theta_1 + E_1)\eta + E_3 = 0$ (вычисления могут быть ускорены за счет того, что $E_1 = E_2 \left(\sqrt{\dfrac{c}{a}} + \dfrac{b}{a} \right)$). Корни преобразованного многочлена (если преобразование имело место) являются корнями многочленов $y^2 + \xi y + \eta_1$ и $y^2 + \xi y + \eta_2$.

3.1.6. Сравнение методов вычисления корней многочлена

Гибридный подход к нахождению корней многочленов заключается в последовательном уменьшении степени многочлена до четырех посредством специального разложения, описанного в параграфе 3.1.3, и применении аналитических методов для вычисления корней многочленов степени меньше четырех, описанного в параграфе 3.1.5.

Для оценки эффективности предложенного подхода было проведено моделирование на языке программирования C++. Умножение элементов в конечном поле $\mathrm{GF}(2^8)$ осуществлялось с использованием таблиц логарифмов и антилогарифмов. Время, затраченное для вычисления значений многочленов в элементах конечного поля $\alpha^0, \ldots, \alpha^{254}$, было усреднено более чем для 100 000 вычислений и показано в табл. 4.

Из результатов моделирования видно, что предложенное разложение многочленов по теореме 3.1 дает больший выигрыш по сравнению с гибридной процедурой Ченя для больших степеней (больших, чем 14). Однако для многочленов любой степени наиболее существенный выигрыш получается для гибридных процедур, основанных или на разложении (27) (для малых степеней многочлена), или на разложении (28) (для больших степеней многочлена). Другими словами, исключение некоторых сложных в вычислительном отношении действий применением предложенных разложений значительно сокращает общую сложность. Кроме того, как следует из формулы (29), комбинация с аналитическими алгоритмами для многочленов малой степени сокращает сложность еще больше. Более того, можно заметить, что в некоторых случаях увеличение степени многочлена не приводит к значительному возрастанию сложности вычислений. Это можно объяснить тем, что сложность вычисления многочленов для предложенных разложений зависит от числа многочленов в разложении, но не от их степени.

Таблица 4

Время вычисления корней многочленов в микросекундах

Степень многочлена	Процедура Ченя	Гибридная процедура Ченя	Разложение по теореме 3.1	Гибридный метод по теореме 3.1	Разложение (27)	Гибридный метод по (27)	Разложение (28)	Гибридный метод по (28)
5	14.6	2.6	13.4	3.0	10.0	2.3	—	—
6	17.2	4.7	14.8	5.2	10.1	3.6	—	—
7	19.6	6.6	14.9	6.6	10.8	4.9	—	—
8	22.2	8.5	15.1	7.9	10.8	5.8	11.4	6.7
9	24.9	10.0	15.4	8.9	—	—	15.5	9.5
10	27.3	11.7	16.9	10.7	—	—	16.4	11.0
11	29.8	13.4	18.0	12.0	—	—	16.4	11.6
12	32.3	15.0	18.1	12.8	—	—	16.4	12.2
13	34.8	16.5	18.2	13.5	—	—	16.9	13.5
14	37.3	18.0	18.2	14.3	—	—	17.4	14.2
15	39.8	19.6	20.1	16.0	—	—	17.4	14.8
16	42.3	21.1	21.3	17.4	—	—	17.9	15.8
17	44.9	22.7	21.3	18.1	—	—	17.9	16.2

3.2. Циклотомический алгоритм

Известно много алгоритмов быстрого преобразования Фурье (БПФ) над полями вещественных и комплексных чисел, но перенос этих алгоритмов в конечные поля не всегда возможен. Кроме того, алгоритм БПФ, построенный специально для данного конечного поля, может оказаться лучше алгоритма, перенесенного из другого поля [7].

Предложенный автором метод [39] состоит в разбиении исходного многочлена на сумму линеаризованных многочленов (30) и вычислении их значений в наборе базисных точек (31). Компоненты преобразования Фурье вычисляются как линейные комбинации этих значений с коэффициентами из простого поля (32).

Подход, основанный на представлении многочлена в виде суммы линеаризованных многочленов, впервые был предложен в работе [96] и обобщен в [71]. Далее будут рассмотрены основные понятия и определения, введено циклотомическое разложение многочленов и представлен алгоритм БПФ, основанный на этом разложении. Алгоритм описывается для полей характеристики 2, но может быть обобщен и на случай произвольных конечных полей.

3.2.1. Основные понятия и определения

Введем ДПФ над многочленами. За исходный многочлен примем многочлен $f(x)$ степени $n-1$, а за результирующий — многочлен $F(x) = \sum_{i=0}^{n-1} F_i x^i$ той же степени.

Определение 3.4. *Преобразованием Фурье многочлена* $f(x) = \sum_{i=0}^{n-1} f_i x^i$ *степени* $\deg f(x) = n-1$, $n \mid 2^m - 1$, *в поле* $\mathrm{GF}(2^m)$ *назы-*

вается набор значений

$$F_j = f(\alpha^j) = \sum_{i=0}^{n-1} f_i \alpha^{ij}, \quad j \in [0, n-1],$$

где α *— элемент порядка* n *в поле* $\mathrm{GF}(2^m)$.

Напомним, что понятие линеаризованных многочленов было введено в определении 3.1. Несложно показать, что для линеаризованных многочленов выполняется равенство $L(a+b) = L(a) + L(b)$. Следствием этого свойства является следующая теорема, являющаяся расширением леммы 3.1.

Теорема 3.2 [5, 11]. *Пусть* $x \in \mathrm{GF}(2^m)$ *и элементы* $\beta_0, \beta_1, \ldots, \beta_{m-1}$ *образуют базис поля.*

$$\textit{Если} \quad x = \sum_{i=0}^{m-1} x_i \beta_i, \quad x_i \in \mathrm{GF}(2), \quad \textit{и} \quad L(x) = \sum_j L_j x^{2^j},$$

$$\textit{то} \quad L(x) = \sum_{i=0}^{m-1} x_i L(\beta_i).$$

Рассмотрим набор циклотомических классов по модулю n над $\mathrm{GF}(2)$:

$$\{0\}, \{k_1, k_1 2, k_1 2^2, \ldots, k_1 2^{m_1-1}\}, \ldots, \{k_l, k_l 2, k_l 2^2, \ldots, k_l 2^{m_l-1}\},$$

где $k_i \equiv k_i 2^{m_i} \bmod n$.

Многочлен $f(x) = \sum_{i=0}^{n-1} f_i x^i, f_i \in \mathrm{GF}(2^m)$, может быть разложен как

$$f(x) = \sum_{i=0}^{l} L_i(x^{k_i}), \quad L_i(y) = \sum_{j=0}^{m_i-1} f_{k_i 2^j \bmod n} y^{2^j}, \tag{30}$$

где $y \in \mathrm{GF}(2^m)$ — независимая переменная.

Действительно, выражение (30) является способом группировки чисел $s \in [0, n-1]$ по циклотомическим классам:

$s \equiv k_i 2^j \bmod n$. Очевидно, что такое разложение существует всегда. Заметим, что при $k_i = 0$ свободный член f_0 можем записать как значение многочлена $L_0(y) = f_0 y$ при $y = x^0$.

Выражение (30) будем называть циклотомическим разложением многочлена $f(x)$.

Пример циклотомического разложения многочлена. Многочлен $f(x) = \sum_{i=0}^{6} f_i x^i, f_i \in \mathrm{GF}(2^3)$, представляется как

$$\begin{aligned} f(x) = &\quad L_0(x^0) + L_1(x) + L_2(x^3); \\ L_0(y) = &\quad f_0 y, \\ L_1(y) = &\quad f_1 y + f_2 y^2 + f_4 y^4, \\ L_2(y) = &\quad f_3 y + f_6 y^2 + f_5 y^4. \end{aligned}$$

3.2.2. Быстрое вычисление преобразования Фурье

В соответствии с разложением (30) запишем $f(\alpha^j) = \sum_{i=0}^{l} L_i(\alpha^{j\,k_i})$. Как известно [33], элемент α^{k_i} является корнем соответствующего минимального многочлена степени m_i и, следовательно, лежит в подполе $\mathrm{GF}(2^{m_i})$, $m_i \mid m$. Таким образом, все величины $(\alpha^{k_i})^j$ принадлежат полю $\mathrm{GF}(2^{m_i})$ и могут быть разложены в каком-либо базисе $(\beta_{i,0}, \ldots, \beta_{i,m_i-1})$ этого поля: $\alpha^{j\,k_i} = \sum_{s=0}^{m_i-1} a_{ijs}\beta_{i,s}, \quad a_{ijs} \in \mathrm{GF}(2)$. Тогда значения каждого из линеаризованных многочленов могут быть вычислены в базисных точках соответствующего подполя по формуле

$$L_i(\beta_{i,s}) = \sum_{p=0}^{m_i-1} \beta_{i,s}^{2^p} f_{k_i 2^p}, \quad i \in [0, l], \quad s \in [0, m_i - 1]. \qquad (31)$$

Базисы $(\beta_{i,0}, \ldots, \beta_{i,m_i-1})$ для каждого из линеаризованных многочленов $L_i(y)$ могут выбираться независимо.

В соответствии с теоремой 3.2 компоненты преобразования Фурье многочлена $f(x)$ являются линейными комбинациями этих значений:

$$F_j = f(\alpha^j) = \sum_{i=0}^{l} \sum_{s=0}^{m_i-1} a_{ijs} L_i(\beta_{i,s}) =$$

$$= \sum_{i=0}^{l} \sum_{s=0}^{m_i-1} a_{ijs} \left(\sum_{p=0}^{m_i-1} \beta_{i,s}^{2^p} f_{k_i 2^p} \right), \quad j \in [0, n-1]. \tag{32}$$

Последнее выражение может быть записано в матричной форме как $\mathbf{F} = AL\mathbf{f}$, где $\mathbf{F} = (F_0, F_1, \ldots, F_{n-1})^T$, $\mathbf{f} = (f_0, f_{k_1}, f_{k_1 2}, f_{k_1 2^2}, \ldots, f_{k_1 2^{m_1-1}}, \ldots, f_{k_l}, f_{k_l 2}, f_{k_l 2^2}, \ldots, f_{k_l 2^{m_l-1}})^T$ есть перестановка вектора коэффициентов исходного многочлена $f(x)$, соответствующая разложению (30); A — матрица, составленная из элементов $a_{ijs} \in \mathrm{GF}(2)$, а L — блочно-диагональная матрица, составленная из элементов $\beta_{i,s}^{2^p}$. Очевидно, что для линеаризованных многочленов одинаковой степени m_i, входящих в разложение (30), можно выбрать одинаковые базисы $(\beta_{i,s})$ в подполях $\mathrm{GF}(2^{m_i})$, вследствие чего матрица L будет содержать много одинаковых блоков.

Таким образом, задача вычисления БПФ разбивается на два этапа: умножение блочно-диагональной матрицы L на исходный вектор $\mathbf{f}$ и умножение двоичной матрицы A на полученный вектор $\mathbf{S} = L\mathbf{f}$:

$$\mathbf{F} = AL\mathbf{f}. \tag{33}$$

Рассмотрим более подробно первый этап преобразования Фурье — задачу вычисления произведения $\mathbf{S} = L\mathbf{f}$. Блочно-диагональная матрица

$$L = \begin{bmatrix} L_0 & 0 & \dots & 0 \\ 0 & L_1 & \dots & 0 \\ \dots & \dots & \ddots & \dots \\ 0 & 0 & \dots & L_l \end{bmatrix}$$

состоит из блоков

$$L_i = \begin{bmatrix} \beta_{i,0} & \beta_{i,0}^2 & \dots & \beta_{i,0}^{2^{m_i-1}} \\ \beta_{i,1} & \beta_{i,1}^2 & \dots & \beta_{i,1}^{2^{m_i-1}} \\ \dots & \dots & \dots & \dots \\ \beta_{i,m_i-1} & \beta_{i,m_i-1}^2 & \dots & \beta_{i,m_i-1}^{2^{m_i-1}} \end{bmatrix}.$$

Пример использования в качестве $(\beta_{i,0}, \dots, \beta_{i,m_i-1})$ стандартного базиса рассмотрен в работе [70]. В случае нормального базиса $\left(\gamma_i, \gamma_i^2, \dots, \gamma_i^{2^{m_i-1}}\right)$ матрица L состоит из блоков вида

$$L_i = \begin{bmatrix} \gamma_i^{2^0} & \gamma_i^2 & \dots & \gamma_i^{2^{m_i-1}} \\ \gamma_i^2 & \gamma_i^4 & \dots & \gamma_i^{2^0} \\ \dots & \dots & \dots & \dots \\ \gamma_i^{2^{m_i-1}} & \gamma_i^{2^0} & \dots & \gamma_i^{2^{m_i-2}} \end{bmatrix}.$$

В силу блочно-диагональной структуры матрицы L вычисление произведения $\mathbf{S} = L\mathbf{f}$ может быть представлено как $\mathbf{S} = (b_0, b_1, \dots, b_l)^T = L\,(a_0, a_1, \dots, a_l)^T$, где $\mathbf{b}_i = (b_{i,0}, b_{i,1}, \dots, b_{i,m_i-1})$ — подвекторы искомого вектора $\mathbf{S}$; $\mathbf{a}_i = (a_{i,0}, a_{i,1}, \dots, a_{i,m_i-1})$ — подвекторы исходного вектора $\mathbf{f}$.

Представим вычисление $\mathbf{b}_i^T = L_i \mathbf{a}_i^T$ как циклическую свертку:

$$\begin{gathered} b_i(x) = b_{i,0} + b_{i,m_i-1}x + \ldots + b_{i,1}x^{m_i-1} = \\ = (\gamma_i + \gamma_i^{2^{m_i-1}}x + \ldots + \gamma_i^2 x^{m_i-1}) \times \\ \times (a_{i,0} + a_{i,1}x + \ldots + a_{i,m_i-1}x^{m_i-1}) \bmod (x^{m_i} - 1). \end{gathered}$$

Для ее вычисления могут быть применены известные алгоритмы [7, 11, 3]. При этом использование свойства нормального базиса $\gamma_i + \gamma_i^2 + \ldots + \gamma_i^{2^{m_i-1}} = 1$ позволяет заметно сократить число операций при вычислении циклической свертки. Отметим, что

вычисление значений линеаризованных многочленов с помощью циклической свертки было описано в монографии [11].

Данный подход позволяет свести задачу умножения блочно-диагональной матрицы L на исходный вектор $\mathbf{f}$ над $\mathrm{GF}(2^m)$ к задаче вычисления $l+1$ циклических сверток малой длины m_i. Существующие алгоритмы вычисления циклических сверток $b_i(x) = \gamma_i(x)a_i(x) \bmod (x^{m_i}-1)$ могут быть записаны в матричном виде как

$$\mathbf{b}_i = \begin{bmatrix} b_{i,0} \\ b_{i,1} \\ \dots \\ b_{i,m_i-1} \end{bmatrix} = Q_i \left(D_i \begin{bmatrix} \gamma_i \\ \gamma_i^{2^{m_i-1}} \\ \dots \\ \gamma_i^2 \end{bmatrix} \cdot (P_i \mathbf{a}_i) \right),$$

где Q_i, D_i и P_i являются двоичными матрицами, а $\mathbf{x} \cdot \mathbf{y}$ обозначает покомпонентное произведение векторов. Очевидно, что вектор $\mathbf{C}_i = D_i \left(\gamma_i, \gamma_i^{2^{m_i-1}}, \dots, \gamma_i^2 \right)^T$ может быть вычислен заранее. Таким образом, выражение (33) может быть переписано как

$$\mathbf{F} = AQ\left(\mathbf{C} \cdot (P\mathbf{f})\right), \tag{34}$$

где Q — двоичная блочно-диагональная матрица объединенных последующих сложений для $l+1$ циклической свертки; $\mathbf{C}$ — объединенный вектор констант; P — двоичная блочно-диагональная матрица объединенных предварительных сложений.

Учитывая формулы (33) и (34), второй этап вычисления БПФ может рассматриваться как умножение двоичной матрицы AQ на вектор $\mathbf{C} \cdot (P\mathbf{f})$. Для вычисления произведения $(AQ)\left(\mathbf{C} \cdot (P\mathbf{f})\right)$ могут быть использованы модифицированный алгоритм “четырех русских” (В. Л. Арлазаров, Е. А. Диниц, М. А. Кронрод, И. А. Фараджев) для умножения булевых матриц со сложностью $O(n^2/\log n)$ сложений над элементами поля $\mathrm{GF}(2^m)$ [1, 2], метод Липницкого и Стройниковой [32] или эвристический алгоритм [95], сложность которого оценить не удалось. Однако для всех рассмотренных примеров сложность эвристического алгоритма меньше сложности модификации алгоритма “четырех русских”.

Приведенные преобразования имеют много общего с преобразованием из работы [20]. Основные отличия состоят в следующем.

1. Матрица L имеет регулярную структуру. Это позволяет свести задачу минимизации числа умножений к классической задаче вычисления циклической свертки.

2. Алгоритм содержит всего два умножения двоичных матриц на векторы, что может быть использовано для более глубокой оптимизации.

3. Используется более эффективный алгоритм оптимизации последовательности сложений.

Предложенный автором алгоритм БПФ эффективен при малых значениях длины преобразования (см. табл. 5), хотя известны асимптотически более эффективные алгоритмы БПФ для конечных полей со сложностью $O(n \log^2 n)$ операций в основном поле [98, 43].

3.2.3. Пример

Продолжим рассмотрение БПФ длины 7 над полем $\mathrm{GF}(2^3)$. Пусть α — корень примитивного многочлена $x^3 + x + 1$ над $\mathrm{GF}(2)$. В качестве базиса поля $\mathrm{GF}(2^3)$ выберем нормальный базис $(\gamma, \gamma^2, \gamma^4)$, где $\gamma = \alpha^3$. Разложим многочлен $f(x)$, как в примере из параграфа 3.2.1, и представим компоненты преобразования Фурье в виде сумм

$$
\begin{aligned}
f(\alpha^0) = \quad & L_0(\alpha^0) + L_1(\alpha^0) + L_2(\alpha^0) = \\
= \quad & L_0(1) + L_1(\gamma) + L_1(\gamma^2) + L_1(\gamma^4) + \\
+ \quad & L_2(\gamma) + L_2(\gamma^2) + L_2(\gamma^4); \\
f(\alpha^1) = \quad & L_0(\alpha^0) + L_1(\alpha) + L_2(\alpha^3) = \\
= \quad & L_0(1) + L_1(\gamma^2) + L_1(\gamma^4) + L_2(\gamma); \\
f(\alpha^2) = \quad & L_0(\alpha^0) + L_1(\alpha^2) + L_2(\alpha^6) = \\
& L_0(1) + L_1(\gamma) + L_1(\gamma^4) + L_2(\gamma^2); \\
f(\alpha^3) = \quad & L_0(\alpha^0) + L_1(\alpha^3) + L_2(\alpha^2) = \\
& L_0(1) + L_1(\gamma) + L_2(\gamma) + L_2(\gamma^4); \\
f(\alpha^4) = \quad & L_0(\alpha^0) + L_1(\alpha^4) + L_2(\alpha^5) = \\
& L_0(1) + L_1(\gamma) + L_1(\gamma^2) + L_2(\gamma^4); \\
f(\alpha^5) = \quad & L_0(\alpha^0) + L_1(\alpha^5) + L_2(\alpha) = \\
& L_0(1) + L_1(\gamma^4) + L_2(\gamma^2) + L_2(\gamma^4); \\
f(\alpha^6) = \quad & L_0(\alpha^0) + L_1(\alpha^6) + L_2(\alpha^4) = \\
& L_0(1) + L_1(\gamma^2) + L_2(\gamma) + L_2(\gamma^2).
\end{aligned}
$$

Эта система может быть записана в матричной форме как

$$
\mathbf{F} = \begin{bmatrix} F_0 \\ F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \\ F_6 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} L_0(1) \\ L_1(\gamma) \\ L_1(\gamma^2) \\ L_1(\gamma^4) \\ L_2(\gamma) \\ L_2(\gamma^2) \\ L_2(\gamma^4) \end{bmatrix} = A\mathbf{S}.
$$

Тогда задачу БПФ можно переписать в виде

$$\mathbf{F} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \gamma^1 & \gamma^2 & \gamma^4 & 0 & 0 & 0 \\ 0 & \gamma^2 & \gamma^4 & \gamma^1 & 0 & 0 & 0 \\ 0 & \gamma^4 & \gamma^1 & \gamma^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \gamma^1 & \gamma^2 & \gamma^4 \\ 0 & 0 & 0 & 0 & \gamma^2 & \gamma^4 & \gamma^1 \\ 0 & 0 & 0 & 0 & \gamma^4 & \gamma^1 & \gamma^2 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_4 \\ f_3 \\ f_6 \\ f_5 \end{bmatrix} =$$

$$= AL\mathbf{f}.$$

Первый этап алгоритма БПФ состоит в вычислении двух циклических сверток

$$\begin{bmatrix} b_{i,0} \\ b_{i,1} \\ b_{i,2} \end{bmatrix} = \begin{bmatrix} \gamma^1 & \gamma^2 & \gamma^4 \\ \gamma^2 & \gamma^4 & \gamma^1 \\ \gamma^4 & \gamma^1 & \gamma^2 \end{bmatrix} \begin{bmatrix} a_{i,0} \\ a_{i,1} \\ a_{i,2} \end{bmatrix}, \quad i = 1, 2,$$

где

$$\mathbf{S} = \begin{bmatrix} L_0(1) \\ L_1(\gamma) \\ L_1(\gamma^2) \\ L_1(\gamma^4) \\ L_2(\gamma) \\ L_2(\gamma^2) \\ L_2(\gamma^4) \end{bmatrix} = \begin{bmatrix} b_{0,0} \\ b_{1,0} \\ b_{1,1} \\ b_{1,2} \\ b_{2,0} \\ b_{2,1} \\ b_{2,2} \end{bmatrix}, \qquad \mathbf{f} = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_4 \\ f_3 \\ f_6 \\ f_5 \end{bmatrix} = \begin{bmatrix} a_{0,0} \\ a_{1,0} \\ a_{1,1} \\ a_{1,2} \\ a_{2,0} \\ a_{2,1} \\ a_{2,2} \end{bmatrix}.$$

Используя алгоритм вычисления трехточечной циклической свертки $b_i(x) = b_{i,0} + b_{i,2}x + b_{i,1}x^2 = (\gamma + \gamma^4 x + \gamma^2 x^2)(a_{i,0} + a_{i,1}x + a_{i,2}x^2) \bmod (x^3 - 1)$, представленный в работе [7], получим

$$\mathbf{b}_i = \begin{bmatrix} b_{i,0} \\ b_{i,1} \\ b_{i,2} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix} \times$$

$$\times \left[\left(\begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} \gamma \\ \gamma^4 \\ \gamma^2 \end{bmatrix} \right) \cdot \left(\begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_{i,0} \\ a_{i,1} \\ a_{i,2} \end{bmatrix} \right) \right] =$$

$$= Q_i \left(\mathbf{C}_i \cdot (P_i a_i) \right), \quad i = 1, 2.$$

С учетом $\gamma + \gamma^2 + \gamma^4 = 1$ видно, что алгоритм требует трех умножений, четырех предварительных и пяти последующих сложений.

