ПРИКЛАДНАЯ МАТЕМАТИКА

УДК 004.056.3

А. В. Маров, А. Ю. Утешев

МАТРИЧНЫЙ ФОРМАЛИЗМ КОДОВ РИДА—СОЛОМОНА

Санкт-Петербургский государственный университет, Российская Федерация,
199034, Санкт-Петербург, Университетская наб., 7–9

Предлагаются модификации алгоритмов кодирования и исправления ошибок (отказов и скрытых повреждений), используемых в кодах Рида—Соломона. Эти модификации используют матричный формализм и основаны на алгоритме обращения матрицы Вандермонда Для матрицы $V = \left[\lambda_j^{i-1}\right]_{i,j=1}^{n}$ предлагаемый алгоритм вычисляет столбцы $V_{[1]}^{-1}, \ldots, V_{[n-1]}^{-1}, V_{[n]}^{-1}$ матрицы V^{-1} рекурсивно, начиная с последнего, по формулам

$$V_{[n]}^{-1} = \Xi_0,\ V_{[j]}^{-1} = \Xi_{n-j} - \sigma_1 V_{[j+1]}^{-1} - \sigma_2 V_{[j+2]}^{-1} - \cdots - \sigma_{n-j} V_{[n]}^{-1},\quad j = n-1, n-2, \ldots, 1,$$

где $\Xi_k = \left[\lambda_1^k/W'(\lambda_1), \ldots, \lambda_n^k/W'(\lambda_n)\right]^\top$, $\sigma_k = \sum_{j=1}^{n} \lambda_j^{n+k-1}/W'(\lambda_j)$, $k = \overline{1,n}$, а $W(x) = \prod_{k=1}^{n}(x-\lambda_k)$. Полученный результат предлагается использовать для реализации систематического кодирования вектора из n информационных блоков посредством операции умножения (в подходящем поле Галуа) его на матрицу $\mathbf{K} = [\widetilde{W}_i(a^{N-j-1})], i = \overline{1,m}, j = \overline{0,n-1}$. Здесь $\widetilde{W}_\ell(x), \ell = \overline{1,m}$, означают базовые интерполяционные полиномы Лагранжа, порожденные степенями примитивного элементами поля, а m — количество служебных блоков (синдромов). В этой же идеологии реализуется и процедура исправления ошибок. Программная реализация на языке C демонстрирует рост производительности в сравнении с известными специализированными программными продуктами, а также допускает возможность параллелизации. Библиогр. 17 назв. Ил. 1.

Ключевые слова: помехоустойчивое кодирование, коды Рида—Соломона, матрица Вандермонда.

A. V. Marov, A. Yu. Uteshev

MATRIX FORMALISM OF THE REED—SOLOMON CODES

St. Petersburg State University, 7–9, Universitetskaya nab.,
St. Petersburg, 199034, Russian Federation

The paper is focused onto modification of the involved algorithms of coding and error (i. e., failures and silent data corruptions) correction in the Reed—Solomon codes. These

Маров Алексей Валерьевич — аспирант, разработчик исследовательской лаборатории, Raidix; alekseymmm@mail.ru

Утешев Алексей Юрьевич — доктор физико-математических наук, профессор; alexeiuteshev@gmail.com

Marov Alexei Valer'evich — postgraduate student, software developer in R&D Lab, Raidix; alekseymmm@mail.ru

Uteshev Alexei Yur'evich — doctor of physical and mathematical sciences, professor; alexeiuteshev@gmail.com

© Санкт-Петербургский государственный университет, 2016

 DOI: 10.21638/11701/spbu10.2016.401

modifications involve matrix formalism and are based on an algorithm for the Vandermonde matrix inversion. For such a matrix $V = \left[\lambda_j^{i-1}\right]_{i,j=1}^{n}$ the suggested algorithm computes the columns $V_{[1]}^{-1}, \ldots, V_{[n-1]}^{-1}, V_{[n]}^{-1}$ of the matrix V^{-1} recursively, starting from the last column, via the formulas

$$V_{[n]}^{-1} = \Xi_0,\ V_{[j]}^{-1} = \Xi_{n-j} - \sigma_1 V_{[j+1]}^{-1} - \sigma_2 V_{[j+2]}^{-1} - \cdots - \sigma_{n-j} V_{[n]}^{-1},\ \ j = n-1, n-2, \ldots, 1\,.$$

Here $\Xi_k = \left[\lambda_1^k/W'(\lambda_1), \ldots, \lambda_n^k/W'(\lambda_n)\right]^\top, \sigma_k = \sum_{j=1}^{n} \lambda_j^{n+k-1}/W'(\lambda_j)$, $k = \overline{1,n}$, and $W(x) = \prod_{k=1}^{n}(x - \lambda_k)$. The obtained result is applied for realization of systematic coding of the n-vector of the information blocks with the aid of multiplication (in an appropriate Galois field) by the matrix $\mathbf{K} = [\widetilde{W}_i(a^{N-j-1})]$, $i = \overline{1,m}$, $j = \overline{0,n-1}$. Here $\widetilde{W}_\ell(x)$, $\ell = \overline{1,m}$, denote the basic Lagrange interpolation polynomials generated by the powers of a primitive element of the field, while m stands for the number of redundancy blocks (syndromes). In the framework of this ideology, an error correcting procedure is also realized. The program implementation in C demonstrates high performance results (compared to existing software) with solid perspectives for parallelization. Refs 17. Fig 1.

Keywords: error-correcting codes, Reed—Solomon codes, Vandermonde matrix.

1. Введение. Рост объемов информации, хранимой человечеством, за последние годы увеличился настолько, что в системах хранения данных (СХД) стали использовать не отдельные накопители («жесткие диски»), но массивы накопителей. Наряду с требованиями, предъявляемыми к скорости записи и чтения, существенным является и обеспечение отказоустойчивости таких массивов. Для последней задачи применяются различные алгоритмы помехоустойчивого кодирования. Изначально они разрабатывались для каналов передачи данных и требовали значительных вычислительных ресурсов при кодировании и декодировании. Широко распространенные в настоящее время технологии хранения данных, известные как RAID (Redundant Array of Independent Disks), также изначально использовали специализированное оборудование (ASIC контроллеры) для кодирования и декодирования. Однако дороговизна разработки и производства такого оборудования сформировала потребность организации процедуры кодирования с помощью процессоров общего назначения. Поэтому актуальной стала задача повышения скорости процессов кодирования за счет оптимизации алгоритмов и их реализаций в RAID.

