ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ им. проф. М. А. БОНЧ-БРУЕВИЧА»
(СПбГУТ)

С. С. Владимиров, О. С. Когновицкий

ПРАКТИКА ПОМЕХОУСТОЙЧИВОГО КОДИРОВАНИЯ

Циклические коды

Практикум

СПб ГУТ)))

Санкт-Петербург
2019

УДК 621.391 (076)
ББК 32.811.7 я73
В 57

Рецензент
доцент кафедры СС и ПД,
кандидат технических наук *А. Н. Глухов*

Рекомендован к печати редакционно-издательским советом СПбГУТ

Владимиров, С. С.

В 57 Практика помехоустойчивого кодирования. Циклические коды : практикум / С. С. Владимиров, О. С. Когновицкий ; СПбГУТ. — СПб, 2019. — 44 с.

Призван ознакомить студентов с циклическими помехоустойчивыми кодами и методами их декодирования. Представленный материал служит справочным и методическим пособием при выполнении курса практических работ по дисциплинам «Практика помехоустойчивого кодирования в современных инфокоммуникационных системах» и «Помехоустойчивое кодирование в инфокоммуникационных системах».

Предназначен для студентов, обучающихся по направлениям 09.03.01 «Информатика и вычислительная техника» и 11.03.02 «Инфокоммуникационные технологии и системы связи».

УДК 621.391 (076)
ББК 32.811.7 я73

Содержание

Практическая работа 1
Циклические коды

1.1. Цель работы

Рассмотреть на примере и получить навыки в исследовании блочных циклических кодов.

1.2. Теоретическая справка

Циклические коды являются подмножеством линейных кодов. Как и ранее, мы остановимся на рассмотрении двоичных циклических кодов.

Линейный (n,k)-код C назывется *циклическим*, если циклический сдвиг любого кодового слова (вектора) $\mathbf{v}$ из C также принадлежит коду C [1].

Циклический сдвиг кодового слова $\mathbf{v} = \{v_0, v_1, \ldots, v_{n-1}\}$ соответствует сдвигу всех элементов слова на одну позицию вправо, в результате чего будет получено кодовое слово $\mathbf{v}^{(1)} = \{v_{n-1}, v_0, v_1, \ldots, v_{n-2}\}$. В результате же i-кратного сдвига получается кодовое слово $\mathbf{v}^{(i)} = \{v_{n-i}, \ldots, v_{n-1}, v_0, v_1, \ldots, v_{n-i-1}\}$ [1].

Аппаратно циклический сдвиг реализуется при помощи n-разрядного регистра сдвига с обратной связью (рис. 1.1) [1].

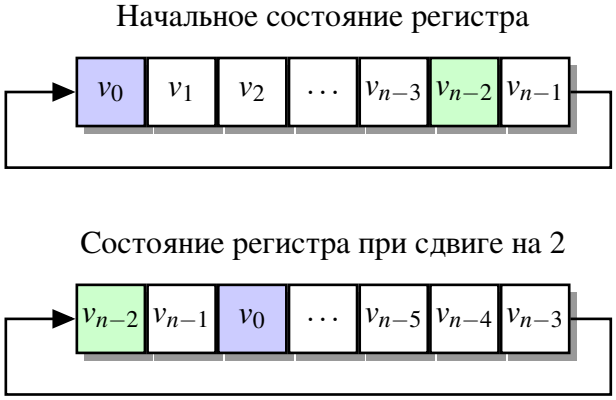

Рис. 1.1. Регистр сдвига с обратной связью

Кодовые слова циклического кода удобно представлять в виде многочленов над полем GF(2) [1]. Так кодовое слово $\mathbf{v} = \{v_0, v_1, \ldots, v_{n-1}\}$ можно представить полиномом

$$v(x) = v_0 x^0 + v_1 x^1 + \ldots + v_{n-1} x^{n-1} = v_0 + v_1 x + \ldots + v_{n-1} x^{n-1}. \qquad (1.1)$$

Циклический сдвиг кодового слова $\mathbf{v}$ на i позиций можно представить в виде полинома, начальную часть которого выделим и обозначим как $q(x)$ [1]:

$$v^{(i)}(x) = \underbrace{v_{n-i} + v_{n-i+1} x + \ldots + v_{n-1} x^{i-1}}_{q(x)} + v_0 x^i + v_1 x^{i+1} + \ldots + v_{n-i-1} x^{n-1}. \qquad (1.2)$$

Если сравнить (1.2) с произведением $v(x)$ на x^i, можно увидеть, что в нем также присутствует $q(x)$ [1]:

$$\begin{aligned} x^i \cdot v(x) &= v_0 x^i + v_1 x^{i+1} + \ldots + v_{n-i-1} x^{n-1} + \\ &+ \underbrace{v_{n-i} x^n + \ldots + v_{n-1} x^{n+i-1}}_{q(x)\cdot x^n}. \end{aligned} \tag{1.3}$$

Сравнивая эти выражения, можно составить равенство

$$x^i \cdot v(x) = q(x) \cdot (x^n + 1) + v^{(i)}(x), \tag{1.4}$$

из которого следует, что *многочлен, соответствующий циклическому сдвигу вектора v на i позиций, можно вычислить как остаток от деления многочлена* $x^i v(x)$ *на* $(x^n + 1)$. Это свойство используется в процедуре эффективного обнаружения ошибок [1].

1.2.1. Порождающий многочлен циклического кода

Для построения циклического (n,k)-кода C используется *порождающий* (или *образующий*) многочлен $g(x)$ степени $r = n - k$

$$g(x) = g_0 + g_1 x + \ldots + g_r x^r. \tag{1.5}$$

Для порождающего многочлена циклического кода верен ряд следующих утверждений [1].

1. $g(x)$ является кодовым многочленом наименьшей степени r.
2. Коэффициент g_0 многочлена $g(x)$ всегда равен 1.
3. Полином $v(x)$ является кодовым тогда и только тогда, когда он кратен полиному $g(x)$.
4. Полином $g(x)$ делит $(x^n + 1)$ без остатка. Верно и обратное утверждение (п. 5).
5. Если некоторый полином $g(x)$ степени $n - k$ делит $(x^n + 1)$ без остатка, то он порождает некоторый циклический (n,k)-код.

В качестве примера определим порождающий многочлен циклического кода $(7,4)$. Как следует из определения, он строится посредством порождающего многочлена $g(x)$ степени $r = n - k = 7 - 3 = 4$, являющегося делителем полинома $x^7 + 1$. В формуле (1.6) показано разложение полинома $x^7 + 1$ на множители:

$$x^7 + 1 = (1 + x)(1 + x + x^3)(1 + x^2 + x^3). \tag{1.6}$$

Видно, что в разложении присутствуют два полинома степени 3. Каждый из них может быть использован для построения циклического кода.

1.2.2. Несистематическое кодирование циклического кода

Несистематическое кодирование циклического (n,k)-кода заключается в умножении информационного полинома $u(x)$ на образующий полином кода. Процесс кодирования можно представить формулой

$$v(x) = u(x) \cdot g(x). \tag{1.7}$$

По аналогии с (1.1) полиному $u(x)$ соответствует двоичный вектор $\mathbf{u}$.

Для примера рассмотрим кодирование вектора $\mathbf{u} = \{1\ 0\ 1\ 0\}$ для циклического кода $(7,4)$, образованного полиномом $g(x) = 1 + x + x^3$. Вектор $\mathbf{u}$ представим в виде полинома

$$u(x) = 1 + x^2.$$

Тогда кодовый полином $v(x)$ и соответствующий ему вектор $\mathbf{v}$ будут равны

$$v(x) = (1+x^2)(1+x+x^3) = 1 + x + x^2 + x^5 \longrightarrow \mathbf{v} = \{1\ 1\ 1\ 0\ 0\ 1\ 0\}.$$

Можно видеть, что в случае несистематического кодирования кодовое слово $\mathbf{v}$ не содержит символов информационного вектора $\mathbf{u}$.

1.2.3. Систематические циклические коды

Информационный полином $u(x)$ для циклического (n,k)-кода имеет степень $k-1$

$$u(x) = u_0 + u_1 x + \ldots + u_{k-1} x^{k-1}, \tag{1.8}$$

из чего следует, что его $(r = n-k)$-кратный сдвиг

$$x^r \cdot u(x) = u_0 x^r + u_1 x^{r+1} + \ldots + u_{k-1} x^{n-1} \tag{1.9}$$

не приводит к переполнению n-разрядного регистра сдвига и соответствует заполнению k правых двоичных разрядов регистра информационным словом. Теперь необходимо заполнить r левых разрядов так, чтобы полученный n-разрядный вектор принадлежал коду. Для этого необходимо представить полином $x^r u(x)$ как

$$x^r \cdot u(x) = a(x) \cdot g(x) + b(x), \tag{1.10}$$

где $b(x)$ — остаток от деления $x^r u(x)$ на $g(x)$ [1].

Из (1.10) следует, что

$$x^r \cdot u(x) + b(x) = a(x) \cdot g(x). \tag{1.11}$$

Таким образом можно составить алгоритм кодирования систематического циклического (n,k)-кода.

1. Умножить информационный полином $u(x)$ (степень $\leq (k-1)$) на x^r.
2. Найти остаток $b(x)$ (степень $\leq (r-1)$) от деления $x^r u(x)$ на $g(x)$.
3. Сложить $b(x)$ и $x^r u(x)$, в результате получив кодовый полином $v(x)$ (степень $\leq (n-1)$).

Из (1.11) следует, что полученный многочлен $v(x)$

$$v(x) = b(x) + x^r \cdot u(x) = \underbrace{b_0 + b_1 x + \ldots + b_{r-1} x^{r-1}}_{r \text{ проверочных символов}} + \\ + \underbrace{u_0 x^r + u_1 x^{r+1} + \ldots + u_{k-1} x^{n-1}}_{k \text{ информационных символов}} \tag{1.12}$$

является кодовым так как делится на $g(x)$ без остатка, а сам код является систематическим, поскольку из (1.12) видно, что старшие k элементов кодового вектора являются информационным вектором (рис. 1.2) [1].

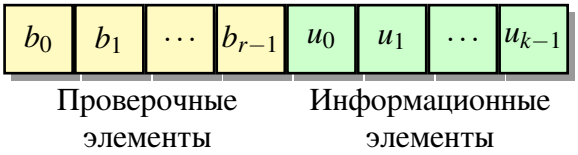

Рис. 1.2. Кодовое слово систематического циклического кода в регистре сдвига

Для примера, как и в случае несистематического кода, рассмотрим кодирование вектора $\mathbf{u} = \{1\ 0\ 1\ 0\}$ для систематического циклического кода $(7,4)$, образованного полиномом $g(x) = 1 + x + x^3$. Вектор $\mathbf{u}$ представим в виде полинома

$$u(x) = 1 + x^2.$$

Умножаем информационный полином на x^3

$$x^3 u(x) = x^3 + x^5.$$

Определяем остаток от деления $x^3 u(x)$ на $g(x)$

$$x^3 u(x) = (x^2) g(x) + \underbrace{x^2}_{b(x)}.$$

Получаем кодовый полином $v(x)$ и соответствующий кодовый вектор $\mathbf{v}$:

$$v(x) = x^3 u(x) + b(x) = x^2 + x^3 + x^5 \longrightarrow \mathbf{v} = \{0\ 0\ 1\ 1\ 0\ 1\ 0\}.$$

По аналогии с рис. 1.2 его можно представить в виде регистра (рис. 1.3).

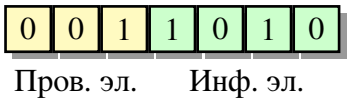

Рис. 1.3. Пример кодового слова систематического циклического кода $(7,4)$ *в регистре сдвига*

1.2.4. Порождающая матрица

Из свойств порождающего полинома циклического кода следует, что каждый кодовый многочлен может быть представлен произведением

$$v(x) = u(x)g(x) = u_0 g(x) + u_1 x g(x) + \ldots + u_{k-1} x^{k-1} g(x). \tag{1.13}$$

Каждое слагаемое в (1.13) содержит сдвиг порождающего полинома $g(x)$, следовательно, кодовый вектор $\mathbf{v}$, соответствующий полиному $v(x)$, можно представить как произведение информационного вектора $\mathbf{u}$ на порождающую матрицу $\mathbf{G}$:

$$\mathbf{v}_{1\times n} = \mathbf{u}_{1\times k} \cdot \mathbf{G}_{k\times n}. \tag{1.14}$$

Порождающая матрица несистематического кода при этом имеет вид

$$\mathbf{G}_{k\times n} = \begin{bmatrix} 1 & g_1 & \ldots & g_{r-1} & g_r & 0 & 0 & \ldots & 0 \\ 0 & 1 & \ldots & g_{r-2} & g_{r-1} & g_r & 0 & \ldots & 0 \\ 0 & 0 & \ldots & g_{r-3} & g_{r-2} & g_{r-1} & g_r & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & 0 \\ 0 & 0 & \ldots & 0 & 1 & g_1 & \ldots & g_{r-1} & g_r \end{bmatrix}. \tag{1.15}$$

Для примера приведем порождающую матрицу несистематического циклического кода $(7,4)$ с порождающим многочленом $g(x) = 1 + x + x^3$:

$$\mathbf{G}_{4\times 7} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}. \tag{1.16}$$

Путем элементарных матричных преобразований матрицу (1.16) можно привести к систематическому виду:

$$\mathbf{G}^{syst}_{4\times 7} = (\mathbf{P}_{4\times 3}\ \mathbf{E}_4) = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{1.17}$$

где $\mathbf{E}_i$ — единичная матрица размера $i \times i$.

Кодовые векторы (полиномы), образующие порождающую матрицу систематического циклического кода, называются *базисными*.

