УДК 519.632.4

Л. Б. Маркеева[1], *И. В. Оселедец*[1,2]

[1]Московский физико-технический институт (национальный исследовательский университет)
[2]Сколковский институт науки и техники

Эффективный алгоритм поиска численных решений уравнений эллиптического типа в QTT-формате с использованием z-kron

Данная работа описывает способ решения дифференциального уравнения эллиптического типа с использованием тензорного разложение Quantized Tensor Train (QTT) в качестве структуры для хранения данных. QTT позволяет хранить разреженные матрицы в компактном представлении в памяти ЭВМ. Преимуществом данного тензорного разложения является наличие эффективных реализаций базовых математических операций, таких как сложение и умножение на число, вектор или матрицу и т.д. Также существуют готовые итерационные методы решения СЛАУ и их программные реализации, сохраняющие одновременно матрицу системы и решение в QTT-формате, такие как AMEN и TT-GMRES. Оба этих решателя используются в данной работе. Для предотвращения экспоненциального роста рангов в QTT представлении использовалась z-перестановка строк и столбцов матрицы.

В качестве метода поиска численного решения взят итерационный метод Дирихле. Данный метод позволяет искать численное решение уравнения на отдельных подобластях задачи параллельно, после чего происходит согласование решения на границах подобластей.

В результате выполнения данной работы был предложен итерационный алгоритм решения дифференциальных уравнений эллиптического типа. Следствием этого алгоритма является реализованный код решателя. Данный алгоритм демонстрирует ограниченное сверху константой число итераций во время поиска решения. Алгоритмическая сложность данного решателя $O\left(nr^2\right)$, где r – это ранг в QTT представлении, n – число узлов сетки.

Ключевые слова: tensor train, z-kron, GMRES.

L. B. Markeeva[1], *I. V. Oseledets*[1,2]

[1]Moscow Institute of Physics and Technology
[2]Skolkovo Institute of Science and Technology

Efficient algorithm for finding the numerical solution of elliptic differential equations using QTT format and z-kron

This paper presents the approach of finding a numerical solution of the elliptical differential equation that uses quantisedtensor train decomposition (QTT) as the main data structure. The QTT-format allows storing multidimensional arrays in a low amount of memory. This format has efficient implementationsof basic math operations like summation, multiplication by a scalar and a vector, etc., which is a big advantage.Additionally,iterative solvers thatstore the coefficientsmatrix and the solution in the QTT format are implemented. For example AMEN solver and TT-GMRESsolver. Both of them are used in the present work. To avoid exponential rank growth in QTT-representation,the z-permutation of rows and columns is used.

The iterative Dirichlet method is used for finding a solution. This method iteratively searches for the solution on each subdomain separately, then it refines a solution on borders

between subdomains. This approach allows calculating the solution on each subdomain in parallel on many computation units.

As a result the iterative algorithm for finding the solution of the elliptical differential equation is given. This algorithm results in the implementation of the necessary approach. The number of iterations of the algorithm for searching the solution is restricted by constant. The algorithmic complexity of the presented approach is $O\left(nr^2\right)$, where r is a rank of QTT-representation n — the number of nodes in a discretization grid.

Key words: tensor train, z-kron, GMRES.

1. Введение

Цель данной работы – разработать численный алгоритм итеративного поиска решения двумерного дифференциального уравнения эллиптического типа с нулевыми условиями на границе при помощи Метода Конечных Элементов (МКЭ) [1], метода декомпозиции области [2] и формата Quantized Tensor Train (QTT-формат) [16]. Одним из требований к данному алгоритму является возможность распараллелить поиск решения на несколько вычислительных узлов. Это условие не было достигнуто в [4]. Второй целью является достижение фиксированного числа итераций решателя во время поиска решений.

Эллиптическая задача выбрана в качестве модельного примера, т.к. результаты, полученные для уравнений этого типа, легко экстраполируются на параболические дифференциальные уравнения.

В качестве структуры для хранения данных выбран Quantized Tensor Train формат (QTT-формат), который является частным случаем Tensor Train формата (TT-формат). Данные форматы представляют собой методы нелинейной малоранговой аппроксимации многомерных массивов [16]. Отличительной особенностью данного формата является наличие эффективно реализованных основных матричных операций и наличие готовых итерационных методов решения СЛАУ и их программные реализации, сохраняющие одновременно матрицу системы и решение в QTT-формате, таких как AMEN [5] и TT-GMRES [6]. Так же стоит отметить, что TT и QTT-форматы имеют множество примеров успешного применения к задачам численного решения дифференциальных уравнений в частных производных и вычислительной физики [7], [9], [10], [11], [12].

В работе [4] был представлен алгоритм поиска численного решения эллиптического дифференциального уравнения при помощи МКЭ и QTT. Вычислительная сложность этого алгоритма равна $O\left(\log\left(n\right)\right)$, $n = 4^d$. К сожалению, исходная структура матриц коэффициентов такова, что приводит к экспоненциальному росту рангов ядер QTT представления, поэтому для решения этой проблемы в [4] вводится специальная операция z-kron, которая переставляет строки и столбцы матрицы в z-перестановке [13]. Данная перестановка отображает многомерный тензор в одномерный массив с сохранением локальности данных, что позволяет получить ограниченные сверху константой ранги представления. Также алгоритм из [4] требовал подачу в решатель матрицы коэффициентов, которая описывала всю область решения целиком, что приводило к поиску решения СЛАУ на всей области целиком единовременно и затрудняло параллельный расчет на нескольких вычислительных узлах.

