Системы, сети и устройства телекоммуникаций

DOI 10.66032/2221-2574-2025-1-4-14-25 УДК 519.725.6

ИССЛЕДОВАНИЕ ВЫЧИСЛИТЕЛЬНОЙ ЭФФЕКТИВНОСТИ ДЕКОДИРОВАНИЯ КОДОВ БОУЗА – ЧОУДХУРИ – ХОКВИНГЕМА С ИСПОЛЬЗОВАНИЕМ АЛГОРИТМА ПИТЕРСОНА — ГОРЕНСТЕЙНА — ЦИРЛЕРА

By Txe Зует
аспирант ФГБОУ ВО «Санкт-Петербургский государственный университет телекоммуникаций им. проф. М.А. Бонч-Бруевича».
E-mail: vu.tz@sut.ru

Глушанков Евгений Иванович
доктор технических наук, профессор, профессор кафедры радиотехники ФГБОУ ВО «Санкт-Петербургский государственный университет телекоммуникаций им. проф. М.А. Бонч-Бруевича».
Адрес: 193232, Российская Федерация, г. Санкт-Петербург, пр. Большевиков, д. 22, корп. 1.

Аннотация: В данной статье проводится анализ вычислительной эффективности различных известных и новой предлагаемой процедуры определения полинома локаторов ошибок в процессе декодирования кодов Боуза — Чоудхури — Хоквингема с использованием алгоритма Питерсона — Горенстейна — Цирлера (ПГЦ). Вначале выполняется теоретический анализ вычислительной сложности рассматриваемых методов. На основе данного анализа осуществлена реализация и моделирование соответствующих алгоритмов декодирования в среде MATLAB, что позволило провести оценку и сравнение степени оптимальности рассмотренных методов применительно к декодированию конкретных кодов БЧХ. Научная новизна исследования заключается в повышении вычислительной эффективности алгоритма декодирования ПГЦ путём использования особой структуры синдромной матрицы, которая обладает свойствами тёплицевой матрицы. Все исследованные методы декодирования показали сопоставимую эффективность в исправлении ошибок. Однако предлагаемый метод, основанный на алгоритме Левинсона, разработанном при использования особых свойств синдромной матрицы, продемонстрировал превосходящую вычислительную эффективность.

Ключевые слова: коды Боуза – Чоудхури – Хоквингема, полином локаторов ошибок, алгоритм Питерсона — Горенстейна — Цирлера, алгоритм декодирования, синдромная матрица, тёплицевая матрица, алгоритм Левинсона, вычислительная эффективность, арифметические операция, MATLAB.

Введение

При передаче информации по физическим каналам связи, таким, как каналы с аддитивным белым гауссовским шумом (АБГШ) или каналы с замираниями (фейдингом), информационные системы подвергаются воздействию искажающих факторов. Это приводит к возникновению ошибок в передаваемых данных и, как следствие, к снижению общей энергетической эффективности. Для противодействия этим негативным явлениям и повышения надёжности передачи данных широко используются помехо-устойчивые коды, представляющие собой эффективный инструмент обеспечения качества функционирования каналов связи. Среди различных типов кодов исправления ошибок коды Боуза — Чоудхури — Хоквингема (БЧХ) выделяются своей способностью исправлять случайные ошибки и значительной гибкостью в практическом применении. Коды БЧХ находят широкое применение во многих областях, включая системы беспроводной связи, спутниковое телевидение и устройства

хранения данных (например, USB-накопители, SD-карта, SSD-накопитель) [1, 2]. Помимо этого, они интегрированы в стандарты цифрового вещания, такие, как DVB-S2 [3]. Основной целью исследования является разработка и оценка процедур, направленных на повышение вычислительной эффективности процесса декодирования, в частности, на этапе определения позиций ошибок путём нахождения полинома локаторов ошибок.

1. Обзор алгоритма декодирования ПГЦ

Среди множества разработанных методов декодирования алгоритм Питерсона — Горенстейна — Цирлера (ПГЦ) [4, 5] является одним из основополагающих и наиболее признанных решений. Ключевой принцип алгоритма ПГЦ для локализации ошибок заключается в формировании и последующем решении системы линейных алгебраических уравнений (СЛАУ). Данная система уравнений строится на основе вычисленных значений синдромов, а количество уравнений в ней напрямую зависит от максимальной корректирующей способности t используемого кода БЧХ.

Предположим, что кодовое слово представлено полиномом $c(x)$ с коэффициентами из конечного поля $GF(q)$, полином ошибок обозначен как $e(x)$, а принятая на приёмной стороне последовательность — как $v(x)$. Если в канале передачи возникают ошибки, то значения $v(\alpha^j)$ при $j = 1,2,\dots 2t$ отражают присутствие полинома ошибок $e(x)$. Именно эти значения и используются для формирования синдромов [6–8]:

$$S_j = v(\alpha^j) = c(\alpha^j) + e(\alpha^j) = 0 + e(\alpha^j) = e(\alpha^j).$$

Пусть в процессе передачи данных произошло ν ошибок, где $\nu \le t$. Положения этих ошибок обозначим как $\{i_1, i_2, \dots, i_\nu\}$. В этом случае компоненты синдрома [6–8] вычисляются следующим образом:

$$S_j = e_{i_1}\alpha^{i_1 j} + e_{i_2}\alpha^{i_2 j} + \dots + e_{i_\nu}\alpha^{i_\nu j},$$

где e_{i_h} представляет собой значение в положении ошибки h.

Поставим $X_h = \alpha^{i_h}$ вместо прямого определения локаторов ошибок X_h, алгоритм нацелен на поиск их обратных величин X_h^{-1}. Это достигается путём нахождения коэффициентов особого полинома, известного как полином локаторов ошибок $\Delta(x)$. Данный полином $\Delta(x)$ строится так, что его корнями служат именно эти обратные величины X_h^{-1} [6–8]:

$$\Delta(x) = \prod_{j=1}^{\nu}(1 - xX_j) = 1 + \Delta_1 x + \dots + \Delta_\nu x^\nu.$$

Коэффициенты полинома локаторов ошибок $\Delta(x)$ определяются путём решения СЛАУ. Данная система строится на основе значений синдромов S_j с применением тождеств Ньютона:

$$\begin{pmatrix} S_1 & S_2 \dots & S_\mu \\ S_2 & S_3 \dots & S_{\mu+1} \\ \vdots & \dots \ddots & \vdots \\ S_\mu & S_{\mu+1} \dots & S_{2\mu-1} \end{pmatrix} \begin{pmatrix} \Delta_\mu \\ \Delta_{\mu-1} \\ \vdots \\ \Delta_1 \end{pmatrix} = \begin{pmatrix} -S_{\mu+1} \\ -S_{\mu+2} \\ \vdots \\ -S_{2\mu} \end{pmatrix}. \quad (1)$$

Из (1) возможно определение набора коэффициентов $\{\Delta_1, \Delta_2, \dots, \Delta_\mu\}$ полинома локаторов ошибок. После того, как полином $\Delta(x)$ определён, его корни позволяют найти элементы — локаторы ошибок X_h. Для этой цели часто применяется алгоритм поиска Чиня [9].