1.2.5. Проверочный полином и проверочная матрица

Для порождающего многочлена $g(x)$ степени $r = n - k$ циклического (n,k)-кода C существует *проверочный многочлен* $h(x)$ степени k такой, что $x^n + 1 = g(x)h(x)$ [1]:

$$h(x) = h_0 + h_1 x + \ldots + h_k x^k. \tag{1.18}$$

Многочлен взаимообратный $h(x)$

$$x^k \cdot h(x^{-1}) = h_k + h_{k-1} x + \ldots + h_0 x^k \tag{1.19}$$

является порождающим многочленом $(n, n-k)$-кода C_d, дуального коду C.

На основе проверочного полинома $h(x)$ строится проверочная матрица $\mathbf{H}_{r\times n}$ несистематического кода:

$$\mathbf{H}_{r\times n} = \begin{bmatrix} h_k & h_{k-1} & \ldots & h_1 & h_0 & 0 & 0 & \ldots & 0 \\ 0 & h_k & \ldots & h_2 & h_1 & h_0 & 0 & \ldots & 0 \\ 0 & 0 & \ldots & h_3 & h_2 & h_1 & h_0 & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & 0 \\ 0 & 0 & \ldots & 0 & h_k & h_{k-1} & \ldots & h_1 & h_0 \end{bmatrix}, \tag{1.20}$$

на основе которой строится уравнение для синдромного декодирования:

$$\mathbf{v} \cdot \mathbf{H}^T = 0. \tag{1.21}$$

Проверочную матрицу $\mathbf{H}^{syst}_{(n-k)\times n}$ систематического циклического кода можно построить из порождающей матрицы $\mathbf{G}^{syst}_{k\times n}$ по рассмотренным ранее правилам:

$$\mathbf{H}_{r\times n} = (\mathbf{E}_r \; \mathbf{P}^T_{k\times r}). \tag{1.22}$$

Для примера рассмотрим циклический код $(7,4)$ с порождающим многочленом $g(x) = 1 + x + x^3$.

Проверочный полином этого кода рассчитывается следующим образом:

$$h(x) = \frac{x^n + 1}{g(x)} = \frac{x^7 + 1}{x^3 + x + 1} = x^4 + x^2 + x + 1. \tag{1.23}$$

Проверочная матрица соответствующего несистематического кода в соответствии с (1.20) имеет вид:

$$\mathbf{H}_{3\times 7} = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}, \tag{1.24}$$

а проверочная матрица систематического кода:

$$\mathbf{H}_{3\times 7}^{syst} = (\mathbf{E}_3\ \mathbf{P}_{4\times 3}^T) = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}. \tag{1.25}$$

Проверочная матрица позволяет определить минимальное кодовое расстояние $d_{\min}$. Оно равно минимальному числу столбцов матрицы $\mathbf{H}$, построчная сумма по mod 2 элементов которых дает нулевой столбец [2]. В рассматриваемом примере для получения нулевого столбца необходимо сложить минимум три столбца матрицы $\mathbf{H}_{3\times 7}$ или матрицы $\mathbf{H}_{3\times 7}^{syst}$, следовательно этот код имеет $d_{\min} = 3$ и, соответственно, может гарантированно исправить $t = 1$ ошибку.

1.2.6. Синдромное декодирование циклического кода

При передаче по каналу связи к кодовому полиному $v(x)$ добавляется полином ошибок $e(x)$, в результате чего полином полученного на приеме слова имеет вид

$$r(x) = v(x) + e(x) \tag{1.26}$$

или

$$r(x) = a(x)g(x) + s(x), \tag{1.27}$$

где $s(x)$ — *синдром* ошибки. Синдром кодового слова равен нулю. Обнаружение и исправление ошибок в систематических кодах может производиться на основе анализа синдрома [1].

Как видно из формулы (1.27), синдром $s(x)$ равен остатку от деления принятого слова $r(x)$ на образующий полином $g(x)$.

Для обнаружения ошибки достаточно посчитать синдром принятого слова. Если он является ненулевым, то принятое слово содержит ошибки.

Схема вычисления синдрома представлена на рис. 1.4.

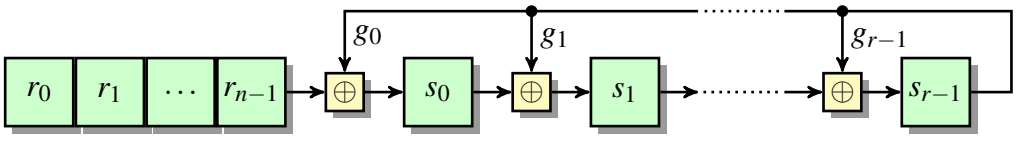

Рис. 1.4. Схема вычисления синдрома для циклического (n,k)-кода

Для исправления ошибок используется тот факт, что каждый вектор ошибки (исправляемой кодом) имеет свой синдром, который может быть получен путем деления полинома ошибки $e(x)$ на порождающий полином кода $g(x)$. При этом синдром не зависит от переданной кодовой комбинации.

На основании приведенного свойства существует метод определения места ошибки по синдрому s_{n-1}, соответствующему наличию ошибки в старшем разряде. Если ошибка произошла в следующем разряде (более низком), то такой же остаток получится в произведении принятого многочлена $r(x)$ на одночлен x [1].

Рассмотрим принцип декодирования на примере систематического кода $(7,4)$, образованного полиномом $g(x) = 1 + x + x^3 = \{1\ 1\ 0\ 1\}$. Пусть передавалась кодовая комбинация $\mathbf{v} = \{0\ 0\ 1\ 1\ 0\ 1\ 0\}$, соответствующая полиному $v(x) = x^2 + x^3 + x^5$. Первые три разряда проверочные. Вектор ошибки $\mathbf{e} = \{0\ 0\ 0\ 1\ 0\ 0\ 0\}$, т. е. ошибка в четвертом слева разряде. Следовательно на вход декодера принята кодовая комбинация $\mathbf{r} = \{0\ 0\ 1\ 0\ 0\ 1\ 0\}$.

Определяем синдром ошибки в крайнем правом разряде как остаток от деления вектора ошибки $\mathbf{e}_6 = \{0\ 0\ 0\ 0\ 0\ 0\ 1\}$ на вектор образующего полинома. В результате получается синдром

$$s_6 = 1. \tag{1.28}$$

Теперь делим принятый вектор $\mathbf{r}$ на вектор образующего полинома до получения остатка $s_6 = 1$ (рис. 1.5). По необходимости приписываем справа нули. Поскольку результат деления нас не интересует, не указываем его.

Из процедуры деления видно, что для получения остатка $s_6 = 1$ потребовалось дописать четыре нуля. Следовательно, ошибка находится на четвертой позиции слева.

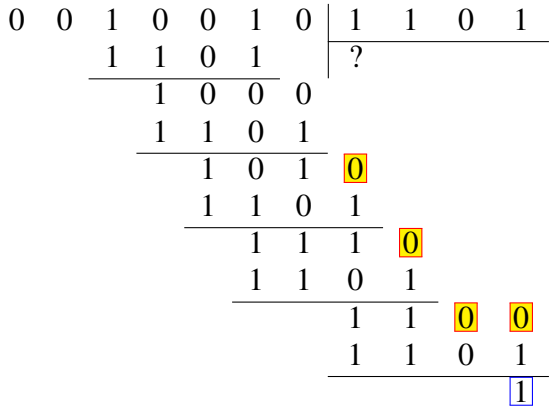

Рис. 1.5. Деление вектора $\mathbf{r}$ *на вектор* $\mathbf{g}$ *образующего полинома*

При такой процедуре декодирования возможны ещё два варианта.

Если при нахождении остатка он сразу получается равным s_{n-1} (s_6 для нашего примера), то ошибка находится в крайнем справа разряде и мы можем сразу ее исправить. Если же при нахождении остатка нам пришлось добавить n нулей и остаток s_{n-1} так и не был обнаружен, то такая ошибка не может быть исправлена. В этом случае декодер должен отметить принятое кодовое

слово как неисправляемое. В случае системы с обратной связью может быть осуществлен перезапрос переданных данных.

Для случая обратной записи кодовых слов и образующего многочлена процедура декодирования остается той же, но меняется положение ошибки в зависимости от числа дописанных нулей.

1.2.7. Оптимальное декодирование на основе анализа веса

Для нахождения ошибочных элементов в циклических кодах с $d_{\min} > 5$ получили распространение методы, основанные на анализе веса остатка. При этом осуществляются следующие процедуры.

1. Принятая кодовая комбинация делится на $g(x)$.

2. Подсчитывается вес остатка от деления ω (т. е. количество единиц в остатке).

3. Если $\omega \le t$, где t — кратность гарантированно исправляемой ошибки, то исправление сводится к сложению принятой кодовой комбинации с остатком от деления.

4. Если $\omega > t$, то производят циклический сдвиг принятой кодовой комбинации вправо на один разряд, а затем делят ее на $g(x)$ и определяют вес остатка. Если $\omega \le t$, то делимое суммируют с остатком, а затем производят циклический сдвиг на один элемент влево. Это и будет исправленная кодовая комбинация.

5. Если после первого сдвига остаток дает $\omega > t$, то процедуру повторяют до тех пор, пока не будет удовлетворяться условие $\omega \le t$. Исправленная комбинация получается в результате сдвига влево суммы последней кодовой комбинации и остатка на столько разрядов, на сколько была сдвинута исходная кодовая комбинация вправо.

Для случая обратной записи кодовых слов и образующего многочлена процедура декодирования остается той же, но сдвиг производится вначале влево, а после исправления ошибки — вправо.

Рассмотрим принцип оптимального декодирования также на примере систематического кода $(7,4)$ с $d_{\min} = 3$ $(t = 1)$, образующий полином $g(x) = 1 + x + x^3 = \{1\ 1\ 0\ 1\}$. Пусть передавалась кодовая комбинация $\mathbf{v} = \{0\ 0\ 1\ 1\ 0\ 1\ 0\}$. Первые три разряда проверочные. Вектор ошибки $\mathbf{e} = \{0\ 1\ 0\ 0\ 0\ 0\ 0\}$, т. е. ошибка во втором слева разряде, следовательно на вход декодера принята кодовая комбинация $\mathbf{r} = \{0\ 1\ 1\ 1\ 0\ 1\ 0\}$.

1. Ищем остаток от деления $\mathbf{r}$ на $\mathbf{g}$:

$$\begin{array}{ccccccc|cccc} 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ \cline{6-11} & & & & & 1 & 1 & ? & & & \end{array}$$

Вес остатка $\omega = 2 > t$.

2. Сдвигаем **r** на разряд вправо $\{0\ 1\ 1\ 1\ 0\ 1\ 0\} \to \{0\ 0\ 1\ 1\ 1\ 0\ 1\}$ и находим остаток:

$$\begin{array}{cccc|ccc|cccc} 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ \hline & & & & 1 & 1 & 1 & ? & & & \end{array}$$

Вес остатка $\omega = 3 > t$.

3. Сдвигаем на разряд вправо $\{0\,0\,1\,1\,1\,0\,1\} \to \{1\,0\,0\,1\,1\,1\,0\}$ и находим остаток:

$$\begin{array}{cccc|ccc|cccc} 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ \hline & & & & 1 & 0 & 1 & ? & & & \end{array}$$

Вес остатка $\omega = 2 > t$.

4. Сдвигаем на разряд вправо $\{1\,0\,0\,1\,1\,1\,0\} \to \{0\,1\,0\,0\,1\,1\,1\}$ и находим остаток:

$$\begin{array}{cccc|ccc|cccc} 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ \hline & & & & 1 & 0 & 0 & ? & & & \end{array}$$

Вес остатка $\omega = 1 \leq t$, следовательно, суммируем делимое с остатком:

$$\begin{array}{cc} & \begin{array}{ccccccc} 0 & 1 & 0 & 0 & 1 & 1 & 1 \end{array} \\ \oplus & \begin{array}{ccccccc} & & & & 1 & 0 & 0 \end{array} \\ \hline & \begin{array}{ccccccc} 0 & 1 & 0 & 0 & 0 & 1 & 1 \end{array} \end{array}$$

5. Поскольку мы трижды сдвигали исходный вектор вправо, сдвигаем его на три разряда влево:

$$\{0\,1\,0\,0\,0\,1\,1\} \to \{1\,0\,0\,0\,1\,1\,0\} \to \{0\,0\,0\,1\,1\,0\,1\} \to \{0\,0\,1\,1\,0\,1\,0\}.$$

В результате получен исправленный кодовый вектор.

1.3. Порядок выполнения задания

Задание выполняется каждым учащимся индивидуально. Вариант задания выбирается согласно номеру студента в журнале группы.

Все расчеты должны быть расписаны максимально подробно.

1. По заданному для циклического (n,k)-кода ($n = 15$) порождающему многочлену $g(x)$ (табл. 1.1) определить:

а) количество информационных элементов k;

б) проверочный многочлен $h(x)$;

в) порождающую и проверочную матрицы несистематического кода: $G^{ns}_{(15,k)}$ и $H^{ns}_{(15,k)}$, соответственно;

г) порождающую и проверочную матрицы систематического кода: $G^{syst}_{(15,k)}$ и $H^{syst}_{(15,k)}$, соответственно.

2. Выбрать информационный полином $u(x)$ из табл. 1.2 согласно номеру своего варианта. Преобразовать выбранный полином в двоичный вектор **u** (младший бит слева), дополнив справа нулями до нужного k.

Пример:

$$\left\{ \begin{array}{l} k = 9, \\ u(x) = x + x^2 + x^4 + x^5 + x^6 \end{array} \right\} \longrightarrow \mathbf{u} = \{0\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 0\}.$$

3. Закодировать полученный в предыдущем пункте информационный полином $u(x)$ несистематическим циклическим кодом. Вначале использовать порождающий полином, затем порождающую матрицу $G^{ns}_{(15,k)}$. Результаты сравнить.