Итерационный метод Дирихле [2] позволяет решить вышеперечисленные проблемы из [4]. Он позволяет искать численное решение дифференциального уравнения путём итеративного поиска решений на подобластях с последующим согласованием решения на границах. Данный подход предполагает построение двух предобуславливателей S_D и S_F. Конструирование S_F требует прямого доступа к элементам матриц жёсткости, что затруднительно в случае QTT-формата, т.к. операция получения элемента ресурсозатратна и имеет сложность $O\left(dr^3\right)$, где r – ранг представления [16]. В данной работе мы предлагаем алгоритм построения всех необходимых матриц и предобуславливателей без прямого доступа к конкретным элементам матриц.

Выбранный подход позволяет сократить число неизвестных в решаемой системе с qn^2 до qn, где $q << n$ – количество подобластей, на которые разбита область. Также в данной работе в качестве итеративного решателя предлагается использовать TT-GMRES [6] вместо стандартного AMEN [14], что также снижает алгоритмическую сложность с $O\left(n^2r^2\right)$ до $O\left(nr^2\right)$ на каждой итерации, где r – это ранг QTT представления.

В разделе 2 рассматривается постановка задачи и кратко описывается мотивация использования Метода Конечных Элементов и приводятся необходимые для данной работы понятия. Раздел 3 даёт краткий обзор QTT-формата и операции z-kron и их преимуществ для данной работы. Раздел 4 описывает итерационный метод Дирихле. Раздел 5 описывает метод построения предобуславливателя для итерационного метода Дирихле в QTT-формате. Раздел 6 кратко описывает весь алгоритм вычисления решения итерационным методом Дирихле в формате QTT. Раздел 7 содержит эксперименты. Раздел 8 заключительный, в нем кратко подводятся итоги данной работы и предлагаются направления дальнейших исследований.

2. Постановка и дискретизация задачи

Рассмотрим уравнение Пуассона, заданное на области Ω:

$$\begin{cases} -\Delta u = f \text{ в } \Omega, \\ u|_{\partial\Omega} = 0, \end{cases} \tag{1}$$

где $\Omega \subseteq \mathbb{R}^2$, Ω многоугольник с границей $\partial\Omega$. $\Omega = \bigcup_{m=1}^{q} \Omega_m$ разбит на q четырехугольников.

Мы хотим найти численно значение u для данной системы.

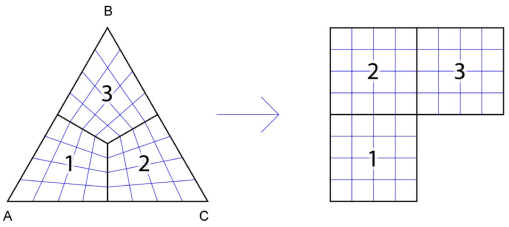

Рис. 1. Способ разбиения исходного домена на подобласти с их последующим отображением на квадраты

Метод Конечных Элементов [1] позволяет свести задачу поиска решения дифференциального уравнения к решению системы линейных алгебраических уравнений (СЛАУ). Делается это путём построения дискретизационной сетки на выбранной области и выписыванием системы линейных уравнений для каждого узла этой сетки. В общем виде система имеет вид

$$Au = f, \tag{2}$$

где A называется матрицей жёсткости и имеет размер $N \times N$, где N – общее число узлов в дискретизационной сетке (в предположении, что сетка построена на четырёхугольнике и на каждую сторону приходится n узлов), f – вектор правой части в узлах дискретизационной сетки или вектор силы, u – вектор неизвестных в узлах сетки.

В случае области со сложной формой, как, например, на рис. 1, исходная область (домен) разбивается на четырёхугольники (подобласти, поддомены), и в каждом из них строится отдельная дискретизационная сетка. Решения на границах согласовываются при по-

мощи специальных матриц-интерфейсов. И система (2) в таком случае примет вид

$$Bu = g, \text{ где } B = \begin{pmatrix} B_{11} & B_{12} & \dots & B_{1q} \\ B_{21} & B_{22} & \dots & B_{2q} \\ \vdots & \vdots & \ddots & \vdots \\ B_{q1} & B_{q2} & \dots & B_{qq} \end{pmatrix}, \; u = \begin{pmatrix} u^{(1)} \\ \vdots \\ u^{(q)} \end{pmatrix}, \; g = \begin{pmatrix} g^{(1)} \\ \vdots \\ g^{(q)} \end{pmatrix}, \tag{3}$$

$$B_{mm} = A^{(m)} - \gamma\Pi_{mm}, \; B_{mp} = \begin{cases} \Pi_{mp}A^{(p)} - \gamma\Pi_{mp}, & \text{если } m \neq p, \text{ и } m \text{ и } p - \\ 0, & \text{соседствующие подобласти,} \end{cases} \tag{4}$$

$$g^{(m)} = f^{(m)} + \sum_{p \neq m} \Pi_{mp} f^{(p)}, \tag{5}$$

где $A^{(p)}$ – матрица жесткости, построенная для подобласти p, $f^{(p)}$ – вектор силы для подобласти p, $u^{(p)}$ – вектор неизвестных для подобласти p. Матрицы Π_{mp} – матрицы интерфейсы, содержат 0, если узел с индексами i, j не находится на границе между доменами m и p, 1 в обратном случае. Подробнее способ построения всех вышеперечисленных матриц разобран в [4].