2. Метод определения полинома локаторов ошибок на основе *LU*-разложения

В первую очередь, для определения коэффициентов $\Delta(x)$ будет рассмотрен один из классических подходов, а именно метод, основанный на методе Гаусса [10]. В этой статье данный подход реализован с помощью техники *LU*-разложения, которое по своей сути является матричной формой алгоритма Гаусса.

Как следует из (1), коэффициенты $\Delta(x)$ можно найти, умножив обратную матрицу к синдромной матрице на вектор S_v,

$$\Delta_v = S^{-1} S_v . \qquad (2)$$

Таким образом, ключевой задачей становится вычисление S^{-1}. В общем случае, когда матрица S не обладает специальной структурой, для её обращения часто прибегают к методу LU-разложения. Этот метод включает применение процедуры гауссова исключения для получения матриц L и U. При LU-разложении квадратная матрица S разлагается на произведение нижней треугольной матрицы L и верхней треугольной матрицы U, то есть $S = LU$:

$$L = \begin{pmatrix} 1 & 0\dots & 0 \\ l_{21} & 1\dots & 0 \\ \dots & \dots\ddots & \dots \\ l_{n1} & l_{n2}\dots & 1 \end{pmatrix}, U = \begin{pmatrix} u_{11} & u_{12}\dots & u_{1n} \\ 0 & u_{22}\dots & u_{2n} \\ \dots & \dots\ddots & \dots \\ 0 & 0\dots & u_{nn} \end{pmatrix}.$$

где l_{ij} — поддиагональные элементы и множители, используемые в процессе гауссова исключения; u_{ij} — наддиагональные элементы, получаемые в ходе гауссова исключения.

По сути LU-разложение представляет собой процесс приведения матрицы S к верхнетреугольному виду U путём выполнения последовательности элементарных строчных преобразований. Теоретические основы данного подхода и методика построения матриц-сомножителей L и U подробно изложены в [11]. Конкретный алгоритм для вычисления элементов этих матриц представлен далее:

```
function [L,U] = LU_Factorization(S)
Вход: матрица S, U=S, L=I (единичная матрица)
Выход: матрицы L,U.
for k=1:n-1
   for i=k+1:n
      L(i,k)= U(i,k)/U(k,k);
      U(i,k:n)= U(i,k:n)-L(i,k)*U(k,k:n);
      U(i,k)=0;
   end
end
```

При стандартном LU-разложении, если на некотором k-м шаге процесса гауссова исключения диагональный элемент оказывается равным нулю, дальнейшее выполнение алгоритма становится невозможным. Для устранения этой проблемы и одновременного повышения численной устойчивости широкое распространение получил алгоритм LU-разложения с выбором ведущего элемента .

Согласно [11], суть данного алгоритма заключается в том, что на каждом k-м шаге процесса исключения выполняется поиск элемента с наибольшим абсолютным значением в текущем k-м столбце, начиная с k-й строки и до конца столбца. Если этот максимальный элемент по модулю находится не в текущей k-й строке, а в некоторой строке p (где $p > k$), то строки k и p меняются местами. После такой перестановки процесс гауссова исключения продолжается обычным образом: производятся элементарные строчные преобразования для обнуления элементов, расположенных под новым ведущим элементом в k-м столбце, как это было описано ранее. В результате общее разложение принимает вид $PS = LU$ где P — это матрица перестановки. Алгоритм этого процесса можно представить следующим образом:

```
function [L,U, P] = performPartialPivotingLU(S)
  for k = 1:n-1
    [~, max_idx_rel] = max(abs(U(k:n, k)));
    max_idx_abs = max_idx_rel + k - 1;
      if max_idx_abs ~= k
       U([k, max_idx_abs], k:n) = U([max_idx_abs, k], k:n);
       P([k, max_idx_abs], :) = P([max_idx_abs, k], :);
   L([k, max_idx_abs], 1:k-1) = L([max_idx_abs, k], 1:k-1);
      end
  end
end
```

Нахождение обратной матрицы S^{-1} эквивалентно решению системы уравнений $Sx = I$ или $PSx = P$, где I - единичная матрица. После того, как выполнено разложение $PS = LU$, система уравнений $PSx = P$ преобразуется к виду $LUx = P$. Для нахождения каждого столбца x_j обратной матрицы S^{-1} необходимо решить систему линейных уравнений:

$$LUx_j = P_j, j = 1,2,\ldots,n. \quad (3)$$

где P_j — j-й столбец матрицы перестановки P; $n \times n$ — размер матрицы S.

Система уравнений (3) решается в два этапа:

1. Пусть $LUx_j = P_j$. Решение системы $Ly_j = P_j$ позволяет найти ромежуточный вектор y_j.

2. Решение систему $Ux_j = y_j$ позволяет найти вектора x_j.

Первая система уравнений $Ly_j = P_j$ в матричной форме представляется следующим образом:

$$\begin{pmatrix} 1 & 0\ldots & 0 \\ l_{21} & 1\ldots & 0 \\ \ldots & \ldots\ddots & \ldots \\ l_{n1} & l_{n2}\ldots & 1 \end{pmatrix} \begin{pmatrix} y_1^{(j)} \\ y_2^{(j)} \\ \ldots \\ y_n^{(j)} \end{pmatrix} = \begin{pmatrix} P_1^{(j)} \\ P_2^{(j)} \\ \ldots \\ P_n^{(j)} \end{pmatrix}.$$

Компоненты вектора $y_i^{(j)}$ вычисляются последовательно сверху вниз (метод прямой подстановки) следующим образом:

- при $i = 1$ (первая строка): $y_1^{(j)} = P_1^{(j)}$ (так как $l_{11} = 1$, а остальные внедиагональные элементы первой строки матрицы L равны нулю);

- при $i = 2$ имеем $l_{21}y_1^{(j)} + y_2^{(j)} = P_2^{(j)}$ и поскольку $y_1^{(j)}$ уже известен, вычисляем $y_2^{(j)} = P_2^{(j)} - l_{21}y_1^{(j)}$;

- при $i = 3$ имеем $l_{31}y_1^{(j)} + l_{32}y_2^{(j)} + y_3^{(j)} = P_3^{(j)}$, вычисляем $y_3^j = P_3^{(j)} - (l_{31}y_1^{(j)} + l_{32}y_2^{(j)})$.

