

УДК 621.396.41

Шифр специальности ВАК: 2.2.13

А.Н. Кравченко, Д.А. Покаместов

Методы кодирования случайных кодов с малой плотностью проверок на четность

Актуальность. Разработка и исследование методов кодирования случайных кодов с малой плотностью проверок на четность (МППЧ) является важной задачей при построении высокоскоростных систем связи, поскольку различные алгоритмы формирования кодовых слов существенно отличаются по вычислительным затратам и требованиям к реализации. **Цель исследования.** Сравнительный анализ методов кодирования МППЧ-кодов, заданных случайными разреженными проверочными матрицами, с оценкой их вычислительной сложности и практической производительности. **Методы.** Используются методы линейной алгебры, аналитическая оценка вычислительной сложности, имитационное моделирование и метод Монте-Карло. В качестве исследуемых методов кодирования рассмотрены метод Нила, алгоритмы на основе LU- и LUP-разложений, приведение проверочной матрицы к ступенчатой форме, а также методы нижней и приближенной нижней триангуляции. **Научная новизна.** Проведено сравнительное исследование методов кодирования случайных МППЧ-кодов, включающее совместную оценку их аналитической вычислительной сложности и времени программной реализации при одинаковых исходных условиях. **Результаты.** Установлены различия в характере роста вычислительных затрат рассмотренных методов при увеличении размера проверочной матрицы. Показано, что методы Нила, LU и приведения к ступенчатой форме характеризуются меньшим числом арифметических операций, тогда как результаты программного моделирования существенно зависят от способа реализации вычислений и возможностей оптимизации матричных операций в MatLab. **Практическая значимость.** Полученные результаты могут использоваться при выборе алгоритма кодирования случайных МППЧ-кодов и разработке программных и аппаратных кодеров для высокоскоростных систем связи.

Ключевые слова: LDPC, МППЧ, случайные матрицы, помехоустойчивое кодирование, вычислительная сложность.

DOI: 10.21293/1818-0442-2026-29-2-64-73

Введение

С развитием систем беспроводной связи наблюдается рост требований к пропускной способности, надежности и энергоэффективности передачи, обусловленный увеличением числа подключенных устройств и разнообразием сценариев применения. В рамках перехода к системам связи шестого поколения (6G) предполагается дальнейшее расширение возможностей сетей, включая поддержку массовых машинных коммуникаций, ультранадежной связи с малыми задержками и интеграцию с интеллектуальными сервисами [1]. В этих условиях ключевую роль играют методы помехоустойчивого кодирования, обеспечивающие эффективную передачу информации в сложных каналах связи. Коды с малой плотностью проверок на четность (МППЧ; англ. LDPC, low density parity check) занимают важное место среди современных кодовых конструкций благодаря своей способности обеспечивать производительность, близкую к пределу Шеннона, при относительно низкой вычислительной сложности декодирования [2]. В системах 5G LDPC-коды используются для кодирования пользовательских данных, а в контексте 6G рассматриваются их дальнейшие модификации и адаптивные версии, ориентированные на работу в сценариях, характерных для перспективных сетей беспроводной связи [3].

В настоящее время существует большое количество книг, публикаций, патентов, посвященных теории, построению, алгоритмам кодирования/декодирования, практической аппаратной реализации

МППЧ-кодов. Основные понятия и определения, характеризующие МППЧ-коды, можно найти в книге П.В. Трифонова [4]. Также в данной книге находится достаточно обширная библиография, позволяющая сформировать представление о МППЧ-кодах. Как известно [5], конструкции кодов можно разделить на две общие категории: случайные и структурированные. В свою очередь, структурированные коды построены на основе алгебраических и комбинаторных методов [5, 6].

Помимо МППЧ-кодов, в теории помехоустойчивого кодирования широко известны совершенные и циклические коды. Совершенные коды для определенных наборов параметров достигают границы Хэмминга и обеспечивают максимально эффективное использование кодового пространства с точки зрения гарантированного исправления заданного числа ошибок. Циклические коды благодаря своей алгебраической структуре допускают эффективную реализацию процедур кодирования и декодирования, в том числе с использованием регистров сдвига с обратными связями. Вместе с тем МППЧ-коды обладают большей гибкостью при выборе длины и скорости кода, допускают построение кодовых слов большой длины и обеспечивают высокую помехоустойчивость, близкую к пределу Шеннона, что обусловило их широкое применение в современных системах связи.

Для случайных МППЧ-кодов, задаваемых разреженными проверочными матрицами, важной задачей является выбор эффективного метода формирования кодового слова.

Случайное построение МППЧ-кодов заключается в конструировании разреженных случайных матриц проверки четности с использованием компьютерного поиска на основе определенного набора правил проектирования. К графам Таннера таких кодов предъявляется ряд требований: распределение степеней символьных и проверочных узлов, а также контроль циклов малой кратности. Классическая работа [7] показала, что коды, построенные на основе разреженных случайных матриц, обладают возможностью практической реализации декодирования и близостью к пределу Шеннона. Использование таких проверочных матриц позволяет формировать ансамбль кодов, для которого при увеличении длины кода вероятность ошибки стремится к нулю при скорости передачи, не превышающей пропускную способность канала.

Разреженность проверочных матриц $\mathbf{H}$ обеспечивает возможность эффективного итеративного декодирования с использованием алгоритма распространения доверия (BP) или его приближений [2]. Ряд работ последних лет посвящен разработке эффективных алгоритмов декодирования МППЧ-кодов [8, 9].

К настоящему времени опубликовано множество работ, посвященных конструированию МППЧ-кодов на основе случайных проверочных матриц. В качестве примера можно привести работу Мак-Кея [7]. Мак-Кей и Нил создали обширную библиотеку разреженных случайных проверочных матриц [10], пригодных для практического использования. Кроме того, коды из данной библиотеки могут быть использованы для оценки эффективности и сравнения различных алгоритмов декодирования.

В данной работе излагаются наиболее эффективные и пригодные для практической реализации алгоритмы и методы кодирования случайных МППЧ-кодов. Несмотря на значительное число исследований, посвященных вопросам построения и декодирования МППЧ-кодов, задачи кодирования для случайных проверочных матриц остаются недостаточно проработанными. В общем случае кодирование сводится к решению системы линейных уравнений большой размерности, что может приводить к высокой вычислительной сложности и численным проблемам.

Современные системы связи предъявляют требования высокой скорости обработки при ограниченных вычислительных ресурсах. В связи с этим разработка эффективных алгоритмов кодирования является важной задачей. В работе проводится анализ методов кодирования случайных МППЧ-кодов с точки зрения их применимости на практике. Рассматриваемые методы сопровождаются подробными примерами, что способствует лучшему пониманию их реализации и особенностей применения.