Каждый RAID массив представляет собой набор жестких дисков, воспринимаемых внешней системой как единое целое. Для повышения надежности массивов на них записываются и хранятся не только информационные данные, но и некоторая избыточная информация (синдромы), которая позволяет восстанавливать данные в случае их частичной утраты. Синдромы рассчитываются на этапе кодирования, а их использование для контроля целостности и восстановления утраченных данных производится на этапе декодирования. Различают два типа потери данных. Первый связан с выходом из строя одного блока, расположение которого обнаруживается системами технического (аппаратного) контроля. В этом случае место утраты данных известно и говорят об *отказе* блока (failed block или failure). Во втором случае данные искажаются в процессе записи, хранения или чтения, но системы аппаратного контроля не выявляют ошибки (ни самого факта ее присутствия, ни ее местоположения). В таком случае говорят о *скрытом повреждении данных* (silent data corruption, SDC).

Различные подходы к восстановлению нескольких отказов описаны в работах [1, 2]. Их суть заключается в применении матрицы Коши или матрицы Вандермонда в качестве матрицы кодирования. Алгоритм обнаружения и исправления одного скрытого повреждения рассматривается в статье [3], но излагаемый в ней подход

не позволяет исправлять большее количество скрытых повреждений. В последние годы были разработаны и другие коды, такие как регенерирующие (Regenerating codes) [4], LRC [5], а также пирамидальные [6]; все они нацелены на уменьшение количества блоков, которые необходимо прочитать, чтобы восстановить отказавший блок.

В случае необходимости восстановления большого количества отказов и скрытых повреждений универсальным решением являются коды Рида—Соломона (RS-коды). Основное их преимущество заключается в обеспечении высокой степени отказоустойчивости при минимальной избыточности, благодаря чему они очень популярны в СХД. Математические основы этих кодов и практические рекомендации по их использованию описаны в монографиях [7–9]. Следует заметить, что с учетом изменения архитектуры процессоров со временем многие задачи, описанные в [7–9], потеряли свою актуальность или нашли новые решения (см., например, [10, 11]). Однако же некоторые особенности остаются еще узкими местами в практике применения RS-кодов. Одну из таких сложностей представляет алгоритм кодирования и декодирования, основанный на операциях с полиномами над конечными полями. Существует несколько способов оптимизации расчетов RS-кодов, например основанный на применении быстрого преобразования Фурье [12].

В настоящей работе предлагается новый подход к RS-кодам, использующий матричный формализм, в том числе алгоритм обращения матрицы Вандермонда. Статья организована следующим образом. В п. 2 изложен алгоритм обращения матрицы Вандермонда. В п. 3 кратко описан стандартный способ построения систематического RS-кода, а в п. 4 — предлагаемый авторами матричный вариант алгоритмов кодирования и исправления ошибок. В п. 5 приведены результаты сравнения производительности этого варианта с реализациями помехоустойчивых кодов из популярных специализированных библиотек.

2. Обращение матрицы Вандермонда. Для наглядности изложим здесь результаты в терминологии бесконечных полей, хотя в п. 3–5 они будут использованы исключительно только для полей конечных.

Пусть задан набор различных элементов $\{\lambda_1, \ldots, \lambda_n\}$ поля. Будем называть матрицу вида

$$V_{m,n}(\lambda_1, \lambda_2, \ldots, \lambda_n) = \begin{pmatrix} 1 & 1 & \ldots & 1 \\ \lambda_1 & \lambda_2 & \ldots & \lambda_n \\ \lambda_1^2 & \lambda_2^2 & \ldots & \lambda_n^2 \\ \vdots & \vdots & & \vdots \\ \lambda_1^{m-1} & \lambda_2^{m-1} & \ldots & \lambda_n^{m-1} \end{pmatrix}$$

(m, n)-матрицей Вандермонда.

Нас прежде всего интересуют способы обращения квадратной матрицы Вандермонда $V_{n,n}(\lambda_1, \lambda_2, \ldots, \lambda_n)$; для краткости будем обозначать ее просто V. С этой целью рассмотрим полиномы

$$W(x) = \prod_{j=1}^{n}(x - \lambda_j), \quad W_j(x) = \frac{W(x)}{x - \lambda_j}, \quad j = \overline{1, n}\,,$$

и вычислим набор базовых интерполяционных полиномов Лагранжа

$$\widetilde{W}_j(x) = \frac{W_j(x)}{W_j(\lambda_j)} = \frac{W_j(x)}{W'(\lambda_j)} = w_{j0} + w_{j1}x + \cdots + w_{j,n-1}x^{n-1}, \quad j = \overline{1, n}\,. \tag{1}$$

Будем говорить о полиномах (1), как о порожденных набором $\{\lambda_1, \ldots, \lambda_n\}$.

Теорема 2.1 ([13]). *Имеет место равенство*

$$V^{-1} = [w_{j,k-1}]_{j,k=1}^{n} = \begin{pmatrix} w_{10} & w_{11} & \dots & w_{1,n-1} \\ w_{20} & w_{21} & \dots & w_{2,n-1} \\ \dots & & & \dots \\ w_{n0} & w_{n1} & \dots & w_{n,n-1} \end{pmatrix} . \tag{2}$$

Доказательство основывается на элементарном свойстве базовых интерполяционных полиномов:

$$\widetilde{W}_j(\lambda_k) = \begin{cases} 1 & \text{при } j = k, \\ 0 & \text{при } j \neq k. \end{cases}$$

Умножение матрицы (2) на матрицу Вандермонда V слева имеет результатом матрицу, состоящую из величин $\widetilde{W}_j(\lambda_k)$. □

Таким образом, обращение матрицы Вандермонда фактически сводится к вычислению коэффициентов базовых интерполяционных полиномов. Задача заключается теперь в параллелизации алгоритма вычисления этих коэффициентов. Сначала вычислим величины

$$\xi_1 = \frac{1}{W'(\lambda_1)}, \dots, \xi_n = \frac{1}{W'(\lambda_n)}$$

и составим из них столбец

$$\Xi_0 = [\xi_1, \dots, \xi_n]^\top$$

(здесь и далее в статье $\top$ означает транспонирование). Умножим его на столбец

$$\Lambda = [\lambda_1, \dots, \lambda_n]^\top \ ,$$

причем умножение векторов производится поэлементно (так называемое *адамарово произведение*):

$$\Xi_1 = \Xi_0 \otimes \Lambda = [\xi_1\lambda_1, \dots, \xi_n\lambda_n]^\top \ .$$