В общем виде для i от 1 до n:

$$y_i^{(j)} = P_i^{(j)} - \sum_{k=1}^{i-1} l_{ik}y_k^{(j)}.$$

После того как вектор $y_j = \left(y_1^{(j)}, y_2^{(j)}, \ldots, y_n^{(j)}\right)^T$ определён, приступают к решению системы уравнений с верхнетреугольной матрицей $Ux_j = y_j$ для нахождения вектора $x_j = \left(x_1^{(j)}, x_2^{(j)}, \ldots, x_n^{(j)}\right)^T$. В матричной форме данная система уравнений имеет следующий вид:

$$\begin{pmatrix} u_{11} & u_{12}\ldots & u_{1n} \\ 0 & u_{22}\ldots & u_{2n} \\ \ldots & \ldots\ddots & \ldots \\ 0 & 0\ldots & u_{nn} \end{pmatrix} \begin{pmatrix} x_1^{(j)} \\ x_2^{(j)} \\ \ldots \\ x_n^{(j)} \end{pmatrix} = \begin{pmatrix} y_1^{(j)} \\ y_2^{(j)} \\ \ldots \\ y_n^{(j)} \end{pmatrix}.$$

Компоненты $x_i^{(j)}$ вектора x_j вычисляются последовательно в порядке снизу вверх (метод обратной подстановки):

- при $i = n$ имеем $u_{nn}x_n^{(j)} = y_n^{(j)}$, откуда

$$x_n^{(j)} = \frac{y_n^{(j)}}{u_{nn}};$$

- при $i = n-1$ имеем

$$u_{n-1,n-1}x_{n-1}^{(j)} + u_{n-1,n}x_n^{(j)} = y_{n-1}^{(j)};$$

- поскольку $x_n^{(j)}$ известен, находим:

$$x_{n-1}^{(j)} = \frac{y_{n-1}^{(j)} - u_{n-1,n}x_n^{(j)}}{u_{n-1,n-1}}.$$

В общем виде, для $i = n, n-1, \ldots, 1$ компоненты $x_i^{(j)}$ вычисляются по формуле:

$$x_i^{(j)} = \frac{1}{u_{ii}}(y_i^{(j)} - \sum_{k=i+1}^{n} u_{ik}x_k^{(j)}).$$

После выполнения двух этапов решения для конкретного j-го получают вектор $x_j = \left(x_1^{(j)}, x_2^{(j)}, \ldots, x_n^{(j)}\right)^T$. Этот вектор и является j-м столбцом искомой обратной матрицы S^{-1}. Повторение данной процедуры для всех n столбцов (т.е. для $j = 1,2,\ldots,n$) позволяет полностью построить матрицу S^{-1}. Имея S^{-1}, можно затем применить формулу (2) для нахождения коэффициентов полинома локаторов ошибок, после чего использовать ранее описанный алгоритм для осуществления процесса декодирования кодов БЧХ.

Разложение матрицы S на произведение нижней треугольной матрицы L и верхней

треугольной матрицы U требует выполнения $\frac{n^3}{3}$ операций умножения и $\frac{n^3}{3}$ операций сложения. Решение двух систем треугольных уравнений $(Ly_j = P_j, Ux_j = y_j)$ требует выполнения примерно n^2 операций умножения и n^2 операций сложения. Эта процедура повторяется для всех n столбцов обратной матрицы, поэтому общее количество операций для определения обратной матрицы составляет n^3 операций умножения и n^3 операций сложения. Следовательно, общая вычислительная сложность метода нахождения обратной матрицы путем LU-разложения и последовательного решения систем составляет $O(8n^3/3)$, что в общем виде представляется как $O(n^3)$.

4. Метод определения полинома локаторов ошибок на основе рекуррентного алгоритма Левинсона

Соотношение (1) можно переписать в следующем виде:

$$\begin{pmatrix} S_\mu & S_{\mu-1}\dots & S_1 \\ S_{\mu+1} & S_\mu\dots & S_2 \\ \vdots & \dots\ddots & \vdots \\ S_{2\mu-1} & S_{2\mu-2}\dots & S_\mu \end{pmatrix} \begin{pmatrix} \Delta_1 \\ \Delta_2 \\ \vdots \\ \Delta_\mu \end{pmatrix} = \begin{pmatrix} -S_{\mu+1} \\ -S_{\mu+2} \\ \vdots \\ -S_{2\mu} \end{pmatrix}. \quad (4)$$

Исходная синдромная матрица трансформируется в тёплицеву матрицу (4). Тёплицевой называется матрица, в которой все элементы на каждой диагонали, параллельной главной, одинаковы**.** Для тёплицевой матрицы A её элемент A_{ij} зависит только от разности индекса строки i и индекса столбца j, то есть $A_{ij} = a_{i-j}$. Благодаря этому свойству, квадратная тёплицева матрица размера $n \times n$ полностью задаётся $2n-1$ своими независимыми элементами. Общий вид тёплицевой матрицы:

$$A = \begin{pmatrix} a_0 & a_{-1}\dots & a_{-n+1} \\ a_1 & a_0\dots & a_{-n+2} \\ \dots & \dots & \dots \\ a_{n-1} & a_{n-2}\dots & a_0 \end{pmatrix}. \quad (5)$$

Для удобства изложения алгоритма будем рассматривать (5) как синдромную. Поскольку A является тёплицевой матрицей, процедура вычисления её обратной матрицы A^{-1} основывается на использовании присущих тёплицевым матрицам математических свойств и их особой структуры.

