Блочный метод Ланцоша-Монтгомери с малым количеством обменов *

Д.А. Желтков[1], Н.Л. Замарашкин[1,2]
ИВМ РАН[1], ВМК МГУ[2]

Предложена параллельная реализация блочного метода Ланцоша-Монтгомери с уменьшенным числом обменов для решения больших разреженных систем линейных уравнений над конечными полями.

1. Введение

Необходимость решения больших разреженных систем линейных уравнений над конечными полями возникает в различных приложениях, например, в таких как факторизация больших составных чисел или решение задач дискретного логарифмирования в большом простом поле. Наиболее эффективными методами решения таких систем считаются алгоритмы двух типов:

1. алгоритмы типа Ланцоша-Монтгомери (см. [3,5,6,9,10,13,14]);
2. алгоритмы типа Видемана-Копперсмита (см. [1,4,7,8]).

Устройство алгоритмов обоих типов имеет много общего. Наиболее трудоемкое вычисление, как правило, связано с многократным (порядка удвоенного размера матрицы) последовательным умножением разреженной матрицы и/или ее транспонированной системы на вектор. Поэтому с точки зрения алгоритмической сложности оба подхода рассматривают как эквивалентные. Однако, при решении задач рекордно больших размеров решающую роль играет возможность эффективной реализации алгоритмов на мощных вычислительных системах с разделенной памятью.

К настоящему времени, по-видимому, сложилось мнение, что методы типа Ланцоша-Монтгомери в целом уступают по параллельной производительности методам Видемана-Копперсмита. При факторизации всех рекордно больших RSA-чисел в качестве алгоритма решения линейных систем над полем $\mathbb{F}_2$, полученных методом решета обобщенного числового поля, использовались различные модификации именно алгоритма Видемана-Копперсмита [7,8].

Привлекательность алгоритмам типа Видемана-Копперсмита дает следующая простая идея построения параллельных вычислений. Вместо последовательного умножения матрицы на один вектор рассматривается умножение на блок, состоящий из K векторов. Поскольку умножение на каждый вектор в блоке можно делать независимо, то тем самым алгоритм приобретает значительный параллельный ресурс.

Безусловно аналогичную идею блочного алгоритма можно применить и к алгоритмам типа Ланцоша-Монтгомери (см., например, [14]). Однако, в этом случае на каждой итерации метода происходит глобальный обмен данными. В реализации метода Монтгомери, предложенной в [14], время на такой обмен не масштабировалось с ростом размера блока, что приводило к заметному замедлению расчетов. Именно проблема немасштабируемого (не идеально масштабируемого) обмена данными, как нам представляется, является главным фактором, сказывающимся на меньшей популярности метода Ланцоша-Монтгомери при решении больших разреженных систем линейных уравнений над конечными полями.

В настоящей работе предлагается реализация блочного метода Ланцоша-Монтгомери, в которой время, необходимое на глобальные обмены, масштабируется с ростом размера

*Работа выполнена при финансовой поддержке Минобрнауки России, соглашение от 17 июня 2014 г. №14.604 .21.0034 (идентификатор соглашения RFMEFI60414X0034).

блока K. Новая реализация является усовершенствованием предложенной в [14]. Полученные в работе теоретические оценки параллельной сложности показывают, что для матриц рекордно больших размеров и при размере блока $K > 50$ временные затраты, требуемые на обмен данными в предлагаемой реализации, становятся меньше, чем временные затраты на идеально параллелизуемые вычисления с плотными блоками.

Интересно заметить, что в существующей практике анализа сложности (параллельной сложности) методов решения больших разреженных систем линейных уравнений над конечными полями вычисления с плотными блоками, как правило, даже не рассматриваются. Предполагается, что наиболее тяжелая вычислительная часть алгоритма, состящая в построении базиса пространства Крылова полной размерности, связана именно с умножением векторов на большую разреженную матрицу системы линейных уравнений. Однако, как мы увидим далее, такое предположение, верно только при использовании блоков умеренного размера K (можно показать, что в этом случае блок не превосходит порядка 10). При необходимости решать задачи быстрее возникает потребность в увеличении размера блока хотя бы до $K = 100$, при этом принебрегать вычислениями с плотными блоками уже нельзя. Время на вычисления оказывается даже больше, времени, необходимого на обмены. Таким образом, оказывается, что более сложные алгоритмы типа Видемана-Копперсмита на практике не имеют существенных преимуществ перед методами типа Ланцоша-Монтгомери, по крайней мере для систем, состоящих из нескольких десятков тысяч распределенных узлов (число вычислительных ядер при этом может достигать миллиона).