Представлена оценка вычислительной сложности, основанная на подсчете числа арифметических операций, что позволяет выполнить сравнение методов с точки зрения их масштабируемости. Кроме того, приведены результаты имитационного моделирования среднего времени кодирования в среде

MatLab, позволяющие оценить практическую производительность алгоритмов при работе со случайными проверочными матрицами различного размера.

Основные методы кодирования двоичных случайных МППЧ-кодов

Коды с малой плотностью проверок на четность представляют собой класс двоичных линейных блочных кодов, задаваемых разреженной проверочной матрицей $\mathbf{H} \in \{0,1\}^{m,n}$, где m – число проверочных (избыточных) битов, n – длина кодового слова. Скорость МППЧ-кодов определяется соотношением $R = k/n = 1 - m/n$, где k – длина сообщения [2].

Для практической реализации кодирования МППЧ-кодов предложен ряд методов, отличающихся по вычислительной сложности, требованиям к структуре проверочной матрицы и особенностям аппаратной реализации. В основе этих подходов лежат различные способы преобразования проверочной матрицы и решения соответствующих систем линейных уравнений. Далее рассмотрим основные из них.

Кодирование методом Нила [11]. Предположим, для кодирования используется проверочная матрица

$$\mathbf{H} = \left[\begin{array}{cccc|cccc} 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \end{array} \right], \quad (1)$$

которую можно разделить на две части $\mathbf{H} = [\mathbf{A}|\mathbf{B}]$, где $\mathbf{A}$ и $\mathbf{B}$ имеют размерности $m \times (n-m)$ и $m \times m$ соответственно. Кодовое слово $\mathbf{c}$ также делится на две части $\mathbf{c} = [\mathbf{u}|\mathbf{p}]$, где $\mathbf{u}$ – вектор из $(n-m)$ информационных битов, а $\mathbf{p}$ – вектор из m проверочных битов. Проверочное соотношение согласно [11] имеет вид

$$[\mathbf{A}|\mathbf{B}] \cdot \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \end{bmatrix} = \mathbf{0}, \quad (2)$$

которое можно записать следующим образом:

$$\mathbf{A} \cdot \mathbf{u} + \mathbf{B} \cdot \mathbf{p} = \mathbf{0} \quad (3)$$

или

$$\mathbf{p} = \mathbf{B}^{-1} \cdot \mathbf{A} \cdot \mathbf{u}. \quad (4)$$

Предварительно можно вычислить элементы матрицы $\mathbf{C}$, которые для проверочной матрицы (1) будут равны

$$\mathbf{C} = \mathbf{B}^{-1} \cdot \mathbf{A} = \left[\begin{array}{ccccc} 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \end{array} \right]. \quad (5)$$

Тогда проверочные биты можно вычислить в соответствии с выражением

$$\mathbf{p} = \mathbf{C} \cdot \mathbf{u}. \quad (6)$$

Пример 1. Допустим, имеется следующая последовательность информационных битов: $\mathbf{u} = [1 \ 1 \ 0 \ 1 \ 0]$. Тогда проверочные биты $\mathbf{p}$, вычисленные согласно уравнению (6), имеют вид $\mathbf{p} = [0 \ 1 \ 1 \ 1 \ 1]$. Вери-

фикация формирования кодового слова выполняется путем проверки $\mathbf{c} \cdot \mathbf{H}^T = \mathbf{0}$, где

$$\mathbf{c} = [1 \ 1 \ 0 \ 1 \ 0 \ 0 \ 1 \ 1 \ 1 \ 1].$$

Метод кодирования, основанный на LU-разложении квадратной матрицы. Как известно [12], LU-разложение широко используется при решении систем линейных алгебраических уравнений. Методы и примеры разложения матрицы $\mathbf{A}$ на подматрицы $\mathbf{L}$ и $\mathbf{U}$ подробно описаны в [12]. Методы решения систем уравнений на основе данного метода описаны в [13]. Кратко опишем суть метода.

LU-разложением квадратной матрицы $\mathbf{A}$ называется представление матрицы в виде произведения $\mathbf{A} = \mathbf{L} \cdot \mathbf{U}$, где $\mathbf{L} = (l_{ij})$ – нижняя треугольная матрица, а $\mathbf{U} = (u_{ij})$ – верхняя треугольная матрица с единицами на главной диагонали, например:

$$\mathbf{A} = \mathbf{L}\mathbf{U} = \begin{bmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}. \quad (7)$$

Допустим, необходимо решить систему линейных уравнений (7) с помощью LU-разложения.

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b} \quad (8)$$

или

$$\mathbf{L} \cdot \mathbf{U} \cdot \mathbf{x} = \mathbf{b}. \quad (9)$$

Перепишем (9) в виде $\mathbf{L} \cdot (\mathbf{U} \cdot \mathbf{x}) = \mathbf{b}$ и обозначим

$$\mathbf{U} \cdot \mathbf{x} = \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}. \quad (10)$$

Первым действием в решении является нахождение $\mathbf{y}$ из уравнения

$$\mathbf{L} \cdot \mathbf{y} = \mathbf{b}, \quad (11)$$

затем нахождение $\mathbf{x}$ из уравнения

$$\mathbf{U} \cdot \mathbf{x} = \mathbf{y}. \quad (12)$$

Уравнение (11) решается методом прямой подстановки [13]

$$y_i = b_i - \sum_{j=1}^{i-1} l_{ij} \cdot y_j. \quad (13)$$

Уравнение (12) решается методом обратной подстановки

$$x_i = y_i - \sum_{j=i+1}^n u_{ij} \cdot x_j. \quad (14)$$

Как отмечается в [13], LU-разложение осуществляется только тогда, когда главные миноры матрицы отличны от нуля. На практике это условие выполняется не всегда. Поэтому в прикладных задачах вместо LU-разложений чаще используются так называемые LUP-разложения [14]. Матрица $\mathbf{A}$ с помощью матрицы перестановок $\mathbf{P}$ модифицируется так, что матрицы $\mathbf{L}$ и $\mathbf{U}$ имеют описанную ранее структуру. Пример решения систем уравнений (9), (10) с учетом матрицы $\mathbf{P}$ будет описан далее.

Применим LU-разложение для кодирования МППЧ-кода.

Пример 2. Для кодирования кода будем использовать матрицу (5) из примера 1. Главные миноры данной матрицы отличны от нуля. Исходя из этого, для разложения будем использовать стандартную функцию $[L, U] = \text{lu}(C)$ из библиотеки MatLab.

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix}, \quad (15)$$

$$\mathbf{U} = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \quad (16)$$

Систему уравнений (11), (12) перепишем в виде

$$\mathbf{y} = \mathbf{U} \cdot \mathbf{u}, \quad (17)$$

где $\mathbf{u}$ – вектор информационных битов.

$$\mathbf{p} = \mathbf{L} \cdot \mathbf{u}, \quad (18)$$

где $\mathbf{p}$ – вектор проверочных битов.