Будем исходить из предположения, что все ведущие миноры матрицы A отличны от нуля. Это является условием её невырожденности и позволяет применять определённые методы обращения. Ведущая главная подматрица A_k размера $(k+1)\times(k+1)$ матрицы A имеет следующий вид:

$$A_k = \begin{pmatrix} a_0 & a_{-1}\dots & a_{-k} \\ a_1 & a_0\dots & a_{-k+1} \\ \dots & \dots & \dots \\ a_k & a_{k-1}\dots & a_0 \end{pmatrix}.$$

Для ведущей подматрицы A_k определяются векторы

$$x^{(k)} = \left(x_0^{(k)}, x_1^{(k)}, \dots, x_k^{(k)}\right)^T,$$

$$y^{(k)} = \left(y_0^{(k)}, y_1^{(k)}, \dots, y_k^{(k)}\right)^T.$$

Утверждается, что эти векторы, представляющие собой первый и последний столбцы обратной матрицы A_k^{-1}, позволяют полностью её определить. Значения векторов $x^{(k)}$ и $y^{(k)}$ вычисляются рекуррентно на основе ряда вспомогательных величин. Эти рекуррентно вычисляемые векторы служат основой для последовательного построения полной обратной матрицы A^{-1} на последнем этапе, при $k = n-1$. Как показано в [12], векторы $x^{(k)}$ и $y^{(k)}$ могут быть представлены в виде линейной комбинации двух других векторов:

$$\alpha x^{(k)} = \begin{pmatrix} x^{(k-1)} \\ 0 \end{pmatrix} + \beta \begin{pmatrix} 0 \\ y^{(k-1)} \end{pmatrix},$$
$$\alpha y^{(k)} = \begin{pmatrix} x^{(k-1)} \\ 0 \end{pmatrix} \gamma + \begin{pmatrix} 0 \\ y^{(k-1)} \end{pmatrix}, \quad (6)$$

где $x^{(k-1)}$ и $y^{(k-1)}$ — это соответственно, первый и последний столбцы обратной матрицы A_{k-1}^{-1} и α, β, γ — коэффициенты линейной комбинаций. Для их нахождения приведённые выше векторные уравнения (6) умножаются слева на матрицу A_k:

$$\alpha A_k x^{(k)} = A_k \begin{pmatrix} x^{(k-1)} \\ 0 \end{pmatrix} + A_k \beta \begin{pmatrix} 0 \\ y^{(k-1)} \end{pmatrix},$$
$$\alpha A_k y^{(k)} = A_k \gamma \begin{pmatrix} x^{(k-1)} \\ 0 \end{pmatrix} + A_k \begin{pmatrix} 0 \\ y^{(n-1)} \end{pmatrix}. \quad (7)$$

Поиск первого и последнего столбцов обратной матрицы по существу сводится к решению СЛАУ. В частности, векторы $x^{(k-1)}$ и $y^{(k-1)}$ являются решениями систем:

$$A_{k-1} x^{(k-1)} = e_1,\ A_{k-1} y^{(k-1)} = e_k, \quad (8)$$

где e_1, e_k — это соответственно первый и последний столбцы единичной матрицы I размера $k \times k$ (матрица A_{k-1} имеет размер $k \times k$).

Применяем (8) в (7), получаются:

$$(\alpha, 0, \ldots, 0)^T = (1, 0, \ldots, 0, F)^T + \\ + (G, 0, \ldots, 0, 1)^T \beta,$$
$$(0, 0, \ldots, \alpha)^T = (1, 0, \ldots, 0, F)^T \gamma + \\ + (G, 0, \ldots, 0, 1)^T. \quad (9)$$

Из (9) определены

$$\alpha = 1 - FG,\ \beta = -F,\ \gamma = -G. \quad (10)$$

Тогда

$$F = a_k x_0^{(k-1)} + a_{k-1} x_0^{(k-1)} + \ldots + a_1 x_{k-1}^{(k-1)},$$
$$G = a_{-1} y_0^{(k-1)} + a_{-2} y_1^{(k-1)} + \ldots + a_{-k} y_{k-1}^{(k-1)}. \quad (11)$$

Векторы $x^{(k)}$, $y^{(k)}$ по (6) представляются в виде:

$$\left(x_0^{(k)}, x_1^{(k)}, \ldots, x_k^{(k)}\right)^T = \left(x_0^{(k-1)}, x_1^{(k-1)}, \ldots x_{k-1}^{(k-1)}, 0\right)^T r_k + \\ + \left(0, y_0^{(k-1)}, y_1^{(k-1)}, \ldots y_{k-1}^{(k-1)}\right)^T s_k,$$
$$\left(y_0^{(k)}, y_1^{(k)}, \ldots y_k^{(k)}\right)^T = \left(x_0^{(k-1)}, x_1^{(k-1)}, \ldots x_{k-1}^{(k-1)}, 0\right)^T t_k + \\ + \left(0, y_0^{(k-1)}, y_1^{(k-1)}, \ldots y_{k-1}^{(k-1)}\right)^T r_k.$$

Коэффициенты определяются как:

$$r_k = 1/(1 - F_k G_k),\ s_k = \frac{\beta}{(1 - F_k G_k)} = -r_k F_k,$$
$$t_k = \frac{\gamma}{(1 - F_k G_k)} = -r_k G_k.$$

Если векторы $x^{(k-1)}$, $y^{(k-1)}$ заранее известны, то на основе (11), (10) вычисляются коэффициенты линейной комбинации α, β, γ, а затем из них вычисляются векторы $x^{(k)}$, $y^{(k)}$. При $k = n-1$ будут найдены $x^{(n-1)}, y^{(n-1)}$ и обратная матрица A_{n-1}^{-1}. Таким образом, суть рекуррентного алгоритма построения обратной матрицы A_k^{-1} заключается в использовании и обновлении информации, полученной из обратной матрицы A_{k-1}^{-1}, вычисленной на предыдущем шаге. Конкретный рекуррентный алгоритм подробно представлен ниже:

$k = 0$:

$$x_0^{(0)} = y_0^{(0)} = 1/a_0;$$

$k = 1, \ldots, n-1$:

$$F_k = \sum_{i=1}^{k} a_i x_{k-i}^{(k-1)} = a_k x_0^{(k-1)} + a_{k-1} x_0^{(k-1)} + \ldots + a_1 x_{k-1}^{(k-1)},$$
$$G_k = \sum_{i=1}^{k} a_{-i} y_{i-1}^{(k-1)} = a_{-1} y_0^{(k-1)} + a_{-2} y_1^{(k-1)} + \ldots + a_{-k} y_{k-1}^{(k-1)}.$$