4. Закодировать полученный в предыдущем пункте информационный полином $u(x)$ систематическим циклическим кодом. Вначале использовать порождающий полином, затем порождающую матрицу $G^{syst}_{(15,k)}$. Результаты сравнить.

5. Наложить на полученную в предыдущем пункте кодовую комбинацию $v(x)$ *систематического* кода полином ошибки $e(x)$ (табл. 1.3) и декодировать полученную комбинацию $r(x)$ вначале с использованием синдрома ошибки в крайнем правом разряде, а затем методом оптимального декодирования на основе анализа веса. Результаты сравнить.

Таблица 1.1

Порождающий полином кода
(выбирается согласно номеру студента в журнале)

№	Порождающий полином $g(x)$
1, 11, 21	$1 + x^3 + x^4 + x^5 + x^6$
2, 12, 22	$1 + x + x^4$
3, 13, 23	$1 + x^4 + x^6 + x^7 + x^8$
4, 14, 24	$1 + x^3 + x^4$
5, 15, 25	$1 + x + x^2 + x^4 + x^8$
6, 16, 26	$1 + x + x^2 + x^3 + x^6$
7, 17, 27	$1 + x + x^4$
8, 18, 28	$1 + x + x^3 + x^4 + x^5 + x^7 + x^8$
9, 19, 29	$1 + x^2 + x^3 + x^4 + x^6$
10, 20, 30	$1 + x + x^2 + x^4 + x^8$

Таблица 1.2

Информационный полином
(выбирается согласно номеру студента в журнале)

№	Информационный полином $u(x)$	№	Информационный полином $u(x)$
1	$1+x+x^2+x^6$	2	$x+x^4+x^5+x^6$
3	$1+x+x^3+x^6$	4	$1+x^2+x^3+x^5$
5	$1+x+x^4+x^6$	6	$1+x^2+x^4+x^5$
7	$1+x+x^5+x^6$	8	$1+x^3+x^4+x^5$
9	$1+x^2+x^3+x^6$	10	$x^2+x^3+x^4+x^5$
11	$1+x^2+x^4+x^6$	12	$x^2+x^3+x^4+x^6$
13	$1+x^2+x^5+x^6$	14	$x^2+x^3+x^5+x^6$
15	$1+x^3+x^4+x^6$	16	$x^2+x^4+x^5+x^6$
17	$1+x^3+x^5+x^6$	18	$x^2+x^3+x^4+x^5$
19	$1+x^4+x^5+x^6$	20	$x+x^2+x^3+x^5$
21	$x+x^2+x^3+x^6$	22	$x+x^2+x^4+x^5$
23	$x+x^2+x^4+x^6$	24	$x+x^3+x^4+x^5$
25	$x+x^2+x^5+x^6$	26	$x^3+x^4+x^5+x^6$
27	$x+x^3+x^4+x^6$	28	$1+x^2+x^3+x^4$
29	$x+x^3+x^5+x^6$	30	$x+x^2+x^3+x^4$

Таблица 1.3

Полином ошибки
(выбирается согласно номеру студента в журнале)

№	Полином ошибки $e(x)$
1, 6, 11, 16, 21, 26	x^3
2, 7, 12, 17, 22, 27	x^4
3, 8, 13, 18, 23, 28	x^5
4, 9, 14, 19, 24, 29	x^6
5, 10, 15, 20, 25, 30	x^7

1.4. Порядок защиты практической работы

Защита работы может осуществляться одним из нижеперечисленных способов или их сочетанием на усмотрение преподавателя.

1. Устный ответ по теме работы.
2. Тестирование по теме работы
3. Задача по теме работы.
4. Иные варианты на усмотрение преподавателя.

Контрольные вопросы

1. Что такое циклические коды?
2. Кодирование несистематическим циклическим кодом.
3. Кодирование систематическим циклическим кодом.
4. Декодирование циклических кодов.

Практическая работа 2
Коды Боуза–Чоудхури–Хоквингема

2.1. Цель работы

Рассмотреть на примере и получить навыки в исследовании циклических кодов Боуза–Чоудхури–Хоквингема (БЧХ).

2.2. Теоретическая справка

Корректирующие свойства циклических кодов могут быть определены на основе двух теорем [3].

Теорема 1. Для любых m и t существует циклический (n,k)-код длиной $n = 2^m - 1$, с кратностью гарантированно исправляемой ошибки t и числом проверочных разрядов $n-k = r$ не более mt [3].

Аналогично, если задана длина кода n, то можно определить необходимое для обеспечения требуемой корректирующей способности число проверочных разрядов.

Например, при заданных $n = 31$ и $m = 5$ для различных t можно получить

$$t = 1,\ r = n-k \le mt = 5\cdot 1 = 5 \longrightarrow \text{Код } (31{,}26);$$
$$t = 2,\ r = n-k \le mt = 5\cdot 2 = 10 \longrightarrow \text{Код } (31{,}21);$$
$$t = 3,\ r = n-k \le mt = 5\cdot 3 = 15 \longrightarrow \text{Код } (31{,}16).$$

Теорема 2. Эта теорема позволяет определить порождающий полином для кода с заданными параметрами. Она часто называется *теоремой БЧХ*, поскольку она была доказана Хоквингемом в 1959 г. и независимо от него Боузом и Чоудхури в 1960 г. [3].

Если среди корней порождающего полинома $g(x)$ циклического кода длиной $n = 2^m - 1$ содержится $d_0 - 1$ последовательных степеней ε^i, ε^{i+1}, …, $\varepsilon^{i+d_{\min}-2}$, то кодовое расстояние $d_{\min} \ge d_0$ [3].

Согласно этому принципу строятся коды Боуза–Чоудхури–Хоквингема (коды БЧХ).

Перед рассмотрением кодов БЧХ примем ряд обозначений. Пусть ε — примитивный элемент поля $\mathrm{GF}(2^m)$, а $m_i(x)$ — минимальный многочлен элемента ε^i [3].

Примитивным кодом БЧХ называется циклический код, порождающий полином $g(x)$ которого равен наименьшему общему кратному минимальных многочленов $m_1(x)$, …, $m_{2t}(x)$:

$$g(x) = \mathrm{НОК}[m_1(x),\ldots,m_{2t}(x)].$$

В таком случае элементы

$$\varepsilon, \varepsilon^2, \ldots, \varepsilon^{2t} \tag{2.1}$$

являются корнями $g(x)$. То есть, $g(\varepsilon^j = 0)$ для $j = 1,2,\ldots,2t$. Тогда по теореме БЧХ кодовое расстояние этого кода $d_{\min} \geq (2t+1)$ [3].

Таким образом, для любого числа $n = 2^m - 1$ и любого $t < 2^{m-1}$ определен примитивный двоичный код БЧХ длиной n, с кодовым расстоянием $d_{\min} \geq (2t+1)$ и числом проверочных символов меньшим или равным mt [3].

Нижнюю границу для минимального кодового расстояния можно увеличить до $2t+2$. Для этого в число корней порождающего многочлена нужно включить единицу, минимальный многочлен для которой равен $x+1$. Таким образом, код БЧХ с порождающим многочленом $g_1(x) = (x-1)g(x)$ имеет кодовое расстояние $d_{\min} \geq (2t+2)$ [3].

Для примера построим порождающий многочлен для кода БЧХ, исправляющего две ошибки $(t = 2)$, с $n = 31$. Из первой теоремы следует, что этот код построен над полем Галуа $\mathrm{GF}(2^5)$ $(m = 5)$ и имеет $r = mt = 5 \cdot 2 = 10$ проверочных символов, откуда следует, что это код $(31,21)$. Будем считать, что образующий полином поля Галуа равен $p(x) = x^5 + x^2 + 1$.

Из теоремы БЧХ следует, что порождающий полином $g_{(31,21)}(x)$ этого кода должен иметь корни ε, ε^2, ε^3, ε^4. С другой стороны, из свойств циклического кода следует, что полином $g_{(31,21)}(x)$ должен быть делителем многочлена $x^n + 1 = x^{31} + 1$. Теперь необходимо определить минимальные многочлены, соответствующие корням полинома $g_{(31,21)}(x)$. Для этого проведем разложение полинома $(x^{31}+1)$ на множители, которые и будут являться минимальными многочленами. Вначале требуется определить корни каждого из этих минимальных многочленов. Механизм разложения представлен формулой:

$$x^{31}+1 = \underbrace{m_1(x)}_{\substack{\varepsilon \\ \varepsilon^2 \\ \varepsilon^4 \\ \varepsilon^8 \\ \varepsilon^{16}}} \cdot \underbrace{m_2(x)}_{\substack{\varepsilon^3 \\ \varepsilon^6 \\ \varepsilon^{12} \\ \varepsilon^{24} \\ \varepsilon^{48} = \varepsilon^{17}}} \cdot \underbrace{m_3(x)}_{\substack{\varepsilon^5 \\ \varepsilon^{10} \\ \varepsilon^{20} \\ \varepsilon^{40} = \varepsilon^9 \\ \varepsilon^{18}}} \cdot \underbrace{m_4(x)}_{\substack{\varepsilon^7 \\ \varepsilon^{14} \\ \varepsilon^{28} \\ \varepsilon^{56} = \varepsilon^{25} \\ \varepsilon^{50} = \varepsilon^{19}}} \cdot \underbrace{m_5(x)}_{\substack{\varepsilon^{11} \\ \varepsilon^{22} \\ \varepsilon^{44} = \varepsilon^{13} \\ \varepsilon^{26} \\ \varepsilon^{52} = \varepsilon^{21}}} \cdot \underbrace{m_6(x)}_{\substack{\varepsilon^{15} \\ \varepsilon^{30} \\ \varepsilon^{60} = \varepsilon^{29} \\ \varepsilon^{58} = \varepsilon^{27} \\ \varepsilon^{54} = \varepsilon^{23}}} \cdot \underbrace{m_7(x)}_{\varepsilon^0} \tag{2.2}$$

Теперь рассчитаем эти многочлены:

$$\begin{aligned}
&m_1(x) = (x+\varepsilon)(x+\varepsilon^2)(x+\varepsilon^4)(x+\varepsilon^8)(x+\varepsilon^{16}) = 1+x^2+x^5;\\
&m_2(x) = (x+\varepsilon^3)(x+\varepsilon^6)(x+\varepsilon^{12})(x+\varepsilon^{24})(x+\varepsilon^{17}) = 1+x^2+x^3+x^4+x^5;\\
&m_3(x) = (x+\varepsilon^5)(x+\varepsilon^{10})(x+\varepsilon^{20})(x+\varepsilon^9)(x+\varepsilon^{18}) = 1+x+x^2+x^4+x^5;\\
&m_4(x) = (x+\varepsilon^7)(x+\varepsilon^{14})(x+\varepsilon^{28})(x+\varepsilon^{25})(x+\varepsilon^{19}) = 1+x+x^2+x^3+x^5;\\
&m_5(x) = (x+\varepsilon^{11})(x+\varepsilon^{22})(x+\varepsilon^{13})(x+\varepsilon^{26})(x+\varepsilon^{21}) = 1+x+x^3+x^4+x^5;\\
&m_6(x) = (x+\varepsilon^{15})(x+\varepsilon^{30})(x+\varepsilon^{29})(x+\varepsilon^{27})(x+\varepsilon^{23}) = 1+x^3+x^5;\\
&m_7(x) = (x+\varepsilon^0) = 1+x.
\end{aligned} \tag{2.3}$$

Из формул (2.2) и (2.3) видно, что корни ε, ε^2 и ε^4 принадлежат полиному $m_1(x) = 1 + x^2 + x^5$, а корень ε^3 — полиному $m_2(x) = 1 + x^2 + x^3 + x^4 + x^5$, следовательно, образующий полином рассматриваемого кода БЧХ $(31,21)$ равен наименьшему общему кратному этих многочленов, а поскольку они являются различными неприводимыми полиномами, их наименьшее общее кратное (НОК) равно их произведению[1] [3]:

$$\begin{aligned} g_{(31,21)}(x) = m_1(x)m_2(x) = (1 + x^2 + x^5)(1 + x^2 + x^3 + x^4 + x^5) = \\ = 1 + x^3 + x^5 + x^6 + x^8 + x^9 + x^{10}. \end{aligned} \tag{2.4}$$

2.2.1. Кодирование кода БЧХ

Механизм кодирования кодом БЧХ полностью аналогичен кодированию обычным циклическим кодом. В случае несистематического кода используется умножение на образующий полином, а в случае систематического кода — поиск остатка от деления $x^r u(x)$, где $u(x)$ — информационный полином, на образующий полином $g(x)$.

2.2.2. Декодирование кода БЧХ

Основным алгоритмом декодирования кодов БЧХ является так называемый *алгебраический* или *синдромный* метод декодирования. В основе этого метода лежит использование элементов поля Галуа для нумерации позиций элементов принятого кодового слова $r(x)$ [4]:

$$\begin{array}{rcccc} \text{Значения:} & [r_0 & r_1 & \ldots & r_{n-1}] \\ \text{Локаторы позиций:} & 1 & \varepsilon & \ldots & \varepsilon^{n-1} \end{array} \tag{2.5}$$

Для определения позиций ошибок необходимо решить систему уравнений над полем Галуа $\mathrm{GF}(2^m)$, использованным для построения кода [4]. Далее рассмотрим, как получается эта система уравнений.

Возьмем полином ошибок $e(x)$, представленный как

$$e(x) = e_{j_1} x^{j_1} + e_{j_2} x^{j_2} + \ldots + e_{j_q} x^{j_q}, \tag{2.6}$$

где $q \leq t$ — число ошибок в принятом слове $r(x)$ [4].

Множество

$$\{e_{j_1}, e_{j_2}, \ldots, e_{j_q}\}, \quad e_j \in \{0,1\} \tag{2.7}$$

называется множеством *значений ошибок*, а соответствующее ему множество

$$\{\varepsilon^{j_1}, \varepsilon^{j_2}, \ldots, \varepsilon^{j_q}\}, \quad \varepsilon^j \in \mathrm{GF}(2^m) \tag{2.8}$$

[1] Аналогично тому, как НОК двух простых чисел равно их произведению [3].