Алгоритм вычисления кодового слова:

1. Формируется вектор сообщения $\mathbf{u}$
 $\mathbf{u} = [1 \ 1 \ 0 \ 1 \ 0]$
2. Вектор $\mathbf{y}$ вычисляется путем обратной подстановки
 $K = m;$
 $\mathbf{y}(1:K) = \mathbf{0};$
for $i = K:-1:1$
 $\text{sum} = 0;$
for $j = K:-1:i$ $\text{sum} = \text{mod}(\text{sum} + \text{mod}(\mathbf{U}(i,j) * \mathbf{u}(j), 2), 2);$
end
 $\mathbf{y}(i) = \text{mod}(\text{sum}, 2);$
end
 $\mathbf{y} = [0 \ 1 \ 0 \ 1 \ 0]$
3. Вектор проверочных битов $\mathbf{p}$ вычисляется путем прямой подстановки
 $\mathbf{p}(1:K) = \mathbf{0};$
for $i = 1:K$
 $\text{sum} = 0;$
for $j = 1:i$ $\text{sum} = \text{mod}(\text{sum} + \text{mod}(\mathbf{L}(i,j) * \mathbf{y}(j), 2), 2);$
end
 $\mathbf{p}(i) = \text{sum};$
end
 $\mathbf{p} = [0 \ 1 \ 1 \ 1 \ 1]$
4. $\mathbf{c} = [1 \ 1 \ 0 \ 1 \ 0 \ 0 \ 1 \ 1 \ 1 \ 1]$
5. $\text{mod}(\mathbf{c} * \mathbf{H}^T, 2) = [0 \ 0 \ 0 \ 0 \ 0]$ % {Проверка достоверности кодового слова %}

Метод кодирования, основанный на LUP-разложении квадратной матрицы. Как отмечалось выше, не для всех квадратных матриц существует LU-разложение. В таких случаях не все главные миноры матрицы отличны от нуля. В таком случае строки/столбцы квадратной матрицы переставляются так, чтобы результирующие матрицы имели строго треугольную форму.

LUP-разложением квадратной матрицы $\mathbf{A}$ называется представление матрицы в виде произведения $\mathbf{PA} = \mathbf{LU}$. В процессе выполнения LUP-разложения генерируется матрица перестановок $\mathbf{P}$, которая ис-

пользуется для коррекции решения системы уравнений. Продемонстрируем данный метод для кодирования МППЧ-кода на примере.

Пример 3. Допустим, при построении кода используется проверочная матрица

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}. \quad (19)$$

Матрица $\mathbf{C}$, вычисленная в соответствии с преобразованием (5), имеет вид

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}. \quad (20)$$

Вычислим LU-разложение для матрицы (20)

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}, \quad (21)$$

$$\mathbf{U} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \quad (22)$$

Как видно из (21), матрица $\mathbf{L}$ имеет нетреугольную форму. Анализ матрицы (20) показывает, что не все главные миноры отличны от нуля. Миноры 2, 3, 4 равны нулю, следовательно, процесс кодирования невозможен. В таком случае вычислим LUP-разложение. Результат разложения имеет следующий вид:

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad (23)$$

$$\mathbf{U} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad (24)$$

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}. \quad (25)$$

Алгоритм вычисления кодового слова:

1. $\%u = \text{double}(\text{rand}(K,1)>0.5);$
 $u = [1 \ 0 \ 0 \ 0 \ 1]$
2. Вектор y вычисляем путем обратной подстановки
 $y = [1 \ 1 \ 1 \ 0 \ 1]$
3. Вектор проверочных битов p (промежуточный результат) вычисляем путем прямой подстановки
 $p = [1 \ 0 \ 0 \ 0 \ 1]$
4. Коррекция проверочных битов
 $p = \text{double}(p)$
 $p = \text{mod}(p * \mathbf{P}, 2)$
 $p = [1 \ 1 \ 0 \ 0 \ 0]$
5. $s = [1 \ 0 \ 0 \ 0 \ 1 \ 1 \ 1 \ 0 \ 0 \ 0] \% \{\text{Вычисленное кодовое слово}\}$
6. $\text{mod}(s * \mathbf{H}', 2) = [0 \ 0 \ 0 \ 0 \ 0] \% \{\text{Проверка достоверности кодового слова}\}$

Алгоритмы кодирования, построенные на методах LU и LUP, являются последовательными. Алгоритм кодирования, основанный на LU-разложении, выполняется за 2 шага. Алгоритм кодирования, основанный на LUP-разложении, выполняется за 3 шага.

Данное свойство дает возможность использовать технику конвейерной обработки данных [15] и позволяет значительно увеличить производительность кодера по сравнению, например, с кодером Нила.

Кроме того, аппаратная реализация данных методов очень простая, не требующая параллельных матричных вычислений. Примеры блоков, реализующих вычисления прямой и обратной подстановки, можно найти в [16].

Метод кодирования, основанный на приведении проверочной матрицы к ступенчатой форме. Для приведения проверочной матрицы $\mathbf{H}$ к систематической ступенчатой $\mathbf{H}_{\text{sys}}$ используется метод исключения Гаусса–Жордана [12]. Систематическая проверочная матрица имеет вид

$$\mathbf{H}_{\text{sys}} = [\mathbf{I}_m \mathbf{P}], \quad (26)$$

где $\mathbf{I}_m$ – единичная матрица, имеющая размерность $m \times m$. Матрица $\mathbf{P}$ имеет размерность $m \times (n-m)$.

Для нахождения $\mathbf{H}_{\text{sys}}$ из $\mathbf{H}$ будем использовать стандартную функцию $R = \text{rref}(A)$ из библиотеки MatLab. Функция $\text{rref}(H)$ приводит матрицу $\mathbf{H}$ к ступенчатой форме $\mathbf{H}_{\text{sys}}$ с использованием исключения Гаусса–Жордана.

Пример 4. Допустим, для кодирования используется проверочная матрица:

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}, \quad (27)$$

$$\mathbf{H}_{\text{sys}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}. \quad (28)$$

Путем перестановки подматриц $\mathbf{I}_m \mathbf{P}$ в (28) получаем стандартную форму проверочной матрицы в ступенчатой форме

$$\mathbf{H}_{\text{std}} = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \quad (29)$$

Используя правила построения порождающей матрицы [17], из стандартной проверочной получаем стандартную порождающую матрицу в ступенчатой форме

$$\mathbf{G} = [\mathbf{I}_m \mathbf{P}^T] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}. \quad (30)$$

Достоверность вычисления порождающей матрицы проверяем с помощью соотношения $[\mathbf{G} \cdot \mathbf{H}_{\text{std}}^T] = \mathbf{0}$.