Продолжив умножение, образуем последовательность

$$\Xi_0, \Xi_1 = \Xi_0 \otimes \Lambda, \Xi_2 = \Xi_1 \otimes \Lambda, \dots, \Xi_{n-1} = \Xi_{n-2} \otimes \Lambda, \dots, \Xi_{2n-2} = \Xi_{2n-3} \otimes \Lambda \ . \tag{3}$$

Лемма. *Сумма элементов любого из столбцов $\Xi_0, \dots, \Xi_{n-2}$ равна 0. Сумма элементов столбца Ξ_{n-1} равна 1.*

Доказательство. Равенства известны как равенства Эйлера—Лагранжа [14]

$$\sum_{j=1}^{n} \frac{\lambda_j^k}{W'(\lambda_j)} = \begin{cases} 0 & \text{при } k < n-1, \\ 1 & \text{при } k = n-1. \end{cases} \tag{4}$$

□

Теорема 2.2. *Вычислим суммы элементов столбцов $\Xi_n, \Xi_{n+1}, \dots, \Xi_{2n-2}$, т. е. величины*

$$\sigma_k = \sum_{j=1}^{n} \frac{\lambda_j^{n+k-1}}{W'(\lambda_j)}, \quad k = \overline{1, n-1} \ . \tag{5}$$

Столбцы матрицы $V^{-1} = \left[V_{[1]}^{-1}, V_{[2]}^{-1}, \ldots, V_{[n]}^{-1}\right]$ *связаны со столбцами* (3) *следующими рекурсивными соотношениями:*

$$\left\{\begin{array}{lcl} V_{[n]}^{-1} & = & \Xi_0, \\ V_{[n-1]}^{-1} & = & \Xi_1 - \sigma_1 V_{[n]}^{-1}, \\ V_{[n-2]}^{-1} & = & \Xi_2 - \sigma_1 V_{[n-1]}^{-1} - \sigma_2 V_{[n]}^{-1}, \\ \ldots & & \ldots \\ V_{[1]}^{-1} & = & \Xi_{n-1} - \sigma_1 V_{[2]}^{-1} - \sigma_2 V_{[3]}^{-1} - \cdots - \sigma_{n-1} V_{[n]}^{-1}. \end{array}\right. \tag{6}$$

Д о к а з а т е л ь с т в о. На основании (4) имеем

$$V \cdot V_{[n]}^{-1} = V \cdot \Xi_0 = \left[\sum_{j=1}^{n} \frac{1}{W'(\lambda_j)}, \sum_{j=1}^{n} \frac{\lambda_j}{W'(\lambda_j)}, \ldots, \sum_{j=1}^{n} \frac{\lambda_j^{n-1}}{W'(\lambda_j)}\right]^{\top} = [0, 0, \ldots, 0, 1]^{\top}.$$

Далее с использованием только что полученного равенства получаем

$$V \cdot V_{[n-1]}^{-1} = V \cdot \Xi_1 - \sigma_1 V \cdot V_{[n]}^{-1} = [0, 0, \ldots, 1, \sigma_1]^{\top} - \sigma_1 [0, 0, \ldots, 0, 1]^{\top} = [0, 0, \ldots, 1, 0]^{\top}.$$

Аналогично, применяя уже два доказанных равенства, доказываем третье из (6):

$$V \cdot V_{[n-2]}^{-1} = V \cdot \Xi_2 - \sigma_1 V \cdot V_{[n-1]}^{-1} - \sigma_2 V \cdot V_{[n]}^{-1} =$$

$$= [0, 0, \ldots, 1, \sigma_1, \sigma_2]^{\top} - \sigma_1 [0, 0, \ldots, 1, 0]^{\top} - \sigma_2 [0, 0, \ldots, 0, 1]^{\top} = [0, 0, \ldots, 1, 0, 0]^{\top}.$$

И так далее. □

Пример 2.1. Вычислить

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 4 & 9 \end{pmatrix}^{-1}.$$

Решение. Здесь

$$\lambda_1 = 1,\ \lambda_2 = 2,\ \lambda_3 = 3,\ W'(1) = 2,\ W'(2) = -1,\ W'(3) = 2\,.$$

Вычисляем последовательность столбцов (3) и для компактности записываем ее в виде матрицы

$$\begin{bmatrix} 1/2 & 1/2 & 1/2 & 1/2 & 1/2 \\ -1 & -2 & -4 & -8 & -16 \\ 1/2 & 3/2 & 9/2 & 27/2 & 81/2 \end{bmatrix}.$$

Суммируем элементы двух последних столбцов:

$$\sigma_1 = 6, \quad \sigma_2 = 25\,.$$

Используем формулы (6):

$$V_{[3]}^{-1} = \begin{bmatrix} 1/2 \\ -1 \\ 1/2 \end{bmatrix},$$

$$V_{[2]}^{-1} = \begin{bmatrix} 1/2 \\ -2 \\ 3/2 \end{bmatrix} - 6 \begin{bmatrix} 1/2 \\ -1 \\ 1/2 \end{bmatrix} = \begin{bmatrix} -5/2 \\ 4 \\ -3/2 \end{bmatrix},$$

$$V_{[1]}^{-1} = \begin{bmatrix} 1/2 \\ -4 \\ 9/2 \end{bmatrix} - 6 \begin{bmatrix} -5/2 \\ 4 \\ -3/2 \end{bmatrix} - 25 \begin{bmatrix} 1/2 \\ -1 \\ 1/2 \end{bmatrix} = \begin{bmatrix} 3 \\ -3 \\ 1 \end{bmatrix}.$$

Замечание 1. Для вещественных матриц Вандермонда формулы (6) можно рассматривать как результат процесса ортогонализации Грама—Шмидта [15], примененного к системе столбцов $\{\Xi_0, \Xi_1, \ldots, \Xi_{n-1}\} \subset \mathbb{R}^n$ при скалярном произведении в $\mathbb{R}^n$, задаваемом формулой $\langle X, Y\rangle = X^\top (V^\top V) Y$.