Теперь можно записать формулу (34) для рассматриваемого примера в матричной форме:

$$\mathbf{F} = \begin{bmatrix} F_0 \\ F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \\ F_6 \end{bmatrix} =$$

$$= \left(\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \right) \times$$

$$\times \left(\begin{bmatrix} 1 \\ 1 \\ \gamma^2 + \gamma^4 \\ \gamma + \gamma^4 \\ \gamma + \gamma^2 \\ 1 \\ \gamma^2 + \gamma^4 \\ \gamma + \gamma^4 \\ \gamma + \gamma^2 \end{bmatrix} \cdot \left(\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_4 \\ f_3 \\ f_6 \\ f_5 \end{bmatrix} \right) \right) =$$

$$= (AQ)(\mathbf{C} \cdot (P\mathbf{f})).$$

Второй этап БПФ состоит в умножении двоичной матрицы AQ на вектор $\mathbf{C} \cdot (P\mathbf{f})$:

$$\mathbf{F} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} (\mathbf{C} \cdot (P\mathbf{f})).$$

Этот этап может быть выполнен за 17 сложений.

Таким образом, БПФ длины 7 сводится к следующей последовательности действий:

выполнение предварительных сложений $P \times \mathbf{f}$:

$$\begin{array}{ll} V_1 = f_2 + f_4, & V_8 = f_6 + f_5, \\ V_2 = f_1 + f_2, & V_9 = f_3 + f_6, \\ V_3 = f_1 + f_4, & V_{10} = f_3 + f_5, \\ V_4 = f_1 + V_1, & V_{11} = f_3 + V_8; \end{array}$$

выполнение умножений на константы $\mathbf{C} \cdot (P\mathbf{f})$:

$$\begin{array}{ll} V_5 = V_1\,\alpha, & V_{12} = V_8\,\alpha, \\ V_6 = V_2\,\alpha^2, & V_{13} = V_9\,\alpha^2, \\ V_7 = V_3\,\alpha^4, & V_{14} = V_{10}\,\alpha^4; \end{array}$$

умножение матрицы AQ на вектор $\mathbf{C} \cdot (P\mathbf{f})$:

$$\begin{array}{ll} T_{10} = V_{12} + V_{14}, & F_2 = T_8 + T_{11}, \\ T_{11} = f_0 + V_{11}, & F_3 = T_7 + T_{14}, \\ T_{14} = f_0 + V_4, & T_{12} = F_2 + T_{10}, \\ T_{15} = V_5 + V_6, & T_{13} = F_3 + T_{15}, \\ T_{16} = V_6 + V_{13}, & F_4 = T_7 + T_{12}, \\ F_0 = V_4 + T_{11}, & F_5 = T_{10} + T_{13}, \\ T_9 = V_{12} + T_{16}, & F_6 = F_5 + T_7, \\ T_7 = V_7 + T_9, & F_1 = F_4 + T_8. \\ T_8 = V_5 + T_9, & \end{array}$$

Общая сложность алгоритма составляет $2 \times 3 = 6$ умножений и $2 \times 4 + 17 = 25$ сложений, что на одно сложение меньше, чем для алгоритма, представленного в работе [20].

3.2.4. Сравнение сложности алгоритмов БПФ

Вычисление $a + b$ (или $a \times b$) будем считать сложением (умножением) только тогда, когда оба слагаемых (сомножителя) принадлежат основному полю [8], т. е. операции в простом подполе не учитываются [7]. В табл. 5 приведена сложность БПФ длины $n = 2^m - 1$ над полями $\mathrm{GF}(2^m)$ в числе умножений N_{mul} и сложений N_{add}. Метод Горнера описан, например, в работе [11], а модификация алгоритма Герцеля для конечных полей рассмотрена в монографии [7].

3.3. Рекуррентный метод

В разделе предложен метод вычисления ДПФ над конечным полем, основанный на разложении матрицы преобразования в произведение двоичной блочно-циркулянтной и диагональной блочно-циркулянтной матриц. Известно, что асимптотически лучший алгоритм вычисления ДПФ описан Вангом, Жу [98] и Афанасьевым [43], однако при малых значениях длины преобразования циклотомический алгоритм [39] является самым эффективным. Идея сведения вычисления ДПФ длины n к вычислению циклической свертки длины $n - 1$ была предложена Рейдером [90] для простых длин. Алгоритм Герцеля, модифицированный для конечных полей [7], в работе [39] интерпретирован как сведение ДПФ длины $n = 2^m - 1$ к вычислению циклических сверток длины m. Л. А. Бассалыго [4] первым обратил внимание на регулярную структуру матрицы преобразования, что дает возможность применять блоковые циклические свертки для вычисления ДПФ. Изложенный в разделе метод предложен автором в работе [41]. Этот метод имеет не большую сложность вычислений, чем циклотомический алгоритм [39], но более простое описание.

Таблица 5

Сложность некоторых алгоритмов БПФ

Длина	Метод Горнера		Алгоритм Герцеля		Алгоритмы из [11] и [3]		Метод Захаровой [20]		Предложенный метод	
n	N_{mul}	N_{add}	N_{mul}	N_{add}	N_{mul}	N_{add}	N_{mul}	N_{add}	N_{mul}	N_{add}
7	36	42	12	42	9	35	6	26	6	25
15	196	210	38	210	20	70	16	100	16	77
31	900	930	120	930	108	645	60	388	54	315
63	3844	3906	282	3906	158	623	97	952	97	805
127	15876	16002	756	16002	594	5770	468	3737	216	2780
255	64516	64770	1718	64770	1225	4715	646	35503	586	7919
511	260100	260610	4044	260610	4374	—	—	—	1014	26643

3.3.1. Основные понятия и определения

Введем ДПФ над векторами, эквивалентное определению 3.4. За исходный вектор примем вектор $\mathbf{f}$ длины n, а за результирующий — вектор $\mathbf{F}$ той же длины.

Определение 3.5. *Дискретным преобразованием Фурье длины* $n \mid 2^m - 1$ *вектора* $\mathbf{f} = (f_i), \quad i \in [0, n-1]$, *в поле* $\mathrm{GF}(2^m)$ *называется вектор* $\mathbf{F} = (F_j)$,

$$F_j = \sum_{i=0}^{n-1} f_i \alpha^{ij}, \quad j \in [0, n-1],$$

где α *— элемент порядка* n *в поле* $\mathrm{GF}(2^m)$.

Запишем ДПФ в матричной форме

$$\mathbf{F} = W\mathbf{f},$$

где $W = (\alpha^{ij}), \quad i, j \in [0, n-1]$, — матрица Вандермонда.

Рассмотрим набор циклотомических классов по модулю n над $\mathrm{GF}(2)$

$$(c_1, c_1 2, c_1 2^2, \ldots, c_1 2^{m_1 - 1}), \ldots, (c_l, c_l 2, c_l 2^2, \ldots, c_l 2^{m_l - 1}),$$

где $c_i \equiv c_i 2^{m_i} \pmod{n}, \quad i \in [1, l]$; l — число циклотомических классов по модулю n над $\mathrm{GF}(2)$.

Пусть

$$\Gamma_i = \left(\gamma_i^{2^0}, \gamma_i^{2^1}, \ldots, \gamma_i^{2^{m_i - 1}}\right) \quad —$$

нормальный базис для подполя $\mathrm{GF}(2^{m_i}) \subset \mathrm{GF}(2^m)$.

Циркулянтной матрицей, или циркулянтом, называется матрица, любая строка которой получается из предыдущей строки путем циклического сдвига на одну позицию влево (вправо). Циркулянтные матрицы могут быть не только квадратными, но и прямоугольными. Если элементами циркулянта являются матрицы, то циркулянт называется блоковым циркулянтом.

Назовем произведение

$$\begin{bmatrix} B_0 & B_1 & \dots & B_{t-1} \\ B_1 & B_2 & \dots & B_0 \\ \dots & \dots & \dots & \dots \\ B_{t-1} & B_0 & \dots & B_{t-2} \end{bmatrix} \begin{bmatrix} \mathbf{b}_0 \\ \mathbf{b}_1 \\ \dots \\ \mathbf{b}_{t-1} \end{bmatrix}$$

t-точечной блоковой циклической сверткой, где блоки $B_i, \quad i \in [0, t-1]$, являются подматрицами, блоки $\mathbf{b}_i, \quad i \in [0, t-1]$, — подвекторами, и все размеры блоков определены корректно.

Ганкелевой матрицей называется матрица вида

$$\begin{bmatrix} a_0 & a_1 & a_2 & \dots & a_{t-1} \\ a_1 & a_2 & a_3 & \dots & a_t \\ a_2 & a_3 & a_4 & \dots & a_{t+1} \\ \dots & \dots & \dots & \dots & \dots \\ a_{t-1} & a_t & a_{t+1} & \dots & a_{2t-2} \end{bmatrix}.$$

Ганкелевы матрицы, так же как и циркулянты, могут быть прямоугольными и могут образовывать блоковые конструкции.

Матрица вида

$$\left(a_i{}^{b_j}\right), \quad i \in [0, t-1], \quad j \in [0, u-1],$$

называется обобщенной матрицей Вандермонда [12]. Если для такой матрицы выполняется свойство $b_j = 2^j, \quad j \in [0, u-1]$, то назовем ее квадратичной обобщенной матрицей Вандермонда.

Через L_i обозначим $m_i \times m_i$ циркулянт, первая строка которого представляет собой нормальный базис, и назовем его базовым циркулянтом

$$L_i = \begin{bmatrix} \gamma_i^{2^0} & \gamma_i^{2^1} & \dots & \gamma_i^{2^{m_i-1}} \\ \gamma_i^{2^1} & \gamma_i^{2^2} & \dots & \gamma_i^{2^0} \\ \dots & \dots & \dots & \dots \\ \gamma_i^{2^{m_i-1}} & \gamma_i^{2^0} & \dots & \gamma_i^{2^{m_i-2}} \end{bmatrix}.$$

Из построения видно, что базовый циркулянт L_i также является квадратичной обобщенной матрицей Вандермонда. Матрица L_i всегда невырожденная.

Через $GCD(a,b)$ обозначим наибольший общий делитель чисел a и b, а через $\varphi(c)$ — функцию Эйлера от натурального числа c.

Для построения эквивалентного преобразования введем множество индексов по $\bmod n$

$$Z = (Z_i), \quad i \in [0, n-1].$$

Определим перестановочную матрицу $\Pi = (\Pi_{ij}), \;\; i,j \in [0, n-1]$,

$$\Pi_{ij} = \begin{cases} 1, & \text{если } j = Z_i, \quad i \in [0, n-1] \\ 0, & \text{иначе} \end{cases}.$$

Заменим ДПФ на эквивалентное преобразование, отличающееся порядком компонент в результирующем и исходном векторах:

$$\begin{aligned} \mathbf{F}_c &= W_c \mathbf{f}_c, \\ \mathbf{F}_c &= \Pi \mathbf{F}, \\ \mathbf{f}_c &= \Pi \mathbf{f}, \end{aligned}$$

где $W_c = \Pi W \Pi^T = (\alpha^{Z_i Z_j}), \;\; i,j \in [0, n-1]$.

3.3.2. Алгоритм Рейдера

Алгоритм Рейдера [90] состоит в преобразовании вычисления ДПФ для простой длины n к вычислению циклической свертки длины $n-1$. Это всегда возможно, так как простое поле $\mathrm{GF}(n)$ содержит примитивный элемент и $\mathrm{GF}(2^m)\backslash\mathbf{0}$ образует мультипликативную циклическую группу.

Поясним работу алгоритма Рейдера на примере.

Пример. Рассмотрим ДПФ длины $n = 7$ над полем $\mathrm{GF}(2^3)$. Поле задается элементом α — корнем примитивного многочлена x^3+x+1 над $\mathrm{GF}(2)$. За ядро ДПФ возьмем примитивный элемент α. Запишем преобразование Фурье $\mathbf{F} = W\mathbf{f}$ в матричной форме

$$\begin{bmatrix} F_0 \\ F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \\ F_6 \end{bmatrix} = \begin{bmatrix} \alpha^0 & \alpha^0 & \alpha^0 & \alpha^0 & \alpha^0 & \alpha^0 & \alpha^0 \\ \alpha^0 & \alpha^1 & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \\ \alpha^0 & \alpha^2 & \alpha^4 & \alpha^6 & \alpha^1 & \alpha^3 & \alpha^5 \\ \alpha^0 & \alpha^3 & \alpha^6 & \alpha^2 & \alpha^5 & \alpha^1 & \alpha^4 \\ \alpha^0 & \alpha^4 & \alpha^1 & \alpha^5 & \alpha^2 & \alpha^6 & \alpha^3 \\ \alpha^0 & \alpha^5 & \alpha^3 & \alpha^1 & \alpha^6 & \alpha^4 & \alpha^2 \\ \alpha^0 & \alpha^6 & \alpha^5 & \alpha^4 & \alpha^3 & \alpha^2 & \alpha^1 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \end{bmatrix}.$$

Для построения эквивалентного преобразования выберем множество индексов по $\bmod n$ как объединение циклической группы и нулевого элемента.

Пусть $g = 3$ — первообразный корень по $\bmod 7$. Тогда он порождает мультипликативную циклическую группу

$$(g^0, g^1, g^2, g^3, g^4, g^5) = (1, 3, 2, 6, 4, 5).$$

Составляем множество индексов

$$Z = (1, 3, 2, 6, 4, 5) \cup (0) =$$

$$= \begin{array}{c} \begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & \end{array} \\ \begin{pmatrix} 1 & 3 & 2 & 6 & 4 & 5 & 0 \end{pmatrix} \end{array}$$

и перестановочную матрицу

$$\Pi = \begin{array}{c} \begin{array}{ccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}.$$

Запишем эквивалентное преобразование Фурье $\mathbf{F}_c = W_c\mathbf{f}_c$ в матричной форме

$$\left[\begin{array}{c} F_1 \\ F_3 \\ F_2 \\ F_6 \\ F_4 \\ F_5 \\ \hline F_0 \end{array}\right] = \left[\begin{array}{cccccc|c} \alpha^1 & \alpha^3 & \alpha^2 & \alpha^6 & \alpha^4 & \alpha^5 & \alpha^0 \\ \alpha^3 & \alpha^2 & \alpha^6 & \alpha^4 & \alpha^5 & \alpha^1 & \alpha^0 \\ \alpha^2 & \alpha^6 & \alpha^4 & \alpha^5 & \alpha^1 & \alpha^3 & \alpha^0 \\ \alpha^6 & \alpha^4 & \alpha^5 & \alpha^1 & \alpha^3 & \alpha^2 & \alpha^0 \\ \alpha^4 & \alpha^5 & \alpha^1 & \alpha^3 & \alpha^2 & \alpha^6 & \alpha^0 \\ \alpha^5 & \alpha^1 & \alpha^3 & \alpha^2 & \alpha^6 & \alpha^4 & \alpha^0 \\ \hline \alpha^0 & \alpha^0 & \alpha^0 & \alpha^0 & \alpha^0 & \alpha^0 & \alpha^0 \end{array}\right] \left[\begin{array}{c} f_1 \\ f_3 \\ f_2 \\ f_6 \\ f_4 \\ f_5 \\ \hline f_0 \end{array}\right].$$

Из построения матрицы эквивалентного преобразования W_c для выбранного множества индексов Z видно, что матрица содержит $(n-1) \times (n-1)$ циркулянтную подматрицу. Умножение подвектора исходного вектора на циркулянтную матрицу и есть $(n-1)$-точечная циклическая свертка.

3.3.3. Свойства матрицы эквивалентного преобразования

Для построения эквивалентного преобразования выберем множество индексов по $\bmod n$ как объединение циклотомических классов:

$$Z = (Z_i) = \begin{array}{ll} (c_1, c_1 2, c_1 2^2, \ldots, c_1 2^{m_1-1}, \ldots, & \\ c_l, c_l 2, c_l 2^2, \ldots, c_l 2^{m_l-1}), & i \in [0, n-1]. \end{array}$$

Из построения матрицы эквивалентного преобразования W_c для выбранного множества индексов Z видно, что матрица состоит из $l \times l$ прямоугольных блоков, которые являются циркулянтными матрицами. Кроме того, каждый блок является квадратичной обобщенной матрицей Вандермонда. Напомним, что базовым циркулянтом размерности m_i называется циркулянт, первая строка которого содержит нормальный базис подполя $GF(2^{m_i})$.

Теорема 3.3. *Матрица эквивалентного преобразования W_c факторизуется в произведение двоичной и блочно-диагональной матриц.*

Доказательство. Составим множество индексов Z как объединение циклотомических классов:

$$Z = (c_1, c_1 2, c_1 2^2, \ldots, c_1 2^{m_1-1}, \ldots, c_l, c_l 2, c_l 2^2, \ldots, c_l 2^{m_l-1}).$$

Каждый циклотомический класс определяется образующим элементом c_i и состоит из m_i, $m_i | m$, $i \in [1, l]$, элементов.

Матрица W_c состоит из подматриц

$$W_c = (W_{ij}), \quad i, j \in [1, l],$$

где подматрица

$$W_{ij} = \left((\alpha^{c_i c_j})^{2^{p+q}} \right), \quad p \in [0, m_i - 1], \quad q \in [0, m_j - 1], \qquad (35)$$

имеет размеры $m_i \times m_j$.

Докажем вспомогательное утверждение.

Лемма 3.2. *Все элементы матрицы W_{ij} принадлежат подполю* $\mathrm{GF}(2^{GCD(m_i, m_j)}) \subset \mathrm{GF}(2^m)$.

Доказательство. Покажем, что элемент $\alpha^{c_i c_j} \in \mathrm{GF}(2^{m_j}) \subset \mathrm{GF}(2^m)$. Из определения циклотомического класса, содержащего образующий элемент c_j, следует, что $c_j \equiv c_j 2^{m_j} \pmod{n}$. Тогда $(\alpha^{c_i c_j})^{2^{m_j}} = \alpha^{c_i c_j}$ и $(\alpha^{c_i c_j})^{2^{m_j}-1} = 1$. Следовательно, порядок элемента $\alpha^{c_i c_j}$ делит число $2^{m_j} - 1$:

$$\mathrm{ord}(\alpha^{c_i c_j}) \mid 2^{m_j} - 1,$$

а сам элемент принадлежит подполю $\alpha^{c_i c_j} \in \mathrm{GF}(2^{m_j})$.

Аналогично можно показать, что $\alpha^{c_i c_j} \in \mathrm{GF}(2^{m_i})$.