Алгоритм вычисления кодового слова:

1. %u = round(rand(1,size(G,1)));
u = [1 0 1 0 1];
2. Вычисление кодового слова:
c = mod(u*G,2);
%{c = [1 0 1 0 1 1 0 0 0]}%
3. mod(c*H_std',2) = [0 0 0 0 0] % {Проверка достоверности кодового слова %}

Методы кодирования на основе эквивалентной проверочной матрицы в нижней треугольной форме

Метод кодирования на основе нижней триангуляции проверочной матрицы. В работе [18] предложена методика кодирования, основанная на приведении проверочной матрицы к эквивалентной проверочной матрице в нижней треугольной форме.

Средствами метода исключения Гаусса [12] проверочная матрица $\mathbf{H}$ преобразуется в эквивалентную нижнетреугольную матрицу $\mathbf{H}_t$. Проверочные биты p_l , $l = 1, \dots, m$ вычисляются в соответствии с формулой

$$p_l = \sum_{j=1}^{m-n} \mathbf{H}_{tl,j} \cdot s_j + \sum_{j=1}^{l-1} \mathbf{H}_{tl,j+n-m} \cdot p_j, \quad (31)$$

где s_j – информационные биты.

Пример 5. Допустим, для кодирования используется проверочная матрица

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}. \quad (32)$$

Применяя исключение Гаусса для проверочной матрицы, получаем нижнюю треугольную матрицу:

$$\mathbf{H}_t = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}. \quad (33)$$

Как видно из (33), матрицу $\mathbf{H}_t$ можно разделить на две части $\mathbf{H}_t = [\mathbf{A} | \mathbf{L}]$, где $\mathbf{A}$ и $\mathbf{L}$ имеют размерности $m \times (n - m)$ и $m \times m$ соответственно. Матрица $\mathbf{A}$ является прямоугольной (в примере 5 квадратной, поскольку скорость кода равна 1/2). Матрица $\mathbf{L}$ является треугольной. Кодовое слово $\mathbf{c}$ делится также на две части $\mathbf{c} = [\mathbf{s} \ \mathbf{p}]$. С учетом данного обстоятельства из анализа выражения (29) видно, что в вычислении кодового слова можно выделить два независимых процесса. Во время первого процесса коэффициенты текущей строки l матрицы $\mathbf{A}$ умножаются на информационные биты $\mathbf{s}$ и суммируются. Во время второго процесса вычисленные предыдущие проверочные биты умножаются на коэффициенты матрицы $\mathbf{L}$ и суммируются. Данный процесс в [18] назван методом обратной подстановки. По окончании данного процесса вычисляется текущий проверочный бит p_l .

Проверочные биты вычисляются последовательно, при этом оба процесса на каждом шаге могут выполняться параллельно. При параллельной реализации вычислений матрицы $\mathbf{A}$ и $\mathbf{L}$ должны храниться в памяти раздельно. Данный метод даёт возможность значительно повысить скорость кодирования. Отметим, что при этом требования к объему памяти выше, чем у кодера Нила или кодера, основанного на LU и LUP-разложениях.

Алгоритм вычисления кодового слова, реализующий выражение (29):

1. %s=double(rand(k,1)>0.5)
s=[1 0 1 1 1]'
for j = 1:m
 p(j) = 0;
end
2. Вычисление кодового слова:
for l = 1:m
 sum1 = 0;
 sum1 = mod(HT(l,1:k)*s,2); % First process
 sum2 = 0;
 for j = 1:l sum2 = mod(sum2 +
mod(HT(l,j+k)*p(j),2),2 % Second process
 end
 p(l) = mod(sum1 + sum2,2);
end
% p = [1 1 1 1 1]
3. c = [s' p]
% c = [1 0 1 1 1 1 1 1 1 1]
mod(H*c',2) = [0 0 0 0 0] % {Проверка достоверности кодового слова %}

Метод кодирования на основе приближенной нижней триангуляции проверочной матрицы. В работе [18] предложена методика кодирования, осно-

ванная на приближенной эквивалентной проверочной матрице в нижней треугольной форме.

Следует отметить, что данный подход применяется при проектировании современных кодеров квазициклических МППЧ-кодов стандарта 5G NR [22, 23]. Структура проверочных матриц кодов 5G NR позволяет использовать нижнетреугольное представление для эффективного последовательного вычисления проверочных битов. Реализации кодеров, основанные на данном принципе, рассмотрены в ряде современных работ [22, 23].

Суть метода заключается в следующем:

1) генерируется или выбирается разреженная проверочная матрица $\mathbf{H}$, имеющая размерность $m \times n$. Матрица должна быть невырожденной;

2) матрица $\mathbf{H}$ делится на блоки (подматрицы) с соответствующими размерностями

$$\mathbf{H} = \begin{bmatrix} \mathbf{A} & \mathbf{B} & \mathbf{T} \\ \mathbf{C} & \mathbf{D} & \mathbf{E} \end{bmatrix}. \quad (34)$$

Подматрицы имеют следующие размерности:

$\mathbf{A} - (m-g) \times (n-m); \quad \mathbf{C} - g \times (n-m);$

$\mathbf{B} - (m-g) \times (g); \quad \mathbf{D} - g \times g;$

$\mathbf{T} - (m-g) \times (m-g); \quad \mathbf{E} - g \times (m-g).$

Значение g должно быть по возможности минимальным [18].

Проверочное соотношение записывается в форме

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} & \mathbf{T} \\ \mathbf{C} & \mathbf{D} & \mathbf{E} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{u} \\ \mathbf{p}_1 \\ \mathbf{p}_2 \end{bmatrix} = \mathbf{0}, \quad (35)$$

где $\mathbf{u}$ – вектор информационных битов, $\mathbf{p}_1$ и $\mathbf{p}_2$ – векторы проверочных битов.

Проверочное соотношение можно записать в иной форме

$$\mathbf{A} \cdot \mathbf{u} + \mathbf{B} \cdot \mathbf{p}_1 + \mathbf{T} \cdot \mathbf{p}_2 = \mathbf{0}, \quad (36)$$

$$\mathbf{C} \cdot \mathbf{u} + \mathbf{D} \cdot \mathbf{p}_1 + \mathbf{E} \cdot \mathbf{p}_2 = \mathbf{0}. \quad (37)$$

Система уравнений (36), (37) решается с помощью блочного исключения Гаусса–Шура [19], позволяющего исключить $\mathbf{p}_2$ в уравнении (37). Данная

задача решается путем «обнуления» подматрицы $\mathbf{E}$ в уравнении (37). Описание процедуры «обнуления» подматрицы в системе уравнений и дополнительные сведения приведены в [20]. Для вычисления дополнения Шура к подматрице $\mathbf{E}$ выполняется умножение на матрицу

$$\begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{E} \cdot \mathbf{T}^{-1} & \mathbf{I} \end{bmatrix}, \quad (38)$$

$$\begin{bmatrix} \mathbf{T} \\ \mathbf{E} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{E} \cdot \mathbf{T}^{-1} & \mathbf{I} \end{bmatrix} = \begin{bmatrix} \mathbf{T} \\ \mathbf{0} \end{bmatrix}, \quad (39)$$