Работа устроена следующим образом. В разделе 2 описываются организация вычислений и способ хранения данных в усовершенствованной реализации алгоритма Ланцоша-Монтгомери. В разделе 3 приводятся теоретические оценки на параллельную сложность вычислений в предлагаемой реализации. Все оценки получены в предположении, что узлы вычислительной системы соединены по технологии «каждый с каждым». Такая структура соединений узлов в настоящий момент является весьма распространненой и реализуется с помощью топологии «толстое дерево». В разделе 4 мы приводим примеры расчетов, полученных с помощью усовершенствованной реализации метода Ланцоша-Монтгомери.

2. Описание усовершенствованного метода Ланцоша-Монтгомери

2.1. Алгоритм Ланцоша для систем над конечными полями

Как и в случае стандартных для вычислительной математики полей $\mathbb{R}$ и $\mathbb{C}$, алгоритм Ланцоша для решения систем линейных уравнений $AX = B$ над конечными полями заключается в следующих шагах:

1. переход к симметризованной системе вида

$$A^T AX = A^T B,$$

2. вычисление $A^T A$-ортогонального базиса V_0, V_1, $\cdots$, V_N пространства Крылова, построенного на векторах B, $A^T AB$, $\left(A^T A\right)^2 B$, $\cdots$, $(A^T A)^{\tilde{N}} B$, где $\tilde{N}$ – целое число, как правило близкое к числу N столбцов матрицы A. На $k+1$ итерации новый вектор V_{k+1} получается из коротких рекуррентных соотношений

$$V_{k+1} = A^T AV_k + \sum_{i=k}^{k-m} V_k C_{k+1,i}, \tag{1}$$

с длиной рекурсии m, где $m = 1$ для больших конечных полей, и $m \geq 2$ для $\mathbb{F}_2$).

3. итерационное уточнение решения

$$X_{k+1} = X_k + V_{k+1}G_{k+1}, \tag{2}$$

с некоторой $K \times K$ матрицей G_{k+1}.

Однако, в реализации алгоритма Ланцоша над конечными полями присутствует ряд особенностей. Первая и самая важная особенность состоит в том, что при решении систем уравнений в конечных полях отстсутствует понятие приближенного решения. В результате с вероятностью 1 число итераций, необходимых для получения решения таково, чтобы размер A^TA-ортогонального базиса пространства Крылова оказался практически равным размеру симметризованной матрицы A^TA. Кроме того, если число элементов конечного поля $\mathbb{F}$ невелико, как, например, в случае поля $\mathbb{F}_2$, то во избежание «обрывов» используются блочным методы (см. [3,5,6,14]). Одной из самых эффективных реализаций в случае поля $\mathbb{F}_2$ считается реализация Монтгомери [5], в которой размер блока равен числу бит машинного слова. Наконец в случае малых полей, для удовлетворения условия A^TA-ортогональности при расчете нового блока векторов направлений приходится увеличивать длину рекурсии до $m \geq 2$.

Так или иначе, общая структура вычислений в блочных (с размером блока K) методах типа Ланцоша-Монтгомери остается неизменной и может быть описана следующим образом:

Структура методов типа Ланцоша-Монтгомери

1. **Начало итерации:** перед началом итерации с номером i известны $N \times K$ блоки V_i, $V_{i-1}, \cdots, V_{i-m}$ с элементами в конечном поле $\mathbb{F}$;

2. **Матрица на блок:** вычисляется произведение $(A^TA)V_i$ симметризованной матрицы A^TA на блок V_i;

3. **Билинейные формы от пары блоков:** вычисляются билинейные формы вида $X^TY \in \mathbb{F}^{K\times K}$ с блоками $X, Y \in \mathbb{F}^{N\times K}$;

4. **Линейные комбинации блоков:** вычисляются линейные комбинация вида $XU+YV$ с блоками $X \in \mathbb{F}^{N\times K}$ и кквадратными $K \times K$ матрицами U, V.

2.2. Организация параллельных вычислений в усовершенствованном методе Ланцоша-Монтгомери

Структура метода Ланцоша из предыдущего раздела, являясь по сути последовательной, определяет набор базовых операций алгоритма (см. [13, 14]):

1. умножение разреженной матрицы на блок;

2. вычисление X^TY для $N \times K$ блоков X и Y;

3. вычисление XU для $N \times K$ блока X и $K \times K$ матрицы U.