Из доказанного и (35) следует, что все элементы матрицы W_{ij} принадлежат подполю $\mathrm{GF}(2^{GCD(m_i, m_j)})$. Ч.т.д.

Пусть P_j есть $m_j \times m_j$ матрица циклического сдвига

$$P_j = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \ldots & \ldots & \ldots & \ddots & \ldots \\ 0 & 0 & \cdots & 0 & 1 \\ 1 & 0 & \cdots & 0 & 0 \end{bmatrix}.$$

Заметим, что $(P_j)^0$ — единичная матрица.

Выделим двоичный сомножитель A_c из матрицы $W_c = A_c L_c$. Для этого представим каждую подматрицу W_{ij} в виде произведения

$$W_{ij} = A_{ij} L_j,$$

где A_{ij} — двоичная $m_i \times m_j$ матрица; $L_j = \left(\gamma_j^{2^{p+q}}\right)$, $p, q \in [0, m_j - 1]$, есть $m_j \times m_j$ базовый циркулянт.

По лемме 3.2 элемент $\alpha^{c_i c_j}$ лежит в подполе $\mathrm{GF}(2^{m_j})$ и может быть разложен по базису Γ_j

$$\alpha^{c_i c_j} = \sum_{s=0}^{m_j - 1} g_{ijs} \gamma_j^{2^s}, \quad g_{ijs} \in \mathrm{GF}(2).$$

Тогда для любого элемента матрицы W_{ij}

$$(\alpha^{c_i c_j})^{2^{p+q}} = \left(\sum_{s=0}^{m_j - 1} g_{ijs} \gamma_j^{2^s} \right)^{2^{p+q}} =$$

$$= \sum_{s=0}^{m_j - 1} (g_{ijs})^{2^{p+q}} (\gamma_j^{2^s})^{2^{p+q}} = \sum_{s=0}^{m_j - 1} g_{ijs} \gamma_j^{2^{s+p+q}}.$$

Запишем циклический сдвиг на s позиций для базового циркулянта:

$$P_j^s L_j = \left(\gamma_j^{2^{p+q+s}}\right), \quad p, q \in [0, m_j - 1].$$

В зависимости от соотношения m_i и m_j возможны три случая представления матрицы W_{ij}.

1. $m_i = m_j$.

$$W_{ij} = \left(\sum_{s=0}^{m_j - 1} g_{ijs} \gamma_j^{2^{s+p+q}} \right)_{p,q \in [0, m_j - 1]} =$$

$$= \sum_{s=0}^{m_j-1} g_{ijs} P_j^s L_j = \left(\sum_{s=0}^{m_j-1} g_{ijs} P_j^s \right) L_j = A_{ij} L_j,$$

где

$$A_{ij} = \sum_{s=0}^{m_j-1} g_{ijs} P_j^s \quad -$$

двоичный циркулянт.

2. $m_i < m_j$.
Дополним матрицу (35) до квадратной матрицы

$$W'_{ij} = \left((\alpha^{c_i c_j})^{2^{p+q}} \right), \quad p, q \in [0, m_j - 1].$$

Как и в случае 1, из матрицы W'_{ij} выделим двоичный сомножитель $W'_{ij} = A'_{ij} L_j$. Тогда $W_{ij} = A_{ij} L_j$, где $A_{ij} = A'_{ij}[m_i, m_j]$ — подматрица матрицы A'_{ij}, состоящая из ее первых m_i строк и m_j столбцов:

$$A_{ij} = \left(\sum_{s=0}^{m_j-1} g_{ijs} P_j^s \right) [m_i, m_j].$$

3. $m_i > m_j$.
Обозначим через $\lceil x \rceil$ наименьшее целое число, большее или равное x. Пусть $r = \left\lceil \dfrac{m_i}{m_j} \right\rceil$. В случае необходимости дополним матрицу W_{ij} до rm_j строк в соответствии с правилом построения матрицы (35):

$$\widetilde{W}_{ij} = \left((\alpha^{c_i c_j})^{2^{p+q}} \right), \quad p \in [0, r-1], q \in [0, m_j - 1],$$

т. е. матрица

$$\widetilde{W}_{ij} = \begin{bmatrix} W'_{ij} \\ W'_{ij} \\ \cdots \\ W'_{ij} \\ W'_{ij} \end{bmatrix}$$

является конкатенацией $m_j \times m_j$ квадратных подматриц W'_{ij}. Как и в случае 1, каждая подматрица W'_{ij} имеет представление $W'_{ij} = A'_{ij}L_j$ и, учитывая особенность нижней подматрицы W'_{ij} матрицы $\widetilde{W}_{ij}$, как в случае 2, имеем

$$W_{ij} = \begin{bmatrix} A'_{ij} \\ A'_{ij} \\ \cdots \\ A'_{ij} \\ A'_{ij}[m_i - (r-1)m_j, m_j] \end{bmatrix} L_j = A_{ij}L_j.$$

Итак, матрица $A_c = (A_{ij}), \quad i, j \in [1, l]$, состоит из подматриц, каждая из которых может быть представлена как сумма циклических сдвигов и является двоичным циркулянтом:

$$W_c = \begin{bmatrix} A_{11} & \cdots & A_{1l} \\ \cdots & \cdots & \cdots \\ A_{l1} & \cdots & A_{ll} \end{bmatrix} \begin{bmatrix} L_1 & 0 & \ldots & 0 \\ 0 & L_2 & \ldots & 0 \\ & & \ddots & \\ \ldots & \ldots & & \ldots \\ 0 & 0 & \ldots & L_l \end{bmatrix} = A_c L_c.$$

Ч.т.д.

Из теоремы 3.3 следует, что двоичная матрица A_c в разложении $W_c = A_c L_c$ полностью состоит из циркулянтов. Структура матрицы A_c определяется упорядочением чисел из полной системы вычетов по $\bmod n$, составляющих множество индексов Z. Методика построения такого упорядочения чисел описывается ниже.

3.3.4. Упорядочение чисел из полной системы вычетов

Для упорядочения чисел из полной системы вычетов по $\bmod n$ представим множество $[0, n-1]$ как конкатенацию циклотомических классов, все образующие элементы которых составляют объединение циклических групп.

Все группы, рассматриваемые в разделе, являются коммутативными конечными группами. Все операции, если модуль не указан в явном виде, выполняются по $\bmod n$. Множество называется упорядоченным, если его элементы пронумерованы последовательными натуральными числами. Неупорядоченные множества заключены в фигурные скобки, упорядоченные — в круглые, множество последовательных целых чисел — в квадратные.

Введем новую операцию — произведение упорядоченных множеств. Произведение упорядоченных множеств вычисляется по следующему правилу:

$$(a_1, a_2, \ldots, a_i) \times (b_1, b_2, \ldots, b_j) =$$
$$= (a_1b_1, a_2b_1, \ldots, a_ib_1; a_1b_2, a_2b_2, \ldots, a_ib_2; \ldots; a_1b_j, a_2b_j, \ldots, a_ib_j).$$

Ниже приведены примеры операций (произведение, конкатенация, умножение на константу) над упорядоченными множествами:

$$(a_1, a_2, a_3) \times (b_1, b_2) = (a_1b_1, a_2b_1, a_3b_1, a_1b_2, a_2b_2, a_3b_2),$$

$$(a_1, a_2, a_3) + (b_1, b_2) = (a_1, a_2, a_3, b_1, b_2),$$
$$i(a_1, a_2, a_3) = (ia_1, ia_2, ia_3).$$

Для нечетного k введем циклическую группу степеней числа 2 по $\bmod k$

$$B(\bmod k) = (2^0, 2^1, 2^2, \ldots, 2^{\delta-1}),$$

где

$$\delta = \arg\min_i (2^i \equiv 1 \pmod k, \quad i > 0) \quad —$$

порядок 2 по $\bmod k$. Также будем использовать обозначение

$$B_i = (2^0, 2^1, 2^2, \ldots, 2^{i-1}).$$

Введем мультипликативную по $\bmod \dfrac{n}{d}$ циклическую группу

$$G\left(i, \bmod \frac{n}{d}\right) = (1, i, i^2, \ldots)$$

с образующим элементом i, где d — делитель числа n. Для случая $d = 1$ будем использовать упрощенное обозначение $G(i) = G(i, \bmod n)$.

Пусть

$$G(i)[j] = (1, i, i^2, \ldots, i^{j-1}) \quad —$$

упорядоченное подмножество, состоящее из первых j элементов группы $G(i)$.

Приведенная система вычетов по $\bmod k$ есть

$$Q(k) = \{i \mid GCD(i, k) = 1,\ i \in [0, k-1]\},$$

причем $Q(k)$ является конечной группой с операцией умножения по $\bmod k$.

Для любого n существует разложение полной системы вычетов по $\bmod n$ в объединение непересекающихся подмножеств

$$[0, n-1] = \bigcup_{d|n} dQ\left(\frac{n}{d}\right),$$

где $Q\left(\frac{n}{d}\right)$ — приведенная система вычетов по $\bmod \frac{n}{d}$; d пробегает все делители числа n.

Для нечетного k группа $B(\bmod k)$ является подгруппой группы $Q(k)$ и, следовательно, группа $Q(k)$ разлагается на непересекающиеся смежные классы по подгруппе $B(\bmod k)$.

Основная теорема о конечных коммутативных группах гласит, что любая конечная коммутативная группа разлагается в прямое произведение циклических групп [10]. Для фиксированных нечетных значений n и d будем искать разложение

$$Q\left(\frac{n}{d}\right) \cong B\left(\bmod \frac{n}{d}\right) \times G\left(i, \bmod \frac{n}{d}\right),$$

где $G\left(i, \bmod \frac{n}{d}\right)$ — некоторая циклическая группа, а через $\cong$ обозначено соответствие между неупорядоченным и упорядоченным множествами, состоящими из одних и тех же элементов.

Заметим, что такого разложения может и не существовать, так как одна из циклических групп, а именно $B\left(\bmod \frac{n}{d}\right)$, была

зафиксирована. В этом случае ослабим требование к разложению и будем искать вместо группы $G\left(i, \bmod \frac{n}{d}\right)$ подмножество, состоящее из первых элементов циклической группы.

Искомое упорядочение чисел построим в два этапа. Вначале разобьем все числа $i \in [0, n-1]$ на непересекающиеся множества в соответствии со значением $GCD(n, i)$. Затем представим все числа, соответствующие фиксированному значению $GCD(n, i)$, как прямое произведение циклических групп (или их подмножеств). Соответствие между полной системой вычетов по $\bmod n$ и объединением упорядоченных прямых произведений циклических групп запишем как

$$[0, n-1] = \bigcup_{j=1}^{\tau} d_j Q\left(\frac{n}{d_j}\right) \cong$$

$$\cong \bigcup_{j=1}^{\tau} e_j \left(B\left(\bmod \frac{n}{d_j}\right) \times G\left(i_j, \bmod \frac{n}{d_j}\right)\right),$$

где $e_j, \quad j \in [1, \tau]$, — некоторые константы, возможно отличные от делителей d_j числа n, а τ — количество делителей числа n. Множество $\{e_i\}$, $i \in [1, \tau]$, построим так, чтобы число различных произведений $e_i e_j \,(\bmod n)$ для всех пар $(i, j), \quad i, j \in [1, \tau]$, было минимально.

Рассмотрим пример упорядоченного представления чисел по $\bmod 7$.

Пример. Число $n = 7$ имеет делители $\{1, 7\}$.

1. Числа $1, 2, 3, 4, 5, 6$ имеют с n наибольший общий делитель, равный $d_1 = 1$.

   Представление этих чисел в виде прямого произведения циклических групп есть

   $$B_3 \times G(6) = (1, 2, 4) \times (1, 6) = (1, 2, 4, 6, 5, 3).$$

2. Число 0 имеет с n наибольший общий делитель, равный $d_2 = 7$.

Таким образом:

$$B_3 \times G(6) + (0) = (1, 2, 4, 6, 5, 3, 0).$$

Приведем еще пример упорядоченного представления чисел по $\bmod 15$.

Пример. Число $n = 15$ имеет делители $\{1, 3, 5, 15\}$.

1. Числа 1, 2, 4, 7, 8, 11, 13, 14 имеют с n наибольший общий делитель, равный $d_1 = 1$.

   Представление этих чисел в виде прямого произведения циклических групп есть

   $$B_4 \times G(11) = (1, 2, 4, 8) \times (1, 11) = (1, 2, 4, 8, 11, 7, 14, 13).$$

2. Числа 3, 6, 9, 12 имеют с n наибольший общий делитель, равный $d_2 = 3$.

   Представление этих чисел в виде сдвига циклотомического класса есть

   $$3Q(5) = 3(1, 2, 4, 3) = (3, 6, 12, 9) \cong 6B_4 =$$

   $$= 6(1, 2, 4, 8) = (6, 12, 9, 3).$$

   Константа $e_2 = 6$ выбрана так, чтобы выполнялось сравнение $e_2^2 \equiv e_2 \pmod n$.

3. Числа 5 и 10 имеют с n наибольший общий делитель, равный $d_3 = 5$.

   Представление этих чисел в виде сдвига циклотомического класса по $\bmod 3$ есть

   $$5Q(3) = 5(1, 2) = (5, 10) \cong 10B_2 = 10(1, 2) = (10, 5).$$

   Константа $e_3 = 10$ выбрана так, чтобы выполнялось сравнение $e_3^2 \equiv e_3 \pmod n$.

4. Число 0 имеет с n наибольший общий делитель, равный $d_4 = 15$.

Таким образом:

$$B_4 \times G(11) + 6B_4 + 10B_2 + 0 =$$

$$= (1,2,4,8) \times (1,11) + 6(1,2,4,8) + 10(1,2) + (0) =$$

$$= (1,2,4,8,11,7,14,13,6,12,9,3,10,5,0).$$

Результаты упорядочения чисел для модулей вида $n = 2^m - 1$ приведены в табл. 6. Для всех рассмотренных значений n, кроме $n = 255$, существуют, построены аналитически и приведены в таблице разложения соответствующих приведенных систем вычетов в прямые произведения циклических групп. Для $n = 15, 63, 511$ множество констант $\{e_i\}$ выбрано, исходя из принципа минимизации числа различных произведений $e_i e_j \pmod n$. Для случая $n = 255$ искомого разложения не существует, а в таблице приведено более слабое разложение в прямое произведение группы B_i и подмножество циклической группы.

Описанное упорядочение чисел будем использовать в качестве множества индексов Z для построения матрицы W_c эквивалентного преобразования.

3.3.5. Быстрое вычисление ДПФ

Эквивалентное преобразование Фурье длины n в поле $\mathrm{GF}(2^m)$

$$\mathbf{F}_c = W_c \mathbf{f}_c = A_c L_c \mathbf{f}_c$$

состоит из двух этапов. На первом этапе вычисляют l циклических сверток, что при $n = 2^m - 1$ требует не более $m2^m \approx n \log n$ сложений и умножений в поле $\mathrm{GF}(2^m)$. На втором этапе вычисляют произведение двоичной матрицы A_c на вектор $L_c \mathbf{f}_c$.

Таблица 6

Упорядочение чисел из полной системы вычетов по $\mathrm{mod}\, n$

n	Делители n	Множество индексов Z
7	$\{1, 7\}$	$B_3 \times G(6) + 0$
15	$\{1, 3, 5, 15\}$	$B_4 \times G(11) + 6B_4 + 10B_2 + 0$
31	$\{1, 31\}$	$B_5 \times G(6) + 0$
63	$\{1, 3, 7, 9, 21, 63\}$	$B_6 \times G(13) + 15(B_6 \times G(13, \mathrm{mod}\, 21)) + 7B_6 +$ $36(B_3 \times G(13, \mathrm{mod}\, 7)) + 42B_2 + 0$
127	$\{1, 127\}$	$B_7 \times G(24) + 0$
255	$\{1, 3, 5, 15, 17, 51, 85, 255\}$	$B_8 \times G(13) \times G(7)[4] + 3(B_8 \times G(13) \times G(7)[2]) +$ $5(B_8 \times G(7)[4]) + 15(B_8 \times G(7)[2]) +$ $17(B_4 \times G(13)[2]) + 51B_4 + 85B_2 + 0$
511	$\{1, 7, 73, 511\}$	$B_9 \times G(10) \times G(510) + 147(B_9 \times G(10, \mathrm{mod}\, 73)) +$ $73(B_3 \times G(510, \mathrm{mod}\, 7)) + 0$

Пусть τ — количество делителей числа n. При рассмотрении структуры матрицы A_c видно, что она содержит τ^2 блоковых циркулянтов $A_{Bij}, \quad i,j \in [1,\tau]$ (возможно, неполных или тривиальных) для всех значений n, кроме $n = 255$. Таким образом, подматрицы $A_{ij}, \quad i,j \in [1,l]$ матрицы A_c образуют двоичные блоковые циркулянты $A_{Bij}, \quad i,j \in [1,\tau]$, а умножение матрицы A_c на вектор $L_c\mathbf{f}_c$ может быть сведено к вычислению τ^2 блоковых циклических сверток. В случае длины $n = 255$ подматрицы A_{ij} составляют блоковые циркулянты, образующие в свою очередь блоковые ганкелевы матрицы, умножения на которые также могут быть выполнены с помощью быстрых алгоритмов.

Оценим сверху число сложений в поле $\mathrm{GF}(2^m)$ при выполнении второго этапа преобразования. Для этого введем следующие функции сложности вычисления циклических сверток. Пусть $CON_{\mathrm{mul}}(i)$ — число умножений при вычислении циклической свертки длины i, а $CON_{\mathrm{add}}(i)$ — число сложений. Через $CIR_{\mathrm{add}}(i)$ обозначим число сложений, требуемое для умножения двоичного циркулянта размера $i \times i$ на вектор длины i из основного поля, а через $\mathrm{add}(i)$ — число сложений элементов из основного поля при сложении двух векторов длины i. Далее определим размеры подматриц $A_{Bij}, \quad i,j \in [1,\tau]$, состоящих из двоичных блоков $A_{ij}, \quad i,j \in [1,l]$. Пусть $\{d_1, d_2, \ldots, d_\tau\}$ — множество делителей числа n. Тогда размер блока δ_i, соответствующий i-му делителю, равен порядку 2 по $\mathrm{mod}\,\dfrac{n}{d_i}$:

$$\delta_i = \arg\min_j \left(2^j \equiv 1 \pmod{\frac{n}{d_i}}, \quad j > 0\right).$$

Число блоков t_i, соответствующее i-му делителю, равно

$$t_i = \frac{\varphi\left(\dfrac{n}{d_i}\right)}{\delta_i}.$$

То есть каждая подматрица A_{Bij} состоит из $t_i \times t_j$ блоков A_{ij}, имеющих размеры $\delta_i \times \delta_j$.

При построении оценки сложности будем дополнять неполные циркулянты до полных. Альтернативно можно строить более точные оценки, если определить функции сложности для неполных циркулянтов.

Оценим сложность вычисления блоковой циклической свертки при умножении подматрицы A_{Bij} на вектор как

$$BCON(i,j) = CON_{\text{mul}}(\max(t_i,t_j))\, CIR_{\text{add}}(\max(\delta_i,\delta_j)) +$$

$$+CON_{\text{add}}(\max(t_i,t_j))\ \text{add}(\max(\delta_i,\delta_j)).$$

Тогда справедлива оценка числа сложений при выполнении второго этапа преобразования

$$N_{\text{add}} = \sum_{\substack{(i,j)\\ i,j\in[1,\tau]}} BCON(i,j) \ + \ (\tau-1)n.$$

3.3.6. Примеры вычисления ДПФ

Пример. Рассмотрим ДПФ длины $n = 7$ над полем $\mathrm{GF}(2^3)$. Поле задается элементом α — корнем примитивного многочлена x^3+x+1 над $\mathrm{GF}(2)$. За ядро ДПФ возьмем примитивный элемент α. Множество индексов Z выберем из соответствующего примера параграфа 3.3.4:

$$Z = (1,2,4,6,5,3,0).$$

Запишем эквивалентное преобразование Фурье $\mathbf{F}_c = W_c\mathbf{f}_c$ в матричной форме

$$\begin{bmatrix} F_1 \\ F_2 \\ F_4 \\ \hline F_6 \\ F_5 \\ F_3 \\ \hline F_0 \end{bmatrix} = \left[\begin{array}{ccc|ccc|c} \alpha^1 & \alpha^2 & \alpha^4 & \alpha^6 & \alpha^5 & \alpha^3 & \alpha^0 \\ \alpha^2 & \alpha^4 & \alpha^1 & \alpha^5 & \alpha^3 & \alpha^6 & \alpha^0 \\ \alpha^4 & \alpha^1 & \alpha^2 & \alpha^3 & \alpha^6 & \alpha^5 & \alpha^0 \\ \hline \alpha^6 & \alpha^5 & \alpha^3 & \alpha^1 & \alpha^2 & \alpha^4 & \alpha^0 \\ \alpha^5 & \alpha^3 & \alpha^6 & \alpha^2 & \alpha^4 & \alpha^1 & \alpha^0 \\ \alpha^3 & \alpha^6 & \alpha^5 & \alpha^4 & \alpha^1 & \alpha^2 & \alpha^0 \\ \hline \alpha^0 & \alpha^0 & \alpha^0 & \alpha^0 & \alpha^0 & \alpha^0 & \alpha^0 \end{array}\right] \begin{bmatrix} f_1 \\ f_2 \\ f_4 \\ \hline f_6 \\ f_5 \\ f_3 \\ \hline f_0 \end{bmatrix}.$$

В поле $\mathrm{GF}(2^3)$ нормальным базисом является $\left(\gamma, \gamma^2, \gamma^4\right) = \left(\alpha^6, \alpha^5, \alpha^3\right)$, а в его простом подполе $\mathrm{GF}(2)$: $\left(\gamma^1 + \gamma^2 + \gamma^4\right) = \left(\alpha^0\right)$.