называется множеством *локаторов ошибок* [4].

Синдромы при декодировании кода БЧХ определяются как значения принятого полинома $r(x)$ в *нулях кода*, под которыми понимаются $2t$ корней с последовательными степенями (2.1) [4]:

$$\begin{aligned} &S_1 = r(\varepsilon) \equiv e_{j_1}(\varepsilon)^{j_1} + e_{j_2}(\varepsilon)^{j_2} + \ldots + e_{j_q}(\varepsilon)^{j_q}, \\ &S_2 = r(\varepsilon^2) \equiv e_{j_1}(\varepsilon^2)^{j_1} + e_{j_2}(\varepsilon^2)^{j_2} + \ldots + e_{j_q}(\varepsilon^2)^{j_q}, \\ &\ldots \\ &S_{2t} = r(\varepsilon^{2t}) \equiv e_{j_1}(\varepsilon^{2t})^{j_1} + e_{j_2}(\varepsilon^{2t})^{j_2} + \ldots + e_{j_q}(\varepsilon^{2t})^{j_q}. \end{aligned} \tag{2.9}$$

Необходимо отметить, что для двоичных кодов верно следующее равенство:

$$S_{2i} = S_i^2. \tag{2.10}$$

Далее вводится *полином локаторов ошибок* $\sigma(x)$:

$$\sigma(x) = \prod_{l=1}^{q}(1 + \varepsilon^{j_l}x) = 1 + \sigma_1 x + \sigma_2 x^2 + \ldots + \sigma_q x^q, \tag{2.11}$$

корни которого равны обратным величинам локаторов ошибок (2.8) [4].

Из приведенных формул следует *ключевое уравнение*, которое показано в формуле (2.12) в матричном виде [4]:

$$\begin{bmatrix} S_{q+1} \\ S_{q+2} \\ \vdots \\ S_{2q} \end{bmatrix} = \begin{bmatrix} S_1 & S_2 & \cdots & S_q \\ S_2 & S_3 & \cdots & S_{q+1} \\ \vdots & \vdots & \ddots & \vdots \\ S_q & S_{q+1} & \cdots & S_{2q-1} \end{bmatrix} \begin{bmatrix} \sigma_q \\ \sigma_{q-1} \\ \vdots \\ \sigma_1 \end{bmatrix}. \tag{2.12}$$

Известны три основных метода решения ключевого уравнения [4].

1. Алгоритм Берлекэмпа–Месси (BMA).
2. Алгоритм Евклида (EA).
3. Алгоритм Питерсона–Горенштейна–Цирлера (PGZ).

2.2.2.1. Общий алгоритм декодирования двоичных кодов БЧХ

Блок-схема декодера двоичных кодов БЧХ в общем виде показана на рис. 2.1. Каждый из блоков декодера реализует определенный шаг алгоритма декодирования [4].

1. Вычисление синдромов.
2. Определение коэффициентов полинома локаторов ошибок $\sigma(x)$, т. е. решение ключевого уравнения.
3. Нахождение позиций ошибок, как обратных значений корней $\sigma(x)$ по алгоритму Ченя.

4. Исправление кодового слова.

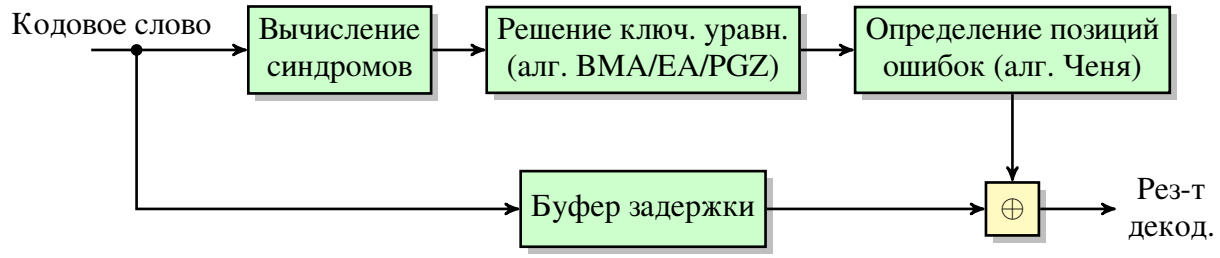

Рис. 2.1. Блок-схема декодера двоичных кодов БЧХ в общем виде

Далее рассмотрим работу декодера на каждом из этапов процедуры декодирования.

2.2.2.2. Вычисление синдромов

Поскольку формулы для вычисления синдромов (2.9) и (2.10) были приведены ранее, рассмотрим их вычисление на примере.

Для примера используем код БЧХ $(15,7)$ над полем $\mathrm{GF}(2^4)$ с образующим полиномом $g_{(15,7)}(x)$:

$$g_{(15,7)}(x) = 1 + x^4 + x^6 + x^7 + x^8 \longrightarrow \mathbf{g}_{(15,7)} = \{1\,0\,0\,0\,1\,0\,1\,1\,1\}. \tag{2.13}$$

Этот код имеет минимальное кодовое расстояние $d_{\min} = 5$ и, следовательно, гарантированно исправляет $t = 2$ ошибки.

Согласно теореме БЧХ образующий полином (2.13) будет иметь корни

$$\{\varepsilon, \varepsilon^2, \varepsilon^3, \varepsilon^4\}. \tag{2.14}$$

Выбранный для примера информационный полином $v_{(15,7)}(x)$ показан в формуле:

$$v_{(15,7)}(x) = x^2 + x^5 \longrightarrow \mathbf{v}_{(15,7)} = \{0\,0\,1\,0\,0\,1\,0\}. \tag{2.15}$$

После кодирования по систематическому алгоритму будет получено кодовое слово $u_{(15,7)}(x)$:

$$\begin{aligned} u_{(15,7)}(x) &= x + x^4 + x^7 + x^{10} + x^{13}, \\ \mathbf{u}_{(15,7)} &= \{0\,1\,0\,0\,1\,0\,0\,1\,0\,0\,1\,0\,0\,1\,0\}. \end{aligned} \tag{2.16}$$

Зададим полином ошибок $e_{(15,7)}(x)$:

$$\begin{aligned} e_{(15,7)}(x) &= x^3 + x^{10}, \\ \mathbf{e}_{(15,7)} &= \{0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\}. \end{aligned} \tag{2.17}$$

В таком случае на вход декодера будет принята комбинация $r_{(15,7)}(x)$:

$$\begin{aligned} &r_{(15,7)}(x) = x + x^3 + x^4 + x^7 + x^{13}, \\ &\mathbf{r}_{(15,7)} = \{0\,1\,0\,1\,1\,0\,0\,1\,0\,0\,0\,0\,0\,1\,0\}. \end{aligned} \tag{2.18}$$

Теперь можно вычислить синдромы по формулам (2.9) и (2.10):

$$\begin{aligned} &S_1 = r(\varepsilon) = \varepsilon + (\varepsilon)^3 + (\varepsilon)^4 + (\varepsilon)^7 + (\varepsilon)^{13} = \varepsilon^{12}, \\ &S_2 = S_1^2 = \varepsilon^9, \\ &S_3 = r(\varepsilon^3) = \varepsilon^3 + (\varepsilon^3)^3 + (\varepsilon^3)^4 + (\varepsilon^3)^7 + (\varepsilon^3)^{13} = \varepsilon^7, \\ &S_4 = S_2^2 = \varepsilon^3. \end{aligned} \tag{2.19}$$

2.2.2.3. Решение ключевого уравнения по алгоритму Берлекэмпа–Месси

Алгоритм Берлекэмпа–Месси (БМ) был предложен в 1968 г. Э. Берлекэмпом и применен к линейным кодам в 1969 г. Дж. Месси [5, 6, 7]. Алгоритм БМ часто рассматривается как итеративный процесс построения минимального линейного регистра сдвига с обратной связью (ЛРОС), который генерирует известную последовательность синдромов (2.9) [4, 8].

Основной целью алгоритма БМ является построение полинома $\sigma^{(i+1)}(x)$, удовлетворяющего уравнению (2.20), выводимому из (2.12):

$$\sum_{j=0}^{l_i+1} S_{k-j}\sigma_j^{(i+1)} = 0, \quad l_i < k < i+1. \tag{2.20}$$

Решение (2.20) эквивалентно тому, что полином

$$\sigma^{(i+1)}(x) = 1 + \sigma_1^{(i+1)}x + \ldots + \sigma_{l_{i+1}}^{(i+1)}x^{l_{i+1}} \tag{2.21}$$

является многочленом обратной связи ЛРОС, генерирующим последовательность синдромов (2.9) [4].

Для определения соответствия между синдромной последовательностью и генерируемой ЛРОС вводится так называемая *несовместность* (*невязка*), определяемая на i-й итерации как

$$d_i = S_{i+1} + S_i\sigma_1^{(i)} + \ldots + S_{i-l_i+1}\sigma_{l_i}^{(i)} \tag{2.22}$$

и содержащая корректирующий множитель для вычисления $\sigma^{(i+1)}$ на следующей итерации [4].

При вычислении значения невязки возможны два варианта:

1) $d_i = 0$. В этом случае из уравнения (2.20) получаем

$$\sigma^{(i+1)}(x) = \sigma^{(i)}(x), \quad l_{i+1} = l_i; \tag{2.23}$$

2) $d_i \neq 0$. В таком случае решение на следующей итерации имеет вид

$$\begin{aligned} &\sigma^{(i+1)}(x) = \sigma^{(i)}(x) + d_i d_m^{-1} x^{i-m} \sigma^{(m)}(x), \\ &l_{i+1} = \max\{l_i, l_m + i - m\}, \end{aligned} \tag{2.24}$$

где $\sigma^{(m)}(x)$ — решение на m-й итерации такое, что $-1 \leq m < i, d \neq 0$ и $(m - l_m)$ максимально [4].

Расчет $\sigma^{(i+1)}(x)$ начинается с $i = 0$ и продолжается пока не будут выполнены одно или оба условия [4]:

$$i \geq l_{i+1} + t - 1 \quad \text{или} \quad i = 2t - 1. \tag{2.25}$$

Начальные условия алгоритма [4]:

$$\begin{aligned} &\sigma^{(-1)}(x) = 1, \quad l_{-1} = 0, \quad d_{-1} = 1, \\ &\sigma^{(0)}(x) = 1, \quad l_0 = 0, \quad d_0 = S_1. \end{aligned} \tag{2.26}$$

Далее рассмотрим процесс решения ключевого уравнения на примере.

Итерация 0. Начальные условия.

$$\begin{aligned} &\sigma^{(-1)}(x) = 1, \quad l_{-1} = 0, \quad d_{-1} = 1, \\ &\sigma^{(0)}(x) = 1, \quad l_0 = 0, \quad d_0 = S_1 = \varepsilon^{12}. \end{aligned}$$

Итерация 1.

$i = 0, d_0 = \varepsilon^{12} \neq 0, m = -1 = \arg(\max(-1 + 0) = -1)$ для $d_{-1} \neq 0$.
$\sigma^{(1)}(x) = \sigma^{(0)}(x) + d_0 d_{-1}^{-1} x^{(0-(-1))} \sigma^{(-1)}(x) = 1 + \varepsilon^{12} x$,
$l_1 = \max\{l_0, l_{-1} + 0 - (-1)\} = 1$,
Проверяем условие: $l_1 + 2 - 1 \leq 0$?
Не выполняется:
$d_1 = S_2 + S_1 \sigma_1^{(1)} = \varepsilon^9 + \varepsilon^{12} \varepsilon^{12} = 0$.

Итерация 2.

$i = 1, d_1 = 0$,
$\sigma^{(2)}(x) = \sigma^{(1)}(x) = 1 + \varepsilon^{12} x$,
$l_2 = l_1 = 1$,
Проверяем условие: $l_2 + 2 - 1 \leq 1$?
Не выполняется:
$d_2 = S_3 + S_2 \sigma_1^{(1)} = \varepsilon^7 + \varepsilon^9 \varepsilon^{12} = \varepsilon^{10}$.

Итерация 3.

$i = 2, d_2 = \varepsilon^{10} \neq 0, m = 0 = \arg(\max(0-0) = 0)$ для $d_0 \neq 0$.
$\sigma^{(3)}(x) = \sigma^{(2)}(x) + d_2 d_0^{-1} x^{(2-0)} \sigma^{(0)}(x) =$
$= (1 + \varepsilon^{12} x) + \varepsilon^{10} \varepsilon^{-12} x^2 = 1 + \varepsilon^{12} x + \varepsilon^{13} x^2,$
$l_3 = \max\{l_2, l_0 + 2 - (0)\} = 2,$
Проверяем условие: $l_3 + 2 - 1 \leq 2$?
Не выполняется:
$d_3 = S_4 + S_3 \sigma_1^{(3)} + S_2 \sigma_2^{(3)} = \varepsilon^3 + \varepsilon^7 \varepsilon^{12} + \varepsilon^9 \varepsilon^{13} = 0.$

Итерация 4.

$i = 3, d_3 = 0,$
$\sigma^{(4)}(x) = \sigma^{(3)}(x) = 1 + \varepsilon^{12} x + \varepsilon^{13} x^2,$
$l_4 = l_3 = 2,$
Проверяем условие: $l_4 + 2 - 1 \leq 3$?
Выполняется: Конец.