Эффективная параллельная реализация этих операций является нашей целью при построении *усовершенствованного метода Ланцоша-Монтгомери.*

В основе предлагаемого нами усовершенствованного метода Ланцоша-Монтгомери лежат оригинальные идеи хранения данных на распределенных узлах вычислительной системы и организации вычислений с этими данными. Новый способ представления данных весьма близок к рассмотренному в [14], но имеет и некотороые отличия. Эти изменения позволяют получить программную реализацию алгоритма с уменьшенным числом обменов. Более того, как будет видно из анализа и из вычислительных экспериментов, с увеличением размера блока время, требуемое на обмены, становится несущественным по сравнению со

временем вычислений в блочных операциях. И это несмотря на то, что арифметические вычисления идеально распараллеливаются.

Перейдем к описанию метода. Итак, пусть заданы два натуральных числа K и S (здесь и далее используем K для обозначения размера блока). В качестве вычислительной системы будем рассматривать систему с распределенной памятью, состоящей из $K \times S$ узлов. Удобно считать, что узлы $\mathcal{N}_{i,j}$ образуют прямоугольную решетку. По одной стороне решетки располагается K узлов, а по другой S. Несмотря на регулярное расположение считаем, что узлы соединены по типу «каждый с каждым».

Чтобы полностью описать способ представления данных, нам необходимо рассмотреть:

1. хранение разреженной матрицы A системы линейных уравнений;
2. хранение $N \times K$ блоков;
3. хранение $K \times K$ плотных матриц.

Начнем с большой разреженной матрицы системы A. Несмотря на то, что в алгоритме рассматривается симметризованная система, ее матрица A^TA не вычисляется явно. Для вычисления произведения $(A^TA)V$ блок V последовательно умножается на A и на A^T.

Для хранения матрицы A используется специальная схема. Будем рассматривать матрицу $A \in \mathbb{F}^{M \times N}$ в виде объединения блоков строк A^i и блоков столбцов A_j

$$A = \begin{bmatrix} A_1 & A_2 & \cdots & A_S \end{bmatrix} = \begin{bmatrix} A^1 \\ A^2 \\ \cdots \\ A^S \end{bmatrix},$$

причем предполагается, что в каждом блоке строк (столбцов) одинаковое количество строк (столбцов) и приблизительно равное количество ненулевых элементов A (этого можно добиться с помощью специальной процедуры предобработки матрицы). В усовершенствованной схеме на узел $\mathcal{N}_{i,j}$ со вторым индексом равным j хранятся и блок строк A_j, и блок столбцов A^j.

Помимо матрицы необходимо распределить в памяти вычислительной системы блоки размера $N \times K$. Хранение таких блоков осуществляется следующим образом: блок V рассматривается как объединение равных по размеру меньших блоков числом KS

$$V = \begin{bmatrix} V_1 \\ V_2 \\ \cdots \\ V_{KS} \end{bmatrix},$$

причем подблок V_l размера $\frac{N}{KS} \times K$ располагается на узле $\mathcal{N}_{i,j}$, для которого $l = K * j + i$.

Что касается $K \times K$ матриц, то в дальнейшем считается, что экземпляр любой такой матрицы хранится на каждом из KS узлов $\mathcal{N}_{i,j}$.

Описанная структура хранения в значительной степени определяет параллельные реализации для различных операций методов типа Ланцоша-Монтгомери.

Замечание 1. *Выше рассматривались лишь самые общие принципы хранения данных. В программной реализации имеется довольно много важных деталей, о которых сказано*

не было. Например, для получения более эффективного кода часть строк и/или столбцов разреженной матрицы A следует рассматривать как плотные, не используя специальные разреженные форматы хранения данных. В случае большого поля плотные строки (столбцы) могут быть двух типов: (а) с элеметами «малыми по модулю»; (б) с элементами «типичными» для большого поля. Разреженная матрица хранится в программе в специальном кэш-независимом формате и т.д. Однако, для целей анализа параллельной сложности усовершенствованного метода Ланцоша-Монтгомери вполне достаточно представленного выше огрубленного описания способа хранения данных.