где

$$\mathbf{T} \cdot (-\mathbf{E} \cdot \mathbf{T}^{-1}) + \mathbf{E} \cdot \mathbf{I} = \mathbf{0}. \quad (40)$$

Соответственно, дополнения Шура к подматрицам $\mathbf{C}$ и $\mathbf{D}$ также вычисляются путем умножения данных подматриц на матрицу (36). Окончательные выражения, полученные авторами, приводятся

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} & \mathbf{T} \\ -\mathbf{E} \cdot \mathbf{T}^{-1} \cdot \mathbf{A} + \mathbf{C} & -\mathbf{E} \cdot \mathbf{T}^{-1} \cdot \mathbf{B} + \mathbf{D} & \mathbf{0} \end{bmatrix}. \quad (41)$$

Совмещая систему уравнений (36), (37) и выражение (41), окончательно получим систему уравнений для вычисления проверочных битов $\mathbf{p}_1$ и $\mathbf{p}_2$:

$$\mathbf{A} \cdot \mathbf{u} + \mathbf{B} \cdot \mathbf{p}_1 + \mathbf{T} \cdot \mathbf{p}_2 = \mathbf{0}, \quad (42)$$

$$(-\mathbf{E} \cdot \mathbf{T}^{-1} \cdot \mathbf{A} + \mathbf{C}) \cdot \mathbf{u} + (-\mathbf{E} \cdot \mathbf{T}^{-1} \cdot \mathbf{B} + \mathbf{D}) \cdot \mathbf{p}_1 = \mathbf{0}. \quad (43)$$

Из уравнения (41) получим выражение для вычисления проверочных битов $\mathbf{p}_1$.

$$\mathbf{p}_1 = (-\mathbf{E} \cdot \mathbf{T}^{-1} \cdot \mathbf{B} + \mathbf{D})^{-1} (-\mathbf{E} \cdot \mathbf{T}^{-1} \cdot \mathbf{A} + \mathbf{C}) \cdot \mathbf{u}. \quad (44)$$

Подставляя вычисленный вектор $\mathbf{p}_1$ в уравнение (40), получаем вектор проверочных битов $\mathbf{p}_2$:

$$\mathbf{p}_2 = \mathbf{T}^{-1} (\mathbf{A} \cdot \mathbf{u} + \mathbf{B} \cdot \mathbf{p}_1). \quad (45)$$

Рассмотрим данную методику кодирования на примере.

Пример 6. В данном примере используем следующую проверочную матрицу.

Данная матрица невырожденная и имеет ранг $r = 10$.

$$\mathbf{H} = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}. \quad (46)$$

Значение g выбираем равным 2. К первым восьми строкам матрицы применен метод исключения Гаусса для нижней треугольной матрицы.

Последние две строки матрицы остаются без изменения. Полученная эквивалентная проверочная матрица $\mathbf{ATM}$ с нижней треугольной формой имеет вид

$$\mathbf{ATM} = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & | & 0 & 0 & | & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & | & 0 & 0 & | & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & | & 1 & 0 & | & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & | & 0 & 0 & | & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & | & 0 & 1 & | & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & | & 1 & 0 & | & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & | & 0 & 0 & | & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & | & 0 & 0 & | & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ - & - & - & - & - & - & - & - & - & - & | & - & - & | & - & - & - & - & - & - & - \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & | & 0 & 0 & | & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & | & 1 & 0 & | & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}. \quad (47)$$

Алгоритм вычисления кодового слова:

```
1. invT = mod( (inv(T)),2) =
[1 0 0 0 0 0 0 0;
0 1 0 0 0 0 0 0;
1 1 1 0 0 0 0 0;
1 0 0 1 0 0 0 0;
1 0 0 0 1 0 0 0;
0 0 0 0 0 1 0 0;
1 0 0 1 1 0 1 0;
0 0 0 1 1 0 0 1]
2. ET1 = mod(-(E*invT),2) = [1 0 0 1 1 0 1 0; 1 0 0 0 1 0 0 0]
3. cD = mod((ET1*B + D),2) = [0 1; 1 1] % { Матрица
невырожденная, что соответствует условию метода %}
4. cC = mod((ET1*A + C),2) =
[1 0 0 1 0 1 1 0 1; 1 0 0 1 0 1 0 1 1 1]
5. tmp = mod(inv(cD) * cC,2) =
[0 0 0 0 1 1 1 0 1 0; 1 0 0 1 1 0 1 1 0 1]
6. %u = round(rand(1,m))
u = [0 1 0 1 1 1 0 0 0 1]
7. p1 = mod(tmp*u',2) = [0 1]
8. p2 = mod(invT*(A*u' + B*p1),2) = [1 1 1 1 1 1 1 0]
9. c = [u p1' p2'] = [0 1 0 1 1 1 0 0 0 1 0 1 1 1 1 1 1 1 0]
10. mod(c*N',2) = [0 0 0 0 0 0 0 0 0 0] % {Проверка
достоверности кодового слова %}
```

Оценка вычислительной сложности методов кодирования

В этом разделе проводится оценка вычислительной сложности (ВС) рассмотренных методов кодирования. Оценка ВС по подготовке матриц, используемых в кодере, не рассматривается.

Под ВС будет пониматься число булевых арифметических операций, участвующих в процессе кодирования (сложений, умножений), которые рассматриваются как равнозначные. Основное вычисление, которое выполняет арифметическое устройство систематического кодера – умножение порождающей матрицы

$$\mathbf{G} = \begin{bmatrix} g_{11} & \cdots & g_{1k} \\ \vdots & \ddots & \vdots \\ g_{m1} & \cdots & g_{mk} \end{bmatrix} \quad (48)$$

на вектор информации

$$\mathbf{u} = [u_1 \ u_2 \ \dots \ u_k]^T. \quad (49)$$

В результате произведения формируется вектор проверочных (избыточных) битов:

$$\mathbf{p} = \mathbf{G}\mathbf{u}, \quad (50)$$

$$p_i = \sum_j g_{ij} u_j. \quad (51)$$

Оценку ВС будем рассматривать на примере LDPC-кода с кодовой скоростью $R = 1/2$. В этом случае число информационных битов k равно числу проверочных битов m , а $\mathbf{G}$ является квадратной матрицей. Для вычисления одного проверочного бита (51) необходимо m операций умножения и $m-1$ операций сложения, или в общем $2m-1$ операций. Для вычисления всех проверочных битов требуется $m(2m-1) = 2m^2 - m$ операций. Обозначим для примера ВС умножения матрицы на вектор как $O(2m^2 - m)$. Основываясь на данном подходе, оценим ВС всех рассмотренных методов кодирования.