Таким образом

$$\sigma^{bma}_{(15,7)}(x) = 1 + \varepsilon^{12} x + \varepsilon^{13} x^2. \tag{2.27}$$

Необходимо отметить, что тот факт, что на нечетных итерациях $d_i = 0$, является закономерностью для двоичных кодов БЧХ, что позволяет сократить алгоритм, выполняя только четные итерации и исправив правило остановки на соотношение:

$$i \geq l_{i+1} + t - 2 \quad \text{или} \quad i = 2t - 1. \tag{2.28}$$

2.2.2.4. Решение ключевого уравнения по алгоритму Питерсона–Горенштейна–Цирлера

Согласно этому алгоритму, для решения ключевого уравнения используется стандартный алгоритм решения системы линейных уравнений. При этом, поскольку неизвестно действительное число ошибок в принятом слове, приходится предварительно проверять гипотезу о том, что действительное количество ошибок равно q, начиная с максимально возможного числа ошибок [4].

Декодер начинает работу с предположения о том, что возникло максимальное число ошибок $q = t$. Вычисляется определитель Δ_q для матрицы синдромов

$$\Delta_q = \det \begin{bmatrix} S_1 & S_2 & \cdots & S_q \\ S_2 & S_3 & \cdots & S_{q+1} \\ \vdots & \vdots & \ddots & \vdots \\ S_q & S_{q+1} & \cdots & S_{2q-1} \end{bmatrix} \tag{2.29}$$

и сравнивается с нулем. Если $\Delta_q = 0$, то действительное число ошибок меньше, чем предполагалось. Тогда q уменьшается на 1 и снова проверяется определитель. Процедура при необходимости повторяется, пока q не станет равным 1. Как только оказывается, что $\Delta_q \neq 0$, вычисляется обратная матрица для матрицы синдромов и вычисляются значения $\sigma_1, \ldots, \sigma_q$. Если $\Delta_q = 0$ для всех q от 1 до t, декодирование считается безуспешным и регистрируется обнаружение неисправляемой комбинации ошибок [4].

Разберем процедуру на примере.

Предположим, что число ошибок $q = t = 2$. В этом случае матрица синдромов имеет вид:

$$\mathbf{S}_2 = \begin{bmatrix} S_1 & S_2 \\ S_2 & S_3 \end{bmatrix} = \begin{bmatrix} \varepsilon^{12} & \varepsilon^{9} \\ \varepsilon^{9} & \varepsilon^{7} \end{bmatrix}. \tag{2.30}$$

Определитель Δ_2 будет иметь вид:

$$\Delta_2 = \det(\mathbf{S}_2) = \begin{vmatrix} \varepsilon^{12} & \varepsilon^{9} \\ \varepsilon^{9} & \varepsilon^{7} \end{vmatrix} = \varepsilon^{12}\varepsilon^{7} + \varepsilon^{9}\varepsilon^{9} = \varepsilon^{7}. \tag{2.31}$$

Так как $\Delta_2 \neq 0$, подтверждается предположение о том, что в принятой комбинации две ошибки. Подставим полученные синдромы в ключевое уравнение (2.12) и получим

$$\begin{bmatrix} S_1 & S_2 \\ S_2 & S_3 \end{bmatrix} \begin{bmatrix} \sigma_2 \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} S_3 \\ S_4 \end{bmatrix} \Leftrightarrow \begin{bmatrix} \varepsilon^{12} & \varepsilon^{9} \\ \varepsilon^{9} & \varepsilon^{7} \end{bmatrix} \begin{bmatrix} \sigma_2 \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} \varepsilon^{7} \\ \varepsilon^{3} \end{bmatrix}. \tag{2.32}$$

Решая систему (2.32), получим

$$\begin{aligned} \sigma_2 &= \Delta_2^{-1} \cdot \begin{vmatrix} \varepsilon^{7} & \varepsilon^{9} \\ \varepsilon^{3} & \varepsilon^{7} \end{vmatrix} = (\varepsilon^{7})^{-1} \cdot (\varepsilon^{7}\varepsilon^{7} + \varepsilon^{9}\varepsilon^{3}) = \varepsilon^{13}, \\ \sigma_1 &= \Delta_2^{-1} \cdot \begin{vmatrix} \varepsilon^{12} & \varepsilon^{7} \\ \varepsilon^{9} & \varepsilon^{3} \end{vmatrix} = (\varepsilon^{7})^{-1} \cdot (\varepsilon^{12}\varepsilon^{3} + \varepsilon^{7}\varepsilon^{9}) = \varepsilon^{12}. \end{aligned} \tag{2.33}$$

Таким образом

$$\sigma^{pgz}_{(15,7)}(x) = 1 + \varepsilon^{12}x + \varepsilon^{13}x^2, \tag{2.34}$$

что совпадает с результатом (2.27), полученным по алгоритму Берлекэмпа–Месси.

2.2.2.5. *Решение ключевого уравнения по алгоритму Евклида*

В основе алгоритма Евклида лежит процедура нахождения наибольшего общего делителя двух полиномов [4].

Вводят *полином значений ошибок* как

$$\Lambda(x) = \sigma(x)S(x), \tag{2.35}$$

где *синдромный полином* $S(x)$ имеет вид

$$S(x) = 1 + S_1 x + \ldots + S_{2t} x^{2t}. \tag{2.36}$$

Из ключевого уравнения (2.12) следует, что

$$\Lambda(x) = \sigma(x)S(x) \ [\mathrm{mod}\ x^{2t+1}]. \tag{2.37}$$

Задача декодирования может быть переформулирована как задача определения полинома $\Lambda(x)$, удовлетворяющего уравнению (2.37). Для этого к полиномам $q_0(x) = x^{2t+1}$ и $q_1(x) = S(x)$ применяют расширенный алгоритм Евклида.

Если на j-м шаге алгоритма получено такое решение

$$q_j(x) = a_j(x)x^{2t+1} + b_j(x)S(x),$$

что степень $q_j(x)$ меньше либо равна t ($\deg[q_j(x)] \leq t$, то $\Lambda(x) = q_j(x)$ и $\sigma(x) = b_j(x)$ [4].

Алгоритм Евклида для решения ключевого уравнения [4].

1. Начальные значения:

$$\begin{array}{ll} q_0(x) = x^{2t+1}, & q_1(x) = S(x), \\ b_0(x) = 0, & b_1(x) = 1. \end{array}$$

2. На шаге $j = 2$ поделить $q_{j-2}(x)$ на $q_{j-1}(x)$:

$$q_{j-2}(x) = f_j(x)q_{j-1}(x) + q_j(x), \quad 0 \leq \deg[q_j(x)] < \deg[q_{j-1}(x)].$$

3. Вычислить

$$b_j(x) = b_{j-2}(x) + f_j(x)b_{j-1}(x).$$

4. Остановить вычисления при

$$\deg[q_j(x)] \leq t.$$

Полином локаторов ошибок при этом равен

$$\sigma(x) = b_j(x).$$

Далее разберем процедуру на примере.

Начальные условия.

$$q_0(x) = x^5, \quad q_1(x) = S(x) = 1 + \varepsilon^{12}x + \varepsilon^9 x^2 + \varepsilon^7 x^3 + \varepsilon^3 x^4,$$
$$b_0(x) = 0, \quad b_1(x) = 1.$$

Шаг $j = 2$.

$$x^5 = (1 + \varepsilon^{12}x + \varepsilon^9 x^2 + \varepsilon^7 x^3 + \varepsilon^3 x^4)(\varepsilon + \varepsilon^{12}x) + (1 + \varepsilon x + \varepsilon^{13}x^2 + \varepsilon^{14}x^3),$$
$$q_2(x) = 1 + \varepsilon x + \varepsilon^{13}x^2 + \varepsilon^{14}x^3,$$
$$f_2(x) = \varepsilon + \varepsilon^{12}x,$$
$$b_2(x) = b_0(x) + f_2(x)b_1(x) = 0 + (\varepsilon + \varepsilon^{12}x) \cdot 1 = \varepsilon + \varepsilon^{12}x,$$
$$\deg[q_2(x)] > t = 2.$$

Шаг $j = 3$.

$$1 + \varepsilon^{12}x + \varepsilon^9 x^2 + \varepsilon^7 x^3 + \varepsilon^3 x^4 = (1 + \varepsilon x + \varepsilon^{13}x^2 + \varepsilon^{14}x^3)(\varepsilon^{13} + \varepsilon^4 x) +$$
$$+ (\varepsilon^6 + \varepsilon^8 x + \varepsilon x^2),$$
$$q_3(x) = \varepsilon^6 + \varepsilon^8 x + \varepsilon x^2,$$
$$f_3(x) = \varepsilon^{13} + \varepsilon^4 x,$$
$$b_3(x) = b_1(x) + f_3(x)b_2(x) = 1 + (\varepsilon^{13} + \varepsilon^4 x)(\varepsilon + \varepsilon^{12}x) = \varepsilon^3 + x + \varepsilon x^2,$$
$$\deg[q_3(x)] \le t = 2 \longrightarrow \text{Завершение}.$$

Получаем полином локаторов ошибок:

$$\sigma^{ea}_{(15,7)}(x) = \varepsilon^3 + x + \varepsilon x^2 = \varepsilon^3(1 + \varepsilon^{12}x + \varepsilon^{13}x^2), \tag{2.38}$$

что с точностью до постоянного множителя, не влияющего на значения корней $\sigma_{(15,7)}(x)$, совпадает с результатом (2.27), полученным согласно алгоритмам Берлекэмпа–Месси и Питерсона–Горенштейна–Цирлера.

2.2.2.6. Определение позиций ошибок по алгоритму Ченя

Для поиска корней $\sigma(x)$ на множестве локаторов позиций кодовых символов используется метод проб и ошибок, получивший название *метод Ченя*. Согласно этому методу, для всех ненулевых элементов поля $\beta \in \mathrm{GF}(2^m)$, проверяется условие $\sigma(\beta^{-1}) = 0$. Далее производится исправление ошибок, что для двоичных кодов БЧХ сводится к сложению позиции с ошибкой с единицей по модулю 2 [4].

Продолжая пример, подставим обратные значения всех возможных элементов поля $\mathrm{GF}(2^4)$ в $\sigma(x)$ и получим, что корни его равны $\varepsilon^5 = \varepsilon^{-10}$ и $\varepsilon^{12} = \varepsilon^{-3}$, что показывает наличие ошибок в третьем и десятом разрядах.

Таким образом, можно восстановить полином ошибок

$$e(x) = x^3 + x^{10}$$

и исправить неверно принятую комбинацию.

2.3. Порядок выполнения задания

Задание выполняется каждым учащимся индивидуально. Вариант задания выбирается согласно номеру студента в журнале группы.

Все расчеты должны быть расписаны максимально подробно.

1. Построить по заданным параметрам (n,k)-код БЧХ, гарантированно исправляющий три ошибки ($t=3$). Код должен быть построен над полем Галуа $\mathrm{GF}(2^m)$, где $m=4$. Порождающий полином поля $p(x)=x^4+x+1$. Элементы поля представлены в табл. 2.1. Для построения кода необходимо:

а) определить n;

б) разбить полином x^n+1 на минимальные многочлены $m_j(x)$, определив корни каждого из минимальных многочленов;

в) определить образующий полином $g(x)$ кода БЧХ;

г) по степени образующего полинома определить k.

Таблица 2.1

Поле Галуа $\mathrm{GF}(2^4)$ *с образующим полиномом* $p(x)=x^4+x+1$.

Элемент поля	Полином	Двоичный вид $(1, \varepsilon, \varepsilon^2, \varepsilon^3)$	Десятичный вид
$\varepsilon^0=1$	1	1000	1
ε	x	0100	2
ε^2	x^2	0010	4
ε^3	x^3	0001	8
ε^4	$1+x$	1100	3
ε^5	$x+x^2$	0110	6
ε^6	x^2+x^3	0011	12
ε^7	$1+x+x^3$	1101	11
ε^8	$1+x^2$	1010	5
ε^9	$x+x^3$	0101	10
ε^{10}	$1+x+x^2$	1110	7
ε^{11}	$x+x^2+x^3$	0111	14
ε^{12}	$1+x+x^2+x^3$	1111	15
ε^{13}	$1+x^2+x^3$	1011	13
ε^{14}	$1+x^3$	1001	9

2. Согласно номеру в журнале выбрать из табл. 2.2 информационный полином $u(x)$ и полином ошибок $e(x)$.

3. Закодировать информационный полином $u(x)$ систематическим кодом БЧХ. Для кодирования использовать образующий полином $g(x)$, полученный в пункте 1.

4. Наложить на полученную в предыдущем пункте кодовую комбинацию $v(x)$ циклического систематического кода полином ошибки $e(x)$ и декодировать полученную комбинацию $r(x)$ синдромным методом декодирования. Задание выполнять в следующем порядке:

а) определить синдромы.

б) решить ключевое уравнение по методу Берлекэмпа–Месси;

в) решить ключевое уравнение по методу Питерсона–Горенштейна–Цирлера;

г) решить ключевое уравнение по методу Евклида;

д) определить позиции ошибок по методу Ченя;

е) исправить ошибки.

Таблица 2.2

Информационный полином $u(x)$ и полином ошибок $e(x)$ (выбираются согласно номеру студента в журнале)

№	$u(x)$	$e(x)$	№	$u(x)$	$e(x)$
1	$1+x$	x^2+x^5	2	$1+x^2$	x^3+x^6
3	$1+x^3$	x^4+x^7	4	$1+x^4$	x^5+x^8
5	$x+x^2$	x^6+x^9	6	$x+x^3$	x^7+x^{10}
7	$x+x^4$	x^8+x^{11}	8	x^2+x^3	x^9+x^{12}
9	x^3+x^4	$x^{10}+x^{12}$	10	$1+x+x^2$	x^2+x^6
11	$1+x+x^3$	x^3+x^7	12	$1+x+x^4$	x^4+x^8
13	$1+x^2+x^3$	x^5+x^9	14	$1+x^2+x^4$	x^6+x^{10}
15	$1+x^3+x^4$	x^7+x^{11}	16	$1+x+x^2+x^3$	x^8+x^{12}
17	$1+x+x^2+x^4$	x^2+x^8	18	$1+x+x^3+x^4$	x^3+x^9
19	$1+x^2+x^3+x^4$	x^1+x^9	20	$x+x^2+x^3$	x^2+x^{10}
21	$x+x^2+x^4$	x^3+x^{11}	22	$x+x^3+x^4$	x^4+x^{12}
23	$x+x^2+x^3+x^4$	x^2+x^8	24	$x^2+x^3+x^4$	x^3+x^9
25	x^2	x^4+x^{10}	26	x	x^5+x^{11}
27	x^3	x^6+x^{12}	28	x^4	x^2+x^9
29	1	x^3+x^{10}	30	$1+x+x^2+x^3+x^4$	x^4+x^{11}

2.4. Порядок защиты практической работы

Защита работы может осуществляться одним из нижеперечисленных способов или их сочетанием на усмотрение преподавателя.