Обратимся к алгоритмам. Рассмотрим процедуру умножения симметризованной матрицы системы на вектор. Она имеет следующий вид:
Алгоритм умножения блока векторов X на симметризованную матрицу A^TA.

1. *на каждом из S узлов $\mathcal{N}_{i,j}$ с фиксированным номером i собрать N вектор X_i, который является i-ым столбцом в блоке X;*
2. *на узле $\mathcal{N}_{i,j}$ вычислить произведение $Y_{i,j} = A^jX_i$;*
3. *на каждом из S узлов $\mathcal{N}_{i,j}$ с фиксированным номером i собрать N вектор Y_i, который является i-ым столбцом вектора $Y = AX$;*
4. *на узле $\mathcal{N}_{i,j}$ вычислить произведение $W_{i,j} = A_jY_i$; части W_{ji} являются требуемым результатом.*
5. *собрать $\frac{N}{KS} \times K$ блоки W_t, $t = 1, \cdots, KS$*

Обмен данными в **Алгоритме умножения блока векторов на симметризованную матрицу** A^TA происходит на шагах 1., 3. и 5. Сборка вектора осуществляется с помощью вызова процедуры коллективных обменов **MPI_Allgather**. Размер данных, которыми обмениваются узлы не превосходит $2\max(N, M)$.

Несложно вычислить затраты на данный алгоритм. Это будет сделано в следующем разделе как для линейных систем, над большим конечным полем, так и для систем над полем $\mathbb{F}_2$.

В случае вычисления билинейных форм алгоритм выглядит следующим образом:
Алгоритм для вычисления билинейных форм вида X^TY.

1. *на каждом из KS узлов $\mathcal{N}_{i,j}$ вычисляется произведение $X_l^TY_l^T$, где $l = Kj + i$;*
2. *собрать результат на каждом из KS узлов $\mathcal{N}_{i,j}$.*

Обмен данными в алгоритме происходит на шаге 2. Сборка результата осуществляется вызовом процедуры глобальных обменов **MPI_Allreduce.**

Наконец вычисление произведения $N \times K$ блока $K \times K$ матрицу:
Алгоритм для произведения XU.

1. *на каждом из KS узлов $\mathcal{N}_{i,j}$ вычисляется произведение $W_l = X_lU$, где $l = Kj + i$; полученные подблоки W_l являются результатом вычисления.*

Обмен данными в данном алгоритме полностью отсуствует.

Замечание 2. *Предполагается, что реализация операций коллективных обменов типа* **MPI_Allgather**, **MPI_Allreduce** *и др. являются эффективными, если узлы распределенной системы соединены друг с другом по типу «каждый с каждым».*

3. Анализ параллельной сложности усовершенствованного метода Ланцоша-Монтгомери

3.1. Оценка сложности метода Ланцоша для систем над большими конечными полями

Оценим сложность параллельных вычислений для метода Ланцоша, применяемого при решении разреженных систем линейных уравнений над большими простыми конечными полями. Будем обозначать через W число машинных слов, необходимых для представления элементов поля, а через p – среденее число ненулей в столбце матрицы A. Так, например, если для задания элементов поля достаточно 512 бит, то $W = 8$. Параметр p может варьироваться в широких пределах, но как правило, его можно оценить как 100. Наши оценки относятся к трем основным операциям, описанным в разделе 2.2.

Параллельная сложность метода Ланцоша.

1. **Сложность умножения разреженной матрицы на блоки векторов:**

$$\{\textbf{сложность выч.}\} = 2W\frac{pN^2}{KS}, \tag{3}$$

$$\{\textbf{сложность обм.}\} = 2W\frac{N^2}{K}. \tag{4}$$

Обратим внимание, что сложность вычислений, которые включают также умножения чисел из большого поля, зависит только от первой степени W, а не от W^2, как этого следовало бы ожидать. Оценка (3) справедлива для приложений, в которых исходная матрица системы A получена методом линейного решета (см., например, [10]). В этом случае практически все элементы A – небольшие «по модулю» числа (более того 90% элементов матрицы это 1 и -1 в поле $\mathbb{F}$). В результате, сложность умножения элемента матрицы на элемент вектора пропорциональна W. Второе важное наблюдение заключается в том, что время на глобальные обмены данными обратно пропорционально размеру блока. Это очень важно, поскольку K – основной ресурс параллельности в алгоритме. Отметим еще, что большое произведение $2WpN^2$ в (3) делится на KS. Собственно говоря, именно эта часть алгоритма, как правило, требовала наибольшего времени. Однако, как будет ясно из дальнейшего, с ростом K ситуация принципиально меняется.