Обозначим базовые циркулянты через

$$L_1 = L_2 = \begin{bmatrix} \gamma^1 & \gamma^2 & \gamma^4 \\ \gamma^2 & \gamma^4 & \gamma^1 \\ \gamma^4 & \gamma^1 & \gamma^2 \end{bmatrix} \quad \text{и} \quad L_3 = \left[\gamma^1 + \gamma^2 + \gamma^4\right] = \left[1\right].$$

Учитывая равенства $\alpha^1 = \gamma^1 + \gamma^2$, $\alpha^6 = \gamma^1$, $\alpha^0 = \gamma^1 + \gamma^2 + \gamma^4$, факторизуем матрицу W_c и имеем

$$W_c = A_c L_c = \left[\begin{array}{c|c|c} (I+P)L_1 & L_2 & \begin{matrix}1\\1\\1\end{matrix} \\ \hline L_1 & (I+P)L_2 & \begin{matrix}1\\1\\1\end{matrix} \\ \hline 1\ 1\ 1 & 1\ 1\ 1 & 1 \end{array}\right] =$$

$$= \left[\begin{array}{c|c|c} I+P & I & \begin{matrix}1\\1\\1\end{matrix} \\ \hline I & I+P & \begin{matrix}1\\1\\1\end{matrix} \\ \hline 1\ 1\ 1 & 1\ 1\ 1 & 1 \end{array}\right] \left[\begin{array}{c|c|c} L_1 & & \\ \hline & L_2 & \\ \hline & & 1 \end{array}\right] =$$

$$= \left[\begin{array}{ccc|ccc|c} 1&1&0&1&0&0&1 \\ 0&1&1&0&1&0&1 \\ 1&0&1&0&0&1&1 \\ \hline 1&0&0&1&1&0&1 \\ 0&1&0&0&1&1&1 \\ 0&0&1&1&0&1&1 \\ \hline 1&1&1&1&1&1&1 \end{array}\right] \left[\begin{array}{ccc|ccc|c} \gamma^1&\gamma^2&\gamma^4 & & & & \\ \gamma^2&\gamma^4&\gamma^1 & & & & \\ \gamma^4&\gamma^1&\gamma^2 & & & & \\ \hline & & & \gamma^1&\gamma^2&\gamma^4 & \\ & & & \gamma^2&\gamma^4&\gamma^1 & \\ & & & \gamma^4&\gamma^1&\gamma^2 & \\ \hline & & & & & & 1 \end{array}\right],$$

где $I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ — единичная матрица; $P = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$ — матрица циклического сдвига.

Матрица A_c состоит из девяти двоичных циркулянтов. Также можно заметить, что подматрица матрицы A_c

$$\left[\begin{array}{ccc|ccc} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ \hline 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{array}\right]$$

является блоковым циркулянтом.

Пример. Рассмотрим ДПФ длины $n = 15$ над полем $\mathrm{GF}(2^4)$. Поле задается элементом α — корнем примитивного многочлена x^4+x+1 над $\mathrm{GF}(2)$. За ядро ДПФ возьмем примитивный элемент α. Множество индексов Z выберем из соответствующего примера параграфа 3.3.4:

$$Z = (1, 2, 4, 8, 11, 7, 14, 13, 6, 12, 9, 3, 10, 5, 0).$$

Запишем эквивалентное преобразование Фурье $\mathbf{F}_c = W_c\mathbf{f}_c$ в матричной форме

$$
\left[\begin{array}{c}
F_1 \\ F_2 \\ F_4 \\ F_8 \\ \hline
F_{11} \\ F_7 \\ F_{14} \\ F_{13} \\ \hline
F_6 \\ F_{12} \\ F_9 \\ F_3 \\ \hline
F_{10} \\ F_5 \\ \hline
F_0
\end{array}\right]
=
\left[\begin{array}{cccc|cccc|cccc|cccc}
\alpha^{1} & \alpha^{2} & \alpha^{4} & \alpha^{8} & \alpha^{11} & \alpha^{7} & \alpha^{14} & \alpha^{13} & \alpha^{6} & \alpha^{12} & \alpha^{9} & \alpha^{3} & \alpha^{10} & \alpha^{5} & \alpha^{0} \\
\alpha^{2} & \alpha^{4} & \alpha^{8} & \alpha^{1} & \alpha^{7} & \alpha^{14} & \alpha^{13} & \alpha^{11} & \alpha^{12} & \alpha^{9} & \alpha^{3} & \alpha^{6} & \alpha^{5} & \alpha^{10} & \alpha^{0} \\
\alpha^{4} & \alpha^{8} & \alpha^{1} & \alpha^{2} & \alpha^{14} & \alpha^{13} & \alpha^{11} & \alpha^{7} & \alpha^{9} & \alpha^{3} & \alpha^{6} & \alpha^{12} & \alpha^{10} & \alpha^{5} & \alpha^{0} \\
\alpha^{8} & \alpha^{1} & \alpha^{2} & \alpha^{4} & \alpha^{13} & \alpha^{11} & \alpha^{7} & \alpha^{14} & \alpha^{3} & \alpha^{6} & \alpha^{12} & \alpha^{9} & \alpha^{5} & \alpha^{10} & \alpha^{0} \\
\hline
\alpha^{11} & \alpha^{7} & \alpha^{14} & \alpha^{13} & \alpha^{1} & \alpha^{2} & \alpha^{4} & \alpha^{8} & \alpha^{6} & \alpha^{12} & \alpha^{9} & \alpha^{3} & \alpha^{5} & \alpha^{10} & \alpha^{0} \\
\alpha^{7} & \alpha^{14} & \alpha^{13} & \alpha^{11} & \alpha^{2} & \alpha^{4} & \alpha^{8} & \alpha^{1} & \alpha^{12} & \alpha^{9} & \alpha^{3} & \alpha^{6} & \alpha^{10} & \alpha^{5} & \alpha^{0} \\
\alpha^{14} & \alpha^{13} & \alpha^{11} & \alpha^{7} & \alpha^{4} & \alpha^{8} & \alpha^{1} & \alpha^{2} & \alpha^{9} & \alpha^{3} & \alpha^{6} & \alpha^{12} & \alpha^{5} & \alpha^{10} & \alpha^{0} \\
\alpha^{13} & \alpha^{11} & \alpha^{7} & \alpha^{14} & \alpha^{8} & \alpha^{1} & \alpha^{2} & \alpha^{4} & \alpha^{3} & \alpha^{6} & \alpha^{12} & \alpha^{9} & \alpha^{10} & \alpha^{5} & \alpha^{0} \\
\hline
\alpha^{6} & \alpha^{12} & \alpha^{9} & \alpha^{3} & \alpha^{6} & \alpha^{12} & \alpha^{9} & \alpha^{3} & \alpha^{6} & \alpha^{12} & \alpha^{9} & \alpha^{3} & \alpha^{0} & \alpha^{0} & \alpha^{0} \\
\alpha^{12} & \alpha^{9} & \alpha^{3} & \alpha^{6} & \alpha^{12} & \alpha^{9} & \alpha^{3} & \alpha^{6} & \alpha^{12} & \alpha^{9} & \alpha^{3} & \alpha^{6} & \alpha^{0} & \alpha^{0} & \alpha^{0} \\
\alpha^{9} & \alpha^{3} & \alpha^{6} & \alpha^{12} & \alpha^{9} & \alpha^{3} & \alpha^{6} & \alpha^{12} & \alpha^{9} & \alpha^{3} & \alpha^{6} & \alpha^{12} & \alpha^{0} & \alpha^{0} & \alpha^{0} \\
\alpha^{3} & \alpha^{6} & \alpha^{12} & \alpha^{9} & \alpha^{3} & \alpha^{6} & \alpha^{12} & \alpha^{9} & \alpha^{3} & \alpha^{6} & \alpha^{12} & \alpha^{9} & \alpha^{0} & \alpha^{0} & \alpha^{0} \\
\hline
\alpha^{10} & \alpha^{5} & \alpha^{10} & \alpha^{5} & \alpha^{5} & \alpha^{10} & \alpha^{5} & \alpha^{10} & \alpha^{0} & \alpha^{0} & \alpha^{0} & \alpha^{0} & \alpha^{10} & \alpha^{5} & \alpha^{0} \\
\alpha^{5} & \alpha^{10} & \alpha^{5} & \alpha^{10} & \alpha^{10} & \alpha^{5} & \alpha^{10} & \alpha^{5} & \alpha^{0} & \alpha^{0} & \alpha^{0} & \alpha^{0} & \alpha^{5} & \alpha^{10} & \alpha^{0} \\
\hline
\alpha^{0} & \alpha^{0} & \alpha^{0} & \alpha^{0} & \alpha^{0} & \alpha^{0} & \alpha^{0} & \alpha^{0} & \alpha^{0} & \alpha^{0} & \alpha^{0} & \alpha^{0} & \alpha^{0} & \alpha^{0} & \alpha^{0}
\end{array}\right]
\left[\begin{array}{c}
f_1 \\ f_2 \\ f_4 \\ f_8 \\ \hline
f_{11} \\ f_7 \\ f_{14} \\ f_{13} \\ \hline
f_6 \\ f_{12} \\ f_9 \\ f_3 \\ \hline
f_{10} \\ f_5 \\ \hline
f_0
\end{array}\right].
$$

В поле $\mathrm{GF}(2^4)$ нормальным базисом является $\left(\gamma^1, \gamma^2, \gamma^4, \gamma^8\right) = \left(\alpha^6, \alpha^{12}, \alpha^9, \alpha^3\right)$, а в его подполях: $\left(\gamma^1 + \gamma^4, \gamma^2 + \gamma^8\right) = \left(\alpha^5, \alpha^{10}\right)$ — для $\mathrm{GF}(2^2) \subset \mathrm{GF}(2^4)$; $\left(\gamma^1 + \gamma^2 + \gamma^4 + \gamma^8\right) = \left(\alpha^0\right)$ — для $\mathrm{GF}(2) \subset \mathrm{GF}(2^4)$.

Заметим, что образующий элемент нормального базиса $\gamma = \alpha^6$ не является примитивным элементом поля.

Обозначим базовые циркулянты через

$$L_1 = L_2 = L_3 = \begin{bmatrix} \gamma^1 & \gamma^2 & \gamma^4 & \gamma^8 \\ \gamma^2 & \gamma^4 & \gamma^8 & \gamma^1 \\ \gamma^4 & \gamma^8 & \gamma^1 & \gamma^2 \\ \gamma^8 & \gamma^1 & \gamma^2 & \gamma^4 \end{bmatrix},$$

$$L_4 = \begin{bmatrix} \gamma^1 + \gamma^4 & \gamma^2 + \gamma^8 \\ \gamma^2 + \gamma^8 & \gamma^1 + \gamma^4 \end{bmatrix} \text{ и } L_5 = \left[\gamma^1 + \gamma^2 + \gamma^4 + \gamma^8\right] = \left[1\right].$$

Введем обозначение $D[a, b]$ для подматрицы матрицы D, состоящей из ее первых a строк и b столбцов. Учитывая равенства $\alpha^1 = \gamma^4 + \gamma^8$, $\alpha^{11} = \gamma^1 + \gamma^4 + \gamma^8$, $\alpha^6 = \gamma^1$, $\alpha^{10} = \gamma^2 + \gamma^8$, $\alpha^0 = \gamma^1 + \gamma^2 + \gamma^4 + \gamma^8$, факторизуем матрицу W_c и имеем

$$W_c = \left[\begin{array}{r|r|r|r|r} P^2+P^3 & P^0+P^2+P^3 & P^0 & (P^1+P^3)[4,2] & J[4,1] \\ \hline P^0+P^2+P^3 & P^2+P^3 & P^0 & (P^0+P^2)[4,2] & J[4,1] \\ \hline P^0 & P^0 & P^0 & J[4,2] & J[4,1] \\ \hline (P^1+P^3)[2,4] & (P^0+P^2)[2,4] & J[2,4] & (P^1+P^3)[2,2] & J[2,1] \\ \hline J[1,4] & J[1,4] & J[1,4] & J[1,2] & J[1,1] \end{array}\right] \times$$

$$\times \left[\begin{array}{c|c|c|c|c} L_1 & & & & \\ \hline & L_2 & & & \\ \hline & & L_3 & & \\ \hline & & & L_4 & \\ \hline & & & & L_5 \end{array}\right] = A_c L_c,$$

где $P = P_1 = P_2 = P_3 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$ — матрица циклического сдвига; $J = P^0 + P^1 + P^2 + P^3$ — матрица из всех единиц. Для упрощения записи будем использовать подматрицы матрицы P вместо матрицы P_4.

Выделим двоичный сомножитель A_c из матрицы W_c

$$W_c = A_c L_c = \left[\begin{array}{c|c|c|c|c} A_{11} & A_{12} & A_{13} & A_{14} & A_{15} \\ \hline A_{21} & A_{22} & A_{23} & A_{24} & A_{25} \\ \hline A_{31} & A_{32} & A_{33} & A_{34} & A_{35} \\ \hline A_{41} & A_{42} & A_{43} & A_{44} & A_{45} \\ \hline A_{51} & A_{52} & A_{53} & A_{54} & A_{55} \end{array}\right] \left[\begin{array}{c|c|c|c|c} L_1 & & & & \\ \hline & L_2 & & & \\ \hline & & L_3 & & \\ \hline & & & L_4 & \\ \hline & & & & L_5 \end{array}\right] =$$

$$
\begin{array}{|cccc|cccc|cccc|cccc|}
\hline
0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\
0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\
\hline
1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\
1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\
0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\
\hline
1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\
\hline
0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\
1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\
1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\
\hline
\end{array}
$$

$$
\begin{array}{|cccc|}
\hline
\gamma^1 & \gamma^2 & \gamma^4 & \gamma^8 \\
\gamma^2 & \gamma^4 & \gamma^8 & \gamma^1 \\
\gamma^4 & \gamma^8 & \gamma^1 & \gamma^2 \\
\gamma^8 & \gamma^1 & \gamma^2 & \gamma^4 \\
\hline
\end{array}
\times
\begin{array}{|cccc|}
\hline
\gamma^1 & \gamma^2 & \gamma^4 & \gamma^8 \\
\gamma^2 & \gamma^4 & \gamma^8 & \gamma^1 \\
\gamma^4 & \gamma^8 & \gamma^1 & \gamma^2 \\
\gamma^8 & \gamma^1 & \gamma^2 & \gamma^4 \\
\hline
\end{array}
\cdot
\begin{array}{|cccc|}
\hline
\gamma^1 & \gamma^2 & \gamma^4 & \gamma^8 \\
\gamma^2 & \gamma^4 & \gamma^8 & \gamma^1 \\
\gamma^4 & \gamma^8 & \gamma^1 & \gamma^2 \\
\gamma^8 & \gamma^1 & \gamma^2 & \gamma^4 \\
\hline
\end{array}
\begin{array}{|cc|}
\hline
\gamma^1 + \gamma^4 & \gamma^2 + \gamma^8 \\
\gamma^2 + \gamma^8 & \gamma^1 + \gamma^4 \\
\hline
\end{array}
\begin{array}{|c|}
\hline
1 \\
\hline
\end{array}
$$

Матрица A_c состоит из двоичных циркулянтов: $A_c = (A_{ij})$, $i, j \in [1, 5]$. Можно заметить, что подматрицы

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \quad \begin{bmatrix} A_{31} & A_{32} \end{bmatrix}, \quad \begin{bmatrix} A_{51} & A_{52} \end{bmatrix}, \quad \begin{bmatrix} A_{13} \\ A_{23} \end{bmatrix}, \quad \begin{bmatrix} A_{15} \\ A_{25} \end{bmatrix}$$

являются блоковыми циркулянтами, подматрицы (A_{33}), (A_{43}), (A_{53}), (A_{34}), (A_{44}), (A_{54}), (A_{35}), (A_{45}), (A_{55}) можно рассматривать как тривиальные блоковые циркулянты, а подматрицы

$$\begin{bmatrix} A_{41} & A_{42} \end{bmatrix} = \left[\begin{array}{cccc|cccc} 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \end{array}\right],$$

$$\begin{bmatrix} A_{14} \\ A_{24} \end{bmatrix} = \left[\begin{array}{c|c} 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ \hline 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{array}\right]$$

образуют более сложную блочно-циркулянтную структуру, в которой блоками циркулянтов являются подматрицы матриц A_{41}, A_{42} или A_{14}, A_{24}.

Эквивалентное преобразование Фурье осуществляют в два этапа. На первом этапе вычисляют три четырехточечные циклические свертки и одну двухточечную с общей сложностью 16 умножений и 42 сложения над полем $\mathrm{GF}(2^4)$. На втором этапе — умножении матрицы A_c на вектор $L_c\mathbf{f}_c$ — вычисляют двухточечную блоковую циклическую свертку и некоторые тривиальные свертки с общей сложностью 49 сложений.

Незначительное увеличение числа сложений эквивалентного преобразования Фурье по сравнению с циклотомическим алгоритмом [39], выполняемого по формуле (34), объясняется следующим образом. Выполнение первого этапа циклотомического алгоритма

становится проще на число последующих сложений при вычислении циклических сверток, поскольку на втором этапе циклотомического алгоритма проводится умножение матрицы AQ на вектор с элементами из основного поля. Однако, в отличие от предложенного рекуррентного алгоритма, структура матрицы AQ в циклотомическом алгоритме описанию не поддается, что затрудняет реализацию вычислений.

3.4. Вычисление синдрома

В разделе рассматривается задача вычисления синдрома от принятого вектора для классических кодов Рида – Соломона с помощью циклотомического и рекуррентного методов. Пусть $S(x) = \sum_{i=0}^{2t-1} S_i x^i$ — синдромный многочлен, где t — корректирующая способность кода. Известно [7], что коэффициенты синдромного многочлена могут быть вычислены как значения принятого многочлена в корнях порождающего многочлена кода Рида – Соломона:

$$S_i = f(\alpha^i), \quad i \in [0, 2t-1],$$

где $f(x) = \sum_{i=0}^{n-1} f_i x^i$ — многочлен, соответствующий принятому вектору.

3.4.1. Вычисление неполного ДПФ с помощью циклотомического алгоритма

Коэффициенты синдромного многочлена S_i могут быть непосредственно вычислены как неполное ДПФ от принятого многочлена $f(x)$. Однако прямое применение циклотомического алгоритма (33) требует вычисления всех циклических сверток. Это

приводит к тому, что число умножений для неполного ДПФ останется таким же, как и для полного циклотомического алгоритма вычисления ДПФ.

Так как обе матрицы A и L в (33) обратимые, то обратное ДПФ может быть записано как

$$\mathbf{f} = L^{-1}A^{-1}\mathbf{F}.$$

Пусть каждый блок в блочно-диагональной матрице L образуется базисом $\beta_i = (\beta_{i,0}, \ldots, \beta_{i,m_i-1})$. Можно показать, что каждый блок в блочно-диагональной матрице L^{-1} также образуется базисом $\widetilde{\beta}_i$, дуальным (двойственным) к β_i [80], т. е. блоки в матрице L^{-1} также будут циркулянтными матрицами. Учитывая связь прямого и обратного преобразований Фурье [11, (3.6)], можно полагать, что последняя формула вычисления обратного ДПФ также приводит к циклотомическому алгоритму вычисления ДПФ.

Если необходимо вычислить только некоторые значения компонент вектора $\mathbf{f}$, то достаточно выполнить умножения на некоторые блоки матрицы L^{-1}. Это значительно сокращает полное число умножений. Более того, можно обратить внимание на то, что нужно вычислять не полное произведение A^{-1} на $\mathbf{F}$, а только значения для элементов, соответствующих необходимым блокам L^{-1}. Это эквивалентно усечению матрицы A^{-1}, что приводит к сокращению числа сложений.