Достаточно легко заметить, что матрица B из уравнения (3) обладает размерами $qN \times qN$, где q – количество подобластей, на которые разбита область, N – число узлов в четырёхугольной дискретизацинной сетке на каждой подобласти. Соответственно, число неизвестных в данной системе qN или qn^2, где n – число узлов на стороне четырёхугольной подобласти. Чтобы сократить число неизвестных, для которых одновременно производится поиск решения, применяется итерационный метод Дирихле, который описан в разделе 4. Также стоит заметить, что матрицы A и матрицы B_{mm} и B_{mp} разрежены. В [4] было показано, что возможно построить представление этих матриц с ограниченными рангами в формате Quantized Tensor Train, что позволяет тем самым сократить количество используемой памяти. Более того, в [15] было показано, что поиск решения на конечно-элементной схеме, представленной в QTT-формате, сходится с экспоненциальной скоростью, и данный подход является стабильным.

Таким образом, чтобы перейти к построению итерационного метода Дирихле в формате QTT, сначала нужно ознакомиться с самим форматом Quantized Tensor Train.

3. Tensor Train

В данном разделе будут описаны основные идеи TT и QTT форматов. Более детальная информация о свойствах TT-формата представлена в [16].

3.1. Quantized Tensor Train

Tensor Train формат (TT-формат) – это нелинейный метод малоранговой аппроксимации. Рассмотрим d-мерный тензор $T \in \mathbb{R}^{n_1 \times \dots \times n_d}$. Элемент с индексом $i_1 \dots i_d$ в T аппроксимируется в TT представлении следующим образом:

$$T_{i_1 \dots i_d} = \sum_{\alpha_1=1}^{r_1} \cdots \sum_{\alpha_{d-1}}^{r_{d-1}} G_1(i_1, \alpha_1) \, G_2(\alpha_1, i_1, \alpha_2) \dots G_{d-1}(\alpha_{d-2}, i_{d-1}, \alpha_{d-1}) \, G_d(\alpha_{d-1}, i_d), \tag{6}$$

где $0 \leq i_k \leq n_k - 1, k \in [1 \dots d]$. G_t называется *ядром*, $t \in [1 \dots d]$. $G_2, \dots, G_{d-1}$ – трёхмерные тензоры с размерами $r_{j-1} \times n_j \times r_j$. G_1, G_d – матрицы размеров $n_1 \times r_1$ и $r_{d-1} \times n_d$ соответственно. Верхние пределы суммирования $r_1 \dots r_{d-1}$ в (6) называются *рангами аппроксимации* или просто *рангами*.

Каждый элемент тензора в TT-формате хранится в неявном виде. Сложность вычисления элемента составляет $O\left(dr^3\right)$ [16]. Как можно заметить, все операции линейны по d.

Также существуют готовые итерационные методы решения СЛАУ и их программные реализации, сохраняющие одновременно матрицу системы и решение в QTT-формате, такие, как AMEN и TT-GMRES. Алгоритм аппроксимации тензоров при помощи TT-формата подробно разобран в [16].

Количество памяти, занимаемое тензором T оценивается следующей формулой:

$$S = r_1 n_1 + \sum_{i=2}^{d-1} r_{i-1} n_i r_i + r_{d-1} n_d. \tag{7}$$

Перейдём к представлению многомерных матриц в TT-формате. Рассмотрим многомерную матрицу $M \in \mathbb{R}^{(n_1 \times \cdots \times n_d) \times (m_1 \times \cdots \times m_d)}$. Элемент с индексом $i_1 \dots i_d; j_1 \dots j_d$ представляется в TT-формате как

$$M_{\substack{i_1 \dots i_d \\ j_1 \dots j_d}} = \sum_{\alpha_1=1}^{r_1} \cdots \sum_{\alpha_{d-1}=1}^{r_{d-1}} V_1(i_1, j_1, \alpha_1) V_2(\alpha_1, i_2, j_2, \alpha_2) \dots \tag{8}$$

$$\dots V_{d-1}(\alpha_{d-2}, i_{d-1}, j_{d-1}, \alpha_{d-1}) V_d(\alpha_{d-1}, i_d, j_d),$$

где $0 \le i_k \le n_k - 1$, $0 \le j_k \le m_k - 1$, V_j при $j \in [2 \dots d-1]$ – это четырехмерные тензоры с размерностями $r_{j-1} \times n_j \times m_j \times r_j$. V_1 и V_d – трехмерные тензоры с размерностями $n_1 \times m_1 \times r_1$ и $r_d \times n_d \times m_d$ соответственно.

В матричной форме (8) может быть представлено как

$$M_{\substack{i_1 \dots i_d \\ j_1 \dots j_d}} = Q_1(i_1, j_1) Q_2(i_2, j_2) \dots Q_{d-1}(i_{d-1}, j_{d-1}) Q_d(i_d, j_d), \tag{9}$$

где $Q_k(i_k, j_k)$ – матрица размера $r_{k-1} \times r_k$, $k \in [1 \dots d]$, а пара i_k, j_k представляет собой «длинный индекс». Важно отметить, что ядра Q_k могут рассматриваться как блочные матрицы размера $r_{k-1} \times r_k$, в которых каждый блок имеет размерность $n_k \times m_k$.

Оценка сложности большинства операций в TT-формате имеет вид dnr^α [16], где $\alpha \in \{2, 3\}$, $n = \max(n_1, \dots n_d)$, $r = \max(r_1, \dots r_{d-1})$.