2. **Сложность вычисления билинейных форм:**

$$\{\textbf{сложность выч.}\} = 3\frac{W^2N^2}{S}, \tag{5}$$

$$\{\textbf{сложность обм.}\} = WKN\left(\log_2\left(KS\right)+1\right). \tag{6}$$

Сложность вычисления билинейных форм (5) получена нами в предположении (а) использования алгоритма Монтгомери для умножения чисел в большом простом поле и (б) использования метода Винограда при умножении плотных матриц. Алгоритм Винограда позволяет вдвое сократить число умножений элементов из большого поля. Это важно, поскольку именно умножения элементов определяют сложность алгоритмов базовой линейной алгебры в случае больших конечных полей. Обратим внимание на то, что (5) не зависит от K. Поэтому при фиксированном значении параметра S, начиная с некоторого K, время на вычисления с блоками превзойдет время на работу с разреженной матрицей. В отношении обменов заметим следующее: несмотря на то, что выражение (6) растет с ростом размера блока K, на практике объем пересылок этого типа незначителен. Действительно, (6) линейно по N. Для систем, где N имеет порядок нескольких миллионов, а K порядка 100 произведение $KN \ll \frac{N^2}{K}$, и значит сложность обменов (4) значительно превосходит (6).

3. **Сложность вычисления линейных комбинаций блоков:**

$$\{\textbf{сложность выч.}\} = \frac{9W^2N^2}{2S}, \quad (7)$$

$$\{\textbf{сложность обм.}\} = 0. \quad (8)$$

Как и в (5), сложность вычислений в (7) получена нами в предположении использования метода Винограда. Величина в (7) не масштабируется с ростом K. Обмены в этой части алгоритма отсутствуют.

Замечание 3. *Как следует из приведенных оценок, вычисления с блоками не масштабируются, поэтому задачи, связанные с эффективной реализацией операций базовой линейной алгебры в конечных полях, являются критическими для получения качественной параллельной программной реализации методов типа Ланцоша. Чем эффективнее эти вычисления, тем больше можно выбирать размер блока.*

3.2. Оценка сложности метода Монтгомери для систем над полем $\mathbb{F}_2$

Выпишем оценки сложности параллельных вычислений для метода Монтгомери, применяемого при решении систем линейных уравнений над конечным полем $\mathbb{F}_2$, состоящем из двух элементов. С точностью до коэффициентов, получаемые здесь выражения совпадают с аналогичными оценками из подраздела 3.1

Параллельная сложность метода Монтгомери.

1. **Сложность умножения матрицы на блок:**

$$\{\textbf{сложность выч.}\} = \frac{1}{32} \cdot \frac{pN^2}{KS}, \quad (9)$$

$$\{\textbf{сложность обм.}\} = \frac{N^2}{32K}. \quad (10)$$

Множитель $\frac{1}{32}$ возникает как следствие того, что метод Монтгомери работает с блоками, размер которых кратен размеру машинных слов (мы предполагаем, что размер машинного слова равен 64). Реальный размер блока задается как $K \cdot 64$. Как нетрудно видеть, оба слагаемых масштабируются с K.

2. **Сложность вычисления биллинейных форм:**

$$\{\textbf{сложность выч.}\} = 3\frac{N^2}{8S}, \quad (11)$$

$$\{\textbf{сложность обм.}\} = KN\left(\log_2(KS) + 1\right). \quad (12)$$

Сложность вычислений (11) получена нами в предположении использования метода Копперсмита. Данный алгоритм позволяет в 8 раз сократить число операций с машинными словами по сравнению с «наивным» алгоритмом.

3. **Сложность вычисления линейных комбинаций блоков:**

$$\{\textbf{сложность выч.}\} = \frac{5N^2}{8S}, \quad (13)$$

$$\{\textbf{сложность обм.}\} = 0. \quad (14)$$

Сложность вычислений (13) получена нами в предположении использования метода 4-х русских для умножения плотных матриц в конечном поле $\mathbb{F}_2$. Данный алгоритм позволяет в 8 раз сократить число операций с машинными словами по сравнению с «наивным» алгоритмом.