Пример. Рассмотрим пример вычисления обратного преобразования Фурье длины 7 над конечным полем $\mathrm{GF}(2^3)$. Для этого вначале напомним пример построения БПФ из параграфа 3.2.3. Пусть α — корень примитивного многочлена $x^3 + x + 1$ над $\mathrm{GF}(2)$. В качестве базиса поля $\mathrm{GF}(2^3)$ выберем нормальный базис $\left(\gamma, \gamma^2, \gamma^4\right)$, где $\gamma = \alpha^3$. Тогда алгоритм БПФ может быть записан следующим образом [39]:

$$\mathbf{F} = \begin{bmatrix} F_0 \\ F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \\ F_6 \end{bmatrix} =$$

$$= \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \gamma^1 & \gamma^2 & \gamma^4 & 0 & 0 & 0 \\ 0 & \gamma^2 & \gamma^4 & \gamma^1 & 0 & 0 & 0 \\ 0 & \gamma^4 & \gamma^1 & \gamma^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \gamma^1 & \gamma^2 & \gamma^4 \\ 0 & 0 & 0 & 0 & \gamma^2 & \gamma^4 & \gamma^1 \\ 0 & 0 & 0 & 0 & \gamma^4 & \gamma^1 & \gamma^2 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_4 \\ f_3 \\ f_6 \\ f_5 \end{bmatrix} =$$

$$= AL\mathbf{f}.$$

Обратное преобразование Фурье вектора $\mathbf{F}$ может быть получено как

$$L^{-1}A^{-1}\mathbf{F} = \mathbf{f} = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_4 \\ f_3 \\ f_6 \\ f_5 \end{bmatrix} =$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \gamma^1 & \gamma^2 & \gamma^4 & 0 & 0 & 0 \\ 0 & \gamma^2 & \gamma^4 & \gamma^1 & 0 & 0 & 0 \\ 0 & \gamma^4 & \gamma^1 & \gamma^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \gamma^1 & \gamma^2 & \gamma^4 \\ 0 & 0 & 0 & 0 & \gamma^2 & \gamma^4 & \gamma^1 \\ 0 & 0 & 0 & 0 & \gamma^4 & \gamma^1 & \gamma^2 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} F_0 \\ F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \\ F_6 \end{bmatrix}.$$

Учитывая связь прямого и обратного преобразований Фурье [11, (3.6)], получим следующую эквивалентную запись алгоритма БПФ:

$$\widetilde{\mathbf{F}} = \begin{bmatrix} F_0 \\ F_6 \\ F_5 \\ F_3 \\ F_4 \\ F_1 \\ F_2 \end{bmatrix} = L^{-1}A^{-1} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \end{bmatrix} = L^{-1}A^{-1}\widetilde{\mathbf{f}},$$

где через $\widetilde{\mathbf{F}}$ и $\widetilde{\mathbf{f}}$ обозначены перестановки векторов коэффициентов $\mathbf{F}$ и $\mathbf{f}$.

Напомним алгоритм вычисления трехточечной циклической свертки $b_i(x) = b_{i,0} + b_{i,2}x + b_{i,1}x^2 = (\gamma + \gamma^4 x + \gamma^2 x^2)(a_{i,0} + a_{i,1}x + a_{i,2}x^2) \bmod (x^3 - 1)$ из параграфа 3.2.3, представленный также в работе [7]:

$$\mathbf{b}_i = \begin{bmatrix} b_{i,0} \\ b_{i,1} \\ b_{i,2} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix} \times$$

$$\times \left[\left(\begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} \gamma \\ \gamma^4 \\ \gamma^2 \end{bmatrix} \right) \cdot \left(\begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_{i,0} \\ a_{i,1} \\ a_{i,2} \end{bmatrix} \right) \right] =$$

$$= Q_i \left(\mathbf{C}_i \cdot (P_i \mathbf{a}_i) \right), \quad i = 1, 2,$$

где Q_i — двоичная матрица последующих сложений для циклической свертки; $\mathbf{C}_i$ — вектор констант; P_i — двоичная матрица предварительных сложений.

С учетом $\gamma + \gamma^2 + \gamma^4 = 1$ видно, что алгоритм требует трех умножений, четырех предварительных и пяти последующих сложений.

Таким образом, получаем следующую запись алгоритма БПФ:

$$\widetilde{\mathbf{F}} = \begin{bmatrix} F_0 \\ F_6 \\ F_5 \\ F_3 \\ F_4 \\ F_1 \\ F_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \left(\begin{bmatrix} 1 \\ 1 \\ \gamma^2 + \gamma^4 \\ \gamma + \gamma^4 \\ \gamma + \gamma^2 \\ 1 \\ \gamma^2 + \gamma^4 \\ \gamma + \gamma^4 \\ \gamma + \gamma^2 \end{bmatrix} \cdot \right.$$

$$\left. \cdot \left(\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \end{bmatrix} \right) \right) =$$

$$= Q(\mathbf{C} \cdot (PA^{-1}\widetilde{\mathbf{f}})),$$

где Q — двоичная блочно-диагональная матрица объединенных последующих сложений; $\mathbf{C}$ — объединенный вектор констант; P — двоичная блочно-диагональная матрица объединенных предварительных сложений.

Или, перемножая матрицы P и A^{-1}, имеем

$$\widetilde{\mathbf{F}} = \begin{bmatrix} F_0 \\ F_6 \\ F_5 \\ F_3 \\ F_4 \\ F_1 \\ F_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \times$$

$$\times \left(\begin{bmatrix} 1 \\ 1 \\ \gamma^2+\gamma^4 \\ \gamma+\gamma^4 \\ \gamma+\gamma^2 \\ 1 \\ \gamma^2+\gamma^4 \\ \gamma+\gamma^4 \\ \gamma+\gamma^2 \end{bmatrix} \cdot \left(\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \end{bmatrix} \right) \right).$$

Таким образом, вычисление БПФ длины 7 сводится к следующей последовательности действий:

выполнение предварительных сложений $(PA^{-1}) \times \widetilde{\mathbf{f}}$:

$$\begin{array}{ll} V_{10} = f_3 + f_5, & V_8 = f_5 + V_9, \\ V_{11} = f_1 + V_{10}, & V_{13} = f_6 + F_0, \\ V_2 = f_2 + V_{11}, & V_1 = V_{10} + V_{13}, \\ V_6 = f_4 + V_{11}, & V_5 = V_1 + V_{12}, \\ V_9 = f_6 + V_2, & V_7 = V_6 + V_8, \\ V_{12} = f_4 + V_9, & V_4 = V_7 + V_{10}, \\ F_0 = f_0 + V_{12}, & V_3 = V_2 + V_4; \end{array}$$

выполнение умножений на константы $\mathbf{C} \cdot (PA^{-1}\widetilde{\mathbf{f}})$:

$$
\begin{array}{ll}
V_{14} = V_2\,\alpha, & V_{17} = V_6\,\alpha, \\
V_{15} = V_3\,\alpha^2, & V_{18} = V_7\,\alpha^2, \\
V_{16} = V_4\,\alpha^4, & V_{19} = V_8\,\alpha^4;
\end{array}
$$

умножение матрицы Q на вектор $\mathbf{C} \cdot (PA^{-1}\widetilde{\mathbf{f}})$:

$$
\begin{array}{ll}
T_1 = V_1 + V_{16}, & T_3 = V_5 + V_{19}, \\
T_2 = V_{14} + V_{15}, & T_4 = V_{17} + V_{18}, \\
F_6 = V_{15} + T_1, & F_4 = V_{18} + T_3, \\
F_5 = V_1 + T_2, & F_1 = V_5 + T_4, \\
F_3 = V_{14} + T_1, & F_2 = V_{17} + T_3.
\end{array}
$$

Общая сложность алгоритма составляет $2 \times 3 = 6$ умножений и $14 + 2 \times 5 = 24$ сложений, что на одно сложение меньше, чем для алгоритма, представленного в параграфе 3.2.3 и работе [39].

Пример. Вычисление синдрома для (7, 5, 3)-кода Рида – Соломона. Пусть имеется $(7, 5, 3)$-код Рида – Соломона над конечным полем $\mathrm{GF}(2^3)$. Поле и базис заданы так же, как в предыдущем примере. Для определения первых двух компонент синдрома F_0 и F_1 запишем неполное ДПФ

$$
\begin{bmatrix} F_0 \\ F_1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix} \times
$$

$$
\times \left(\begin{bmatrix} 1 \\ 1 \\ \gamma^2 + \gamma^4 \\ \gamma + \gamma^4 \end{bmatrix} \cdot \left(\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \end{bmatrix} \right) \right).
$$

Эта запись приводит к следующей последовательности действий:

$$
\begin{aligned}
T_0 &= f_5 + f_6, & T_7 &= f_3 + T_3,\\
T_1 &= f_2 + f_4, & T_8 &= T_1 + T_6,\\
T_2 &= f_0 + f_3, & T_9 &= f_5 + T_7,\\
T_3 &= f_1 + f_4, & T_{10} &= \alpha T_9,\\
T_4 &= T_0 + T_2, & T_{11} &= \alpha^2 T_5,\\
T_5 &= T_0 + T_1, & T_{12} &= T_{10} + T_{11},\\
T_6 &= f_1 + T_4, &&
\end{aligned}
$$

$$
F_0 = T_8, \quad F_1 = T_4 + T_{12}.
$$

В табл. 7 приведена сложность вычисления синдрома для некоторых кодов Рида – Соломона в поле $\mathrm{GF}(2^8)$ в числе умножений N_{mul} и сложений N_{add}. Кроме предложенного метода, указана сложность непосредственного вычисления синдрома по методу Горнера и результат выполнения программы Захаровой FFTDesigner, основанный на ее работах [21, 20].

Предложенный метод базируется на тех же свойствах конечных полей, что и метод Захаровой [21, 20], однако он приводит к некоторому уменьшению числа операций за счет выполнения неполных циклических сверток и оптимизации сложений. Из представленной таблицы видно, что, начиная с вычисления шести синдромных компонент, предложенный метод проще не только метода Горнера, но и метода Захаровой.

Таблица 7

Сложность вычисления синдрома
для некоторых кодов Рида – Соломона

Параметры кода	Предложенный метод		Метод Горнера		Метод Захаровой	
(n, k, d)	N_{mul}	N_{add}	N_{mul}	N_{add}	N_{mul}	N_{add}
$(255, 253, 3)$	7	508	254	508	7	529
$(255, 251, 5)$	17	905	762	1016	18	875
$(255, 249, 7)$	27	1250	1270	1524	30	1268
$(255, 247, 9)$	37	1643	1778	2032	41	1652
$(255, 245, 11)$	45	1909	2286	2540	51	2036
$(255, 243, 13)$	55	2350	2794	3048	62	2391
$(255, 241, 15)$	65	2689	3302	3556	74	2789
$(255, 239, 17)$	75	2938	3810	4064	85	2989
$(255, 223, 33)$	149	5046	7874	8128	167	5440

3.4.2. Вычисление неполного ДПФ с помощью рекуррентного метода

Применим вычисление неполного ДПФ с помощью рекуррентного метода для нахождения синдрома при декодировании алгебраических кодов [41]. Из свойств матрицы эквивалентного преобразования W_c и теоремы 3.3 следует

$$W_c = W_c^T = A_c L_c = L_c^T A_c^T,$$

что позволяет эффективно использовать предложенный метод для вычисления неполного ДПФ.

Пусть требуется вычислить компоненты ДПФ с индексами из множества X. Расширим множество X до множества объединения циклотомических классов $Y \supset X$, т. е. для $j \in X$ содержащий j

циклотомический класс включим во множество Y. Тогда из $W_c = L_c^T A_c^T$ имеем формулу для вычисления неполного ДПФ

$$\mathbf{F}_Y = \begin{bmatrix} L_{j_1}^T & 0 & \dots & 0 \\ 0 & L_{j_2}^T & \dots & 0 \\ \dots & \dots & \ddots & \dots \\ 0 & 0 & \dots & L_{j_v}^T \end{bmatrix} \begin{bmatrix} A_{1j_1}^T & A_{2j_1}^T & \dots & A_{lj_1}^T \\ A_{1j_2}^T & A_{2j_2}^T & \dots & A_{lj_2}^T \\ \dots & \dots & \dots & \dots \\ A_{1j_v}^T & A_{2j_v}^T & \dots & A_{lj_v}^T \end{bmatrix} \mathbf{f}_c,$$

где $\mathbf{F}_Y = (F_i), \quad i \in Y$; $j_1, j_2, \dots, j_v$ — номера циклотомических классов, входящих во множество Y; v — число циклотомических классов, составляющих множество Y.

В последней формуле необходимо вычислить всего $v < l$ циклических сверток. Заметим, что компоненты преобразования с индексами из множества $Y \setminus X$ отбрасывают, а дальнейшее уменьшение числа умножений можно достичь за счет вычисления неполных сверток.

Пример. Вычисление синдрома для (7, 5, 3)-кода Рида – Соломона. Рассмотрим (7, 5, 3)-код Рида – Соломона над конечным полем $\mathrm{GF}(2^3)$. Поле и базис заданы так же, как в примере из параграфа 3.4.1. Пусть

$$f(x) = \sum_{i=0}^{n-1} f_i x^i \quad —$$

многочлен, соответствующий принятому вектору длины $n = 7$.

Первые две компоненты синдрома для этого кода есть $S_i = F_i = f(\alpha^i), \quad i \in [0, 1]$. Составим множество $X = \{0, 1\}$ и его расширение до множества объединения циклотомических классов $Y = \{0, 1, 2, 4\}$.

Запишем эквивалентное преобразование Фурье

$$\left[\begin{array}{c} F_1 \\ F_2 \\ F_4 \\ \hline F_6 \\ F_5 \\ F_3 \\ \hline F_0 \end{array}\right] = \left[\begin{array}{ccc|ccc|c} \gamma^1 & \gamma^2 & \gamma^4 & & & & \\ \gamma^2 & \gamma^4 & \gamma^1 & & & & \\ \gamma^4 & \gamma^1 & \gamma^2 & & & & \\ \hline & & & \gamma^1 & \gamma^2 & \gamma^4 & \\ & & & \gamma^2 & \gamma^4 & \gamma^1 & \\ & & & \gamma^4 & \gamma^1 & \gamma^2 & \\ \hline & & & & & & 1 \end{array}\right] =$$

$$= \left[\begin{array}{ccc|ccc|c} 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ \hline 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right] \left[\begin{array}{c} f_1 \\ f_2 \\ f_4 \\ \hline f_6 \\ f_5 \\ f_3 \\ \hline f_0 \end{array}\right].$$

Тогда неполное ДПФ есть

$$\left[\begin{array}{c} F_1 \\ F_2 \\ F_4 \\ \hline F_0 \end{array}\right] = \left[\begin{array}{ccc|c} \gamma^1 & \gamma^2 & \gamma^4 & \\ \gamma^2 & \gamma^4 & \gamma^1 & \\ \gamma^4 & \gamma^1 & \gamma^2 & \\ \hline & & & 1 \end{array}\right] \left[\begin{array}{ccc|ccc|c} 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right] \left[\begin{array}{c} f_1 \\ f_2 \\ f_4 \\ \hline f_6 \\ f_5 \\ f_3 \\ \hline f_0 \end{array}\right].$$

Запишем трехточечную циклическую свертку

$$\left[\begin{array}{c} F_1 \\ F_2 \\ F_4 \end{array}\right] = \left[\begin{array}{ccc} \gamma^1 & \gamma^2 & \gamma^4 \\ \gamma^2 & \gamma^4 & \gamma^1 \\ \gamma^4 & \gamma^1 & \gamma^2 \end{array}\right] \left[\begin{array}{c} t_0 \\ t_1 \\ t_2 \end{array}\right]$$

и алгоритм Винограда [8] для нее

$$\begin{bmatrix} F_1 \\ F_2 \\ F_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & & & \\ & \gamma^2+\gamma^4 & & \\ & & \gamma+\gamma^4 & \\ & & & \gamma+\gamma^2 \end{bmatrix} =$$

$$= \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} t_0 \\ t_1 \\ t_2 \end{bmatrix}.$$

Неполная циклическая свертка имеет вид

$$\begin{bmatrix} F_1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & & \\ & \gamma+\gamma^4 & \\ & & \gamma+\gamma^2 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} t_0 \\ t_1 \\ t_2 \end{bmatrix}.$$

Запишем окончательный вид неполного ДПФ

$$\left[\begin{array}{c} F_1 \\ \hline F_0 \end{array}\right] = \left[\begin{array}{cccc} 1 & 1 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \left[\begin{array}{ccc|c} 1 & & & \\ & \gamma+\gamma^4 & & \\ & & \gamma+\gamma^2 & \\ \hline & & & 1 \end{array}\right] \times$$

$$\times \left[\begin{array}{ccccccc} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ \hline 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right] \begin{bmatrix} f_1 \\ f_2 \\ f_4 \\ f_6 \\ f_5 \\ f_3 \\ f_0 \end{bmatrix}.$$

Пример. Вычисление синдрома для (15, 11, 5)-кода Рида – Соломона. Пусть имеется (15, 11, 5)-код Рида – Соломона над полем $\mathrm{GF}(2^4)$, а $\mathbf{f}$ — принятый вектор длины

$n = 15$. Поле и базис заданы так же, как в примере из параграфа 3.3.6. Для определения первых четырех компонент синдрома $S_i = F_i = f(\alpha^i), \quad i \in [1,4]$, составим множество $X = \{1,2,3,4\}$ и его расширение до множества объединения циклотомических классов $Y = \{1,2,4,8,6,12,9,3\}$. Тогда формула для вычисления неполного ДПФ принимает вид

$$\mathbf{F}_Y = \left[\begin{array}{c|c} L_1^T & \\ \hline & L_3^T \end{array}\right] \left[\begin{array}{c|c|c|c|c} A_{11}^T & A_{21}^T & A_{31}^T & A_{41}^T & A_{51}^T \\ \hline A_{13}^T & A_{23}^T & A_{33}^T & A_{43}^T & A_{53}^T \end{array}\right] \mathbf{f}_c$$

или

$$\begin{bmatrix} F_1 \\ F_2 \\ F_4 \\ F_8 \\ \hline F_6 \\ F_{12} \\ F_9 \\ F_3 \end{bmatrix} = \left[\begin{array}{cccc|cccc} \gamma^1 & \gamma^2 & \gamma^4 & \gamma^8 & & & & \\ \gamma^2 & \gamma^4 & \gamma^8 & \gamma^1 & & & & \\ \gamma^4 & \gamma^8 & \gamma^1 & \gamma^2 & & & & \\ \gamma^8 & \gamma^1 & \gamma^2 & \gamma^4 & & & & \\ \hline & & & & \gamma^1 & \gamma^2 & \gamma^4 & \gamma^8 \\ & & & & \gamma^2 & \gamma^4 & \gamma^8 & \gamma^1 \\ & & & & \gamma^4 & \gamma^8 & \gamma^1 & \gamma^2 \\ & & & & \gamma^8 & \gamma^1 & \gamma^2 & \gamma^4 \end{array}\right] \times$$

$$\times \left[\begin{array}{cccc|cccc|cccc|cccc} 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ \hline 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array}\right] \left[\begin{array}{c} f_1 \\ f_2 \\ f_4 \\ f_8 \\ \hline f_{11} \\ f_7 \\ f_{14} \\ f_{13} \\ \hline f_6 \\ f_{12} \\ f_9 \\ f_3 \\ \hline f_{10} \\ f_5 \\ \hline f_0 \end{array}\right].$$

Уменьшение числа умножений за счет отбрасывания компонент F_8, F_6, F_{12}, F_9 может быть предметом последующей оптимизации вычислений, непосредственно не связанной с предложенным методом.

Замечания к главе

Материал главы основан на работах автора (совместно с П. В. Трифоновым и другими) в области вычисления корней многочленов и ДПФ. Методы вычисления корней многочленов введены в работах [71, 72, 57], циклотомический алгоритм вычисления ДПФ — в работе [39], рекуррентный метод — в работе [41], а вычисление неполного преобразование Фурье предложено в работах [58, 41].

4. Декодирование по кодовым решеткам

Кодовая решетка есть представление линейного блокового кода в виде набора путей, последовательно соединяющих множество состояний на решетке с номерами от 0 до длины кода n. Каждому кодовому слову взаимно-однозначно соответствует путь на решетке. Если код имеет группу симметрии, то построенная для такого кода решетка будет обладать регулярной структурой. В главе рассматривается описание кодов с заданной группой симметрии, построение хороших циклических замкнутых решеток для квадратично-вычетных кодов, вводится новый тип звездных решеток, подходящих для описания и декодирования кода Голея и кодов Рида – Соломона.