QTT-формат является особым случаем TT-формата, когда индексы исходного объекта представляются n-арным кодом. Рассмотрим пример с бинарным кодированием для вектора $v \in R^{2^d}$. Данный вектор может быть представлен в виде тензора V с размерностями $\underbrace{2 \times 2 \times \cdots \times 2}_{\text{d раз}}$. После этого мы можем получить представление V в TT-формате как d-мерного тензора. Превращение вектора v в d-мерный тензор эквивалентно кодированию индекса этого вектора $0 \le i \le 2^d - 1$ в бинарном формате:

$$i = \overline{i_1, i_2, \dots i_d} = \sum_{k=1}^{d} 2^{k-1} i_k \;\leftrightarrow\; (i_1, i_2, \dots i_d), \tag{10}$$

где $i_k \in \{0, 1\}$, $k \in [1 \dots d]$. Аналогичная идея используется для представления матрицы $\mathbf{M} \in R^{2^d \times 2^d}$ в QTT-формате.

3.2. Операция z-kron

Рассмотрим две QTT-матрицы:

$$K_{\substack{i_1, \dots, i_d \\ j_1, \dots, j_d}} = K_1(i_1, j_1) \dots K_d(i_d, j_d), \; L_{\substack{i'_1, \dots, i'_d \\ j'_1, \dots, j'_d}} = L_1\left(i'_1, j'_1\right) \dots L_d\left(i'_d, j'_d\right), \text{ где } i_k \in \{0, 1\}. \tag{11}$$

Кронекерово произведение двух d-мерных QTT-матриц представляет собой $2d$-мерную QTT-матрицу [16], где каждый элемент вычисляется по формуле

$$M_{\substack{i_1,\ldots,i_d,i'_1,\ldots,i'_d \\ j_1,\ldots,j_d,j'_1,\ldots,j'_d}} = K_{\substack{i_1,\ldots,i_d \\ j_1,\ldots,j_d}} L_{\substack{i'_1,\ldots,i'_d \\ j'_1,\ldots,j'_d}} = K_1(i_1,j_1)\ldots K_d(i_d,j_d)\, L_1\left(i'_1,j'_1\right)\ldots L_d\left(i'_d,j'_d\right). \tag{12}$$

Определим операцию z-крон и обозначим ее как $\oslash$. Данная операция применяется к двум d-мерным QTT-матрицам из (11). Результатом операции является QTT-матрица $M \in \mathbb{R}^{4^d \times 4^d}$. Каждый элемент матрицы определяется как

$$M_{\substack{z_1,\ldots,z_d \\ z'_1,\ldots,z'_d}} = K_{\substack{i_1,\ldots,i_d \\ j_1,\ldots,j_d}} L_{\substack{i'_1,\ldots,i'_d \\ j'_1,\ldots,j'_d}}, \text{ где } z_k = i_k + 2j_k,\ k \in [1\ldots d]. \tag{13}$$

В QTT-формате матрицу $M_{\substack{z_1,\ldots,z_d \\ z'_1,\ldots,z'_d}}$, полученную как $M_{\substack{z_1,\ldots,z_d \\ z'_1,\ldots,z'_d}} = K_{\substack{i_1,\ldots,i_d \\ j_1,\ldots,j_d}} \oslash L_{\substack{i'_1,\ldots,i'_d \\ j'_1,\ldots,j'_d}}$, можно формально записать как

$$M_{\substack{z_1,\ldots,z_d \\ z'_1,\ldots,z'_d}} = M_1\left(z_1,z'_1\right) M_2\left(z_2,z'_2\right)\ldots M_d\left(z_d,z'_d\right), \tag{14}$$

В работе [17] показано, что построенные таким образом матрицы отличаются от матриц из (12) тем, что строки и столбцы у них переставлены в z-последовательности [13].

В работе [4] было показано, что применение этой операции для построения матриц жёсткости и вектора силы позволяет получить QTT представление с ограниченными рангами.

4. Итерационный Метод Дирихле

В случае итерационного метода Дирихле задача (1) сводится к следующей системе:

$$S\begin{bmatrix} u \\ \varphi \end{bmatrix} = \begin{bmatrix} f \\ 0 \end{bmatrix}, \tag{15}$$

где $u = \begin{bmatrix} u^{(1)}, \\ \vdots \\ u^{(q)} \end{bmatrix}$ – вектор неизвестных, $f = \begin{bmatrix} f^{(1)}, \\ \vdots \\ f^{(q)} \end{bmatrix}$ – вектор правой части или вектор силы, φ – вектор потока через границы подобластей.

Для наглядности матрица S будет выписана для треугольника, изображённого на рис. 1. Ее конструкция проста и легко расширяется на другое количество областей или иное разбиение.

$$S = \begin{bmatrix} A & \Pi \\ \Pi^T & 0 \end{bmatrix} = \begin{bmatrix} A^{(1)} & 0 & 0 & \Pi_{1\to2} & 0 & -\Pi_{1\to3} \\ 0 & A^{(2)} & 0 & -\Pi_{2\to1} & \Pi_{2\to3} & 0 \\ 0 & 0 & A^{(3)} & 0 & -\Pi_{3\to2} & \Pi_{3\to1} \\ \Pi^T_{1\to2} & -\Pi^T_{2\to1} & 0 & 0 & 0 & 0 \\ 0 & \Pi^T_{2\to3} & -\Pi^T_{3\to2} & 0 & 0 & 0 \\ -\Pi^T_{1\to3} & 0 & \Pi^T_{3\to1} & 0 & 0 & 0 \end{bmatrix}, \tag{16}$$

где $A^{(m)}$ – матрица жёсткости на домене с номером m размера $4^d \times 4^d$, $\Pi_{l\to p}$ – матрицы интерфейсов между подобластями, на которые разделена изначальная область. $\Pi_{l\to p}$ содержит 0, если узел с индексами i, j не находятся на границе, и 1 в обратном случае. Размер $\Pi_{l\to p}$ равен $4^d \times 2^d$. Если области l и p не имеют общей стороны, то $\Pi_{l\to p} = 0$. Подробно конструкция $\Pi_{l\to p}$ рассмотрена в [4].