$k = 1$:

$$F_1 = a_1 x_0^{(0)}, G_1 = a_{-1} y_0^{(0)},\ r_1 = 1/(1 - F_1 G_1),$$
$$s_1 = -r_1 F_1, t_1 = -r_1 G_1.$$

Осуществляется обновление: $\left(x_0^{(1)}, x_1^{(1)}\right)^T$, $\left(y_0^{(1)}, y_1^{(1)}\right)^T$:

$$\begin{pmatrix} x_0^{(1)} \\ x_1^{(1)} \end{pmatrix} = \begin{pmatrix} x_0^{(0)} \\ 0 \end{pmatrix} r_1 + \begin{pmatrix} 0 \\ y_0^{(0)} \end{pmatrix} s_1,$$

$$\begin{pmatrix} y_0^{(1)} \\ y_1^{(1)} \end{pmatrix} = \begin{pmatrix} x_0^{(0)} \\ 0 \end{pmatrix} t_1 + \begin{pmatrix} 0 \\ y_0^{(0)} \end{pmatrix} r_1.$$

$k = 2$:

$$F_2 = a_2 x_0^{(1)} + a_1 x_1^{(1)}, G_2 = a_{-1} y_0^{(1)} + a_{-2} y_1^{(1)},$$

$$r_2 = 1/(1 - F_2 G_2), s_2 = -r_2 F_2, t_2 = -r_2 G_2.$$

Осуществляется обновление:

$$\begin{pmatrix} x_0^{(2)} \\ x_1^{(2)} \\ x_2^{(2)} \end{pmatrix} = \begin{pmatrix} x_0^{(1)} \\ x_1^{(1)} \\ 0 \end{pmatrix} r_2 + \begin{pmatrix} 0 \\ y_0^{(1)} \\ y_1^{(1)} \end{pmatrix} s_2,$$

$$\begin{pmatrix} y_0^{(2)} \\ y_1^{(2)} \\ y_2^{(2)} \end{pmatrix} = \begin{pmatrix} x_0^{(1)} \\ x_1^{(1)} \\ 0 \end{pmatrix} t_2 + \begin{pmatrix} 0 \\ y_0^{(1)} \\ y_1^{(1)} \end{pmatrix} r_2,$$

до тех пор, пока $k = n - 1$:

$$\begin{pmatrix} x_0^{(n-1)} \\ x_1^{(n-1)} \\ \dots \\ x_{n-1}^{(n-1)} \end{pmatrix} = \begin{pmatrix} x_0^{(n-2)} \\ x_1^{(n-2)} \\ \dots \\ 0 \end{pmatrix} r_{n-1} + \begin{pmatrix} 0 \\ \dots \\ y_0^{(n-2)} \\ y_{n-1}^{(n-2)} \end{pmatrix} s_{n-1},$$

$$\begin{pmatrix} y_0^{(n-1)} \\ y_1^{(n-1)} \\ \dots \\ y_{n-1}^{(n-1)} \end{pmatrix} = \begin{pmatrix} x_0^{(n-2)} \\ x_1^{(n-2)} \\ \dots \\ 0 \end{pmatrix} t_{n-1} + \begin{pmatrix} 0 \\ \dots \\ y_0^{(n-2)} \\ y_{n-1}^{(n-2)} \end{pmatrix} r_{n-1}.$$

По завершении итерационного процесса вычисляются векторы:

$$x^{(n-1)} = \left(x_0^{(n-1)}, x_1^{(n-1)}, \dots, x_{n-1}^{(n-1)} \right)^T,$$

$$y^{(n-1)} = \left(y_0^{(n-1)}, y_1^{(n-1)}, \dots, y_{n-1}^{(n-1)} \right)^T.$$

Эти векторы играют ключевую роль в последующем построении всех остальных элементов обратной матрицы, что возможно благодаря специфическим свойствам тёплицевых матриц. Согласно [12], выражение для определения обратной матрицы A^{-1} имеет следующий вид:

$$A^{-1} = x_0^{-1} \left\{ \begin{pmatrix} x_0 & 0 \dots & 0 \\ x_1 & x_0 \dots & 0 \\ \dots & \dots\dots & \\ x_{n-1} & x_{n-2} \dots & x_0 \end{pmatrix} \begin{pmatrix} y_{n-1} & y_{n-2} \dots & y_0 \\ & y_{n-1} \dots & y_1 \\ 0 & \dots\dots & \dots \\ 0 & 0 \dots\dots & y_{n-1} \end{pmatrix} - \right.$$

$$\left. - \begin{pmatrix} 0 & \dots & 0 \\ y_0 & 0 & \\ \dots & \dots & 0 \\ y_{n-2} & y_{n-3} \dots & 0 \end{pmatrix} \begin{pmatrix} 0 & x_{n-1} \dots & x_1 \\ & \dots\dots & \dots \\ 0 & \dots 0 & x_{n-1} \\ 0 & \dots & 0 \end{pmatrix} \right\}. \quad (12)$$

Если целью является построение обратной матрицы A^{-1} путём поэлементного вычисления на основе формулы (12), то необходимо принимать во внимание вычислительную сложность такого подхода. Формирование обратной матрицы размера $n \times n$ общими методами обычно сопряжено с вычислительными затратами порядка $O(n^3)$.

В работе [12] указывается, что если произвольная матрица может быть представлена в виде суммы t слагаемых, каждое из которых является произведением верхнетреугольной и нижнетреугольной матриц, то это может свидетельствовать о её специфической структуре, допускающей применение более эффективных алгоритмов.

$$\begin{pmatrix} \alpha_1^{(S)} & \dots & 0 \\ \alpha_2^{(S)} & \alpha_1^{(S)} \dots & \\ \dots & \dots\dots & 0 \\ \alpha_{n-1}^{(S)} & \alpha_{n-2}^{(S)} \dots & \alpha_1^{(S)} \end{pmatrix} \begin{pmatrix} \beta_1^{(S)} & \beta_2^{(S)} \dots & \beta_n^{(S)} \\ & \beta_1^{(S)} \dots & \beta_{n-1}^{(S)} \\ 0 & \dots\dots & \dots \\ & \dots\dots & \beta_1^{(S)} \end{pmatrix}.$$