4.1. Представления кодов с заданной группой симметрии

4.1.1. Две конструкции кодов

В работе [33] описана конструкция C_1, позволяющая строить код Голея. Порождающая матрица кода Голея конструкции C_1 задается следующим образом:

$$G = \begin{bmatrix} G_1 & 0 & G_1 \\ 0 & G_1 & G_1 \\ G_2 & G_2 & G_2 \end{bmatrix},$$

где G_1 и G_2 — порождающие матрицы эквивалентных самодуальных (8, 4, 4)-кодов Хэмминга. Использование этой конструкции для построения кодовой решетки специального вида позволяет получить алгоритм декодирования кода Голея [93]. Конструкция позволяет также описать ряд хороших кодов. Вопрос о существо-

вании других конструкций, подобных C_1, которые дают хорошие коды, остается открытым [33].

Для обобщения этой конструкции введем необходимые определения.

Матрица вида

$$C = \begin{bmatrix} c_0 & c_1 & \dots & c_{v-1} \\ c_{v-1} & c_0 & \dots & c_{v-2} \\ \vdots & \vdots & \ddots & \vdots \\ c_1 & c_2 & \dots & c_0 \end{bmatrix}$$

называется циркулянтной матрицей. Если элементы $c_0, c_1, \dots, c_{v-1}$ матрицы C в свою очередь являются матрицами, то матрицу C назовем блочно-циркулянтной. Пусть P есть перестановка, сохраняющая линейный блоковый (n, k, d)-код $\mathcal{G}$, причем порядком перестановки называется такое минимальное целое число $l = \mathrm{ord}(P) \geq 1$, что $P^l = I$, где I — тождественная перестановка. Тогда можно попытаться найти представление порождающей (или проверочной) матрицы кода в блочно-циркулянтном виде. Это представление может использоваться при построении кодовых решеток и для декодирования.

Рассмотрим двоичные коды с перестановкой порядка 3, к которым относятся все квадратично-вычетные и некоторые БЧХ-коды [33]. Тогда блочно-циркулянтное представление для порождающих матриц квадратично-вычетных кодов [66] в виде конструкции C_2 примет вид

$$G = \begin{bmatrix} G_1 & G_2 & 0 \\ 0 & G_1 & G_2 \\ G_2 & 0 & G_1 \end{bmatrix},$$

G_1 и G_2 — матрицы размера $\dfrac{k}{3} \times \dfrac{n}{3}$, причем для дважды четных самодуальных квадратично-вычетных кодов эти матрицы ортогональны, а не самоортогональны, как в конструкции C_1.

Введенная конструкция C_2 позволяет исследовать свойства квадратично-вычетных кодов, разрабатывать алгоритмы декоди-

рования, а также строить по порождающей матрице кодовые решетки. Кроме того, эта конструкция позволяет строить новые коды с хорошими свойствами. Среди кодов, удовлетворяющих конструкции C_2, существуют коды, лежащие на границе Варшамова – Гилберта.

Рассмотрим подстановку ST на множестве $\{0, 1, \ldots, n-2, \infty\}$. Так как подстановка S определена как $y \to y + 1 \pmod{n}$, а подстановка T — как $y \to -y^{-1}$, то подстановка ST: $y \to -(y+1)^{-1}$ имеет порядок 3 [33].

Пример. Подстановка ST при $n = 7$. При $n = 7$ первое отображение есть $ST(0, 1, 2, 3, 4, 5, 6, \infty) = (6, 3, 2, 5, 4, 1, \infty, 0)$. Второе — $ST(6, 3, 2, 5, 4, 1, \infty, 0) = (\infty, 5, 2, 1, 4, 3, 0, 6)$. Третье — $ST(\infty, 5, 2, 1, 4, 3, 0, 6) = (0, 1, 2, 3, 4, 5, 6, \infty)$.

Так как $n = 7$ не делится на 3, то подстановка распадается в произведение четырех циклов: $ST = (0, 6, \infty)(1, 3, 5)(2)(4)$ длины 3 и 1.

Пусть n — длина кода $\mathcal{G}$, $k = \dfrac{n}{2}$ — число информационных символов кода, d — минимальное расстояние Хэмминга кода. Для существования квадратично-вычетных кодов в виде конструкции C_2 необходимо, чтобы подстановка ST, сохраняющая квадратично-вычетный код, разбивала множество позиций кода $\{0, 1, \ldots, n-2, \infty\}$ на $\dfrac{n}{3}$ цикла длины 3 [33]. Такая подстановка существует для всех рассмотренных квадратично-вычетных кодов при $n < 200$, если n делится на 3.

В табл. 8 [66] приведены параметры расширенных двоичных квадратично-вычетных кодов, представленные в виде конструкции C_2, а также параметры подкодов $\mathcal{G}_1$ и $\mathcal{G}_2$, задаваемых порождающими матрицами G_1 и G_2 соответственно.

Таблица 8

Параметры расширенных двоичных квадратично-вычетных кодов

код $\mathcal{G}$			код $\mathcal{G}_1$			код $\mathcal{G}_2$		
n	k	d	n_1	k_1	d_1	n_2	k_2	d_2
18	9	6	6	3	2	6	3	2
24	12	8	8	4	3	8	4	3
42	21	10	14	7	4	14	7	4
48	24	12	16	8	4	16	8	4
72	36	12	24	12	5	24	12	5

Очевидно, что для подматриц самодуальных квадратично-вычетных кодов выполняются соотношения $G_1 G_2^T = 0$ и $G_1 G_1^T = G_2 G_2^T$. Заметим, что для квадратично-вычетных кодов имеется много различных представлений конструкции C_2, а подкоды $\mathcal{G}_1$ и $\mathcal{G}_2$ можно выбрать с различными минимальными расстояниями d_1 и d_2.

Для кодов с параметрами (24, 12, 8) и (48, 24, 12) найдено представление с эквивалентными подкодами $\mathcal{G}_1$ и $\mathcal{G}_2$.

Пример. Код Голея (24, 12, 8).

$$G_1 = \begin{bmatrix} 10001101 \\ 01000110 \\ 00101100 \\ 00011011 \end{bmatrix}, \quad G_2 = \begin{bmatrix} 00100111 \\ 10110110 \\ 11101010 \\ 11100100 \end{bmatrix}.$$

Пример. (48, 24, 12)-код.

$$G_1 = \begin{bmatrix} 1000001001010101 \\ 0100001000101010 \\ 0010001001100001 \\ 0001001001100110 \\ 0000101000110111 \\ 0000010001110011 \\ 0000000100101011 \\ 0000000011111101 \end{bmatrix}, \quad G_2 = \begin{bmatrix} 0011110110011011 \\ 1000101101000110 \\ 1110001101000001 \\ 1000011110100000 \\ 1100110010101011 \\ 0110000010011001 \\ 0110010101010010 \\ 1010101111100100 \end{bmatrix}.$$

Пример. (35, 15, 8)-код. Иногда коды имеют представление, близкое к блочно-циркулянтному. Так порождающая матрица БЧХ-кода с параметрами (35, 15, 8) приводится к виду

$$G = \begin{bmatrix} G_1 & G_2 & 0 & G_3 \\ 0 & G_1 & G_2 & G_3 \\ G_2 & 0 & G_1 & G_3 \end{bmatrix},$$

где G_1 и G_2 — порождающие матрицы $(10, 5, 3)$-кодов, а G_3 — $(5, 5, 1)$-кода.

4.1.2. Приведение матриц кода к квазициклическому представлению

Рассмотрим приведение порождающей матрицы G квадратично-вычетного кода к квазициклическому представлению конструкции C_2.

В соответствии со структурой циклов подстановки ST разобьем множество позиций кода на три непересекающихся подмножества J_0, J_1, J_2 так, что

$$\{0, 1, \ldots, n-2, \infty\} = J_0 \cup J_1 \cup J_2,$$
$$J_i = \{c_{i,1}, c_{i,2}, \ldots, c_{i,\frac{n}{3}}\}, \quad i \in [0, 2],$$
$$ST(c_{i,j}) = c_{((i+1) \bmod 3),j}.$$

В соответствии с рассмотренным разбиением переставляем столбцы порождающей матрицы G и получаем порождающую матрицу G' эквивалентного кода:

$$G' = \begin{matrix} J_0 & J_1 & J_2 \\ [G(J_0) & G(J_1) & G(J_2)], \end{matrix}$$

где $G(J_i)$ — подматрица матрицы G, составленная из столбцов с номерами из множества J_i.

Приводим матрицу G' к единичному виду на первых $\frac{n}{3}$ позициях. В результате получаем матрицу

$$G'' = \begin{bmatrix} N_0 & N_1 & N_2 \\ 0 & G_1 & G_2 \end{bmatrix},$$

где N_0, N_1, N_2 — некоторые $\left(\frac{2k}{3} \times \frac{n}{3}\right)$ матрицы; G_1, G_2 — $\left(\frac{k}{3} \times \frac{n}{3}\right)$ матрицы.

Очевидно, что $\operatorname{rank}[G_1 G_2] = \frac{k}{3}$. Применяя подстановку для $\frac{k}{3}$ нижних строк матрицы G'', получаем порождающую матрицу ее подкода

$$G^\star = \begin{bmatrix} G_1 & G_2 & 0 \\ G_2 & 0 & G_1 \\ 0 & G_1 & G_2 \end{bmatrix}.$$

Если

$$\det \begin{bmatrix} G_1 \\ G_2 \end{bmatrix} \neq 0, \tag{36}$$

то $\operatorname{rank} G^\star = \operatorname{rank} \begin{bmatrix} G_1 \\ G_2 \end{bmatrix} + \operatorname{rank}[G_1 G_2] = k$, а коды, задаваемые матрицами G'' и $G^\star$, совпадают, т. е. представление C_2 построено.

Если условие (36) не выполняется, то это означает, что подкоды $\mathcal{G}_1$ и $\mathcal{G}_2$, задаваемые порождающими матрицами G_1 и G_2

соответственно, имеют общие ненулевые кодовые слова. Заметим, что для самодуальных квадратично-вычетных кодов будет выполняться соотношение $G_1 G_2^T = 0$, т. е. коды $\mathcal{G}_1$ и $\mathcal{G}_2$ ортогональны. Учитывая это, а также запись матрицы G_1 в систематическом виде, переписываем матрицу G' в виде

$$G'' = \begin{bmatrix} N_0 & & N_1 & & N_2 & \\ 0 & I & F & MF^T & & M \end{bmatrix}, \tag{37}$$

где I — единичная $\left(\frac{k}{3} \times \frac{k}{3}\right)$ матрица; $G_1 = [IF]$; $G_2 = M[F^T I]$; M — невырожденная $\left(\frac{k}{3} \times \frac{k}{3}\right)$ матрица; F — некоторая $\left(\frac{k}{3} \times \frac{k}{3}\right)$ матрица.

Пусть подкоды $\mathcal{G}_1$ и $\mathcal{G}_2$ имеют общее ненулевое слово $\mathbf{m}G_1 \in \mathcal{G}_1, \mathcal{G}_2$. Тогда $\mathbf{m}G_1 {G_1}^T = 0$. Или, после преобразований $\mathbf{m}[IF] \begin{bmatrix} I \\ F^T \end{bmatrix} = 0$; $\mathbf{m}[I + FF^T] = 0$, что возможно тогда и только тогда, когда $\det [I + FF^T] = 0$.

Итак, если

$$\det [I + FF^T] \neq 0, \tag{38}$$

то коды $\mathcal{G}_1$ и $\mathcal{G}_2$ не имеют общих ненулевых слов и условие (36) выполняется.

Алгоритм приведения порождающей матрицы G квадратично-вычетного кода к виду конструкции C_2 состоит из следующих шагов.

Шаг 1. В соответствии со структурой циклов подстановки ST разбиваем множество позиций кода на три непересекающихся подмножества J_0, J_1, J_2.

Шаг 2. Записываем эквивалентное представление порождающей матрицы G'' кода $\mathcal{G}$ в виде (37).

Шаг 3. Вычисляем $\operatorname{rank} [I + FF^T]$. Если условие (38) выполнилось, то $G^\star$ — искомое представление порождающей матрицы.

Шаг 4. Иначе модифицируем подмножества

$$J_0 = (J_0 \backslash c_{0,l}) \cup c_{1,l};$$
$$J_1 = (J_1 \backslash c_{1,l}) \cup c_{2,l};$$
$$J_2 = (J_2 \backslash c_{2,l}) \cup c_{0,l}$$

для некоторого $l \in \left[1, \frac{n}{3}\right]$ и переходим к шагу 2.

При увеличении $\operatorname{rank}[I + FF^T]$ сохраняем модифицированные подмножества, иначе игнорируем последнюю модификацию и проводим новую. Таким образом, после некоторого числа шагов получим представление с полным рангом матрицы $I + FF^T$.

4.2. Циклические замкнутые решетки

В работе [50] рассмотрены два различных вида решеток для линейных блоковых кодов: обычные решетки (conventional trellis) и циклические замкнутые решетки (tail-biting trellis). Заметим, что обычные решетки допускают декодирование по минимуму расстояния. Для описания обычных решеток определим ось времени как $I = \{0, 1, \ldots, n\}$, где n — длина кода. Циклические замкнутые решетки имеют ось времени $I = \{0, 1, \ldots, n-1\}$, где индексы вычисляются по $\bmod n$.

Фактор-граф кодовой решетки есть граф, вершинами которого являются пространства состояний кодовой решетки S_i, $i \in [0, n]$, а ребро соединяет вершины S_i и S_j, если на кодовой решетке найдется некоторое состояние из пространства состояний S_i, которое соединяется кодовым символом с некоторым состоянием из пространства состояний S_j.

Фактор-граф обычной решетки изображен на рис. 2, а фактор-граф циклической замкнутой решетки — на рис. 3. Пространство состояний S_i обозначено кругом.

Метод построения циклической замкнутой решетки по порождающей матрице блокового кода описан в работе [50].

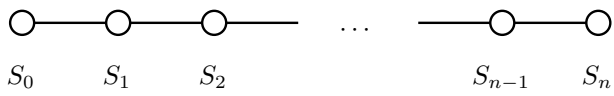

Рис. 2. *Фактор-граф решетки*

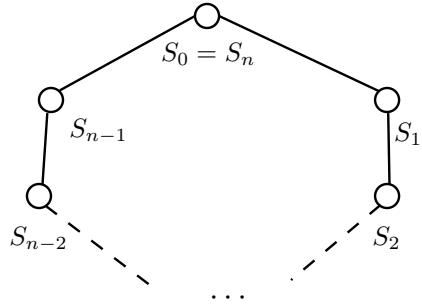

Рис. 3. *Фактор-граф циклической замкнутой решетки*

Пусть P — перестановка порядка l, сохраняющая линейный блоковый (n, k, d)-код. Будем искать порождающую матрицу кода в следующем виде:

$$G = \begin{bmatrix} G_1 & G_2 & \dots & G_l \\ G_l & G_1 & \dots & G_{l-1} \\ \vdots & \vdots & \ddots & \vdots \\ G_2 & G_3 & \dots & G_1 \end{bmatrix},$$

где G_i — некоторые $\left(\dfrac{k}{l} \times \dfrac{n}{l}\right)$ матрицы.

Понятно, что не для всех кодов такое представление существует. Предположим, что $l|n$ и $l|k$.

Таким образом, можно построить циклические замкнутые решетки для квазициклических кодов, какими являются все коды, имеющие представление в виде конструкции C_2 при $l = 3$. При этих условиях порождающая матрица кода будет иметь вид

$$G = \begin{bmatrix} G_1 & G_2 & 0 \\ 0 & G_1 & G_2 \\ G_2 & 0 & G_1 \end{bmatrix}.$$

Напомним, что побочно единичная матрица имеет единицы на побочной диагонали и нули на остальных позициях. Например, побочно единичная 4×4 матрица есть

$$\begin{bmatrix} 0001 \\ 0010 \\ 0100 \\ 1000 \end{bmatrix},$$

а блоковая побочно единичная 8×8 матрица, составленная из 2×2 единичных блоков, есть

$$\left[\begin{array}{cc|cc|cc|cc} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ \hline 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right].$$

Пример. Код Голея (24, 12, 8). В работе автора [30] построены такие матрицы G_1 и G_2, что

$$G_1 = I_{b_k}\, G_2\, I_{b_n},$$

где I_{b_k} — побочно единичная $k \times k$ матрица; I_{b_n} — блоковая побочно единичная $n \times n$ матрица, составленная из 2×2 единичных блоков:

$$G_1 = \begin{bmatrix} 11011011 \\ 00110111 \\ 00001110 \\ 00000011 \end{bmatrix}, \quad G_2 = \begin{bmatrix} 11000000 \\ 10110000 \\ 11011100 \\ 11100111 \end{bmatrix}.$$

Из построения видно, что циклическая замкнутая решетка для кода Голея будет минимальной решеткой, т. е. решетка удовлетворяет условию минимальности числа состояний.

Пример. (48, 24, 12)-код. В работе автора [62] построены такие матрицы G_1 и G_2, что

$$G_1 = I_{b_k}\, G_2\, I_{b_n},$$

где I_{b_k} — побочно единичная $k \times k$ матрица; I_{b_n} — побочно единичная $n \times n$ матрица:

$$G_1 = \begin{bmatrix} 1111010011001011 \\ 0011001011111110 \\ 0000110001101011 \\ 0000001110100001 \\ 0000000011111011 \\ 0000000000111110 \\ 0000000000001100 \\ 0000000000000011 \end{bmatrix}, \quad G_2 = \begin{bmatrix} 1100000000000000 \\ 0011000000000000 \\ 0111110000000000 \\ 1101111100000000 \\ 1000010111000000 \\ 1101011000110000 \\ 0111111101001100 \\ 1101001100101111 \end{bmatrix}.$$

Из построения видно, что циклическая замкнутая решетка будет минимальной решеткой.

Пример. (72, 36, 12)-код. Профиль сложности состояний циклической замкнутой решетки, построенный по порождающей матрице G, имеет максимум, равный 13:

$$G_1 = \begin{bmatrix} 000011011011001011000110 \\ 011101011110110001101010 \\ 001001000111100011110110 \\ 000000111011000011011101 \\ 000000000000001110011001 \\ 000000001111110010110100 \\ 000000000000110010000000 \\ 000000000011010111110000 \\ 000000000000000011001011 \\ 000000000000000000001101 \\ 000000000000000000000011 \\ 000000000000000000110101 \end{bmatrix},$$

$$G_2 = \begin{bmatrix} 100000000000000000000000 \\ 010100000000000000000000 \\ 101011000000000000000000 \\ 110010110000000000000000 \\ 001110101100000000000000 \\ 100010110011000000000000 \\ 111100010110110000000000 \\ 000011111010001100000000 \\ 101101010110011011000000 \\ 110011000101101111110000 \\ 100111011010111001101100 \\ 111010000001100110101011 \end{bmatrix}.$$

4.3. Звездные решетки

4.3.1. Представление кода Голея в виде звездной решетки

В работе автора [93] введен новый тип решетки — звездная решетка (star trellis). Она может быть построена, в частности, для кода Голея длины $n = 24$.

Пусть ось времени состоит из нескольких частей $I = \bigcup_j I_j$, которые сходятся в одной точке, например $I_j = \{0, (j-1)n/3+1, \ldots, jn/3, \infty\}$ для трех частей, где ∞ обозначает, что в этот момент сходятся все части оси времени. Здесь проводится согласование всех информационных символов кода.

Заметим, что вершиной фактор-графа также может быть специальное пространство состояний, в котором происходит согласование информационных символов кода. Пространство состояний S_i обозначено кругом, а согласование информационных символов кода — квадратом.

Фактор-граф звездной решетки изображен на рис. 4.

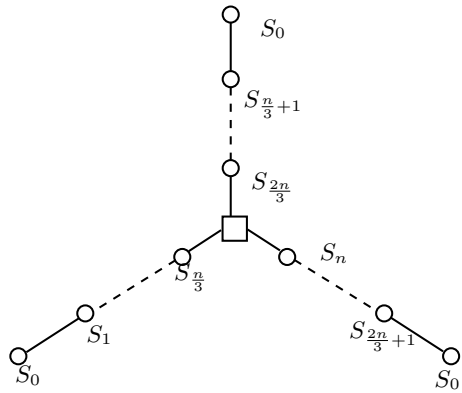

Рис. 4. *Фактор-граф звездной решетки*

Каждая часть оси времени ассоциируется с укорочением обычной решетки. Звездная решетка состоит из объединения всех укороченных решеток в одной точке ∞ с несколькими конечными состояниями. Пример звездной решетки для кода Голея представлен на рис. 5.