Систему (15) можно переписать в эквивалентном виде:

$$\begin{cases} Au + \Pi\varphi = f \\ \Pi^T u = 0 \end{cases} \Rightarrow \begin{cases} \Pi^T A^{-1}\Pi\varphi = \Pi^T A^{-1} f \\ \Pi^T u = 0 \end{cases} \Rightarrow \\ \Rightarrow \begin{cases} S_D\varphi = F \\ \Pi^T u = 0 \end{cases} \tag{17}$$

S_D известен как предобуславливатель Дирихле [2].

Решить систему (17) можно следующим образом:

1)

$$\begin{cases} Au + \Pi\varphi = f \\ \Pi^T u = 0 \end{cases} \Rightarrow \\ \begin{cases} y^{(1)} = f^{(1)} - \Pi_{1\to2}\varphi^{(1)} + \Pi_{1\to3}\varphi^{(3)} \\ y^{(2)} = f^{(2)} + \Pi_{2\to1}\varphi^{(1)} - \Pi_{2\to3}\varphi^{(2)} \\ y^{(3)} = f^{(3)} + \Pi_{3\to2}\varphi^{(2)} - \Pi_{3\to1}\varphi^{(3)} \end{cases} \Rightarrow \\ \begin{cases} u^{(1)} = \text{tt.amen}\left(A^{(1)}, y^{(1)}\right) \\ u^{(2)} = \text{tt.amen}\left(A^{(2)}, y^{(2)}\right) \\ u^{(3)} = \text{tt.amen}\left(A^{(3)}, y^{(3)}\right) \end{cases}, \tag{18}$$

где $\text{tt.amen}\left(A^{(1)}, y^{(1)}\right)$ это поиск решения системы вида $A^{(m)}u^{(m)} = y^{(m)}$ при помощи итеративного решателя СЛАУ AMEN [14], предназначенного специально для QTT-формата.

2) Подсчитываем невязку решений на границе:

$$\begin{cases} r^{(1)} = \Pi^T_{1\to2}u^{(1)} - \Pi^T_{2\to1}u^{(2)} \\ r^{(2)} = \Pi^T_{2\to3}u^{(2)} - \Pi^T_{3\to2}u^{(3)} \\ r^{(3)} = -\Pi^T_{1\to3}u^{(1)} + \Pi^T_{3\to1}u^{(3)} \end{cases} \tag{19}$$

3) Применяем решатель, например Якоби/CG/GMRES:

$$\begin{cases} r^{(1)} = \Pi^T_{1\to2}u^{(1)} - \Pi^T_{2\to1}u^{(2)} \\ r^{(2)} = \Pi^T_{2\to3}u^{(2)} - \Pi^T_{3\to2}u^{(3)} \\ r^{(3)} = -\Pi^T_{1\to3}u^{(1)} + \Pi^T_{3\to1}u^{(3)} \end{cases}. \tag{20}$$

В данной работе используется TT-GMRES [6] из библиотеки ttpy [18].

5. Предобуславливатель для итерационного метода Дирихле в формате QTT

Согласно [2], система (17) слабо обусловлена. Поэтому вводится дополнительный предобуславливатель системы S_F:

$$S_F = diag\left\{S_F^{(1)} \dots S_F^{(q)}\right\}, \tag{21}$$

$$S_F^{(m)} = A_{\Gamma\Gamma}^{(m)} - A_{\Gamma i}^{(m)}\left(A_{ii}^{(m)}\right)^{-1} A_{i\Gamma}^{(m)}, \tag{22}$$

где $A_{\Gamma\Gamma}^{(m)}$ – элементы матрицы жёсткости, описывающие взаимоотношение узлов, лежащих на границе подобласти с другим подобластями. $A_{ii}^{(m)}$ – элементы матрицы жёсткости, которые находятся внутри домена и не являются соседними с узлами на границе, $A_{\Gamma i}$ и $A_{i\Gamma}$ – элементы, описывающие взаимодействия узла на границе подобласти с некоторым внутренним узлом и наоборот [2].

Вычисление $A_{\Gamma\Gamma}^{(m)}$ производится по следующей формуле:

$$A_{\Gamma\Gamma}^{(m)} = \sum_{t=1}^{T} \Pi_{m_t}^T A^{(m)} \Pi_{m_t}, \tag{23}$$

где T – число границ с другими подобластями у домена с номером m. Π_{m_t} – интерфейсная матрица, выделяющая элементы границы с номером t.

Вычисление же $A_{\Gamma i}^{(m)} \left(A_{ii}^{(m)}\right)^{-1} A_{i\Gamma}^{(m)}$ затруднительно, т.к. требует доступ к конкретным элементам матрицы $A^{(m)}$, что в QTT имеет сложность $O\left(dr^3\right)$, где r – максимальный из всех рангов ядер [16].

Ниже приведен способ вычисления $A_{\Gamma i}^{(m)} \left(A_{ii}^{(m)}\right)^{-1} A_{i\Gamma}^{(m)}$, который не требует доступа к конкретным элементам.