Замечание 2. Можно доказать, что величина (5) может быть вычислена как сумма всевозможных мономов степени k от элементов $\lambda_1, \ldots, \lambda_n$:

$$\sigma_k = \sum_{\substack{j_1 \geqslant 0, j_2 \geqslant 0, \ldots j_n \geqslant 0 \\ j_1 + j_2 + \cdots + j_n = k}} \lambda_1^{j_1} \lambda_2^{j_2} \times \cdots \times \lambda_n^{j_n}.$$

Теорема 2.3. *Пусть задан набор неотрицательных разных целых чисел $\{k_0, k_1, \ldots, k_{m-1}\}$ и элемент λ поля такой, что элементы $\lambda^{k_0}, \lambda^{k_1}, \ldots, \lambda^{k_{m-1}}$ все различны. Пусть*

$$g(x) = \prod_{j=0}^{m-1} (x - \lambda^j).$$

Для произвольного полинома $f(x)$ над полем существует единственная пара полиномов $\mathcal{Q}(x)$ и

$$\mathcal{R}(x) = C_0 x^{k_0} + C_1 x^{k_1} + \cdots + C_{m-1} x^{k_{m-1}} \tag{7}$$

такая, что

$$f(x) \equiv \mathcal{Q}(x) g(x) + \mathcal{R}(x). \tag{8}$$

Доказательство. Если подставить в тождество (8) значения $1, \lambda, \lambda^2, \ldots, \lambda^{m-1}$, то получим систему линейных уравнений относительно коэффициентов $C_0, C_1, \ldots, C_{m-1}$. Матричный вид этой системы

$$V_{m,m}(\lambda^{k_0}, \lambda^{k_1}, \ldots, \lambda^{k_{m-1}}) \mathbf{C} = \left[f(1), f(\lambda), \ldots, f(\lambda^{m-1})\right]^\top \tag{9}$$

при $\mathbf{C} = [C_0, C_1, \ldots, C_{m-1}]^\top$.

По условию теоремы матрица системы (9) невырождена. Следовательно, система имеет единственное решение, которое однозначно определяет полином $\mathcal{R}(x)$. Тогда разность $f(x) - \mathcal{R}(x)$ нацело делится на полином $g(x)$, и полином $\mathcal{Q}(x)$ определяется как частное этого деления. □

В случае $k_j = j - 1$, $j = \overline{1, r}$, теорема 2.3 фактически предоставляет способ осуществления деления $f(x)$ на $g(x)$. Нас, однако, будут интересовать упрощения вычислений «обобщенного остатка» (7), например, для случая полинома $f(x)$ вида

$$f(x) = A_1 x_1^{n_1} + \cdots + A_r x^{n_r} \quad \text{при} \quad \{n_1, \ldots, n_r\} \subset \mathbb{Z},\ 0 \leqslant n_1 < \cdots < n_r.$$

С этой целью решение системы (9) запишем так:

$$[C_0, C_1, \ldots, C_{m-1}]^\top = \left[V_{m,m}(\lambda^{k_0}, \lambda^{k_1}, \ldots, \lambda^{k_{m-1}})\right]^{-1} \left[f(1), f(\lambda), \ldots, f(\lambda^{m-1})\right]^\top =$$

$$= \left[V_{m,m}(\lambda^{k_0}, \lambda^{k_1}, \ldots, \lambda^{k_{m-1}})\right]^{-1} V_{m,r}(\lambda^{n_1}, \ldots, \lambda^{n_r}) \left[A_1, \ldots, A_r\right]^{\top} . \qquad (10)$$

На основании теоремы 2.1 приходим к следующему результату.

Теорема 2.4. *Коэффициенты полинома* (7) *находятся из равенства*

$$\begin{pmatrix} C_0 \\ C_1 \\ \vdots \\ C_{m-1} \end{pmatrix} = \begin{pmatrix} \widetilde{W}_1(\lambda^{n_1}) & \widetilde{W}_1(\lambda^{n_2}) & \ldots & \widetilde{W}_1(\lambda^{n_r}) \\ \widetilde{W}_2(\lambda^{n_1}) & \widetilde{W}_2(\lambda^{n_2}) & \ldots & \widetilde{W}_2(\lambda^{n_r}) \\ \vdots & \vdots & & \vdots \\ \widetilde{W}_m(\lambda^{n_1}) & \widetilde{W}_m(\lambda^{n_2}) & \ldots & \widetilde{W}_m(\lambda^{n_r}) \end{pmatrix} \begin{pmatrix} A_1 \\ \vdots \\ A_r \end{pmatrix} .$$

Здесь $\widetilde{W}_1(x), \ldots, \widetilde{W}_m(x)$ *— базовые интерполяционные полиномы Лагранжа, порожденные набором* $\{\lambda^{k_0}, \lambda^{k_1}, \ldots, \lambda^{k_{m-1}}\}$.

З а м е ч а н и е 3. Предположение о неотрицательности показателей $n_j, j = \overline{1,r}$, и $k_s, s = \overline{0,m-1}$, в формулировках теорем 2.3 и 2.4 не является существенным: эти результаты позволяют определить операцию деления и для обобщенных полиномов (Лорана), т. е. линейных комбинаций мономов с произвольными целыми показателями.

3. Систематическое кодирование кодов Рида—Соломона. РС-коды представляют собой циклические коды, позволяющие исправлять ошибки в блоках данных. От исходного набора (вектора) информационных кодовых блоков $(D_0, D_1, \ldots, D_{n-1})$ по специальным формулам рассчитываются *синдромы*, т. е. служебная информация в виде вектора блоков $(C_0, C_1, \ldots, C_{m-1})$. Формулы расчета синдромов подбираются таким образом, чтобы гарантированно исправить определенное количество ошибок, которые могут возникнуть в процессе хранения или передачи.

Различают два способа расчета синдромов РС-кодов: систематическое кодирование и несистематическое кодирование. При систематическом кодировании сохраняемая или передаваемая информация представляется в виде конкатенации информационных блоков и блоков синдромов $(D_0, D_1, \ldots, D_{n-1}, C_0, C_1, \ldots, C_{m-1})$ в единый кодовый вектор. В дальнейшем будем рассматривать только такое кодирование как наиболее востребованное на практике. При его применении в случае отсутствия повреждений данных не требуется осуществлять декодирование и, следовательно, тратится меньше вычислительных ресурсов на извлечение информационных блоков.

Расчет синдромов осуществляется с использованием арифметики конечных полей. Каждый кодовый блок D_j, представленный в виде w-битного вектора, считается элементом некоторого поля Галуа $GF(2^w)$. Мы не останавливаемся здесь подробно на описании арифметики полей Галуа [7–9]. Напомним только, что множество ненулевых элементов поля $GF(2^w)$ замкнуто относительно умножения, и в нем существует *примитивный элемент* a, т. е. такой, что все его степени $a^0, a^1, \ldots, a^{2^w-2}$ представляют все эти элементы поля.