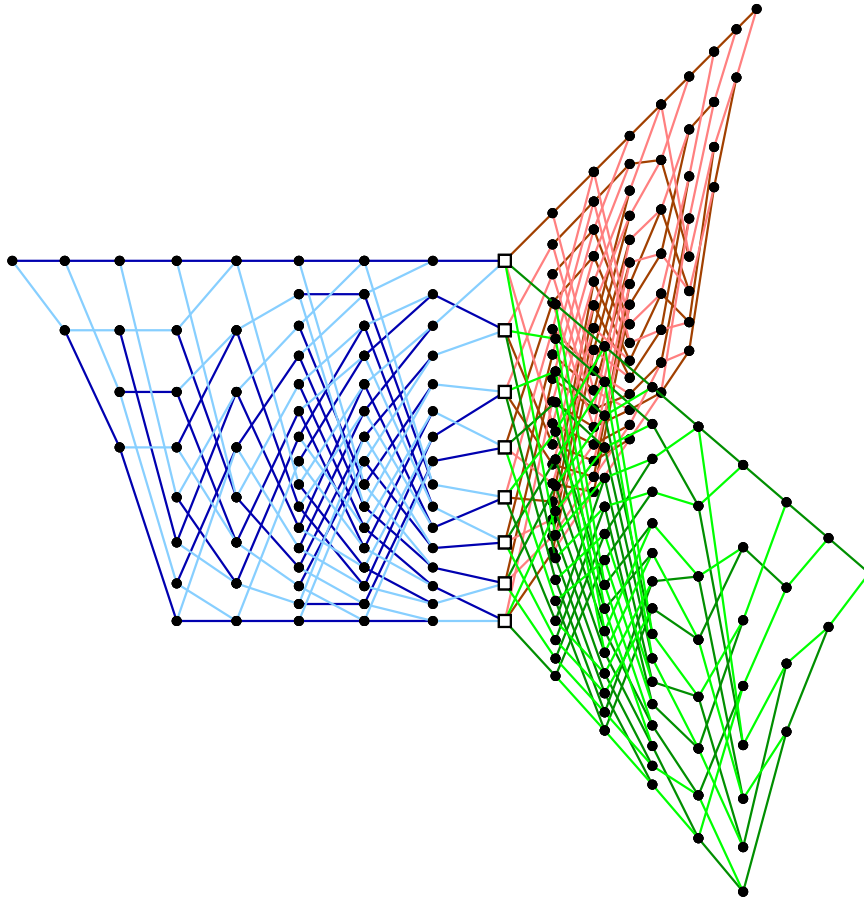

***Рис.* 5.** *Звездная решетка для кода Голея*

Представим порождающую матрицу G_{Golay} кода Голея в виде

$$G_{Golay} = \begin{array}{c} \\ j_1 \\ j_2 \\ j_3 \\ j_4 \\ j_5 \\ j_6 \\ j_7 \\ j_8 \\ j_9 \\ j_{10} \\ j_{11} \\ j_{12} \end{array} \left[\begin{array}{c|c|c} 11110000 & 00000000 & 11110000 \\ 01011010 & 00000000 & 01011010 \\ 00111100 & 00000000 & 00111100 \\ \hline 00000000 & 11110000 & 11110000 \\ 00000000 & 01011010 & 01011010 \\ 00000000 & 00111100 & 00111100 \\ \hline 11111111 & 00000000 & 00000000 \\ 00000000 & 11111111 & 00000000 \\ 00000000 & 00000000 & 11111111 \\ \hline 10011010 & 10011010 & 10011010 \\ 11001001 & 11001001 & 11001001 \\ 01111000 & 01111000 & 01111000 \end{array}\right]. \quad (39)$$

Этот вид получен перестановкой из конструкции Турина [33, 18.7.4]. Слева от порождающей матрицы указаны информационные символы кода Голея.

Обозначим первые p секций обычной решетки $\mathcal{T}$ через $\mathcal{T}^{(0:p)}$, где $\mathcal{T}^{(0:p)}$ — укороченная решетка с одним начальным состоянием и несколькими конечными. Поставим в соответствие таким решеткам расширенные порождающие матрицы с $p+1$ столбцами, причем знак ∞ в последнем столбце будет показывать наличие нескольких конечных состояний в решетке.

Пусть $J = (j_1, j_2, \ldots, j_{12})$ есть множество информационных символов исходного кода Голея, а $J_1 = (j_1, j_2, j_3, j_7, j_{10}, j_{11}, j_{12})$, $J_2 = (j_4, j_5, j_6, j_8, j_{10}, j_{11}, j_{12})$, $J_3 = (j_{1,4}, j_{2,5}, j_{3,6}, j_9, j_{10}, j_{11}, j_{12})$ — подмножества J, соответствующие частям оси времени I_i, $i \in [1, 3]$. Каждой из трех частей оси времени будет соответствовать расширенная порождающая матрица:

$$
G_1 = \begin{array}{c} j_1 \\ j_2 \\ j_3 \\ j_7 \\ j_{10} \\ j_{11} \\ j_{12} \end{array}
\left[\begin{array}{c|c} 11110000 & 0 \\ 01011010 & 0 \\ 00111100 & 0 \\ \hline 11111111 & 0 \\ \hline 10011010 & \infty \\ 11001001 & \infty \\ 01111000 & \infty \end{array}\right],
$$

$$
G_2 = \begin{array}{c} j_4 \\ j_5 \\ j_6 \\ j_8 \\ j_{10} \\ j_{11} \\ j_{12} \end{array}
\left[\begin{array}{c|c} 11110000 & 0 \\ 01011010 & 0 \\ 00111100 & 0 \\ \hline 11111111 & 0 \\ \hline 10011010 & \infty \\ 11001001 & \infty \\ 01111000 & \infty \end{array}\right],
$$

$$
G_3 = \begin{array}{c} j_{1,4} \\ j_{2,5} \\ j_{3,6} \\ j_9 \\ j_{10} \\ j_{11} \\ j_{12} \end{array}
\left[\begin{array}{c|c} 11110000 & 0 \\ 01011010 & 0 \\ 00111100 & 0 \\ \hline 11111111 & 0 \\ \hline 10011010 & \infty \\ 11001001 & \infty \\ 01111000 & \infty \end{array}\right].
$$

Слева от расширенной порождающей матрицы указаны информационные символы кода Голея.

Каждая укороченная решетка $\mathcal{T}^{(0:8)}$ состоит из $n/3 = 8$ секций и имеет одно начальное состояние и восемь конечных, соответствующих последним трем строкам порождающей матрицы кода Голея G_{Golay} (или G_i, $i \in [1,3]$). Объединяя три одинаковые укороченные решетки $\mathcal{T}^{(0:8)}$ во множество из восьми состояний, получаем звездную решетку для кода Голея.

Запишем условие согласования информационных символов:

$$
\begin{aligned}
j_{1,4} &= j_1 + j_4; \\
j_{2,5} &= j_2 + j_5; \\
j_{3,6} &= j_3 + j_6.
\end{aligned}
\tag{40}
$$

Таким образом, каждому кодовому слову кода Голея взаимнооднозначно соответствует путь на звездной решетке, состоящий из объединения трех путей, начинающихся в трех начальных состояниях укороченных решеток и сходящиеся в одном конечном состоянии, для которого выполняется условие согласования информационных символов.

Заметим, что максимальный профиль сложности состояний для звездной решетки составляет всего 16, что меньше соответствующей характеристики циклической замкнутой решетки кода Голея [93].

4.3.2. Декодирование кода Голея по звездной решетке

Укороченная решетка $\mathcal{T}^{(0:8)}$ имеет восемь конечных состояний. Эти состояния соответствуют трем информационным символам j_{10}, j_{11}, j_{12}. Это означает, что в точке согласования информационных символов ($\square$-состояние на фактор-графе звездной решетки) три укороченные решетки объединяются по восьми конечным состояниям в звездную решетку.

Из восьми возможных путей на звездной решетке только для кодовых слов кода Голея выполнится условие согласования информационных символов (40).

Запишем алгоритм декодирования по звездной решетке.

Шаг 1. Декодируем принятый вектор по трем укороченным решеткам $\mathcal{T}^{(0:8)}$ в восемь конечных $\square$-состояний.

Шаг 2. Выберем из восьми конечных $\square$-состояний два информационных множества J_i, $i \in [1,3]$, для которых соответствующие пути на решетке имеют наименьшее расстояние от подвектора принятого вектора.

Шаг 3. Определим для восьми конечных $\square$-состояний последний информационный символ и получаем до восьми полных наборов информационных символов.

Шаг 4. Строим кодовые слова по наборам информационных символов и выбираем из них ближайший к принятому вектору.

Теорема 4.1. *Предложенный алгоритм исправляет три ошибки в коде Голея.*

Доказательство. Всего возможно три случая расположения ошибок на трех частях принятого вектора, если учитывать симметрию частей: a) (1, 1, 1), b) (2, 1, 0) и c) (3, 0, 0). В корректном $\square$-состоянии известно истинное конечное состояние укороченной решетки $\mathcal{T}^{(0:8)}$, т. е. смежный класс (8, 4, 4)-кода Хэмминга в каждой части принятого вектора. Напомним, что первые четыре строки каждой матрицы G_i, $i \in [1, 3]$, образуют порождающую матрицу кода Хэмминга

$$G_{Hamming} = \begin{bmatrix} 11110000 \\ 01011010 \\ 00111100 \\ 11111111 \end{bmatrix}.$$

Следовательно, в случае a) все наборы информационных символов J_i, $i \in [1, 3]$, корректны. В случаях в) и с) первый набор информационных символов J_1 ошибочен и соответствующая ему часть вектора имеет расстояние с частью принятого вектора не менее $4 - 2 = 2$. Два других набора информационных символов J_2 и J_3 корректны. Таким образом, после второго шага в корректном $\square$-состоянии по меньшей мере два набора информационных символов корректны и множество информационных символов кода Голея может быть найдено. Ч.т.д.

Сложность алгоритма определяется шагом 1 (тремя декодированиями в кодах с порождающими матрицами G_i, $i \in [1, 3]$, по решетке $\mathcal{T}^{(0:8)}$ с максимальным профилем сложности состояний 16) и шагом 3 (восьмью перекодированиями (8, 3, 4)-подкода (8, 4, 4)-кода Хэмминга с порождающей матрицей

$$\begin{bmatrix} 11110000 \\ 01011010 \\ 00111100 \end{bmatrix}$$

и декодировании в (8, 1, 8)-коде с повторением). Шаг 4 не содержит никаких существенных вычислений, так как по всем частям расстояния с принятым вектором уже были вычислены на шагах 1 и 3. Заметим, что простой проверкой на четность и остановкой вычислений, если число найденных ошибок превысило 3, сложность шага 3 снижается максимально до двух перекодирований/декодирований.

4.3.3. Замечания к разделу

Предложенный алгоритм декодирования кода Голея по звездной решетке [93] имеет меньшую сложность, чем алгоритм декодирования по циклической замкнутой решетке, и гарантирует исправление ошибок до корректирующей способности кода. В алгоритме было введено согласование информационных символов в особом $\square$-состоянии на фактор-графе. Это согласование заключается в вычислении линейной зависимости между наборами информационных символов. В алгоритме рассмотрено только одно согласование информационных символов, что задает фактор-граф в звездной форме и приводит к звездной решетке. Однако можно предложить общий случай фактор-графов с большим числом согласований информационных символов в надежде построить декодер с небольшой сложностью.

4.4. Декодирование кодов Рида – Соломона по звездным решеткам

4.4.1. Разложение Варди – Беэри

В работе [97] предложено отображение произвольного кода Рида – Соломона $\mathcal{RS}$ в его двоичный образ $\mathrm{Im}(\mathcal{RS})$.

Введем обозначения для кода Рида – Соломона. $\mathcal{RS}(N, K, D)$ — код Рида – Соломона над $\mathrm{GF}(2^m)$ с параметрами $N = 2^m - 1$, $D = N - K + 1$, где $\alpha, \alpha^2, \ldots, \alpha^{D-1}$ — корни

порождающего многочлена $G(x)$, а α — примитивный элемент $\mathrm{GF}(2^m)$.

Аналогично введем БЧХ-код с той же проверочной матрицей, что и у кода Рида – Соломона. $\mathcal{BCH}(n,k,d)$ — двоичный БЧХ-код, причем выполняются следующие соотношения для параметров $n = N$, $k \le K$, $d \ge D$.

Порождающий многочлен БЧХ-кода $g(x) \in \mathrm{GF}(2)[x]$ имеет корни $\alpha, \alpha^2, \ldots, \alpha^{D-1}$ и их сопряженные относительно $\mathrm{GF}(2)$ элементы. G_{BCH} — порождающая матрица БЧХ-кода.

Очевидно, что $G(x) \mid g(x)$.

Рассмотрим $\{\gamma_1, \gamma_2, \ldots, \gamma_m\}$ — произвольный базис $\mathrm{GF}(2^m)$.

Так как $\alpha^j = \sum_{i=1}^{m} a_i \gamma_j$, то введем $\mathrm{Im}(\alpha^j) = (a_1, a_2, \ldots, a_m)$ — двоичный образ элемента α^j; причем $\mathrm{Im}(0) = (0, 0, \ldots, 0)$ — двоичный образ 0.

Для упрощения изложения будем использовать стандартный базис $\{\alpha^0, \alpha^1, \alpha^2, \ldots, \alpha^{m-1}\}$.

Для любого слова кода Рида – Соломона справедливы следующие соотношения:

$$\mathbf{c} = (c_0, c_1, \ldots, c_{n-1}) \in \mathcal{RS};$$

$$\gamma_i \mathbf{c} = (\gamma_i c_0, \gamma_i c_1, \ldots, \gamma_i c_{n-1}) \in \mathcal{RS},\ i \in [1, m];$$

$$\left[\begin{array}{c|c|c|c} \gamma_1 c_0 & \gamma_1 c_1 & \cdots & \gamma_1 c_{n-1} \\ \hline \gamma_2 c_0 & \gamma_2 c_1 & \cdots & \gamma_2 c_{n-1} \\ \hline \cdots & \cdots & \cdots & \cdots \\ \hline \gamma_m c_0 & \gamma_m c_1 & \cdots & \gamma_m c_{n-1} \end{array}\right] \in \mathcal{RS};$$

$$\left[\begin{array}{c|c|c|c} \mathrm{Im}(\gamma_1 c_0) & \mathrm{Im}(\gamma_1 c_1) & \cdots & \mathrm{Im}(\gamma_1 c_{n-1}) \\ \hline \mathrm{Im}(\gamma_2 c_0) & \mathrm{Im}(\gamma_2 c_1) & \cdots & \mathrm{Im}(\gamma_2 c_{n-1}) \\ \hline \cdots & \cdots & \cdots & \cdots \\ \hline \mathrm{Im}(\gamma_m c_0) & \mathrm{Im}(\gamma_m c_1) & \cdots & \mathrm{Im}(\gamma_m c_{n-1}) \end{array}\right] \in \mathrm{Im}(\mathcal{RS}).$$

Любое слово БЧХ-кода также является словом кода Рида – Соломона.

Для

$$\mathbf{b} = (b_0, b_1, \ldots, b_{n-1}) \in \mathcal{BCH},\ b_i \in \mathrm{GF}(2);$$

$$\mathbf{b} = (b_0, b_1, \ldots, b_{n-1}) \in \mathcal{RS};$$

$$\gamma_i \mathbf{b} = (\gamma_i b_0, \gamma_i b_1, \ldots, \gamma_i b_{n-1}) \in \mathcal{RS},\ i \in [1, m].$$

Выберем стандартный базис $\gamma_i = \alpha^{i-1}$, $i \in [1, m]$, тогда

$$\mathrm{Im}(\gamma_{i+1} b_j) = \mathrm{Im}(\alpha^i b_j) =$$

$$= \begin{pmatrix} 0 & \ldots & i-1 & i & i+1 & \ldots & m-1 \\ 0 & \ldots & 0 & b_j & 0 & \ldots & 0 \end{pmatrix};$$

$$\gamma_{i+1} \mathbf{b} = (\alpha^i b_0, \alpha^i b_1, \ldots, \alpha^i b_{n-1}) \in \mathcal{RS},\ i \in [0, m-1];$$

$$I_b = \left[\begin{array}{c|c|c|c} b_0 0 \cdots 0 & b_1 0 \cdots 0 & \cdots & b_{n-1} 0 \cdots 0 \\ \hline 0 b_0 \cdots 0 & 0 b_1 \cdots 0 & \cdots & 0 b_{n-1} \cdots 0 \\ \hline \cdots & \cdots & \cdots & \cdots \\ \hline 00 \cdots b_0 & 00 \cdots b_1 & \cdots & 00 \cdots b_{n-1} \end{array}\right] \in \mathrm{Im}(\mathcal{RS}).$$

Введем перестановку для столбцов порождающей матрицы двоичного образа кода Рида – Соломона

$\mathrm{Per}\Big((0,0),(0,1),\ldots,(0,m-1) \mid (1,0),(1,1),\ldots,(1,m-1) \mid \ldots \mid (n-1,0),(n-1,1),\ldots,(n-1,m-1)\Big) = \Big((0,0),(1,0),\ldots,(n-1,0) \mid (0,1),(1,1),\ldots,(n-1,1) \mid \ldots \mid (0,m-1),(1,m-1),\ldots,(n-1,m-1)\Big)$.

Тогда

$$\mathrm{Per}(I_b) = \left[\begin{array}{c|c|c|c} b_0 b_1 \cdots b_{n-1} & 00 \cdots 0 & \cdots & 00 \cdots 0 \\ \hline 00 \cdots 0 & b_0 b_1 \cdots b_{n-1} & \cdots & 00 \cdots 0 \\ \hline \cdots & \cdots & \cdots & \cdots \\ \hline 00 \cdots 0 & 00 \cdots 0 & \cdots & b_0 b_1 \cdots b_{n-1} \end{array}\right] \in$$

$$\in \mathrm{Per}(\mathrm{Im}(\mathcal{RS})).$$

Это верно для каждого вектора $\mathbf{b} \in \mathcal{BCH}$. Таким образом, порождающая матрица перестановки двоичного образа кода Рида – Соломона имеет вид

$$G_{\mathrm{Per}(\mathrm{Im}(\mathcal{RS}))} = \left[\begin{array}{c|c|c|c} G_{BCH} & 0 & \cdots & 0 \\ \hline 0 & G_{BCH} & \cdots & 0 \\ \hline \cdots & \cdots & \cdots & \cdots \\ \hline 0 & 0 & \cdots & G_{BCH} \\ \hline \multicolumn{4}{c}{\text{glue vectors}} \end{array}\right], \qquad (41)$$

где подматрица связывающих слов (“glue vectors”) имеет размерность $m(K-k) \times Nm$.

Пример. Код Рида – Соломона (7, 5, 3). Перестановка столбцов порождающей матрицы двоичного образа кода $\mathcal{RS}(7, 5, 3)$ приводит к (21, 15, 3)-коду с порождающей матрицей [78]

$$G_{\mathrm{Per}(\mathrm{Im}(\mathcal{RS}))} = \left[\begin{array}{c|c|c} 1101000 & 0000000 & 0000000 \\ 0110100 & 0000000 & 0000000 \\ 0011010 & 0000000 & 0000000 \\ 0001101 & 0000000 & 0000000 \\ \hline 0000000 & 1101000 & 0000000 \\ 0000000 & 0110100 & 0000000 \\ 0000000 & 0011010 & 0000000 \\ 0000000 & 0001101 & 0000000 \\ \hline 0000000 & 0000000 & 1101000 \\ 0000000 & 0000000 & 0110100 \\ 0000000 & 0000000 & 0011010 \\ 0000000 & 0000000 & 0001101 \\ \hline 1000000 & 0000100 & 0010000 \\ 0100000 & 0000010 & 0001000 \\ 0010000 & 0000001 & 0000100 \end{array}\right].$$

4.4.2. Метод декодирования

В работе автора [65] было отмечено сходство вида порождающей матрицы кода Голея (39) и порождающей матрицы перестановки двоичного образа кода Рида – Соломона (41). Следовательно, звездная решетка существует не только для кода Голея, но и для кода Рида – Соломона.

Для любого кода Рида – Соломона можно построить звездную решетку, состоящую из m частей, каждой из которых соответствует укороченная решетка с одним начальным состоянием и $2^{m(K-k)}$ конечными. Конечным состояниям соответствуют "glue vectors".

На первом этапе проводится классическое декодирование с мягкими решениями, например [37], по m укороченным решеткам. Результатом этого этапа является список, состоящий из не более $2^{m(K-k)}$ слов кода Рида – Соломона. Из этого списка на втором этапе производится выбор ближайшего к принятому вектору.

Для кода $\mathcal{RS}(7, 5, 3)$ зависимость вероятности ошибки на информационный бит (BER) и на кодовое слово (CER) от отношения сигнал/шум (SNR) в канале с аддитивным белым гауссовым шумом (AWGN) для предлагаемого метода и для классического декодирования показана в табл. 9. Под классическим методом декодирования кодов Рида – Соломона будем понимать алгоритм с исправлением ошибок и стираний, например, метод Форни или Чейза [74, 52], использующий алгоритм Берлекэмпа – Месси для решения ключевого уравнения.

Предложенный метод декодирования обеспечивает возможность получения выигрыша на информационный бит до 2–3 дБ в канале с AWGN по сравнению с классическим декодером кодов Рида – Соломона [65]. Такой выигрыш обеспечивается тем, что все подкоды двоичного образа кода Рида – Соломона (41) декодируются по минимуму расстояния.