1. Метод Нила для вычисления избыточных битов согласно (6)

$$O(2m^2 - m). \quad (52)$$

2. Метод кодирования, основанный на LU-разложении квадратной матрицы:

$$O(m_1, m_2) = O\left(2\left(\sum_{j=1}^m j + \sum_{j=1}^m (j-1)\right)\right), \quad (53)$$

где первая сумма определяет число умножений, вторая – число сложений.

3. Метод кодирования, основанный на LUP-разложении квадратной матрицы:

$$O(m_1, m_2) = O\left(2\left(\sum_{j=1}^m j + \sum_{j=1}^m (j-1)\right) + 2m^2 - m\right), \quad (54)$$

где третье слагаемое определяет дополнительные вычисления для коррекции избыточных битов путем перемножения на перестановочную матрицу $\mathbf{P}$.

4. Метод кодирования, основанный на приведении проверочной матрицы к ступенчатой форме:

$$O(2m + (2m^2 - m)). \quad (55)$$

5. Метод кодирования на основе нижней триангуляции проверочной матрицы

$$O\left(2m^2 - m + \left(\sum_{j=1}^m j + \sum_{j=1}^m (j-1)\right)\right), \quad (56)$$

где первое слагаемое определяет вычисления на матрице **A**, второе – на матрице **L**.

6. Метод кодирования на основе приближенной нижней триангуляции проверочной матрицы:

1) вычисление выражения (44)

$$O_1((2m-1)g). \quad (57)$$

2) вычисление выражения (45) состоит из операций, имеющих следующую сложность:

$$\mathbf{A} \cdot \mathbf{u} : O_2((m-g)(2m-1)), \quad (58)$$

$$\mathbf{B} \cdot \mathbf{p}_1 : O_3(g(2(m-g)-1)). \quad (59)$$

Сумма произведений

$$\mathbf{A} \cdot \mathbf{u} + \mathbf{B} \cdot \mathbf{p}_1 : O_4(m-g). \quad (60)$$

Последняя операция матричного умножения в (45)

$$\mathbf{T}^{-1}(\mathbf{A} \cdot \mathbf{u} + \mathbf{B} \cdot \mathbf{p}_1) : O_5(g(2(m-g)^2 - m - g)). \quad (61)$$

Таким образом, алгоритм имеет следующую вычислительную сложность:

$$O(O_1 + O_2 + O_3 + O_4 + O_5). \quad (62)$$

Для сопоставления вычислительной сложности рассмотренных методов выполнено моделирование зависимости числа арифметических операций от размера проверочной матрицы m при фиксированной скорости (рис. 1).

Как видно из рис. 1, наименьшую сложность демонстрируют методы Нила, LU и RREF (приведение проверочной матрицы к ступенчатой форме), имеющие близкие зависимости с квадратичным ростом. Методы Triang (нижняя триангуляция проверочной матрицы), ALT (приближенная нижняя триангуляция проверочной матрицы) и LUP демонстрируют более быстрый рост вычислительной сложности.

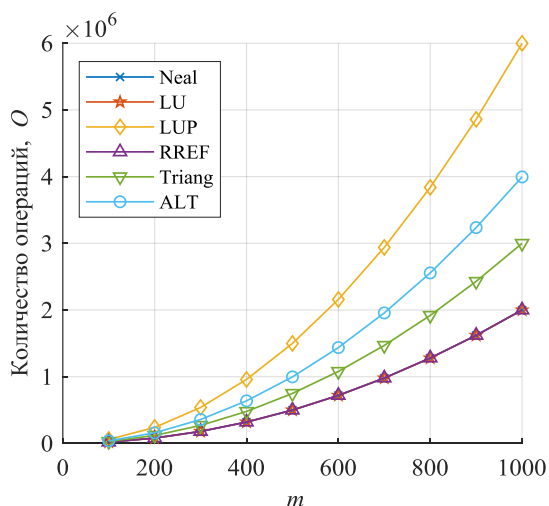


Рис. 1. Зависимость числа арифметических операций от размера проверочной матрицы m

Оценка времени вычислений

В дополнение к проведенной аналитической оценке ВС методов кодирования случайных МППЧ-кодов рассмотрим их временную сложность при работе в единой программной среде.

Для оценки среднего времени выполнения операций кодирования T_{enc} описанных методов была разработана имитационная модель в среде MatLab. Во всех сценариях моделирования рассматривались проверочные матрицы МППЧ-кодов с фиксированной скоростью кодирования $R = 1/2$. Каждая строка матрицы имела фиксированный вес, равный трем, а изменение размеров матрицы осуществлялось за счет управления параметром m (при $n = 2m$). Результаты усреднялись по 1000 случайным реализациям матриц, удовлетворяющих условиям реализуемости рассматриваемых методов. Для каждой матрицы имитировалось кодирование ста случайных сообщений. Результаты моделирования приведены на рис. 2.

Как видно из рис. 2, наибольшее время кодирования демонстрируют методы, основанные на LU- и LUP-разложениях, а также метод Triang. Это объясняется тем, что кодирование в реализациях алгоритмов этих методов основано на циклах, в то время как методы Нила, RREF и ALT реализованы с использованием матричных умножений. Среда MatLab оптимизирована для выполнения матричных умножений, что позволяет обеспечить низкий рост времени вычислений с увеличением размеров проверочных матриц. Вместе с тем операции матричного умножения требуют значительных аппаратных ресурсов при реализации в программируемых логических интегральных схемах (ПЛИС).

По этой причине время вычислений различными методами при аппаратной реализации в ПЛИС будет отличаться от проведенной оценки.

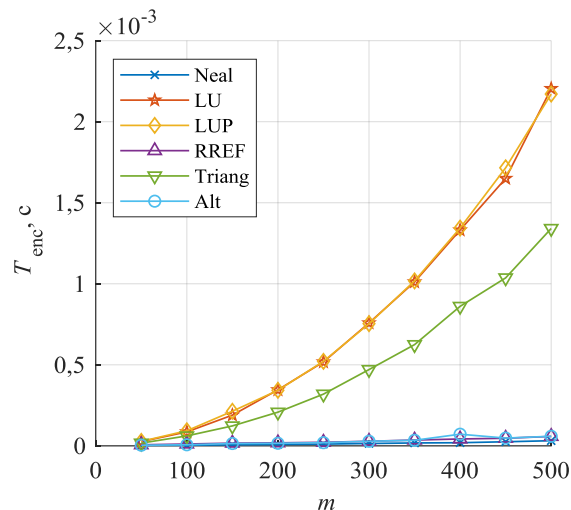


Рис. 2. Зависимость среднего времени кодирования от размера проверочной матрицы m

Заключение

В работе проведен анализ методов кодирования случайных МППЧ-кодов, основанных на различных подходах к преобразованию проверочной матрицы и решению систем линейных уравнений. Показано, что задачи эффективного кодирования для случайных разреженных матриц остаются актуальными, особенно в контексте практической реализации.