Пусть код должен быть устойчив к отказу ℓ блоков и дополнительно к появлению самое большее t скрытых повреждений. Тогда избыточность кода должна состоять, как минимум, из $m = \ell + 2t$ синдромов. Для их расчета выбирается порождающий полином кода, например в виде

$$g(x) = \prod_{j=0}^{m-1} (x + a^j) .$$

По информационным кодовым блокам $D_0, D_1, \ldots, D_{n-1}$ составляется полином

$$G(x) = D_0 x^{n-1} + D_1 x^{n-2} + \cdots + D_{n-2} x + D_{n-1} .$$

Для расчета синдромов разделим полином $\widetilde{G}(x) \equiv G(x)x^m$ на $g(x)$ и обозначим через $\mathcal{Q}(x)$ частное, а через $\mathcal{R}(x)$ — остаток от деления:

$$\widetilde{G}(x) \equiv \mathcal{Q}(x)g(x) + \mathcal{R}(x), \quad \deg \mathcal{R} < m\,.$$

В качестве синдромов берутся коэффициенты остатка

$$\mathcal{R}(x) = C_0 x^{m-1} + C_1 x^{m-2} + \cdots + C_{m-1}\,,$$

а в качестве кодового вектора выбирается вектор

$$(Y_0, \ldots, Y_{N-1}) = (D_0, D_1, \ldots, D_{n-1}, C_0, \ldots, C_{m-1}), \quad N = n + m, \tag{11}$$

коэффициентов полинома $\overline{G}(x) = \widetilde{G}(x) + \mathcal{R}(x)$. Для этого полинома выполняются условия

$$\overline{G}(1) = 0, \quad \overline{G}(a) = 0, \ldots, \ \overline{G}(a^{m-1}) = 0\,.$$

Обратно, если необходимо проверить некоторый вектор $(Y_0, \ldots, Y_{N-1})$ на наличие в нем ошибок, то составляем полином $\overline{G}(x) = Y_0 x^{N-1} + \cdots + Y_{N-1}$ и вычисляем его значения на степенях примитивного элемента поля:

$$S_j = \overline{G}(a^j) = \sum_{i=0}^{N-1} Y_i a^{(N-1-i)j}, \quad j = \overline{0, m-1}\,. \tag{12}$$

Если хотя бы одно из этих значений *контрольных сумм* ненулевое, то это свидетельствует о наличии в векторе ошибок. Значения $S_0, \ldots, S_{m-1}$ используются тогда для восстановления данных, как будет показано в п. 4.

4. Кодирование и исправление ошибок: матричный подход. Нахождение синдромов $C_0, \ldots, C_{m-1}$ способом, описанным в п. 3, требует существенных машинных ресурсов, поскольку использует трудоемкую операцию деления полиномов над конечными полями. В альтернативу этому предлагается применить для кодирования изложенный в п. 2 подход к вычислению остатков, основанный на матричном формализме. По теореме 2.4 синдромы связаны с информационными блоками соотношением

$$\begin{pmatrix} C_0 \\ C_1 \\ \vdots \\ C_{m-1} \end{pmatrix} = \begin{pmatrix} \widetilde{W}_1(a^{N-1}) & \widetilde{W}_1(a^{N-2}) & \cdots & \widetilde{W}_1(a^m) \\ \widetilde{W}_2(a^{N-1}) & \widetilde{W}_2(a^{N-2}) & \cdots & \widetilde{W}_2(a^m) \\ \vdots & \vdots & & \vdots \\ \widetilde{W}_m(a^{N-1}) & \widetilde{W}_m(a^{N-2}) & \cdots & \widetilde{W}_m(a^m) \end{pmatrix} \begin{pmatrix} D_0 \\ D_1 \\ \vdots \\ D_{n-1} \end{pmatrix}, \tag{13}$$

представляющим процедуру кодирования в матричном виде. Здесь $\widetilde{W}_1(x), \ldots, \widetilde{W}_m(x)$ означают базовые интерполяционные полиномы Лагранжа, порожденные элементами поля $\{a^{m-1}, a^{m-2}, \ldots, 1\}$. Матрицу из формулы (13)

$$\mathbf{K} = [\widetilde{W}_i(a^{N-j-1})], \quad i = \overline{1, m}, \quad j = \overline{0, n-1}\,,$$

назовем *матрицей кодирования*. При фиксированных числах m и n ее расчет производится однократно и позволяет осуществлять процедуру кодирования для произвольного вектора из n информационных блоков — и в этом заключается его преимущество перед традиционным подходом, основанном на полиномиальном формализме. Заметим также, что каждый синдром рассчитывается за $O(n)$ операций и вычисление

всего набора синдромов может производиться параллельно, т. е. независимо друг от друга.

Использовать свойства матрицы Вандермонда можно и для восстановления утраченных данных. Предположим, что для блоков данных $D_0, D_1, \ldots, D_{n-1}$ были рассчитаны m синдромов $C_0, C_1, \ldots, C_{m-1}$, например, по формуле (13). Пусть в кодовой последовательности (11) произошло $\ell \leqslant m$ отказов, т. е. повреждений блоков с известными номерами $k_1, \ldots, k_\ell$. Обозначим

$$\mathbf{Y}_{\text{ғ}} = [Y_{k_1}, \ldots, Y_{k_\ell}]^\top$$

столбец отказавших блоков, а

$$\mathbf{Y}_{\text{ѕ}} = [Y_{r_1}, \ldots, Y_{r_{N-\ell}}]^\top,$$

$$\{r_1, \ldots, r_{N-\ell}\} = \{0, \ldots, N-1\} \setminus \{k_1, \ldots, k_\ell\}, \quad r_1 < r_2 < \cdots < r_{N-\ell},$$

— столбец неповрежденных блоков.

Тогда алгоритм восстановления отказавших блоков заключается в следующем. Сначала пересчитываются значения первых ℓ контрольных сумм из (12) с пропуском при расчете отказавших блоков:

$$\widehat{S}_j = \sum_{\substack{i=\overline{0,N-1} \\ i \notin \{k_1, \ldots, k_\ell\}}} Y_i a^{(N-1-i)j}, \quad j = \overline{0, \ell-1}\,. \tag{14}$$

В матричном виде это можно записать так:

$$\widehat{\mathbf{S}} = V_{\ell,N-\ell}(a^{N-r_1-1}, a^{N-r_2-1}, \ldots, a^{N-r_{N-\ell}-1})\mathbf{Y}_{\text{ѕ}}, \tag{15}$$

где $\widehat{\mathbf{S}} = \left[\widehat{S}_0, \widehat{S}_1, \ldots, \widehat{S}_{\ell-1}\right]^\top$.