1. Устный ответ по теме работы.
2. Тестирование по теме работы
3. Задача по теме работы.
4. Иные варианты на усмотрение преподавателя.

Контрольные вопросы

1. Теоремы БЧХ.
2. Построение порождающего полинома кода БЧХ.
3. Кодирование кодом БЧХ.
4. Алгоритм Берлекэмпа–Месси.
5. Алгоритм Евклида.
6. Алгоритм Питерсона–Горенштейна–Цирлера.
7. Алгоритм (метод) Ченя.

Практическая работа 3
Коды Рида–Соломона

3.1. Цель работы

Рассмотреть на примере и получить навыки в исследовании кодов Рида–Соломона.

3.2. Теоретическая справка

Коды Рида–Соломона (коды РС) представляют из себя недвоичные циклические коды Боуза–Чоудхури–Хоквингема, символы которых представляют собой элементы поля Галуа $\mathrm{GF}(q)$, где $q = 2^m$ — порядок поля, а m — степень поля Галуа [4].

Коды РС (n,k) определены на всех m-битовых символах при всех n и k, для которых верно неравенство:

$$0 < k < n \leq 2^m - 1,$$

где k — число информационных символов, подлежащих кодированию, а n — число кодовых символов в кодируемом блоке. Для большинства кодов РС

$$(n,k) = (q-1, q-1-2t), \tag{3.1}$$

где t — количество ошибок, исправляемых кодом, а $n-k = 2t = r$ — число контрольных символов [9].

Если $n < q-1$, то код РС называют укороченным кодом. Если $n = q$ или $n = q+1$, то код РС называется расширенным на 1 или 2 символа, соответственно [8].

Для задания циклических кодов РС используется порождающий многочлен $g(x)$ степени $d_{\min} - 1 = n - k = r$ вида [4, 10]:

$$g(x) = \prod_{j=b}^{b+2t-1} (x + \varepsilon^j), \tag{3.2}$$

где b — целое число. Обычно $b = 0$ или $b = 1$.

Коды РС, у которых $b = 1$, т. е. их образующий полином равен

$$g(x) = (x+\varepsilon)(x+\varepsilon^2)\ldots(x+\varepsilon^{n-k}), \tag{3.3}$$

называют *кодами РС в узком смысле* [11].

На основе порождающего многочлена можно построить проверочный многочлен $h(x)$ степени k, удовлетворяющий условию [8]:

$$h(x)g(x) \equiv 0 \ [\mathrm{mod}\ x^n + 1].$$

При этом все кодовые слова циклического кода кратны $g(x)$, а именно

$$s(x)h(x) \equiv 0 \ [\mathrm{mod}\, x^n + 1] \ \text{ и } \ f(x) \equiv 0 \ [\mathrm{mod}\, g(x)]. \tag{3.4}$$

Код РС обладает наибольшим минимальным кодовым расстоянием, возможным для линейного кода с одинаковыми n и k. Для недвоичных кодов расстояние между двумя кодовыми словами определяется по аналогии с расстоянием Хемминга как число символов, которыми отличаются последовательности. Для кодов РС минимальное расстояние определяется как [12]:

$$d_{\min} = n - k + 1. \tag{3.5}$$

Максимальное количество ошибок, исправляемых кодом РС можно выразить следующим образом [9]:

$$t = \left\lfloor \frac{d_{\min} - 1}{2} \right\rfloor = \left\lfloor \frac{n-k}{2} \right\rfloor. \tag{3.6}$$

Здесь $\lfloor x \rfloor$ означает наибольшее целое, не превышающее x.

Как следует из уравнений (3.1) и (3.6), коды РС, исправляющие t символьных ошибок, требуют $2t$ контрольных символов, т. е. декодер использует $n-k$ избыточных символов, количество которых вдвое превышает количество исправляемых ошибок [9].

Способность кода РС к коррекции стираний выражается следующим образом [9]:

$$\rho \leq d_{\min} - 1 = n - k, \tag{3.7}$$

т. е. код РС способен исправить любой набор $n-k$ или менее стираний в кодовом слове.

Возможность одновременной коррекции ошибок и стираний можно выразить как требование [9]:

$$2t' + \vartheta < d_{\min} < n - k, \tag{3.8}$$

где t' — число символьных ошибок, которые можно исправить, а ϑ — количество исправляемых символьных стираний.

В качестве примера рассмотрим циклический код РС $(n,k) = (7,3)$ над полем $\mathrm{GF}(2^3)$ с примитивным элементом ε, образующий полином которого имеет вид $p(x) = x^3 + x + 1$.

Согласно формуле (3.5), минимальное кодовое расстояние для данного кода рассчитывается как

$$d_{\min} = n - k + 1 = 5. \tag{3.9}$$

Количество гарантированно исправляемых кодом ошибок рассчитаем по формуле (3.6):

$$t = \left\lfloor \frac{n-k}{2} \right\rfloor = 2. \tag{3.10}$$

По формуле (3.2) построим порождающий многочлен кода РС ($b = 1$):

$$\begin{aligned} g_{(7,3),b=1}(x) &= \prod_{j=1}^{4} (x+\varepsilon^j) = (x+\varepsilon)(x+\varepsilon^2)(x+\varepsilon^3)(x+\varepsilon^4) = \\ &= x^4 + \varepsilon^3 x^3 + x^2 + \varepsilon x + \varepsilon^3. \end{aligned} \tag{3.11}$$

Таким образом, был получен порождающий многочлен кода, который можно использовать для кодирования информационных комбинаций.

Если взять $b = 0$, то получится другой полином:

$$\begin{aligned} g_{(7,3),b=0}(x) &= \prod_{j=0}^{3} (x+\varepsilon^j) = (x+1)(x+\varepsilon)(x+\varepsilon^2)(x+\varepsilon^3) = \\ &= x^4 + \varepsilon^2 x^3 + \varepsilon^5 x^2 + \varepsilon^5 x + \varepsilon^6, \end{aligned} \tag{3.12}$$

который также может быть использован для построения кода РС.

3.2.1. Кодирование кодов РС

Кодирование с помощью кода РС может быть реализовано двумя способами: систематическим и несистематическим.

Представим исходное информационное слово как информационный полином $u(x)$ степени $k-1$, а кодовое слово кода РС в виде полинома $v(x)$ степени $n-1$.

В случае несистематического кодирования кодовое слово находится из соотношения

$$v(x) = u(x)g(x)$$

где $g(x)$ — порождающий многочлен кода РС.

Данный алгоритм реализуется кодирующим устройством, которое показано на рис. 3.1 в общем виде.

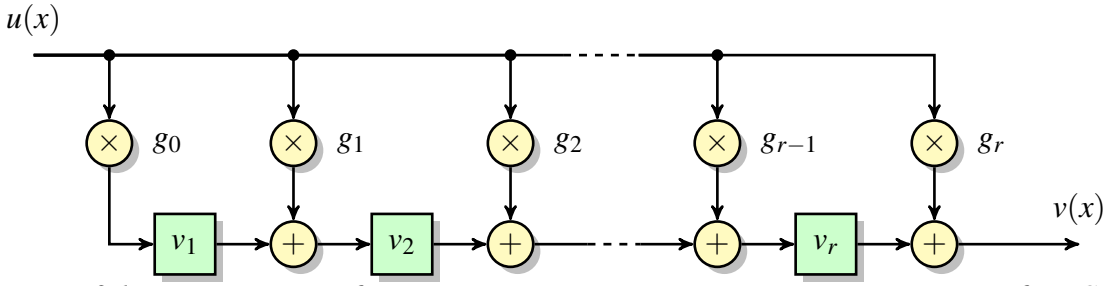

Рис. 3.1. Блок-схема кодирующего устройства несистематического кода РС

Данное устройство представляет из себя обычную схему перемножения двух многочленов.

При этом результирующая кодовая комбинация не содержит в явном виде исходных информационных элементов.

При систематическом кодировании используется типовой алгоритм кодирования для систематических циклических кодов [9]:

1) осуществляется сдвиг информационного полинома $u(x)$ в крайние старшие k разрядов кодового слова путем умножения полинома $u(x)$ на x^{n-k};

2) полученный полином $x^{n-k}u(x)$ делится на порождающий многочлен $g(x)$ для получения остатка от деления $r(x)$;

3) искомое кодовое слово $v(x)$ определяется как $v(x) = x^{n-k}u(x) + r(x)$.

На рис. 3.2 приведена схема, реализующая вышеприведенный алгоритм кодирования [13].

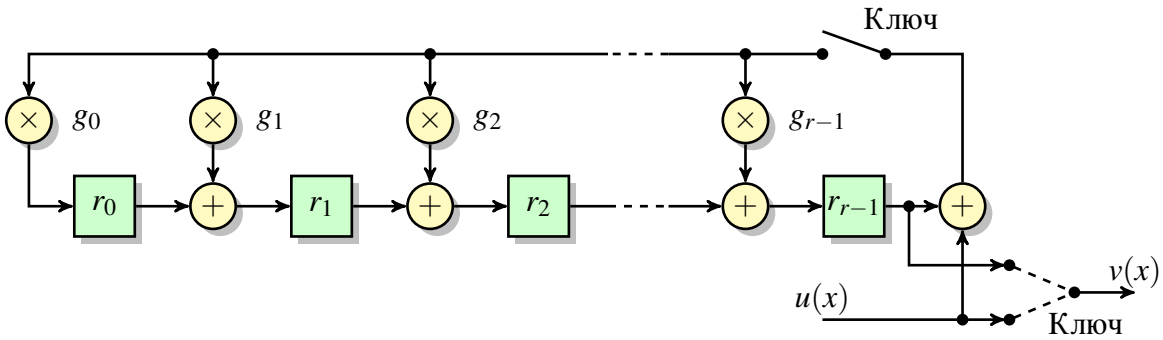

Рис. 3.2. Блок-схема кодирующего устройства систематического кода РС

При использовании данного метода, результирующее кодовое слово содержит в явном виде исходные информационные элементы.

Также существует алгоритм кодирования, не требующий использования порождающего полинома [4]. При использовании данного метода, искомое кодовое слово $v(x)$ определяется как вектор

$$\mathbf{v} = \{u(1)u(\varepsilon)u(\varepsilon^2)\ldots u(\varepsilon^{q-2})\},$$

где ε — примитивный элемент поля Галуа $\mathrm{GF}(q)$. Можно видеть, что этот метод кодирования также является несистематическим.

Приведем примеры кодирования информационного полинома $u(x) = \varepsilon^4 + \varepsilon^3 x^2$ всеми тремя способами, для кода РС (7,3), образованного посредством полинома $g(x)$ при $b = 0$ (3.12).

Несистематическое кодирование:

$$\begin{aligned} v(x) = u(x)g(x) &= (\varepsilon^4 + \varepsilon^3 x^2)(x^4 + \varepsilon^2 x^3 + \varepsilon^5 x^2 + \varepsilon^5 x + \varepsilon^6) = \\ &= \varepsilon^3 + \varepsilon x + \varepsilon^4 x^2 + \varepsilon^2 x^3 + \varepsilon^5 x^4 + \varepsilon^5 x^5 + \varepsilon^3 x^6. \end{aligned}$$

Систематическое кодирование:

$$\begin{aligned} v(x) &= x^4 u(x) + \mathrm{mod}\left(\frac{x^4 u(x)}{g(x)}\right) = \\ &= (\varepsilon^4 x^4 + \varepsilon^3 x^6) + \mathrm{mod}\left(\frac{\varepsilon^4 x^4 + \varepsilon^3 x^6}{x^4 + \varepsilon^2 x^3 + \varepsilon^5 x^2 + \varepsilon^5 x + \varepsilon^6}\right) = \\ &= \varepsilon^2 + \varepsilon^2 x + \varepsilon x^2 + \varepsilon^4 x^3 + \varepsilon^4 x^4 + \varepsilon^3 x^6, \end{aligned}$$

где mod — функция, обозначающая остаток от деления.

Кодирование без использования порождающего полинома:

$$\begin{aligned} \mathbf{v} &= \{u(1)u(\varepsilon)u(\varepsilon^2)u(\varepsilon^3)u(\varepsilon^4)u(\varepsilon^5)u(\varepsilon^6)\} = \\ &= \{\varepsilon^6\ 1\ \varepsilon^5\ \varepsilon\ 0\ \varepsilon^3\ \varepsilon^2\}, \\ v(x) &= \varepsilon^6 + x + \varepsilon^5 x^2 + \varepsilon x^3 + \varepsilon^3 x^5 + \varepsilon^2 x^6. \end{aligned}$$

3.2.2. Алгебраический метод декодирования

Алгебраический метод декодирования кодов Рида–Соломона аналогичен процедуре декодирования двоичных кодов БЧХ. Отличием является то, что элементами кода являются элементы соответствующего поля Галуа и помимо позиций ошибок необходимо определить их значения.

1. Вычисление синдрома $S(x)$ принятого слова.
2. Решение ключевого уравнения

$$S(x)\sigma(x) \equiv \omega(x) \ \ [\mathrm{mod}\, x^d],$$

где $\sigma(x)$ — многочлен локаторов ошибок степени t (его корни являются обратными величинами локаторов искаженных позиций); $\omega(x)$ — многочлен значений ошибок со степенью всегда меньшей степени $\sigma(x)$ [14].