Сначала вычислим $A_{i\Gamma}^{(m)}$ для каждой границы между доменами t:

$$A_{i_t\Gamma_t}^{(m)} = A^{(m)} \Pi_{m_t}. \tag{24}$$

Далее введём вспомогательные матрицы-проекторы:

$$P_{m_t} = \Pi_{m_t} \Pi_{m_t}^T = \begin{bmatrix} 1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ \vdots & \ddots & \vdots \\ 0 & 1 & 0 \end{bmatrix}, \ \hat{P}_{m_t} = E - P_{m_t}. \tag{25}$$

Матрица $A_t^{(m)}$ состоит из четырёх блоков из четырёх типов элементов, где t – конкретная граница для данной подобласти m:

$$A^{(m)} = \begin{bmatrix} A_{i_t i_t}^{(m)} & A_{i_t\Gamma_t}^{(m)} \\ A_{\Gamma_t i_t}^{(m)} & A_{\Gamma_t\Gamma_t}^{(m)} \end{bmatrix}, \tag{26}$$

Отсюда получаем:

$$\begin{gathered} \hat{A}_t^{(m)} = A^{(m)} \hat{P}_{m_t} + \gamma P_{m_t} = \begin{bmatrix} A_{i_t i_t}^{(m)} & 0 \\ A_{\Gamma_t i_t}^{(m)} & \gamma E \end{bmatrix} \Rightarrow \\ \Rightarrow \left(\hat{A}_t^{(m)}\right)^{-1} = \begin{bmatrix} \left(A_{i_t i_t}^{(m)}\right)^{-1} & 0 \\ * & \dfrac{1}{\gamma} E \end{bmatrix}. \end{gathered} \tag{27}$$

Из (24), (27) следует

$$\begin{gathered} \begin{bmatrix} \left(A_{i_t i_t}^{(m)}\right)^{-1} A_{i_t\Gamma_t}^{(m)} \\ \left(A^{(m)}\right)_{\Gamma_t i_t}^{-1} A_{i_t\Gamma_t}^{(m)} + \dfrac{1}{\gamma} A_{\Gamma_t\Gamma_t}^{(m)} \end{bmatrix} = \left(\hat{A}^{(m)}\right)^{-1} A^{(m)} \Pi_{m_t} \Rightarrow \\ \Rightarrow \left(\hat{A}^{(m)}\right)^{-1} = tt.amen\left(\hat{A}_t^{(m)}, A^{(m)} \Pi_{m_t} \beta\right), \end{gathered} \tag{28}$$

где β – это некоторый произвольный вектор, например $\beta = S_D F$, $tt.amen\left(\hat{A}_t^{(m)}, A^{(m)}\Pi_{m_t}\beta\right)$ – это поиск решения системы вида $\hat{A}_t^{(m)} u^{(m)} = A^{(m)}\Pi_{m_t}\beta$ при помощи итеративного решателя СЛАУ AMEN [14], предназначенного специально для QTT-формата.

Далее, чтобы выделить верхний блок матрицы $\left(\hat{A}^{(m)}\right)^{-1}$, домножим её на несколько вспомогательных матриц и просуммируем для всех границ и получим

$$\begin{aligned} S_F\beta = &\sum_{t=1}^{T} \Pi_{m_t}^T A^{(m)} \Pi_{m_t}\beta - \\ &- \sum_{t=1}^{T} \Pi_{m_t}^T A^{(m)} \hat{P}_{m_t}\, tt.amen\left(\hat{A}_t^{(m)}, A^{(m)}\Pi_{m_t}\beta\right). \end{aligned} \tag{29}$$

Соответственно, наша система примет вид

$$\begin{aligned} S_F S_D\varphi = &\sum_{t=1}^{T} \Pi_{m_t}^T A^{(m)} \Pi_{m_t}\beta - \\ &- \sum_{t=1}^{T} \Pi_{m_t}^T A^{(m)} \hat{P}_{m_t}\, tt.amen\left(\hat{A}_t^{(m)}, A^{(m)}\Pi_{m_t} S_D\varphi\right), \end{aligned} \tag{30}$$

$$\begin{aligned} S_F F = &\sum_{t=1}^{T} \Pi_{m_t}^T A^{(m)} \Pi_{m_t}\beta - \\ &- \sum_{t=1}^{T} \Pi_{m_t}^T A^{(m)} \hat{P}_{m_t}\, tt.amen\left(\hat{A}_t^{(m)}, A^{(m)}\Pi_{m_t} F\right). \end{aligned} \tag{31}$$

Поиск численного решения производится при помощи TT-GMRES с реализованными операциями матрично-векторного произведения для S_D и S_F.

6. Алгоритм поиска решения итерационным методом Дирихле

Алгоритм 1 описывает последовательность действий для поиска решения итерационным методом Дирихле в QTT-формате[1].

Рассмотрим шаги 4, 6, 8, 11 в Алгоритме 1. Каждый из этих шагов может быть вычислен на отдельной области m независимо от остальных подобластей. Более того, все эти шаги подразумевают поиск решения СЛАУ при помощи AMEN решателя, что является дорогостоящей операцией. Так как вычисления на этих шагах не зависят от других областей, их можно осуществлять параллельно. Максимальное число параллельно запущенных процессов равно q – числу подобластей.

7. Эксперименты

В качестве экспериментов задача (1) решалась описанным выше методом на областях различной формы. Число подобластей варьировалось от 2 до 10. Максимальное число итераций TT-GMRES было установлено в 400, на каждом шаге максимальное число итераций 20. Параметры AMEN решателя, который используется в матрично-векторных операциях в данном методе, следующие: максимальное число итераций 20, kick rank равный 4, максимальный размер локальной матрицы решателя составлял 1000. Число узлов на каждой подобласти $n = 4^d$, $d = [2 \dots 13]$. Во время эксперимента замерялось количество итераций TT-GMRES и максимальное число итераций AMEN решателя. Также измерялось

[1]Код доступен на GitHub: https://github.com/RerRayne/qtt-laplace

количество потребляемой программой памяти. Для сравнения по потреблению памяти был взят классический программный пакет для поиска численных решений дифференциальных уравнений FEniCS [19]. Независимо от выбранной конфигурации домена, была получена одинаковая картина, которая представлена на рис. 2.