Рассмотренные методы, включая подход Нила, алгоритмы на основе LU- и LUP-разложений, приве-

дение матрицы к ступенчатой форме, а также методы нижней и приближенной триангуляции, демонстрируют различия по вычислительной сложности, требованиям к структуре матрицы и возможностям аппаратной реализации. Установлено, что методы, основанные на треугольных преобразованиях, позволяют реализовать последовательные алгоритмы кодирования с возможностью конвейерной обработки, что важно для высокоскоростных систем связи.

Показано, что выбор конкретного метода кодирования определяется компромиссом между сложностью реализации, требованиями к памяти и структурой проверочной матрицы. Представленные в работе примеры иллюстрируют практическую применимость рассмотренных алгоритмов и позволяют лучше понять особенности их реализации.

Выполнен анализ вычислительной сложности рассмотренных методов кодирования, основанный на оценке числа арифметических операций, а также проведено моделирование времени кодирования в среде MatLab. Полученные результаты демонстрируют различия между методами с точки зрения их практической реализуемости и эффективности при увеличении размерности проверочной матрицы.

Полученные результаты могут быть использованы при разработке эффективных кодеров для систем связи следующих поколений, включая 5G- и перспективные 6G-системы.

Литература

1. 6G communication systems: concept, trends, physical level technologies / D.A. Pokamestov, Y.V. Kryukov, R.R. Abenov, E.V. Rogozhnikov, A.A. Brovkin, A.S. Shinkevich, G.N. Shalin // Radio Engineering and Electronics. – 2024. – Vol. 69, № 1. – P. 3–33.
2. Gallager R. Low-density parity-check codes // IRE Transactions on Information Theory. – 1962. – Vol. 8, No. 1. – P. 21–28.
3. Channel Coding Toward 6G: Technical Overview and Outlook / M. Rowshan, M. Qiu, Y. Xie, X. Gu, J. Yuan // IEEE Open Journal of the Communications Society. – 2024. – Vol. 5. – P. 2585–2685.
4. Трифонов П.В. Основы помехоустойчивого кодирования: учеб. пособие. – СПб.: Университет ИТМО, 2022. – 231 с.
5. Lin S. Structured Low-Density Parity-Check Codes: Algebraic Constructions // IEEE 802.3an Task Force Meeting. – IEEE: Portland, OR, 13–15, 2004. – P. 1–20.
6. Donovan D. High-rate LDPC codes from partially balanced incomplete block designs / D. Donovan, A. Price, A. Rao, E. Üsküplü, E.Ş. Yazıcı // Journal of Algebraic Combinatorics. – 2022. – Vol. 55. – P. 259–275.
7. MacKay D.J.C. Good Error-Correcting Codes Based on Very Sparse Matrices // IEEE Transactions on Information Theory. – 1999. – Vol. 45, No. 2. – P. 399–431.
8. Кравченко А.Н. Снижение сложности декодирования низкоплотного кода // Цифровая обработка сигналов. – 2010. – № 2. – С. 35–41.
9. Subhi M.I. Enhancing Data Communication Performance: A Comprehensive Review and Evaluation of LDPC Decoder Architectures / M.I. Subhi, Q. Al-Doori, O. Alani // Ingénierie des Systèmes d'Information. – 2023. – Vol. 28, No. 5. – P. 1113–1125.
10. MacKay D.J.C. Encyclopedia of Sparse Graph Codes [Электронный ресурс]. – URL: <http://www.inference.org.uk/mackay/codes/data.html> (дата обращения: 19.08.2026).
11. Neal R.M. Sparse Matrix Methods and Probabilistic Inference Algorithms // IMA Program on Codes, Systems and Graphical Models. – 1999. – P. 1–16.
12. Strang G. Introduction to Linear Algebra. – 6th ed. – Wellesley, MA, USA: Wellesley-Cambridge Press, 2023. – 537 p.
13. Вержбицкий В.М. Вычислительная линейная алгебра. – М.: Высшая школа, 2009. – 352 с.
14. LUP-разложение квадратных матриц [Электронный ресурс]. – URL: https://study.sfu-kras.ru/DATA/docs/ProgramTheory/recurs/lup_expn.htm (дата обращения: 19.08.2026).
15. Harris S.L. Digital design and computer architecture, RISC-V edition / S.L. Harris, D. Harris. – Elsevier: Morgan Kaufmann, 2021. – 592 p.
16. A Flexible Hardware Encoder for Low-Density Parity-Check Codes / D.-U. Lee, W. Luk, C. Wang, C. Jones, M. Smith, J. Villasenor // Proceedings of the 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines. – 2004. – P. 101–111.
17. Research and implementation of high-efficiency and low-complexity LDPC-coding algorithm / X. Liao, J. Guo, Z. Luo, Y. Xu, Y. Chu // Electronics. – 2023. – Vol. 12, No. 17. – P. 1–20.
18. Efficient Encoding of Low-Density Parity-Check Codes / T.J. Richardson, R.L. Urbanke // IEEE Transactions on Information Theory. – 2001. – Vol. 47, No. 2. – P. 638–656.
19. Уоткинс Д.С. Основы матричных вычислений / пер. с англ. В.Е. Кондрашова, С.Б. Королева. – М.: БИНОМ. Лаборатория знаний, 2006. – 664 с.
20. Partitioned Matrices and the Schur Complement [Электронный ресурс]. – URL: https://quickfem.com/wp-content/uploads/IFEM.AppP_.pdf (дата обращения: 19.08.2026).
21. Golub G.H. Matrix Computations / G.H. Golub, C.F. Van Loan. – 4th ed. – Baltimore: Johns Hopkins University Press, 2013. – 756 p.
22. Tian Y. Low-Latency QC-LDPC Encoder Design for 5G NR / Y. Tian, Y. Bai, D. Liu // Sensors. – 2021. – Vol. 21, No. 18. – P. 6266.
23. Petrović V.L. Flexible 5G New Radio LDPC Encoder Optimized for High Hardware Usage Efficiency / V.L. Petrović, D.M. El Mezeni, A. Radošević // Electronics. – 2021. – Vol. 10, No. 9. – P. 1106.

Кравченко Александр Николаевич

Канд. техн. наук, системный архитектор Ubiso GmbH, Hermann-Schwer-Str. 3, VS-Villingen, Germany, 78048
Тел.: +49 (0) 7-721-944-76-77
Эл. почта: alexander.kravtchenko@ubiso.com

Покаместов Дмитрий Алексеевич

Канд. техн. наук, доцент каф. телекоммуникаций и основ радиотехники Томского государственного ун-та систем управления и радиоэлектроники (ТУСУР)
Ленина пр-т, 40, г. Томск, Россия, 634050
ORCID: 0000-0002-9405-4573
Тел.: +7-952-809-43-75
Эл. почта: dmaltomsk@mail.ru

Поступила в редакцию: 06.04.2026.

Принята к публикации: 24.08.2026.

Kravtchenko A.N., Pokamestov D.A.