Для элементов каждого слагаемого может быть записано:

$$a_{i,j} = a_{i-1,j-1} + \sum_{S=1}^{t} \alpha_i^{(S)} \beta_j^{(S)}.$$

В рамках данной статьи основное внимание уделяется компонентам матрицы, которая представляет собой произведение одной пары специальных тёплицевых матриц: нижней треугольной $L(u)$ и верхней треугольной $U(v)$. В этом частном рекуррентное соотношение для вычисления элементов $a_{i,j}$ произведения $L(u)U(v)$ значительно упрощается и принимает вид:

$$a_{i,j} = a_{i-1,j-1} + \alpha_i \beta_j. \quad (13)$$

Компоненты, формирующие A^{-1} в рамках такого представления, могут быть полностью вычислены посредством рекуррентного

соотношения (13). Из (12) определяют всю обратную матрицу A^{-1}. Алгоритм (13) реализуется следующими конкретными шагами:

```
function resultMatrix = computeLU_Product(uVector, vVector, matrixSize)
    Вход:
    uVector: Вектор-столбец L(u),
    vVector: Вектор-строка U(v),
    matrixSize: Размер квадратной матрицы (n*n)
    Выход:
    resultMatrix: L(u)U(v)

    %Шаг 1: Вычислить первую строку resultMatrix
    firstU = uVector(1); % Первый элемент uVector
    for col = 1:matrixSize
        resultMatrix(1, col) = firstU * vVector(col);
    end
    %Шаг 2: Вычислить первый столбец resultMatrix
    firstV = vVector(1); % Первый элемент vVector
    for row = 2:matrixSize
        resultMatrix(row, 1) = uVector(row) * firstV;
    end
    %Шаг 3: Вычислить оставшиеся элементы
    for row = 2:matrixSize
     for col = 2:matrixSize
      resultMatrix(row, col)= resultMatrix(row-1,col-1)+...
                              + uVector(row) * vVector(col);
      end
     end
end
```

Процесс нахождения обратной матрицы A^{-1} для тёплицевой матрицы размера $n \times n$ включает два основных этапа: первый заключается в вычислении векторов $x^{(n-1)}, y^{(n-1)}$, а второй — в построении всех элементов матрицы A^{-1}. На первом этапе рекуррентный алгоритм вычисления векторов $x^{(n-1)}, y^{(n-1)}$ требует выполнения порядка $3n^2$ операций умножения и $2n^2$ операций сложения. Второй этап, то есть восстановление полной матрицы A^{-1} по этим векторам, также оценивается примерно в $3n^2$ умножений и $3n^2$ сложений.

Таким образом, изложенный метод обращения тёплицевых матриц обладает общей вычислительной сложностью $O(n^2)$, что демонстрирует его существенное преимущество в эффективности по сравнению с общими методами обращения матриц, имеющими сложность $O(n^3)$.

3. Результаты моделирования и их оценка

После теоретического анализа и оценки вычислительной сложности двух подходов к декодированию кодов БЧХ на основе алгоритма ПГЦ следующим этапом является оценка их практической эффективности.

Будет проведено сравнение эффективности исправления ошибок для двоичных кодов БЧХ$(15,11)$ с возможностью исправления $t=1$ ошибки, БЧХ$(15,7)$ с $t=2$ ошибками и БЧХ$(15,5)$ с $t=3$ ошибками. Декодирование этих кодов будет осуществляться посредством алгоритма ПГЦ, в котором полином локаторов ошибок будет вычисляться каждым из двух упомянутых методов. Также, для сопоставления будет оценена производительность стандартного алгоритма Берлекэмпа–Месси, используемого в качестве эталонного. Моделирование планируется провести по каналам передачи АБГШ с применением квадратурной фазовой манипуляции (QPSK).

На рис. 1. представлены результаты по зависимости коэффициента битовых ошибок (BER, *аббр. от англ* Bit Error Rate) от отношения сигнал/шум (ОСШ/SNR, *аббр. от англ* Signal-to-Noise Ratio). Результаты моделирования показывают, что для одного и того же кода БЧХ и идентичных условий канала передачи, эффективность исправления ошибок при применении различных