Если

$$\deg \sigma(x) \geq r/2,$$

то происходит отказ и переход к пункту 5.

3. Определение множества локаторов ошибок по процедуре Ченя. Если число локаторов ошибок не совпадает со степенью $\sigma(x)$, то происходит отказ и переход к пункту 5.

4. Вычисление значений ошибок и исправление искаженных позиций по алгоритму Форни.

5. Восстановление информационных символов или выдача признака отказа от декодирования. Этап 5 при отсутствии отказа состоит в восстановлении информационных символов из полученного кодового слова. При систематическом коде РС этот этап сводится к выдаче символов на информационных позициях кода. Для несистематического кода выполняется преобразование, определяющее информационные элементы кода, например, дискретное преобразование Фурье.

3.2.2.1. *Вычисление синдромов*

Как и в случае двоичных кодов БЧХ, при декодировании кода РС синдромы вычисляются, как значения принятого полинома $r(x)$ в *нулях кода*:

$$S_j = r(\varepsilon^j), \tag{3.13}$$

где $j = b \ldots b + 2t - 1$, $b = 0$ или $b = 1$, ε^j — корни порождающего полинома кода РС $g(x)$.

Предположим, что из канала связи получена комбинация рассмотренного ранее кода РС (7,3) с образующим полиномом $g(x)$ при $b = 0$ (3.12), представляемая многочленом

$$r(x) = v(x) + e(x) = \varepsilon x^2 + \varepsilon^5 x^4. \tag{3.14}$$

Тогда, согласно формуле (3.13), получим синдромы

$$\begin{aligned} S_0 &= r(1) = \varepsilon(1)^2 + \varepsilon^5(1)^4 = \varepsilon^6, \\ S_1 &= r(\varepsilon) = \varepsilon(\varepsilon)^2 + \varepsilon^5(\varepsilon)^4 = \varepsilon^5, \\ S_2 &= r(\varepsilon^2) = \varepsilon(\varepsilon^2)^2 + \varepsilon^5(\varepsilon^2)^4 = \varepsilon, \\ S_3 &= r(\varepsilon^3) = \varepsilon(\varepsilon^3)^2 + \varepsilon^5(\varepsilon^3)^4 = \varepsilon. \end{aligned} \tag{3.15}$$

3.2.2.2. *Алгоритм Берлекэмпа–Месси*

В применении к кодам РС рассмотрим вариант алгоритма Берлекэмпа–Месси, известный как алгоритм Месси [4].

Исходные данные: синдромы S_j, где $j = b \ldots b + 2t - 1$, $b = 0$ или $b = 1$.

Начальные условия: $\sigma(x) = 1$, корректор $\rho(x) = x$, счетчик $i = 0$, длина регистра $l = 0$.

1. Вычисляем различие $\Delta = \sum_{j=0}^{l} \sigma_j S_{i-j-1}$.
2. Проверяем различие Δ: если $\Delta = 0$, то переходим к 7.
3. Модифицируем многочлен локаторов ошибок: $\sigma_{new}(x) = \sigma(x) + \Delta\rho(x)$.
4. Проверяем длину регистра: если $2l \geq i$, то переходим к 6.
5. Исправляем длину регистра и заменяем корректор: $l = i - l$, $\rho(x) = \sigma(x)/\Delta$.
6. Обновляем многочлен локаторов ошибок: $\sigma(x) = \sigma_{new}(x)$.
7. Обновляем корректор: $\rho(x) = x\rho(x)$.
8. Обновляем счетчик: $i = i + 1$.
9. Если $i < d$, то переходим к 1, иначе — останавливаемся.

Рассмотрим декодирование по алгоритму Мэсси на примере ранее рассмотренного кода РС $(7,3)$ с образующим полиномом $g(x)$ (3.12) и на примере полученной на вход декодера комбинации (3.14) [4]. Синдромы для выбранной комбинации показаны в формуле (3.15).

Шаг $i = 1$:

Начальные условия: $\sigma(x) = 1,\ \rho(x) = x,\ l = 0;$

$$\Delta = \sum_{j=0}^{0} \sigma_j S_{1-j-1} = \sigma_0 S_0 = \varepsilon^6;$$

$$\sigma_{new}(x) = \sigma(x) + \Delta\rho(x) = 1 + \varepsilon^6 x;$$

$$2l = 0 < i;$$

$$l \to i - l = 1 - 0 = 1;\quad \rho(x) = \sigma(x)/\Delta = 1/\varepsilon^6 = \varepsilon^{-6} = \varepsilon;$$

$$\sigma(x) = \sigma_{new}(x) = 1 + \varepsilon^6 x;$$

$$\rho(x) \to x\rho(x) = \varepsilon x.$$

Шаг $i = 2$:

$$\Delta = \sum_{j=0}^{1} \sigma_j S_{2-j-1} = \sigma_0 S_1 + \sigma_1 S_0 = \varepsilon^5 + \varepsilon^6\varepsilon^6 = 0;$$

$$\rho(x) \to x\rho(x) = \varepsilon x^2.$$

Шаг $i = 3$:

$$\Delta = \sum_{j=0}^{1} \sigma_j S_{3-j-1} = \sigma_0 S_2 + \sigma_1 S_1 = \varepsilon + \varepsilon^6\varepsilon^5 = \varepsilon^2;$$

$$\sigma_{new}(x) = \sigma(x) + \Delta\rho(x) = 1 + \varepsilon^6 x + \varepsilon^3 x^2;$$

$$2l = 2 < i;$$

$$l \to i - l = 3 - 1 = 2;\quad \rho(x) = \sigma(x)/\Delta = \varepsilon^5 + \varepsilon^4 x;$$

$$\sigma(x) = \sigma_{new}(x) = 1 + \varepsilon^6 x + \varepsilon^3 x^2;$$

$$\rho(x) \to x\rho(x) = \varepsilon^5 x + \varepsilon^4 x^2.$$

Шаг $i = 4$:

$$\Delta = \sum_{j=0}^{2} \sigma_j S_{4-j-1} = \sigma_0 S_3 + \sigma_1 S_2 + \sigma_2 S_1 = \varepsilon + \varepsilon^6\varepsilon + \varepsilon^3\varepsilon^5 = 1;$$

$$\sigma_{new}(x) = \sigma(x) + \Delta\rho(x) = 1 + \varepsilon^6 x + \varepsilon^3 x^2 + \varepsilon^5 x + \varepsilon^4 x^2 = 1 + \varepsilon x + \varepsilon^6 x^2;$$

$$2l = 4 = i;$$

$$\sigma(x) = \sigma_{new}(x) = 1 + \varepsilon x + \varepsilon^6 x^2;$$

$$\rho(x) \to x\rho(x) = \varepsilon^5 x^2 + \varepsilon^4 x^3.$$

Шаг $i = 5 = d$: Стоп.

Теперь по ключевому уравнению найдем многочлен значений ошибок:

$$\omega(x) = S(x)\sigma(x) \pmod{x^5} = 1 + \varepsilon^5 x + \varepsilon^3 x^2.$$

3.2.2.3. *Алгоритм Евклида*

В основе этого алгоритма лежит процедура нахождения наибольшего общего делителя (НОД) двух полиномов [4]. Для решения ключевого уравнения расширенный алгоритм Евклида применяется к многочленам $r_0(x) = x^d$ и $r_1(x) = S(x)$. Если на j-м шаге алгоритма получено решение

$$r_j(x) = a_j(x)x^d + b_j(x)S(x),$$

такое, что $\deg[r_j(x)] \leq (d-1)/2$, то $\omega(x) = r_j(x)$ и $\sigma(x) = b_j(x)$.

Функция $\deg[f(x)]$ определяет степень полинома $f(x)$.

Расширенный алгоритм Евклида состоит в следующем [4].

1. Начальные условия:

$$r_0(x) = x^d, \quad r_1(x) = S(x), \quad \deg[r_0(x)] \geq \deg[r_1(x)];$$

$$a_0(x) = 1, \quad b_0(x) = 0, \quad a_1(x) = 0, \quad b_1(x) = 1.$$

2. На шаге j $(j \geq 2)$ применяем длинное деление к многочленам $r_{j-2}(x)$ и $r_{j-1}(x)$:

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

3. Вычисляем:

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

4. Останавливаем вычисления на итерации j_{last}, когда

$$\deg[r_{last}(x)] \leq r/2.$$

5. Выход: НОД$[r_0(x), r_1(x)] = r_k(x)$, где k — наибольшее ненулевое целое, для которого выполняются условия $r_k(x) \neq 0$ и $k < j_{last}$.

Данный алгоритм отличается большей простотой аппаратной реализации по сравнению с алгоритмом Берлекэмпа–Месси, но требует больше операций в конечном поле [4].

Важно отметить, что алгоритм Евклида вычисляет $\sigma(x)$ и $\omega(x)$ одновременно, как $\sigma(x) = b_{last}(x)$ и $\omega(x) = r_{last}(x)$.

Рассмотрим декодирование по алгоритму Евклида на примере ранее рассмоpенного кода РС $(7,3)$ с образующим полиномом $g(x)$ (3.12) и на примере полученной на вход декодера комбинации (3.14) [4]. Синдромы для выбранной комбинации приведены в формуле (3.15).

Начинаем декодирование по алгоритму Евклида [4].

Шаг 1. Начальные условия:

$r_0(x) = x^5$,

$r_1(x) = S(x) = 1 + \varepsilon^6 x + \varepsilon^5 x^2 + \varepsilon x^3 + \varepsilon x^4$,

$b_0(x) = 0$,

$b_1(x) = 1$.

Шаг 2:

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

$r_2(x) = \varepsilon^5 x^3 + x^2 + \varepsilon x + \varepsilon^6$,

$q_2(x) = \varepsilon^6 x + \varepsilon^6$,

$b_2(x) = 0 + (\varepsilon^6 x + \varepsilon^6)(1) = \varepsilon^6 x + \varepsilon^6$.

Шаг 3:

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

$r_3(x) = \varepsilon^6 x^2 + \varepsilon x + \varepsilon^3$,

$q_3(x) = \varepsilon^3 x + \varepsilon^2$,

$b_3(x) = 1 + (\varepsilon^3 x + \varepsilon^2)(\varepsilon^6 x + \varepsilon^6) = \varepsilon^3 + \varepsilon^4 x + \varepsilon^2 x^2$.

Алгоритм останавливается, так как $\deg[r_3(x)] = 2 = r/2$.

Следовательно:

$\sigma(x) = \varepsilon^3 + \varepsilon^4 x + \varepsilon^2 x^2 = \varepsilon^3(1 + \varepsilon x + \varepsilon^6 x^2)$,

$\omega(x) = \varepsilon^3 + \varepsilon x + \varepsilon^6 x^2 = \varepsilon^3(1 + \varepsilon^5 x + \varepsilon^3 x^2)$.

3.2.2.4. Процедура Ченя

Как и в случае двоичных кодов БЧХ, для поиска корней $\sigma(x)$ на множестве локаторов позиций кодовых символов используется метод Ченя, согласно которому для всех ненулевых элементов $\beta \in \mathrm{GF}(2^m)$, проверяется условие $\sigma(\beta^{-1}) = 0$ [4].

Для примера, рассмотренного выше, решением ключевого уравнения будут элементы поля ε^{-2} и ε^{-4}, т. е. $\beta_{j_1} = \varepsilon^2$ и $\beta_{j_2} = \varepsilon^4$. Следовательно, полином локаторов ошибок можно представить в виде произведения полиномов первой степени:

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

Таким образом, ошибки произошли на позициях $j_1 = 2$ и $j_2 = 4$.

3.2.2.5. Алгоритм Форни

Согласно алгоритму Форни значения ошибок вычисляются по формуле

$$e_{j_l} = \beta_{j_l}^{1-b} \omega(\beta_{j_l}^{-1})[\sigma'(\beta_{j_l}^{-1})]^{-1}. \tag{3.16}$$

Для рассматриваемого примера это уравнение будет иметь вид

$$e_{j_l} = \beta_{j_l}(1+\varepsilon^5\beta_{j_l}^{-1}+\varepsilon^3\beta_{j_l}^{-2})[\varepsilon]^{-1},$$

тогда

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

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

$$\hat{v}(x) = r(x) + e(x) = 0.$$

Исправлены две ошибки. Была передана нулевая кодовая комбинация.

3.3. Порядок выполнения задания

Задание выполняется каждым учащимся индивидуально. Вариант задания выбирается согласно номеру студента в журнале группы.

Все расчеты должны быть расписаны максимально подробно.

1. Построить по заданным параметрам (n,k)-код РС. Код должен быть построен над полем $\mathrm{GF}(2^m)$, где $m = 4$. Порождающий полином поля Галуа $p(x) = x^4 + x + 1$. Элементы поля представлены в табл. 2.1. Код должен гарантированно исправлять три ошибки ($t = 3$). Считать, что $b = 1$. Задание выполнять в следующем порядке:
   а) определить n и k;
   б) определить образующий полином кода РС;
   в) нарисовать схему систематического кодера РС.
2. Согласно номеру в журнале выбрать из табл. 3.1 информационный вектор $\mathbf{u}$ и полином ошибок $e(x)$. Преобразовать вектор $\mathbf{u}$ в полином $u(x)$. Считать, что вектор $\mathbf{u}$ записан, начиная с младшей степени.
3. Закодировать полученный в предыдущем пункте информационный полином $u(x)$ систематическим кодом РС. Для кодирования использовать порождающий полином или схему кодера.
4. Наложить на полученный в предыдущем пункте кодовый полином $v(x)$ кода РС двукратную ошибку $e(x)$ и декодировать полученную комбинацию $r(x) = v(x) + e(x)$ синдромным методом декодирования. Для решения ключевого уравнения использовать любой из трех алгоритмов (Берлекэмпа–Месси, Питерсона–Горенштейна–Цирлера, Евклида). Задание выполнять в следующем порядке:
   а) определить синдромы;
   б) решить ключевое уравнение;
   в) определить позиции ошибки по методу Ченя;
   г) определить значения ошибок по алгоритму Форни;
   д) исправить ошибки.