Для восстановления отказавших блоков составим систему линейных уравнений суммированием соответствующих уравнений (12) и (14):

$$\sum_{i=1}^{\ell} Y_{k_i} a^{(N-1-k_i)j} = \widehat{S}_j, \quad j = \overline{0, \ell-1}\,.$$

Вектор $\mathbf{Y}_{\text{ғ}}$ выражается отсюда посредством обращения матрицы Вандермонда:

$$\mathbf{Y}_{\text{ғ}} = [V_{\ell,\ell}(a^{N-k_1-1}, a^{N-k_2-1}, \ldots, a^{N-k_\ell-1})]^{-1}\widehat{\mathbf{S}}\,. \tag{16}$$

Подставив (15) в (16), получим выражение отказавших блоков через неповрежденные:

$$\mathbf{Y}_{\text{ғ}} = [V_{\ell,\ell}(a^{N-k_1-1}, \ldots, a^{N-k_\ell-1})]^{-1} V_{\ell,N-\ell}(a^{N-r_1-1}, a^{N-r_2-1}, \ldots, a^{N-r_{N-\ell}-1})\mathbf{Y}_{\text{ѕ}}\,. \tag{17}$$

В соответствии с формулой (10) можно утверждать, что отказавшие блоки составляют набор коэффициентов обобщенного остатка

$$c_1 x^{N-1-k_1} + c_2 x^{N-1-k_2} + \cdots + c_\ell x^{N-1-k_\ell}$$

от деления полинома

$$Y_{r_1} x^{N-1-r_1} + Y_{r_2} x^{N-1-r_2} + \cdots + Y_{r_{N-\ell}} x^{N-1-r_{N-\ell}}$$

на полином

$$\prod_{i=0}^{\ell-1}(x+a^i)\,.$$

На основании теоремы 2.4 можно записать произведение матриц, стоящих в правой части (17) в виде

$$\mathbf{R}_{\ell,N-\ell}=\begin{pmatrix}\widetilde{W}_1(a^{N-r_1-1}) & \widetilde{W}_1(a^{N-r_2-1}) & \dots & \widetilde{W}_1(a^{N-r_{N-\ell}-1})\\ \widetilde{W}_2(a^{N-r_1-1}) & \widetilde{W}_2(a^{N-r_2-1}) & \dots & \widetilde{W}_2(a^{N-r_{N-\ell}-1})\\ \vdots & \vdots & & \vdots\\ \widetilde{W}_\ell(a^{N-r_1-1}) & \widetilde{W}_\ell(a^{N-r_2-1}) & \dots & \widetilde{W}_\ell(a^{N-r_{N-\ell}-1})\end{pmatrix}$$

или компактно —

$$\mathbf{R}=[\widetilde{W}_i(a^{N-j-1})],\quad i=\overline{1,\ell},\quad j\in\{0,\dots,N-1\}\setminus\{k_1,\dots,k_\ell\}\,;$$

здесь $\widetilde{W}_1(x),\dots,\widetilde{W}_\ell(x)$ — базовые интерполяционные полиномы Лагранжа, порожденные набором $\{a^{N-k_1-1},\dots,a^{N-k_\ell-1}\}$.

Окончательно значения отказавших блоков могут быть восстановлены по формуле

$$\mathbf{Y}_{\mathfrak{f}}=\mathbf{R}\cdot\mathbf{Y}_{\mathfrak{s}}\,. \tag{18}$$

Матрицу $\mathbf{R}$ естественно назвать *матрицей восстановления* (декодирования). Данный способ восстановления имеет то достоинство, что восстановление отказавших блоков происходит напрямую из неповрежденных и не требует дополнительной памяти для хранения промежуточных вычислений. Также следует отметить, что при восстановлении данных, например в СХД, расчет матрицы восстановления $\mathbf{R}$ выполняется однократно для всех векторов, в которых необходимо восстановить данные. В связи с этим после расчета матрицы для ее элементов могут быть вычислены вспомогательные значения, необходимые для быстрого умножения, описанного в [10], что приведет к увеличению скорости восстановления.

Описанные способы кодирования и декодирования корректно восстанавливают данные при выполнении условия $m+n<2^w$, т. е. общее число блоков не превышает порядка используемого поля. При необходимости исправления большего количества ошибок требуется переход к полям бóльших порядков.

PC-коды замечательны еще и тем, что позволяют не только исправлять отказы, но также обнаруживать наличие и исправлять стирания, или скрытые повреждения (Silent Data Corruption, SDC), т. е. такие повреждения исходных данных, которые не отслеживаются аппаратными средствами.

Приведем здесь лишь краткое описание алгоритма обнаружения и исправления скрытых повреждений, поскольку он подробно рассмотрен в литературе [7, 8]. Пусть в последовательности блоков (11) имеется ℓ отказов в блоках с номерами $k_1,\dots,k_\ell$ и t скрытых повреждений в блоках с номерами $j_1,j_2,\dots,j_t$, которые априори неизвестны (как неизвестно и само их количество). В предположении, что

выполняется условие $\ell + 2t \leqslant m$, все скрытые повреждения могут быть обнаружены и исправлены. Чтобы обнаружить присутствие и найти расположение скрытых повреждений в таком случае необходимо:

1) пересчитать значения контрольных сумм с пропуском отказавших блоков по формуле (15);

2) (в случае $\ell = 0$ этот шаг пропускается) построить вспомогательный полином

$$z(x) = \prod_{i=1}^{\ell} (x + a^{N-k_i-1}) = z_0 + z_1 x + \cdots + z_\ell x^\ell ;$$

3) посчитать величины

$$U_i = \begin{cases} \sum\limits_{j=0}^{\ell} z_j \widehat{S}_{i+j}, & \text{если } \ell > 0, \\ \widehat{S}_i, & \text{если } \ell = 0, \end{cases} \qquad i = \overline{0, m-\ell-1};$$

4) если хотя бы одно из условий $U_0 = 0, \ldots, U_{m-\ell-1} = 0$ не выполнено, то делается вывод о наличии скрытых повреждений;