3.3. Анализ параллельной сложности методов типа Ланцоша-Монтгомери

В этом разделе мы воспользуемся оценками из подразделов 3.1 и 3.2, чтобы показать, что основным препятствием в использовании все более мощных вычислителей в методах типа Ланцоша-Монтгомери являются не обмены, а вычисления с плотными матрицами и блоками.

Начнем со следующего наблюдения. Выясним, при каких значениях параметров K и S сложность вычисления умножения разреженной матрицы на блоки векторов и сложность вычисления билинейных форм совпадают. Это легко сделать, приравняв, например, выражения (3) и (5). После сокращений получим, что в этом случае справедливо соотношение

$$\frac{p}{K} = 3W. \tag{15}$$

Положив $p = 200$, а $W = 8$, получим, что $K \leq 9$. Другими словами, уже при K порядка 10 слагаемое (5) будет иметь параллельную сложность не ниже, чем (3), что говорит об ограниченности K как параллельного ресурса.

Замечание 4. *На самом деле ситуация несколько лучше. Дело в том, что разреженная матрица имеет случайную структуру. В связи с этим скорость операций при работе с разреженной матрицей ограничивается, например, менее эффективным доступом к памяти. Поэтому на практике более корректная оценка на K, при которой времена для (3) и (5) совпадают, находится в районе $K = 100$. Кроме того, можно значительно ускорить операции базовой линейной алгебры (BLAS), используя специализированные ускорители. В случае базовой алгебры над большими полями в качестве таких ускорителей могут рассматриваться стандартные графические ускорители. Получаемое в BLAS ускорение позволяет увеличивать размер блока. Наконец, при дальнейшем увеличении блока K значительное ускорение BLAS можно получить благодаря использованию быстрых алгоритмов. В этом случае оценка (5) должна быть уточнена.*

Сравним теперь время на обмены и время на вычисления с плотными блоками. Для этого нам необходимо сделать предположения о вычислительной системе. Будем считать, что имеется следующая распределенная вычислительная система:

1. коммуникационная сеть вычислительной системы эффективно соединяет узлы по типу «каждый с каждым» (например, с помощью технологии «толстое дерево»)

2. скорость обмена данными (в одну сторону) оценивается 20 Gb/c, что приблизительно равно $300 \cdot 10^8 \frac{W}{c}$.

3. будем считать, что вычислительная система оборудована 8-и ядерными узлами, работающими на частоте $3 \cdot 10^8$ ГГц. Таким образом, каждый узел сбособен выполнять $2.4 \cdot 10^{10} \frac{W}{c}$ операций с машинными словами в секунду.

Наиболее важным следствием из приведенного описания является тот факт, что по сути отдельный многоядерный узел делает вычисления в 100 раз быстрее, чем передает данные.

Приравняв (4) и (5), выясним, при каких параметрах K и S сравнимы времена на обмен данными и вычисления с плотными блоками

$$\frac{K}{S} = 100\frac{2}{3W}.$$

Учитывая, что, как правило, $S \geq 10$, получаем $K \geq 100$. Таким образом, обмен данными в усовершенствованном алгоритме налагает на K менее строгие ограничения, чем вычисления с плотными блоками.

Повторными вычислениями можно показать, что в случае поля $\mathbb{F}_2$ справедливы те же самые выводы, которые мы получили в случае большого поля. Заметим, что эти выводы не

зависят от размера матриц. Однако, они зависят от среднего числа ненулей в строке p. Чем больше p, тем большие значения K допустимы, тем выше параллельная эффективность усовершенствованного метода.

4. Численные эксперименты

Приведем таблицы с абсолютными временами, полученными в численных экспериментах при решении линейных систем над конечными полями с помощью разработанных усовершенствованных программных реализаций методов типа Ланцоша-Монтгомери. В таблицах приводятся число ядер, на которых решалась система. Вычисления производились на суперкомпьютере Ломоносов. Использовавшиеся матрицы эквивалентны матрицам, полученным с помощью методов решета. В случае системы линейных уравнений над большим полем элементы поля кодировались с помощью 512 бит. Обратим внимание на то, что в экспериментах имеются запуски с довольно большим размером блока $K = 32$. Как следует из таблиц, получаемое ускорение практически линейно зависит от числа ядер вплоть до 2.000 ядер. Затем наблюдается зависимость близкая к корню квадратному из числа ядер.