Таблица 3.1

Информационный вектор **u** *и полином ошибок* $e(x)$
(выбираются согласно номеру студента в журнале)

№	Информационный вектор **u**	$e(x)$
1	$\varepsilon^8\ \varepsilon^5\ \varepsilon^{14}\ \varepsilon^{10}\ \varepsilon^7\ \varepsilon^7\ \varepsilon^3\ \varepsilon^0\ \varepsilon^4$	$\varepsilon^5 x^3 + \varepsilon^{14} x^{13}$
2	$\varepsilon^8\ \varepsilon^3\ \varepsilon^8\ \varepsilon^{13}\ \varepsilon^4\ \varepsilon^3\ \varepsilon^{14}\ \varepsilon^{14}\ \varepsilon^{12}$	$\varepsilon^{11} x^2 + \varepsilon^{12} x^6$
3	$\varepsilon^{13}\ \varepsilon^2\ \varepsilon^8\ \varepsilon^1\ \varepsilon^4\ \varepsilon^5\ \varepsilon^{10}\ \varepsilon^{10}\ \varepsilon^4$	$\varepsilon^4 x^9 + \varepsilon^8 x^{10}$
4	$\varepsilon^{13}\ \varepsilon^9\ \varepsilon^7\ \varepsilon^{14}\ \varepsilon^5\ \varepsilon^3\ \varepsilon^2\ \varepsilon^8\ \varepsilon^{14}$	$\varepsilon^{13} x^2 + \varepsilon^0 x^8$
5	$\varepsilon^3\ \varepsilon^{13}\ \varepsilon^0\ \varepsilon^{13}\ \varepsilon^7\ \varepsilon^4\ \varepsilon^{13}\ \varepsilon^5\ \varepsilon^5$	$\varepsilon^{11} x^{11} + \varepsilon^4 x^{14}$
6	$\varepsilon^{11}\ \varepsilon^{11}\ \varepsilon^2\ \varepsilon^3\ \varepsilon^9\ \varepsilon^0\ \varepsilon^{11}\ \varepsilon^{14}\ \varepsilon^7$	$\varepsilon^2 x^9 + \varepsilon^6 x^{14}$
7	$\varepsilon^7\ \varepsilon^5\ \varepsilon^{14}\ \varepsilon^4\ \varepsilon^{12}\ \varepsilon^{12}\ \varepsilon^5\ \varepsilon^5\ \varepsilon^{13}$	$\varepsilon^{11} x^2 + \varepsilon^{10} x^9$
8	$\varepsilon^0\ \varepsilon^{13}\ \varepsilon^1\ \varepsilon^7\ \varepsilon^{11}\ \varepsilon^{14}\ \varepsilon^{13}\ \varepsilon^4\ \varepsilon^4$	$\varepsilon^0 x^5 + \varepsilon^{13} x^6$
9	$\varepsilon^3\ \varepsilon^1\ \varepsilon^{11}\ \varepsilon^5\ \varepsilon^{14}\ \varepsilon^{10}\ \varepsilon^9\ \varepsilon^{14}\ \varepsilon^9$	$\varepsilon^{11} x^2 + \varepsilon^0 x^5$
10	$\varepsilon^{13}\ \varepsilon^{13}\ \varepsilon^{14}\ \varepsilon^9\ \varepsilon^{10}\ \varepsilon^3\ \varepsilon^7\ \varepsilon^{11}\ \varepsilon^4$	$\varepsilon^7 x^2 + \varepsilon^{10} x^6$
11	$\varepsilon^9\ \varepsilon^4\ \varepsilon^5\ \varepsilon^0\ \varepsilon^0\ \varepsilon^0\ \varepsilon^6\ \varepsilon^3\ \varepsilon^6$	$\varepsilon^5 x^4 + \varepsilon^4 x^7$
12	$\varepsilon^3\ \varepsilon^8\ \varepsilon^{10}\ \varepsilon^0\ \varepsilon^{11}\ \varepsilon^1\ \varepsilon^6\ \varepsilon^1\ \varepsilon^8$	$\varepsilon^6 x^2 + \varepsilon^{14} x^4$
13	$\varepsilon^{14}\ \varepsilon^2\ \varepsilon^{12}\ \varepsilon^{10}\ \varepsilon^1\ \varepsilon^2\ \varepsilon^2\ \varepsilon^4\ \varepsilon^{10}$	$\varepsilon^2 x^2 + \varepsilon^2 x^{12}$
14	$\varepsilon^0\ \varepsilon^0\ \varepsilon^1\ \varepsilon^0\ \varepsilon^0\ \varepsilon^{14}\ \varepsilon^{12}\ \varepsilon^{11}\ \varepsilon^6$	$\varepsilon^2 x^5 + \varepsilon^1 x^9$
15	$\varepsilon^5\ \varepsilon^4\ \varepsilon^{12}\ \varepsilon^6\ \varepsilon^2\ \varepsilon^8\ \varepsilon^{11}\ \varepsilon^3\ \varepsilon^{11}$	$\varepsilon^{13} x^9 + \varepsilon^{10} x^{13}$
16	$\varepsilon^{13}\ \varepsilon^{11}\ \varepsilon^{13}\ \varepsilon^4\ \varepsilon^{13}\ \varepsilon^{10}\ \varepsilon^8\ \varepsilon^3\ \varepsilon^9$	$\varepsilon^5 x^5 + \varepsilon^4 x^{10}$
17	$\varepsilon^{10}\ \varepsilon^3\ \varepsilon^6\ \varepsilon^8\ \varepsilon^{13}\ \varepsilon^0\ \varepsilon^3\ \varepsilon^5\ \varepsilon^{10}$	$\varepsilon^{13} x^6 + \varepsilon^0 x^{13}$
18	$\varepsilon^0\ \varepsilon^5\ \varepsilon^{12}\ \varepsilon^{14}\ \varepsilon^0\ \varepsilon^9\ \varepsilon^{13}\ \varepsilon^5\ \varepsilon^1$	$\varepsilon^{10} x^6 + \varepsilon^{13} x^{14}$
19	$\varepsilon^{10}\ \varepsilon^{13}\ \varepsilon^5\ \varepsilon^8\ \varepsilon^9\ \varepsilon^2\ \varepsilon^7\ \varepsilon^7\ \varepsilon^7$	$\varepsilon^{14} x^3 + \varepsilon^0 x^{14}$
20	$\varepsilon^4\ \varepsilon^2\ \varepsilon^2\ \varepsilon^{10}\ \varepsilon^{12}\ \varepsilon^{14}\ \varepsilon^6\ \varepsilon^1\ \varepsilon^5$	$\varepsilon^{13} x^4 + \varepsilon^7 x^5$
21	$\varepsilon^{13}\ \varepsilon^6\ \varepsilon^{13}\ \varepsilon^{12}\ \varepsilon^{13}\ \varepsilon^2\ \varepsilon^{13}\ \varepsilon^{13}\ \varepsilon^{14}$	$\varepsilon^{12} x^4 + \varepsilon^2 x^{10}$
22	$\varepsilon^8\ \varepsilon^5\ \varepsilon^7\ \varepsilon^6\ \varepsilon^1\ \varepsilon^1\ \varepsilon^5\ \varepsilon^3\ \varepsilon^6$	$\varepsilon^2 x^{10} + \varepsilon^{14} x^{12}$
23	$\varepsilon^{14}\ \varepsilon^{13}\ \varepsilon^1\ \varepsilon^2\ \varepsilon^7\ \varepsilon^{12}\ \varepsilon^9\ \varepsilon^{11}\ \varepsilon^{13}$	$\varepsilon^0 x^3 + \varepsilon^9 x^{14}$
24	$\varepsilon^4\ \varepsilon^0\ \varepsilon^9\ \varepsilon^{14}\ \varepsilon^9\ \varepsilon^{12}\ \varepsilon^0\ \varepsilon^9\ \varepsilon^0$	$\varepsilon^4 x^4 + \varepsilon^{13} x^{10}$
25	$\varepsilon^5\ \varepsilon^1\ \varepsilon^4\ \varepsilon^2\ \varepsilon^6\ \varepsilon^4\ \varepsilon^0\ \varepsilon^8\ \varepsilon^2$	$\varepsilon^0 x^5 + \varepsilon^{14} x^7$
26	$\varepsilon^8\ \varepsilon^1\ \varepsilon^1\ \varepsilon^6\ \varepsilon^7\ \varepsilon^{14}\ \varepsilon^{14}\ \varepsilon^2\ \varepsilon^7$	$\varepsilon^1 x^3 + \varepsilon^4 x^6$
27	$\varepsilon^8\ \varepsilon^3\ \varepsilon^6\ \varepsilon^4\ \varepsilon^6\ \varepsilon^{13}\ \varepsilon^4\ \varepsilon^{12}\ \varepsilon^6$	$\varepsilon^9 x^6 + \varepsilon^3 x^7$
28	$\varepsilon^6\ \varepsilon^7\ \varepsilon^{13}\ \varepsilon^{10}\ \varepsilon^{13}\ \varepsilon^1\ \varepsilon^{11}\ \varepsilon^1\ \varepsilon^{14}$	$\varepsilon^1 x^5 + \varepsilon^9 x^{12}$
29	$\varepsilon^{14}\ \varepsilon^8\ \varepsilon^7\ \varepsilon^2\ \varepsilon^{13}\ \varepsilon^{12}\ \varepsilon^{11}\ \varepsilon^{10}\ \varepsilon^9$	$\varepsilon^5 x^6 + \varepsilon^1 x^{11}$
30	$\varepsilon^{13}\ \varepsilon^7\ \varepsilon^0\ \varepsilon^3\ \varepsilon^9\ \varepsilon^9\ \varepsilon^3\ \varepsilon^{12}\ \varepsilon^0$	$\varepsilon^9 x^7 + \varepsilon^4 x^{11}$

3.4. Порядок защиты практической работы

Защита работы может осуществляться одним из нижеперечисленных способов или их сочетанием на усмотрение преподавателя.

1. Устный ответ по теме работы.
2. Тестирование по теме работы
3. Задача по теме работы.
4. Иные варианты на усмотрение преподавателя.

Контрольные вопросы

1. Что такое код Рида–Соломона?
2. Построение порождающего полинома кода РС.
3. Алгоритм Берлекэмпа–Месси.
4. Алгоритм Евклида.
5. Алгоритм Форни.

Список литературы

[1] Вернер, М. Основы кодирования : учебник для ВУЗов / М. Вернер. Мир программирования. — М. : Техносфера, 2004.

[2] Когновицкий, О. С. Основы циклических кодов : учебное пособие / О. С. Когновицкий. — Л. : ЛЭИС, 1990.

[3] Шварцман, В. О. Теория передачи дискретной информации : учебник для вузов связи / В. О. Шварцман, Г. А. Емельянов. — М. : Связь, 1979.

[4] Морелос-Сарагоса, Р. Искусство помехоустойчивого кодирования. Методы, алгоритмы, применение. / Р. Морелос-Сарагоса. Мир связи. — М. : Техносфера, 2005.

[5] Блейхут, Р. Теория и практика кодов, контролирующих ошибки / Р. Блейхут. — М. : Мир, 1986.

[6] Берлекэмп, Э. Алгебраическая теория кодирования. / Э. Берлекэмп. — М. : Мир, 1972.

[7] Massey, J. L. Shift-Register Synthesis and BCH Decoding / J. L. Massey // Information Theory, IEEE Transactions on. — 1969. — Jan. — Vol. 15. — P. 122–127.

[8] Габидуллин, Э. М. Кодирование в радиоэлектронике / Э. М. Габидуллин, В. Б. Афанасьев. — М. : Радио и связь, 1986.

[9] Скляр, Б. Цифровая связь. Теоретические основы и практическое применение / Б. Скляр ; под ред. А.В. Назаренко. — М. : Издательский дом «Вильямс», 2003.

[10] Золотарёв, В. В. Помехоустойчивое кодирование. Методы и алгоритмы : справочник / В. В. Золотарёв, Г. В. Овечкин ; под ред. чл.-корр. РАН Ю. Б. Зубарева. — М. : Горячая линия–Телеком, 2004.

[11] Варгаузин, В. А. Методы повышения энергетической и спектральной эффективности цифровой радиосвязи : учебное пособие / В. А. Варгаузин, И. А. Цикин. — СПб. : БХВ-Петербург, 2013.

[12] Галлагер, Р. Теория информации и надежная связь / Р. Галлагер ; под ред. М. С. Пинскера, Б. С. Цыбакова. — М. : «Сов. радио», 1974.

[13] Lin, S. Error Control Coding: Fundamentals and Applications / S. Lin, D. J. Costello. — New Jersey : Printice-Hall, 1983.

[14] Теория электрической связи : учебное пособие / К. К. Васильев, В. А. Глушков, А. В. Дормидонтов, А. Г. Нестеренко ; под ред. К. К. Васильева. — Ульяновск : УлГТУ, 2008.

Владимиров Сергей Сергеевич
Когновицкий Олег Станиславович

ПРАКТИКА ПОМЕХОУСТОЙЧИВОГО КОДИРОВАНИЯ. ЦИКЛИЧЕСКИЕ КОДЫ

Практикум

Редактор *Л. К. Паршина*

План издания 2019, п. 50

Подписано к печати 27.12.2019
Объем 2,75 печ. л. Тираж 12 экз. Заказ 1016

Редакционно-издательский отдел СПбГУТ
193232 СПб., пр. Большевиков, 22
Отпечатано в СПбГУТ