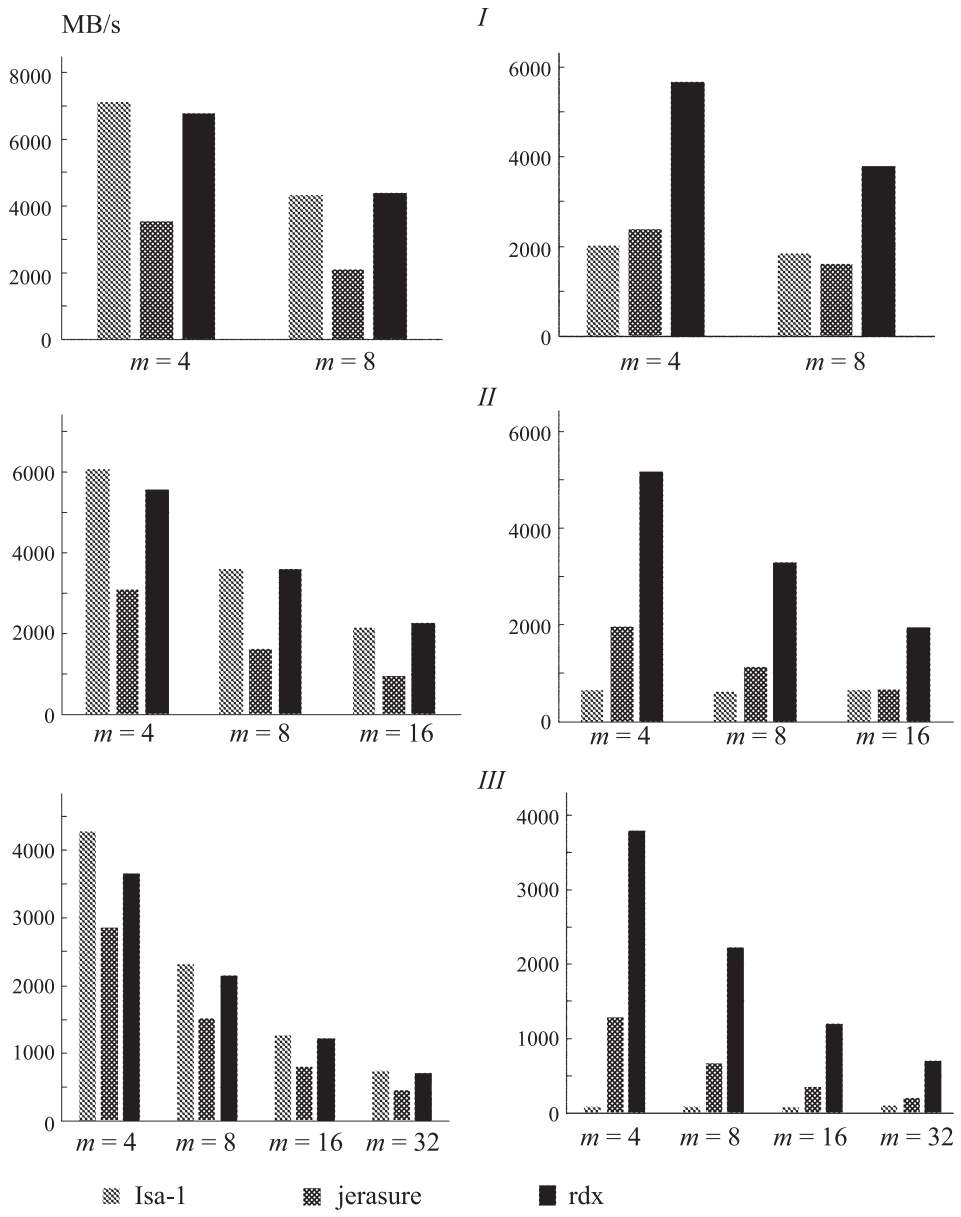
5) построить полином локаторов ошибок — например, по алгоритму Берлекампа—Месси. На вход алгоритма подается последовательность $U_{m-\ell-1}, U_{m-\ell-2}, \ldots, U_0$. На выходе получается полином степени t. Если количество скрытых повреждений в точности равно t, то корнями этого полинома являются a^{N-j_i-1}, $i = \overline{1,t}$. С определением этих корней устанавливаются места скрытых повреждений.

Когда все они определены, задача исправления поврежденных данных оказывается сведенной к случаю наличия в последовательности (11) исключительно только отказов. Последние могут быть исправлены, например, по формуле (18).

5. Результаты вычислительных экспериментов. Для практической реализации описанных алгоритмов кодирования и декодирования был выбран язык C, поскольку он обеспечивает доступ к векторным инструкциям процессора, и в то же время компиляторы языка C реализуют различные алгоритмы оптимизации исполняемого кода. Было произведено сравнение описанного в п. 4 алгоритма с наиболее популярными библиотеками помехоустойчивого кодирования ISA-L [16], Jerasure [17]. Производилось сравнение только скорости работы алгоритма кодирования или декодирования, без учета считывания данных с дисков, на системе со следующей конфигурацией:

- ОС: Debian 8;
- CPU: Intel Core i7-2600 3.40GHz;
- RAM: 8GB;
- компилятор gcc 4.8.

На рисунке, *I–III* представлены результаты сравнений алгоритмов с большим количеством контрольных сумм (n — количество информационных блоков, m — количество контрольных сумм, $N = n + m$ — общее количество блоков). Матричная версия алгоритма кодирования из п. 4 обозначена как “rdx”. Можно увидеть, что эта версия показывает примерно такие же показатели скорости кодирования, как и ISA-L, но более высокие значения скорости исправления ошибок. Следует заметить, что в отличие от упомянутых библиотек разработанная авторами программная реализация алгоритма позволяет исправлять не только отказы, но и скрытые повреждения.



Сравнение скорости кодирования (слева) и декодирования (справа)
при $n = 16$ (I), 32 (II) и 96 (III)

6. Заключение. В настоящей работе решается проблема повышения эффективности применения кодов Рида—Соломона для исправления поврежденных данных. Предложенный алгоритм, основанный на матричных операциях с использованием матрицы Вандермонда, программно реализован на языке C. В сравнении с существующими специализированными пакетами эта программная реализация показывает высокие значения скорости кодирования и декодирования, притом что дополнительно обеспечивает высокую степень отказоустойчивости за счет возможности

исправления как отдельно отказов или скрытых повреждений, так их произвольных комбинаций.

Литература

1. *Plank J.* A Tutorial on Reed—Solomon coding for fault-tolerance in RAID-like systems // Software-Practice & Experience. 1997. Vol. 27(9). P. 995–1012.

2. *Plank J., Xu L.* Optimizing cauchy Reed—Solomon codes for fault-tolerant network storage applications // The 5th IEEE Intern. symposium on network computing and applications. 2006. P. 173–180.

3. *Plank J., Blaum M., Hafner J.* SD codes: erasure codes designed for how storage systems really fail // FAST-2013: 11th USENIX conference on file and storage technologies. 2013. P. 95–104.

4. *Papailiopoulos D., Luo J., Dimakis A., Huang C., Li J.* Simple regenerating codes: network coding for cloud storage // INFOCOM. Proceedings IEEE. 2012. P. 2801–2805.