Таблица 1. Ускорение и время на вычисление одного вектора пространства Крылова над большим конечным полем, блочный метод Ланцоша (матрица $2 \cdot 10^6$, с 84 ненулевыми элементами в столбце)

Число ядер	1	2	4	8	16	32	64	128
Ускорение	1	1.99	3.95	7.86	15.48	30.49	56.04	103.04
Время, с	38.1	19.2	9.6	4.85	2.46	1.25	0.67	0.37
$S \times K$	1×1	1×1	1×1	1×1	2×1	2×2	4×2	8×4

Таблица 2. Ускорение и время на вычисление одного вектора пространства Крылова над большим конечным полем, блочный метод Ланцоша (матрица $2 \cdot 10^6$, с 84 ненулевыми элементами в столбце)

Число ядер	256	512	1024	2048	4096	8192
Ускорение	159.48	231.01	391.34	608.96	847.13	1211,4
Время, с	0.239	0.165	0.0974	0.0625	0.0451	0.0316
$S \times K$	16×4	16×8	32×8	32×16	64×16	64×32

Таблица 3. Ускорение и время на вычисление одного вектора пространства Крылова над полем $\mathbb{F}_2$, блочный метод Ланцоша (матрица $2 * 10^6$, 300 ненулевых элементов в строке)

Число ядер	1	2	4	8	16	32	64	128	256	512	1024
Ускорение	1	1.88	3.6	6.8	12.4	23.8	43.9	82.4	153.54	268.2	342.0
Время, мс	57.1	30.4	15.9	8.39	4.59	2.40	1.30	0.693	0.372	0.213	0.167
$S \times K$	1×1	1×1	1×1	1×1	2×1	4×1	4×2	8×2	8×4	16×4	16×8

Таблица 4. Ускорение и время на вычисления одного вектора пространства Крылова над полем $\mathbb{F}_2$, блочный метод Ланцоша (матрица 10^7, 800 ненулевых элементов в строке)

Число ядер	1024	2048	4096	8192
Ускорение	342.0	605.1	813.7	1140.2
Время, мс	1.38	0.780	0.579	0.414
$S \times K$	16×8	32×8	64×16	64×32

Литература

1. Wiedemann D.H. Solving sparse linear equations over finite fields // IEEE Trans. Inform. Theory 1986 Vol. 32, N. 1.
2. Lanczos C. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators // J. Res. Nat. Bur. Standards. 1950. Vol. 45, P. 255–282.
3. Coppersmith D. Solving linear equations over $GF(2)$: Block Lanczos algorithm. // Linear Algebra Appl. 1993. Vol. 193. P. 33–60.
4. Coppersmith D. Solving homogeneous linear equations over $GF(2)$ via block Wiedemann algorithm // Mathematics of Computation 1994. Vol. 62, N. 205.
5. Montgomery P. A block Lanczos algorithm for finding dependencies over GF(2) // Springer-Verlag, 1995. Vol. 921.
6. Peterson M., Monico C. $\mathbb{F}_2$ Lanczos revisited. // Linear Algebra Appl. 2008. Vol. 428. P. 1135–1150.
7. Kleinjung T. et al. Factorization of a 768-bit RSA modulus // Lecture Notes in Computer Science, Springer-Verlag, 2010. V. 6233. P. 333–350.
8. Thome E. et al. Factorization of RSA-704 with CADO-NFS // Preprint, 2012 P. 1–4.
9. Дорофеев А.Я. Вычисление логарифмов в конечном простом поле методом линейного решета. // Труды по дискретной математике. 2002. Т. 5. Р. 29–50.

10. Дорофеев А.Я. Решение систем линейных уравнений при вычислении логарифмов в конечном простом поле. // Математические вопросы криптографии. 2012. Vol. 3. N. 1. P. 5–51.

11. Черепнев М.А. Блочный алгоритм типа Ланцоша решений разреженных систем линейных уравнений. // Дискретная математика. 2008. Vol. 20. N. 1.

12. М.А. Черепнев. Version of block Lanczos-type algorithm for solving sparse linear systems. *Bull.Math.Soc.Sci.Math.Roumanie*, V.53(101), N.3, 2010, p.225-230, http://rms.unibuc.ro/bulletin.

13. И.А. Поповян Ю.В. Нестеренко, Е.А. Гречников. Вычислительно сложные задачи теории чисел. Учебное пособие. Издательство Московского Университета, 2012.

14. Н.Л. Замарашкин. Алгоритмы для разреженных систем линейных уравнений в GF(2). Учебное пособие. Издательство Московского Университета, 2013.