Алгоритм 1 Итерационный метод Дирихле в QTT-формате

1: Построить матрицу жёсткости $A^{(m)}$ и вектор силы $f^{(m)}$ в z-перестановке для каждой подобласти m в QTT-формате, как это описано в [4].
2: Применить граничные условия к $A^{(m)}$ и $f^{(m)}$ согласно описанию в [4].
3: Построить матрицы-проекторы P_{m_t} согласно формуле (25).
4: Вычислить правую часть F согласно формуле (31).
5: Проинициализировать вектор потока через границы подобластей φ нулями.
6: Построить функцию умножения вектора на предобуславлеватель S_D, $sd_m\_matvec(x)$ для каждой подобласти m, используя формулу (17), а также $A^{(m)}$ и P_{m_t} из шагов 2 и 3.
7: Сохранить функции с предыдущего шага для всех подобластей с массивом sd_matvec.
8: Построить функцию умножения вектора на предобуславлеватель S_F, $sf_m\_matvec(x)$ для каждой подобласти m, используя формулу (30), а также $A^{(m)}$ и P_{m_t} из шагов 2 и 3.
9: Сохранить функции с предыдущего шага для всех подобластей с массивом sf_matvec.
10: $\varphi = \text{TT-GMRES}\,(sd\_matvec, sf\_matvec, F, \varphi)$.
11: Вычислить $u^{(m)}$ для каждой подобласти m, пользуясь формулой (18).
12: Сохранить решения для всех подобластей в массив u.
13: **return** u

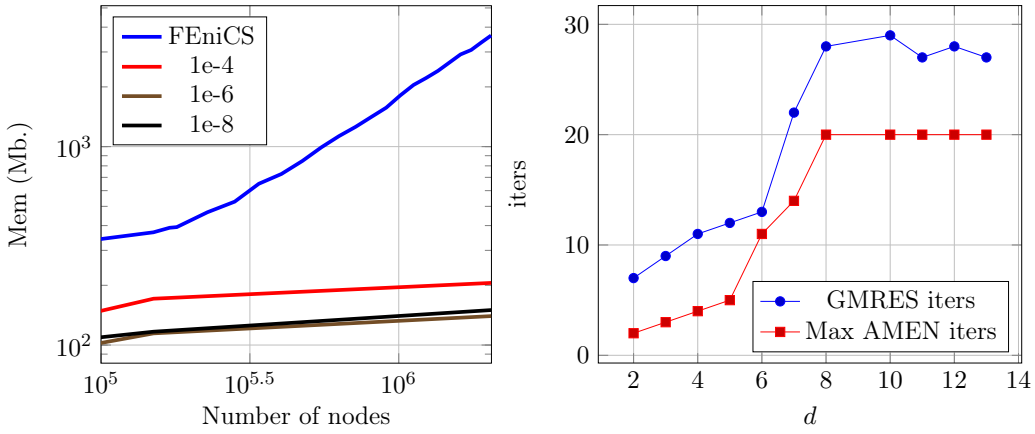

Рис. 2. Слева представлена динамика потребления памяти нашим методом по сравнению с пакетом поиска численных решений дифференциальных уравнений FEniCS. Справа представлена динамика изменения числа итераций GMRES и AMEN при нахождении численного решения уравнения Пуассона итерационным методом Дирихле в зависимости от d

Как можно увидеть слева на рис. 2, предложенный итеративный метод потребляет меньше памяти и демонстрирует линейный рост занимаемой памяти от числа узлов на дискретизационной сетке, в отличие от FEniCS, который демонстрирует экспоненциальный рост.

Справа на рис. 2 можно увидеть, что для $d \geq 8$ количество итераций TT-GMRES находится примерно на одном и том же уровне, AMEN решатель достигает максимального разрешённого ему числа итераций.

К сожалению, при $d > 13$ теряется сходимость решения. Это связано с накапливаемой алгоритмической ошибкой AMEN решателя. Для решения этой проблемы предлагается использовать решатель из [20], который специально разработан для поиска решений СЛАУ на сверх-мелких сетках.

8. Заключение

В данной работе был предложен метод поиска численного решения уравнений эллиптического типа итерационным методом Дирихле в формате QTT. Для предотвращения экспоненциального роста рангов от d была применена z-перестановка строк и столбцов матриц коэффициентов. Эксперименты были проведены на областях различной формы, которые были разбиты на четырёхугольные подобласти. В результате при $13 \geq d \geq 8$ количество итераций TT-GMRES находится примерно на одном и том же уровне, AMEN решатель достигает максимального разрешённого ему числа итераций, равного 20. Предложенный метод демонстрирует гораздо меньшее потребление памяти по сравнению с классическим пакетом для поиска численных решений дифференциальных уравнений FEniCS.

В дальнейших исследованиях предлагается использовать решатель, представленный в [20], вместо AMEN, для поиска решения при $d > 13$.