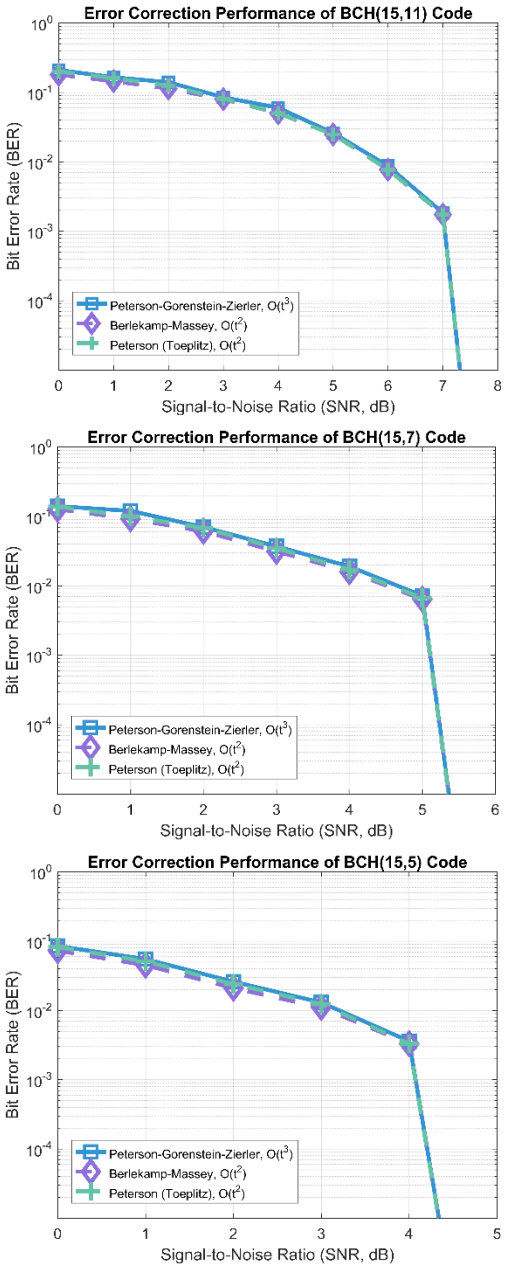

Рис. 1. Эффективность исправления ошибок кодов БЧХ

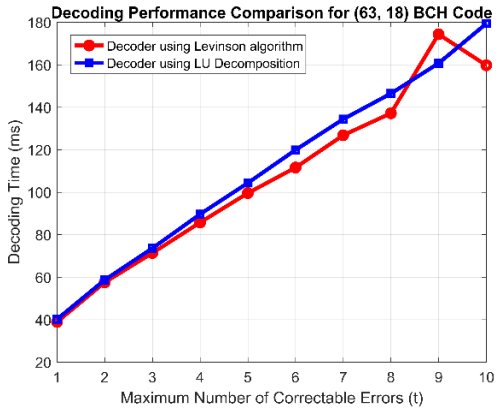

Рис. 2. Зависимость времени декодирования от числа исправляемых ошибок t

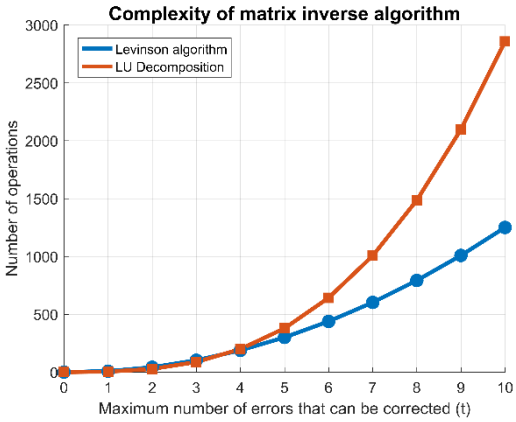

Рис. 3. Зависимость количества операций от числа исправляемых ошибок t

исследуемых алгоритмов декодирования является сопоставимой во всём рассмотренном диапазоне ОСШ. Коды с большей конструктивной корректирующей способностью t закономерно демонстрируют лучшую помехоустойчивость. Например, для кода БЧХ(15,5) достижение уровня $BER = 10^{-4}$ требует ОСШ порядка 4,3 дБ, в то время как для кода БЧХ(15,7) необходим ОСШ = 5,2 дБ, а для БЧХ(15,11) – ОСШ = 7,2 дБ.

Для детализации теоретического анализа вычислительной сложности и оценки реальной производительности двух методов обработки синдромной матрицы в рамках алгоритма ПГЦ было проведено численное моделирование. В ходе него осуществлялся сбор данных о времени декодирования и количестве элементарных арифметических операций в процессе определения полинома локаторов ошибок. Сравнение времени обработки представлено на рис. 2, а рис. 3 иллюстрирует различия в количестве арифметических операций для этих двух подходов при изменении количества исправляемых ошибок t кодов БЧХ. В рамках данного исследования проводится оценка производительности

двоичного кода БЧХ(63,18), обладающего максимальной корректирующей способностью $t_{\max} = 10$ [13]. Для создания контролируемой экспериментальной среды в качестве передаваемой кодовой комбинации принимается нулевая кодовая комбинация. Ошибки вводятся преднамеренно путём сложения вектора ошибок $e(x)$, имеющего вес Хэмминга t, с нулевой кодовой комбинацией. Производительность декодера исследуется путём варьирования фактического числа ошибок t в диапазоне от 1 до 10 со способностью исправления *t*. Моделирование было выполнено несколько раз, чтобы получить среднее время выполнения для каждого значения *t*.

Из рис. 2 видно, что реальное время обработки для обоих рассматриваемых методов увеличивается с ростом значений t. При малых значениях *t* (например, *t* = 1, 2, 3) разница во времени обработки между двумя подходами незначительна. Однако, с увеличением t кривая, соответствующая методу на основе *LU*-разложения $O(t^3)$, демонстрирует более выраженный рост времени обработки по сравнению с методом, использующим рекуррентные вычисления $O(t^2)$. Указанное различие в скорости роста сложности отражает принципиальную разницу в вычислительной трудоёмкости сравниваемых алгоритмических частей, хотя итоговое реальное время обработки также зависит от множества факторов реализации и аппаратной платформы.

На рис. 3 представлена зависимость вычислительной сложности процесса поиска полинома локаторов ошибок от числа исправляемых ошибок t. При $t \le 4$ количество арифметических операций для обоих методов практически совпадает. При $t = 7$ метод, ассоциируемый со сложностью $O(t^2)$, оказывается эффективнее примерно на 400 арифметических операций по сравнению с методом, сложность которого для данного этапа оценивается как $O(t^3)$. При $t = 10$ это преимущество становится ещё более заметным, демонстрируя превосходство алгоритма с меньшей вычислительной сложностью при обработке кодов с большей конструктивной корректирующей способностью.

Заключение

Алгоритм декодирования Питерсона — Горенстейна — Цирлера (ПГЦ) — это эффективный алгебраический метод, широко применяемый для циклических кодов, таких как коды БЧХ и Рида-Соломона, основанный на их структуре и фундаментальных математических свойствах. Процесс декодирования по этому алгоритму включает следующие ключевые этапы:

1. Вычисление вектора синдромов на основе принятого кодового слова; значения синдромов несут информацию о наличии (или отсутствии) ошибок.

2. Построение полинома локаторов ошибок. Этот полином составляется и затем решается для отыскания его корней. Данные корни напрямую указывают на местоположения ошибок в принятом кодовом слове.

3. Определение величин ошибок и их исправление.

В статье был проведён детальный анализ и сравнение вычислительной сложности двух подходов к определению полинома локаторов ошибок в рамках алгоритма декодирования ПГЦ. Теоретический анализ количества операций показал, что рекуррентный метод на основе алгоритма Левинсона обладает явным преимуществом по сравнению с общим методом *LU*-разложения. Хотя эффективность исправления ошибок для обоих подходов при декодировании одного и того же кода БЧХ является сопоставимой, разница в вычислительных затратах весьма существенна, особенно для больших значений t. Данное обстоятельство открывает перспективы для дальнейших исследований, нацеленных на

комплексную оптимизацию всего процесса декодирования.

Литература

1. *Sun F., Rose K., Zhang T.* On the Use of Strong BCH Codes for Improving Multilevel NAND Flash Memory Storage Capacity // IEEE Workshop on Signal Processing Systems (SiPS): Design and Implementation, 2006. Vol. 5. Pp. 1–5.

2. *Micheloni R., Marelli A., Ravasio R.* Error correction codes for non-volatile memories. Springer Science & Business Media, 2008. 337 p.

3. *Chen Z., Zhang Y, Ying Y., Wu C., Zeng. X.* An area-efficient and degree-computationless BCH decoder for DVB-S2 // 2009 IEEE 8th International Conference on ASIC. DOI: 10.1109/ASICON.2009.5351625.

4. *Peterson W.* Encoding and error-correction procedures for the Bose-Chaudhuri codes // IRE Transactions on Information Theory. 1960. Vol. 6. Iss. 4. Pp. 459–470.