Encoding Methods for Random Low-Density Parity-Check Codes

Relevance. The development and investigation of encoding methods for random low-density parity-check (LDPC) codes is an important task in the design of high-speed communication systems, since different codeword generation algorithms differ significantly in computational cost and implementation requirements. **Purpose of the study.** Comparative analysis of encoding methods for LDPC-codes defined by random sparse parity-check matrices, with an assessment of their computational complexity and practical performance. **Methods.** The study employs methods of linear algebra, analytical estimation of computational complexity, simulation modeling, and the Monte Carlo method. The encoding approaches under study include Neal's method, algorithms based on LU and LUP-decompositions, transformation of the parity-check matrix to row-echelon form, as well as lower and approximate lower triangulation methods. **Scientific novelty.** A comparative study of encoding methods for random LDPC codes was carried out, including a joint assessment of their analytical computational complexity and software implementation time under identical initial conditions. **Results.** Differences in the growth patterns of computational costs among the considered methods as the size of the parity-check matrix increases were identified. It was shown that Neal's method, LU decomposition, and the method based on transformation to row-echelon form require fewer arithmetic operations, whereas the results of software simulations depend significantly on the implementation approach and the optimization capabilities for matrix operations in MatLab. **Practical significance.** The obtained results can be used to select encoding algorithms for random LDPC-codes and developing software and hardware encoders for high-speed communication systems. **Keywords:** LDPC, low-density parity-check codes, random matrices, error-correcting coding, computational complexity. **DOI:** 10.21293/1818-0442-2026-29-2-64-73

References

1. Pokamestov D.A., Kryukov Y.V., Abenov R.R., Rogozhnikov E.V., Brovkin A.A., Shinkevich A.S., Shalin G.N. 6G communication systems: concept, trends, physical level technologies. *Radio Engineering and Electronics*, 2024, vol. 69, no. 1, pp. 3–33.
2. Gallager R.G. Low-density parity-check codes. *IRE Transactions on Information Theory*, 1962, vol. 8, no. 1, pp. 21–28.
3. Rowshan M., Qiu M., Xie Y., Gu X., Yuan J. Channel Coding Toward 6G: Technical Overview and Outlook. *IEEE Open Journal of the Communications Society*, 2024, vol. 5, pp. 2585–2685.
4. Trifonov P.V. *Osnovy pomekhoustoichivogo kodirovaniya: uchebnoe posobie* [Fundamentals of error-correcting coding: textbook]. Saint-Petersburg, ITMO University Publ., 2022, 231 p. (in Russ.).
5. Lin S. Structured Low-Density Parity-Check Codes: Algebraic Constructions. IEEE 802.3an Task Force Meeting, IEEE, Portland, OR, July 13–15, 2004.
6. Donovan D., Price A., Rao A., Üsküplü E., Yazıcı E.Ş. High-rate LDPC codes from partially balanced incomplete block designs. *Journal of Algebraic Combinatorics*, 2022, vol. 55, pp. 259–275.
7. MacKay D.J.C. Good error-correcting codes based on very sparse matrices. *IEEE Transactions on Information Theory*, 1999, vol. 45, no. 2, pp. 399–431.
8. Kravchenko A.N. [Reduced-complexity decoding LDPC code]. *Digital Signal Processing*, 2010, no. 2, pp. 35–41 (in Russ.).

9. Subhi M.I., Al-Doori Q., Alani O. Enhancing data communication performance: a comprehensive review and evaluation of LDPC decoder architectures. *Ingénierie des Systèmes d'Information*, 2023, vol. 28, no. 5, pp. 1113–1125.

10. MacKay D.J.C. Encyclopedia of Sparse Graph Codes. Available at: <http://www.inference.org.uk/mackay/codes/data.html> (accessed: 19 August 2026).

11. Neal R.M. Sparse matrix methods and probabilistic inference algorithms. *IMA Program on Codes, Systems and Graphical Models*, 1999, pp. 1–16.

12. Strang G. Introduction to Linear Algebra. 6th ed. Wellesley, MA, USA: Wellesley-Cambridge Press, 2023, 537 p.

13. Verzhbitskii V.M. *Vychislitel'naya lineinaya algebra* [Computational linear algebra]. Moscow, Vysshaya shkola Publ., 2009, 352 p. (in Russ.).

14. LUP-razlozhenie kvadratnykh matrits [LUP decomposition of square matrices]. Available at: https://study.sfu-kras.ru/DATA/docs/ProgramTheory/recurs/lup_expn.htm (accessed: 19 August 2026).

15. Harris S.L., Harris D. Digital design and computer architecture: RISC-V edition. Elsevier, Morgan Kaufmann, 2021, 592 p.

16. Lee D.-U., Luk W., Wang C., Jones C., Smith M., Vilasenor J. A flexible hardware encoder for low-density parity-check codes. *Proc. 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, 2004, pp. 101–111.

17. Liao X., Guo J., Luo Z., Xu Y., Chu Y. Research and implementation of high-efficiency and low-complexity LDPC coding algorithm. *Electronics*, 2023, vol. 12, no. 17, pp. 1–20.

18. Richardson T.J., Urbanke R.L. Efficient encoding of low-density parity-check codes. *IEEE Transactions on Information Theory*, 2001, vol. 47, no. 2, pp. 638–656.

19. Watkins D.S. *Osnovy matrichnykh vychislenii* [Fundamentals of matrix computations]. Moscow, BINOM. Laboratoriya znanii Publ., 2006, 664 p. (in Russ.).

20. Partitioned matrices and the Schur complement. Available at: https://quickfem.com/wp-content/uploads/IFEM.AppP_.pdf (accessed: 19 August 2026).

21. Golub G.H., Van Loan C.F. Matrix computations. 4th ed. Baltimore, Johns Hopkins University Press, 2013, 756 p.

22. Tian Y., Bai Y., Liu D. Low-latency QC-LDPC encoder design for 5G NR. *Sensors*, 2021, vol. 21, no. 18, pp. 6266.

23. Petrović V.L., Mezeni D.M., Radošević A. Flexible 5G New Radio LDPC encoder optimized for high hardware usage efficiency. *Electronics*, 2021, vol. 10, no. 9, pp. 1106.

Alexander N. Kravchenko

PhD, System Architecture, Ubiso GmbH (retired)
3, Hermann-Schwer-Str., VS-Villingen, Germany, 78048
Phone: +49 (0) 7-721-944-76-77
Email: alexander.kravtchenko@ubiso.com

Dmitriy A. Pokamestov

Candidate of Engineering Sciences, Associate Professor,
Department of Telecommunications and Fundamentals
of Radio Engineering, TUSUR
40, Lenin Ave., Tomsk, Russia, 634050
ORCID: 0000-0002-9405-4573
Phone: +7-952-809-43-75
Email: dmaltomsk@mail.ru

Received: 06.04.2026.

Accepted: 24.08.2026.