5. *Huang C., Simitci H., Xu Y., Ogus A., Calder B., Gopalan P., Li J., Yekhanin S.* Erasure coding in windows azure storage // USENIX annual technical conference. 2012. URL: https://www.usenix.org/system/files/conference/atc12/atc12-final181_0.pdf (дата обращения: 11.07.2016).

6. *Huang C., Chen M., Li J.* Pyramid Codes: flexible schemes to trade space for access efficiency in reliable data storage systems // NCA–2007: IEEE Intern. symposium on network computing and applications. 2007. P. 79–86.

7. *Peterson W., Weldon E.* Error-correcting codes. Cambridge: MIT Press, 1972. 572 p.

8. *Berlekamp E.* Algebraic coding theory. Singapore: World scientific Publishing Co, 2015. 501 p.

9. *Lidl R., Niederreiter H.* Finite fields. Vol. 1. 2nd ed. Cambridge: Cambridge University Press, 1997. 755 p.

10. *Plank J., Greenan K., Miller E.* Screaming fast galois field arithmetic using Intel SIMD instructions // FAST–2013: 11th USENIX conference on file and storage technologies. 2013. P. 299–306.

11. *Plank J.* XOR's lower bounds and MDS codes for storage // IEEE Information theory workshop. 2011. P. 529–551.

12. *Trifonov P.* Low-complexity implementation of RAID based on Reed–Solomon codes // ACM transactions on storage. 2015. URL: http://dl.acm.org/citation.cfm?id=2700308 (дата обращения: 26.10.2016).

13. *Knuth D.* The art of computer programming. Vol. 1. 3rd ed. Reading (Massachusetts): Addison-Wesley Press, 1997. 672 p.

14. *Утешев А. Ю., Калинина Е. А.* Лекции по высшей алгебре. Ч. II: учеб. пособие. СПб.: Соло, 2007. 279 с.

15. *Хорн Р., Джонсон Ч.* Матричный анализ / пер. с англ. Х. Д. Икрамова и др.; под ред. Х. Д. Икрамова. М: Мир, 1989. 655 с. (*Horn R. A., Johnson Ch. R.* Matrix analysis.)

16. Intel Intelligent Storage Acceleration Library // URL: https://01.org/intel%C2%AE-storage-acceleration-library-open-source-version (дата обращения: 11.07.2016).

17. Jerasure: Erasure Coding Library // URL: http://jerasure.org/ (дата обращения: 11.07.2016).

Для цитирования: Маров А. В., Утешев А. Ю. Матричный формализм кодов Рида—Соломона // Вестник Санкт-Петербургского университета. Сер. 10. Прикладная математика. Информатика. Процессы управления. 2016. Вып. 4. С. 4–17. DOI: 10.21638/11701/spbu10.2016.401

References

1. Plank J. A Tutorial on Reed–Solomon coding for fault-tolerance in RAID-like systems. *Software–practice & Experience*, 1997, vol. 27(9), pp. 995–1012.

2. Plank J., Xu L. Optimizing cauchy Reed–Solomon codes for fault-tolerant network storage applications. *The 5th IEEE Intern. symposium on network computing and applications*, 2006, pp. 173–180.

3. Plank J., Blaum M., Hafner J. SD codes: erasure codes designed for how storage systems really fail. *FAST-2013: 11th USENIX Conference on File and Storage Technologies*, 2013, pp. 95–104.

4. Papailiopoulos D., Luo J., Dimakis A., Huang C., Li J. Simple regenerating codes: network coding for cloud storage. *INFOCOM, Proceedings IEEE*, 2012, pp. 2801–2805.

5. Huang C., Simitci H., Xu Y., Ogus A., Calder B., Gopalan P., Li J., Yekhanin S. Erasure coding in windows azure storage. *USENIX annual technical conference*, 2012. URL: https://www.usenix.org/system/files/conference/atc12/atc12-final181_0.pdf (accessed: 11.07.2016).

6. Huang C., Chen M., Li J. Pyramid codes: flexible schemes to trade space for access efficiency in reliable data storage systems. *NCA–2007: IEEE Intern. symposium on network computing and applications*, 2007, pp. 79–86.

7. Peterson W., Weldon E. *Error-Correcting Codes*. Cambridge, MIT Press, 1972, 572 p.

8. Berlekamp E. *Algebraic coding theory*. Singapore, World scientific Publishing Co, 2015, 501 p.

9. Lidl R., Niederreiter H. *Finite Fields*. Vol. 1. 2nd ed. Cambridge, Cambridge University Press, 1997, 755 p.

10. Plank J., Greenan K., Miller E. Screaming fast galois field arithmetic using Intel SIMD instructions. *FAST–2013: 11th USENIX Conference on file and storage technologies*, 2013, pp. 299–306.

11. Plank J. XOR's lower bounds and MDS codes for storage. *IEEE Information Theory Workshop*, 2011, pp. 529–551.

12. Trifonov P. Low-complexity implementation of RAID based on Reed–Solomon codes. *ACM Transactions on Storage*, 2015. URL: http://dl.acm.org/citation.cfm?id=2700308 (accessed: 26.10.2016).

13. Knuth D. *The art of computer programming*. Vol. 1. 3rd ed. Reading (Massachusetts), Addison-Wesley Press, 1997, 672 p.

14. Uteshev A., Kalinina E. *Lekcii po algebre* [*Lectures in algebra*]. Pt II. Saint Petersburg, Solo Press, 2007, 279 p. (In Russian).

15. Horn R. A., Johnson C. R. *Matrix analysis*. Cambridge, Cambridge University Press, 1986, 612 p. (Russ ed.: Horn R. A., Johnson C. R. *Matricnij analiz*. Moscow, Mir Publ., 1989, 655 p.)

16. *Intel intelligent storage acceleration library*. URL: https://01.org/intel%C2%AE-storage-acceleration-library-open-source-version (accessed: 11.07.2016).

17. *Jerasure: Erasure coding library*. URL: http://jerasure.org/ (accessed: 11.07.2016).

For citation: Marov A. V., Uteshev A. Yu. Matrix formalism of the Reed—Solomon codes. *Vestnik of Saint Petersburg University. Series 10. Applied mathematics. Computer science. Control processes*, 2016, issue 4, pp. 4–17. DOI: 10.21638/11701/spbu10.2016.401

Статья рекомендована к печати проф. Л. А. Петросяном.

Статья поступила в редакцию 15 июля 2016 г.

Статья принята к печати 29 сентября 2016 г.