5. *Gorenstein D., Zierler N.* A Class of Error-Correcting Codes in pm Symbols // Journal of the Society for Industrial and Applied Mathematics (SIAM). 1961. Vol. 9. No. 2. Pp. 207–214.

6. *Кудряшов Б.Д.* Основы теории кодирования: учеб. пособие. СПб.: БХВ-Петербург, 2015. 400 с.

7. *Joiner L.L., Komo J.J.* Decoding binary BCH codes // Proceedings IEEE Southeastcon'95. Visualize the Future. DOI: 10.1109/SECON.1995.513059.

8. *Burton. H.* Inversionless decoding of binary BCH codes // IEEE Transactions on Information Theory. 1971. Vol. 17. Iss. 4. Pp. 464–466.

9. *Chien R.* Cyclic decoding procedures for Bose-Chaudhuri-Hocquenghem codes // IEEE Transactions on Information Theory. 1964. Vol. 10. Iss. 4. Pp. 357–363.

10. *Hong J., Vetterli. M.* Simple algorithms for BCH decoding // IEEE Transactions on Communications. 1995. Vol. 43. Iss. 8. Pp. 2324–2333.

11. *Даутов Р.З., Тимербаев М.Р.* Численные методы. Решение задач линейной алгебры и дифференциальных уравнений: учебное пособие. Казань: КФУ, 2021. 168 с.

12. *Воеводин В.В., Тыртышников Е.Е.* Вычислительные процессы с теплицевыми матрицами. М.: Наука. Гл. ред. физ-мат. лит., 1987. 320 с.

13. *Варгаузин В.А., Цикин И.А.* Методы повышения энергетической и спектральной эффективности цифровой радиосвязи: учебное пособие. СПб.: БХВ-Петербург, 2013. 352 с.

Поступила 5 июля 2025 г.

English

INVESTIGATION OF THE COMPUTATIONAL EFFICIENCY OF DECODING BOSE–CHAUDHURI–HOCQUENGHEM CODES USING THE PETERSON–GORENSTEIN–ZIERLER ALGORITHM

Vu The Duyet — Postgraduate Student, Department of Radio Engineering, The Bonch-Bruevich Saint-Petersburg State University of Telecommunications.
E-mail: vu.tz@sut.ru

Evgeny Ivanovich Glushankov — Grand Dr. in Engineering, Professor, Department of Radio Engineering, The Bonch-Bruevich Saint Petersburg State University of Telecommunications.

Address: 193232, Russian Federation, Saint Petersburg, Bolshevikov ave., 22/1.

Abstract: This paper presents an analysis of the computational efficiency of various existing methods and a novel proposed approach for determining the error locator polynomial in the decoding of Bose–Chaudhuri–Hocquenghem codes using the Peterson–Gorenstein–Zierler (PGZ) algorithm. Initially, a theoretical analysis of the computational complexity of the methods under consideration is conducted. Based on this analysis, the corresponding decoding algorithms are implemented and simulated in the MATLAB environment, which allows for an evaluation and comparison of the optimality of the considered methods for decoding specific BCH codes. The scientific novelty of this research lies in enhancing the computational efficiency of the PGZ decoding algorithm by exploiting the special structure of the syndrome matrix, which possesses the properties of a Toeplitz matrix. All investigated decoding methods demonstrated comparable effectiveness in error correction. However, the method based on the Levinson algorithm, designed to exploit the special properties of the syndrome matrix, exhibited superior computational efficiency.

Keywords: Bose–Chaudhuri–Hocquenghem codes, error locator polynomial, Peterson–Gorenstein–Zierler algorithm, decoding algorithm, syndrome matrix, Toeplitz matrix, Levinson algorithm, computational efficiency, arithmetic operations, MATLAB.

References

1. *Sun F., Rose K., Zhang T.* On the Use of Strong BCH Codes for Improving Multilevel NAND Flash Memory Storage Capacity. IEEE Workshop on Signal Processing Systems (SiPS): Design and Implementation, 2006. Vol. 5. Pp. 1–5.

2. *Micheloni R., Marelli A., Ravasio R.* Error correction codes for non-volatile memories. Springer Science & Business Media, 2008. 337 p.

3. *Chen Z., Zhang Y, Ying Y., Wu C., Zeng. X.* An area-efficient and degree-computationless BCH decoder for DVB-S2 // 2009 IEEE 8th International Conference on ASIC. DOI: 10.1109/ASICON.2009.5351625.

4. *Peterson W.* Encoding and error-correction procedures for the Bose-Chaudhuri codes. IRE Transactions on Information Theory. 1960. Vol. 6. Iss. 4. Pp. 459–470.

5. *Gorenstein D., Zierler N.* A Class of Error-Correcting Codes in pm Symbols. Journal of the Society for Industrial and Applied Mathematics (SIAM). 1961. Vol. 9. No. 2. Pp. 207–214.

6. *Kudryashov B.D.* Fundamentals of coding theory: textbook. stipend. St. Petersburg: BHV-Petersburg, 2015. 400 p.

7. *Joiner L.L., Komo J.J.* Decoding binary BCH codes. Proceedings IEEE Southeastcon'95. Visualize the Future. DOI: 10.1109/SECON.1995.513059.

8. *Burton. H.* Inversionless decoding of binary BCH codes. IEEE Transactions on Information Theory. 1971. Vol. 17. Iss. 4. Pp. 464–466.

9. *Chien R.* Cyclic decoding procedures for Bose- Chaudhuri-Hocquenghem codes. IEEE Transactions on Information Theory. 1964. Vol. 10. Iss. 4. Pp. 357–363.

10. *Hong J., Vetterli. M.* Simple algorithms for BCH decoding. IEEE Transactions on Communications. 1995. Vol. 43. Iss. 8. Pp. 2324–2333.

11. *Dautov R.Z., Timerbaev M.R.* Numerical methods. Solving problems of linear algebra and differential equations: a textbook. Kazan: KFU, 2021. 168 p.

12. *Voevodin V.V., Tyrtyshnikov E.E.* Computational Processes with Teplic Matrix. Moscow: Nauka. the main editorial office of the physical and mathematical literature, 1987. 320 p.

13. *Vargauzin V.A., Tsikin I.A.* Methods of Increasing Energy and Spectral Efficiency of Digital Radio Communication: Textbook. Saint-Petersburg: BHV-Peterburg, 2013. 352 p.

.