Таблица 9

Зависимость вероятности ошибки от отношения сигнал/шум

SNR	BER Предложенный метод	BER Классический метод	CER Предложенный метод	CER Классический метод
1	0.0460	0.0640	0.20	0.45
2	0.0173	0.0433	0.10	0.36
3	0.0153	0.0227	0.09	0.17
4	0.0047	0.0080	0.03	0.07
5	0.0000	0.0040	0.00	0.03

Замечания к главе

Материал главы представляет собой результаты автора (совместно с Е. А. Круком и У. Зоргером) в области кодовых решеток. Представления кодов с заданной группой симметрии рассматривались в работе [66], построение циклических замкнутых решеток для квадратично-вычетных кодов — в работах [62, 30], звездные решетки введены в работе [93], а идея декодирования кодов Рида – Соломона по звездным решеткам предложена в работе [65].

Заключение

В настоящей работе рассмотрены и исследованы методы кодирования и декодирования линейных блоковых кодов, а также связанные с ними алгоритмы вычисления дискретного преобразования Фурье над конечным полем.

Основные результаты работы можно сформулировать следующим образом.

1. Введены понятия обобщенной информационной совокупности и табличного декодирования.

2. Выявлена “генетическая” связь различных алгебраических методов декодирования, представлен оригинальный вывод и доказательство корректности алгоритма Гао.

3. Предложены принципиально новые алгоритмы вычисления дискретного преобразования Фурье над конечным полем.

4. Введен новый тип решеток для линейных блоковых кодов.

Предложенная работа обобщает опыт развития методов быстрых алгоритмов в современной теории кодирования.

Библиографический список

1. *Арлазаров В. Л., Диниц Е. А., Кронрод М. А., Фараджев И. А.* Об экономном построении транзитивного замыкания графа// Докл. АН СССР. 1970. Т. 194. № 3. С. 487–488.

2. *Ахо А., Хопкрофт Дж., Ульман Дж.* Построение и анализ вычислительных алгоритмов. М.: Мир, 1979. 535 с.

3. *Афанасьев В. Б., Грушко И. И.* Алгоритмы БПФ для полей GF(2^m)// Сб. "Помехоустойчивое кодирование и надежность ЭВМ". М.: Наука, 1987. С. 33–55.

4. *Бассалыго Л. А.* Частное сообщение, 2003.

5. *Берлекэмп Э.* Алгебраическая теория кодирования. М.: Мир, 1971. 479 с.

6. *Бояринов И. М., Кабатянский Г. А.* Обобщенные коды Гоппы// Тр. IV Международного симпозиума по теории информации. Тез. докл. Москва – Ленинград, 1976. Ч. 2. С. 21–23.

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

8. *Блейхут Р.* Быстрые алгоритмы цифровой обработки сигналов. М.: Мир, 1989. 448 с.

9. *Блиновский В. М.* Нижняя асимптотическая граница для числа слов линейного кода в произвольной сфере с заданным радиусом из F_q^n// Проблемы передачи информации. 1987. Т. 23. № 2. С. 50–53.

10. *ван дер Варден Б. Л.* Алгебра. М.: Наука, 1976. 648 с.

11. *Габидулин Э. М., Афанасьев В. Б.* Кодирование в радиоэлектронике. М.: Радио и связь, 1986. 176 с.

12. *Гантмахер Ф. Р.* Теория матриц. М.: Наука, 1988. 552 с.

13. *Гоппа В. Д.* Рациональное представление кодов и (L, g)-коды// Проблемы передачи информации. 1971. Т. 7. № 3. С. 41–49.

14. *Грэхем Р., Кнут Д., Паташник О.* Конкретная математика. Основание информатики. М.: Мир, 1998. 703 с.

15. *Додунеков С. М.* Минимальная блоковая длина линейного q-ичного кода с заданными размерностью и кодовым расстоянием// Проблемы передачи информации. 1984. Т. 20. № 4. С. 11–22.

16. *Думер И. И.* Частное сообщение, 1986.

17. *Думер И. И.* Два алгоритма декодирования линейных кодов// Проблемы передачи информации. 1989. Т. 25. № 1. С. 24–32.

18. *Евсеев Г. С.* О сложности декодирования линейных кодов// Проблемы передачи информации. 1983. Т. 19. № 1. С. 3–8.

19. *Евсеев Г. С., Крук Е. А.* Об одном алгоритме декодирования КВ кодов// Тр. VI симпозиума по проблеме избыточности в информационных системах. Тез. докл. Л., 1974. Ч. 1. С. 26–30.

20. *Захарова Т. Г.* Вычисление преобразования Фурье в полях характеристики 2// Проблемы передачи информации. 1992. Т. 28. № 2. С. 62–77.

21. *Захарова Т. Г.* Применение преобразования Фурье в декодировании кодов Рида – Соломона// Радиотехника. 1996. № 12. С. 55–57.

22. *Кларк Дж., мл., Кейн Дж.* Кодирование с исправлением ошибок в системах цифровой связи. М.: Радио и связь, 1987. 392 с.

23. *Кнут Д. Э.* Искусство программирования. Т. 2. М.: Вильямс, 2000. 832 с.

24. *Крук Е. А.* Граница для сложности декодирования линейных блоковых кодов// Проблемы передачи информации. 1989. Т. 25. № 3. С. 103–107.

25. *Крук Е. А.* Комбинаторное декодирование линейных блоковых кодов: Монография. СПб.: ГУАП, 2007. 238 с.

26. *Крук Е. А., Мирончиков Е. Т., Федоренко С. В.* Декодирование блоковых линейных кодов по обобщенным информационным совокупностям// Радиотехника. 1997. № 2. С. 88–90.

27. *Крук Е. А., Трояновский Б. К., Федоренко С. В.* О программной реализации декодеров// Техника средств связи. Сер. ОТ. Вып. 4. М., 1987. С. 5–13.

28. *Крук Е. А., Федоренко С. В.* Декодирование по обобщенным информационным совокупностям// Проблемы передачи информации. 1995. Т. 31. № 2. С. 54–61.

29. *Крук Е. А., Федоренко С. В.* Комбинаторное декодирование в связи и криптографии// Научно-технические ведомости СПбГТУ. СПб.: Изд. СПбГТУ. 2002. № 3. С. 69–77.

30. *Крук Е. А., Федоренко С. В.* Самодуальные квазициклические коды// Научно-технические ведомости СПбГПУ. СПб.: Изд. СПбГТУ. 2004. № 1. С. 163–167.

31. *Лидл Р., Нидеррайтер Г.* Конечные поля. В 2 т. М.: Мир, 1988. 822 с.

32. *Липницкий В. А, Стройникова Е. Д.* О вычислении дискретного преобразования Фурье в полях Галуа// Сибирский журнал индустриальной математики. 2002. Т. V. № 3(11). С. 131–138.

33. *Мак-Вильямс Ф. Дж., Слоэн Н. Дж. А.* Теория кодов, исправляющих ошибки. М.: Связь, 1979. 744 с.

34. *Мирончиков Е. Т., Федоренко С. В.* Декодирование (L, g)-кодов в стирающем канале// Тр. V совещания по распределенным вычислительным системам и сетям. Тез. докл. М., 1992. С. 190–191.

35. *Мирончиков Е. Т., Федоренко С. В.* Декодирование (L, g)-кодов по обобщенным информационным совокупностям// Проблемы передачи информации. 1993. Т. 29. № 4. С. 94–98.

36. *Мирончиков Е. Т., Федоренко С. В.* Об алгебраическом декодировании циклических кодов// Проблемы передачи информации. 1999. Т. 35. № 1. С. 44–48.

37. *Морелос-Сарагоса Р.* Искусство помехоустойчивого кодирования. Методы, алгоритмы, применение. М.: Техносфера, 2005. 320 с.

38. *Питерсон У., Уэлдон Э.* Коды, исправляющие ошибки. М.: Мир, 1976. 596 с.

39. *Трифонов П. В., Федоренко С. В.* Метод быстрого вычисления преобразования Фурье над конечным полем// Проблемы передачи информации. 2003. Т. 39. № 3. С. 3–10.

40. *Федоренко С. В.* Сложность декодирования линейных блоковых кодов// Проблемы передачи информации. 1993. Т. 29. № 4. С. 18–23.

41. *Федоренко С. В.* Метод вычисления дискретного преобразования Фурье над конечным полем// Проблемы передачи информации. 2006. Т. 42. № 2. С. 81–93.

42. *Эрдёш П., Спенсер Дж.* Вероятностные методы в комбинаторике. М.: Мир, 1976. 133 с.

43. *Afanasyev V.* On complexity of FFT over finite field: Proc. of the Sixth Joint Swedish-Russian International Workshop on Information Theory, Molle, Sweden. August 1993. P. 315–319.

44. *Asnis I. L., Fedorenko S. V.* Tables of coverings for decoding by S-sets// The Workshop on Information Protection. M., 1993. P. 22.

45. *Asnis I. L., Fedorenko S. V., Krouk E. A., Mironchikov E. T.* Tables of coverings for decoding by S-sets// Error control, cryptology, and speech compression: Lecture notes in computer science. Springer-Verlag. 1994. Vol. 829. P. 97–102.

46. *Assmus E. F. Jr., Mattson H. F. Jr.* New 5-designs// Journal of Combinatorial Theory. 1969. Vol. 6. N. 6. P. 122–151.

47. *Barg A.* Complexity issues in coding theory// Handbook of Coding Theory. Vol. 1/ Pless V., Huffman W. C., Eds., Amsterdam, The Netherlands: Elsevier Science, 1998. P. 649–754.

48. *Barg A., Krouk E., van Tilborg H. C. A.* On the complexity of minimum distance decoding of long linear codes// IEEE Transactions on Information Theory. July 1999. Vol. 45. P. 1392–1405.

49. *Bossert M.* Channel coding for telecommunications. John Wiley & Sons, Ltd, 1999. 496 p.

50. *Calderbank A. R., Forney G. D., Vardy A.* Minimal tail-biting trellises: the Golay code and more// IEEE Transactions on Information Theory. 1999. Vol. 45. N. 5. P. 1435–1455.

51. *Chambers W. G.* Solution of Welch – Berlekamp key equation by Euclidean algorithm// Electronics Letters. 1993. Vol. 29. N. 11. P. 1031.

52. *Chase D.* A class of algorithms for decoding block codes with channel measurement information// IEEE Transactions on Information Theory. Jan. 1972. Vol. 18. P. 170–182.

53. *Chen C.-L.* Formulas for the solutions of quadratic equations over GF(2^m)// IEEE Transactions on Information Theory. 1982. Vol. 28. N. 5. P. 792–794.

54. *Chien R. T.* Cyclic decoding procedures for Bose – Chaudhuri – Hocquenghem codes// IEEE Transactions on Information Theory. 1964. Vol. 10. N. 4. P. 357–363.

55. *Chien R. T., Cunningham B. D., Oldham I. B.* Hybrid methods for finding roots of a polynomial with application to BCH decoding// IEEE Transactions on Information Theory. 1969. Vol. 15. N. 2. P. 329–335.

56. *Coffey J. T., Goodman R. M. F.* The complexity of information set decoding// IEEE Transactions on Information Theory. 1990. Vol. 35. N. 5. P. 1031–1037.

57. *Costa E., Fedorenko S., Krouk E., Lott M., Schulz E., Trifonov P.* Method and device for a communication system for finding roots of an error locator polynomial. European patent application, EP1367727 A1 dated 03.12.2003.

58. *Costa E., Fedorenko S. V., Trifonov P. V.* On computing the syndrome polynomial in Reed – Solomon decoder// European Transactions on Telecommunications. 2004. Vol. 15. N. 4. P. 337–342.

59. *Dumer I.* On minimum distance decoding of linear codes: Proc. of the Fifth Joint Soviet-Swedish International Workshop on Information Theory at Moscow, USSR, January 1991. P. 50–52.

60. *Elia M.* Algebraic decoding of the (23,12,7) Golay code// IEEE Transactions on Information Theory. 1987. Vol. 33. N. 1. P. 150–151.

61. *Eklund C., Marks R. B., Stanwood K. L., Wang S.* IEEE standard 802.16: a technical overview of the WirelessMAN air interface for broadband wireless access// IEEE Communications Magazine. June 2002. Vol. 40. N. 6. P. 98–107.

62. *Fedorenko S.* On the structure of linear block codes given the group of symmetry: Proc. of IEEE International Workshop on Concatenated codes, Ulm, Germany, 1999. P. 1–2.

63. *Fedorenko S. V.* A simple algorithm for decoding Reed – Solomon codes and its relation to the Welch — Berlekamp algorithm// IEEE Transactions on Information Theory. Mar. 2005. Vol. 51. N. 3. P. 1196–1198.

64. *Fedorenko S. V.* Correction to "A simple algorithm for decoding Reed – Solomon codes and its relation to the Welch – Berlekamp algorithm"// IEEE Transactions on Information Theory. 2006. Vol. 52. N. 3. P. 1278.

65. *Fedorenko S. V.* The star trellis decoding of Reed – Solomon codes: Proc. of the XI international symposium on problems of redundancy in information and control systems at St.Petersburg, Russia, July 2007. P. 58–61.

66. *Fedorenko S., Krouk E.* About block circulant representation of linear codes: Proc. of Sixth International Workshop on Algebraic and Combinatorial Coding Theory, Pskov, Russia, 1998. P. 116–118.

67. *Fedorenko S., Krouk E.* The table decoders of quadratic-residue codes: Proc. of the Seventh International Workshop on Algebraic and Combinatorial Coding Theory at Bansko, Bulgaria, June 2000. P. 137–140.

68. *Fedorenko S., Krouk E.* A survey of the hard decision decoding for linear block codes: Proc. of the workshop on concepts in information theory, Breisach, Germany, June 2002. P. 15–18.

69. *Fedorenko S., Krouk E.* Decoding beyond the designed error correcting capability on the basis of supercodes: Proc. of the IEEE International Symposium on Information Theory at Lausanne, Switzerland, 2002. P. 89.

70. *Fedorenko S., Trifonov P.* On computing the fast Fourier transform over finite fields: Proc. of the Eighth International Workshop on Algebraic and Combinatorial Coding Theory at Tsarskoe Selo, Russia, September 2002. P. 108–111.

71. *Fedorenko S., Trifonov P.* Finding roots of polynomials over finite fields// IEEE Transactions on Communications. 2002. Vol. 50. N. 11. P. 1709–1711.

72. *Fedorenko S., Trifonov P., Costa E.* Improved hybrid algorithm for finding roots of error-locator polynomials// European Transactions on Telecommunications. 2003. Vol. 14. N. 5. P. 411–416.

73. *Feng G.-L., Tzeng K. K.* A new procedure for decoding cyclic and BCH codes up to actual minimum distance// IEEE Transactions on Information Theory. 1994. Vol. 40. N. 5. P. 1364–1374.

74. *Forney G. D. Jr.* Generalized minimum distance decoding// IEEE Transactions on Information Theory. Apr. 1966. Vol. 12. P. 125–131.

75. *Freudenberger J.* On bounded distance list decoding based on supercodes: Proc. of the Seventh International Symposium on Communication Theory and Applications, Ambleside, UK, 2003. P. 7–12.

76. *Gao S.* A new algorithm for decoding Reed – Solomon codes// Communications, Information and Network Security/ Bhargava V., Poor H. V., Tarokh V., Yoon S., Eds. Norwell, MA: Kluwer, 2003. Vol. 712. P. 55–68.

77. *Gemmell P., Sudan M.* Highly resilient correctors for polynomials// Information Processing Letters. 1992. Vol. 43. N. 4. P. 169–174.

78. *Halford T. R., Ponnampalam V., Grant A. J., Chugg K. M.* Soft-in soft-out decoding of Reed – Solomon codes based on Vardy and Be'ery's decomposition// IEEE Transactions on Information Theory. Dec. 2005. Vol. 51. P. 4363–4368.

79. *Helgert H. J., Stinaff R. D.* Minimum-Distance Bounds for Binary Linear Codes// IEEE Transactions on Information Theory. 1973. Vol. 19. N. 3. P. 344–356.

80. *Hong J., Vetterli M.* Computing m DFT's over GF(q) with one DFT over GF(q^m)// IEEE Transactions on Information Theory. January 1993. Vol. 39. N. 1. P. 271–274.

81. *Justesen J.* On the complexity of decoding Reed – Solomon codes// IEEE Transactions on Information Theory. Mar. 1976. Vol. 22. N. 2. P. 237–238.

82. *Kabatiansky G., Krouk E., Semenov S.* Error correcting coding and security for data networks: Analysis of the superchannel concept. John Wiley & Sons, Ltd, 2005. 278 p.

83. *Kasami T.* A Decoding Procedure for Multiple-Error-Correcting Cyclic Codes// IEEE Transactions on Information Theory. 1964. Vol. 10. N. 2. P. 134–138.

84. *Krouk E. A., Mironchikov E. T., Fedorenko S. V.* Decoding by S-sets: Proc. of the Fifth Joint Soviet-Swedish International Workshop on Information Theory at Moscow, USSR, January 1991. P. 113–115.

85. *Levitin L., Hartmann C. R. P.* A new approach to the general minimum distance decoding problem: The zero-neighbors algorithm// IEEE Transactions on Information Theory. 1985. Vol. 31. N. 3. P. 378–384.

86. *MacWilliams F. J.* Permutation Decoding of Systematic Codes// Bell System Technical Journal. 1964. Vol. 43. P. 485–505.

87. *Moenck R. T.* Fast computation of GCDs: Proc. of the 5th Annual ACM Symposium on Theory of Computing, Austin, TX, 1973. P. 142–151.

88. *Morii M., Kasahara M.* Generalized key-equation of remainder decoding algorithm for Reed – Solomon codes// IEEE Transactions on Information Theory. Nov. 1992. Vol. 38. N. 6. P. 1801–1807.

89. *Prange E.* The Use of Information Sets in Decoding Cyclic Codes// IRE Transactions on Information Theory. 1962. Vol. 8. N. 5. P. S5–S9.

90. *Rader C. M.* Discrete Fourier transforms when the number of data samples is prime: Proc. of the IEEE. 1968. Vol. 56. N. 6. P. 1107–1108.

91. *Rudolph L. D., Mitchell M. E.* Implementation of Decoders for Cyclic Codes// IEEE Transactions on Information Theory. 1964. Vol. 10. N. 3. P. 259–260.

92. *Shiozaki A.* Decoding of redundant residue polynomial codes using Euclid's algorithm// IEEE Transactions on Information Theory. Sep. 1988. Vol. 34. N. 5. P. 1351–1354.

93. *Sorger U., Fedorenko S.* The "Star Trellis" of the Golay Code: Proc. of Seventh International Workshop on Algebraic and Combinatorial Coding Theory at Bansko, Bulgaria, June 2000. P. 288–292.

94. *Sugiyama Y., Kasahara M., Hirasawa S., Namekawa T.* A method for solving key equation for decoding Goppa codes// Information and Control. 1975. Vol. 27. P. 87–99.

95. *Trifonov P.* Matrix-Vector Multiplication via Erasure Decoding: Proc. of the XI international symposium on problems of redundancy in information and control systems at St.Petersburg, Russia, July 2007. P. 104–108.

96. *Truong T.-K., Jeng J.-H., Reed I. S.* Fast algorithm for computing the roots of error locator polynomials up to degree 11 in Reed – Solomon decoders// IEEE Transactions on Communications. 2001. Vol. 49. N. 5. P. 779–783.

97. *Vardy A., Be'ery Y.* Bit-level soft-decision decoding of Reed – Solomon codes// IEEE Transactions on Communications. Mar. 1991. Vol. 39. P. 440–444.

98. *Wang Y., Zhu X.* A fast algorithm for the Fourier transform over finite fields and its VLSI implementation// IEEE Journal on Selected Areas in Communications. Apr. 1988. Vol. 6. N. 3. P. 572–577.

99. *Welch L., Berlekamp E.R.* Error correction for algebraic block codes. U.S. Patent 4,633,470, Sep. 27, 1983.

100. *Williamson C. J.* Apparatus and method for error correction. U.S. Patent 5,905,740, May 1999.

101. *Wolfmann J.* A Permutation Decoding of the (24,12,8) Goley Code// IEEE Transactions on Information Theory. 1983. Vol. 29. N. 5. P. 748–751.

Научное издание

Федоренко Сергей Валентинович

Методы быстрого декодирования
линейных блоковых кодов

Монография

Редактор *А. Г. Ларионова*

Подписано к печати 26.03.08. Формат бумаги 60 × 84 1/16.
Бумага офсетная. Печать офсетная. Усл. печ. л. 12,5. Уч.-изд. л. 9,5.
Тираж 150 экз.
Заказ №

Отпечатано с оригинал-макета, подготовленного кафедрой
безопасности информационных систем Санкт-Петербургского
государственного университета аэрокосмического приборостроения
Редакционно-издательский центр ГУАП
190000, Санкт-Петербург, Б. Морская ул., 67