Литература

1. *Strang G., Fix G.* An analysis of the finite element method. Prentice-hall Englewood Cliffs, NJ, 1973.
2. *Василевский Ю.В., Ольшанский М.А.* Краткий курс по многосеточным методам и методам декомпозиции области. Краткий курс по многосеточным методам и методам декомпозиции области, Москва : МАКС Пресс, 2007.
3. *Oseledets I.* Tensor-train decomposition // SIAM Journal on Scientific Computing. 2011 Sep 22; 33(5):2295–317.
4. *Markeeva L., Tsybulin I., Oseledets I.* QTT-isogeometric solver in two dimensions // arXiv preprint arXiv:1802.02839, 2018
5. *Dolgov S.V., Savostyanov D.V.* Alternating minimal energy methods for linear systems in higher dimensions // SIAM Journal on Scientific Computing. 2014. 36(5):A2248–A2271.
6. *Dolgov S.* TT-GMRES: solution to a linear system in the structured tensor format // Russian Journal of Numerical Analysis and Mathematical Modelling. 2013. 28(2):149–172.
7. *Khoromskij B.N.* Tensor numerical methods in scientific computing. 19, 2018, Walter de Gruyter GmbH & Co KG.
8. *Grasedyck L., Kressner D., Tobler Ch.* A literature survey of low-rank tensor approximation techniques // GAMM-Mitteilungen. 2013. 36(1):53–78.
9. *Matveev S., Zheltkov D., Tyrtyshnikov E, Smirnov A.* Tensor train versus Monte Carlo for the multicomponent Smoluchowski coagulation equation // Journal of Computational Physics. 2016. 316:164–179.
10. *Rakhuba M., Novikov A., Oseledets I.* Low-rank Riemannian eigensolver for high-dimensional Hamiltonians // Journal of Computational Physics. 2019. 396:718–737.
11. *Chertkov A., Oseledets I., Rakhuba M.* Robust discretization in quantized tensor train format for elliptic problems in two dimensions // 2016. arXiv preprint arXiv:1612.01166
12. *Dolgov S., Khoromskij B, Oseledets I* Fast solution of parabolic problems in the tensor train/quantized tensor train format with initial application to the Fokker–Planck equation // SIAM Journal on Scientific Computing. 2012. 34(6):A3016–A3038.
13. *Morton M.* A computer oriented geodetic data base and a new technique in file sequencing. International Business Machines Company New York, 1966.
14. *Oseledets I, Dolgov S.* Solution of linear systems and matrix inversion in the TT-format // SIAM Journal on Scientific Computing. 2012. 34(5):A2718–A2739.
15. *Kazeev V., Schwab Ch.* Quantized tensor-structured finite elements for second-order elliptic PDEs in two dimensions // Numerische Mathematik. 2018. 128(1):133–190.

16. *Oseledets I.* Tensor-train decomposition // SIAM Journal on Scientific Computing. 2011 Sep 22. 33(5):2295–317.

17. *Маркеева Л.Б., Цыбулин И.В.* Построение z-переставленных матриц в QTT-формате // Журнал вычислительной математики и математической физики, специальный выпуск конференции Matrix Methods in Mathematics 2019. 2020.

18. *Oseledets I.V.* // TTPY. 2013. https://github.com/oseledets/ttpy;

19. *Logg A., Mardal K.-A., Wells G.* Automated solution of differential equations by the finite element method: The FEniCS book. Springer Science & Business Media, 2012. 84 p.

20. *Bachmayr M., Kazeev V.* Stability of low-rank tensor representations and structured multilevel preconditioning for elliptic PDEs. 1–62, Foundations of Computational Mathematics. Springer, 2020.

References

1. *Strang G., Fix G.* An analysis of the finite element method. Prentice-hall Englewood Cliffs, NJ, 1973.

2. *Vasylevsky U.V., Olshansky M.A.* A brief course of multigrid and decomposition methods. Moscow : MAKS PRESS, 2007. (in Russian).

3. *Oseledets I.* Tensor-train decomposition. SIAM Journal on Scientific Computing. 2011 Sep 22; 33(5):2295–317.

4. *Markeeva L., Tsybulin I., Oseledets I.* QTT-isogeometric solver in two dimensions. arXiv preprint arXiv:1802.02839, 2018

5. *Dolgov S.V., Savostyanov D.V.* Alternating minimal energy methods for linear systems in higher dimensions. SIAM Journal on Scientific Computing. 2014. 36(5):A2248–A2271.

6. *Dolgov S.* TT-GMRES: solution to a linear system in the structured tensor format. Russian Journal of Numerical Analysis and Mathematical Modelling. 2013. 28(2):149–172.

7. *Khoromskij B.N.* Tensor numerical methods in scientific computing. 19, 2018, Walter de Gruyter GmbH & Co KG.

8. *Grasedyck L., Kressner D., Tobler Ch.* A literature survey of low-rank tensor approximation techniques. GAMM-Mitteilungen. 2013. 36(1):53–78.

9. *Matveev S., Zheltkov D., Tyrtyshnikov E, Smirnov A.* Tensor train versus Monte Carlo for the multicomponent Smoluchowski coagulation equation. Journal of Computational Physics. 2016. 316:164–179.

10. *Rakhuba M., Novikov A., Oseledets I.* Low-rank Riemannian eigensolver for high-dimensional Hamiltonians. Journal of Computational Physics. 2019. 396:718–737.

11. *Chertkov A., Oseledets I., Rakhuba M.* Robust discretization in quantized tensor train format for elliptic problems in two dimensions. 2016. arXiv preprint arXiv:1612.01166

12. *Dolgov S., Khoromskij B, Oseledets I* Fast solution of parabolic problems in the tensor train/quantized tensor train format with initial application to the Fokker–Planck equation. SIAM Journal on Scientific Computing. 2012. 34(6):A3016–A3038.

13. *Morton M.* A computer oriented geodetic data base and a new technique in file sequencing. International Business Machines Company New York, 1966.

14. *Oseledets I, Dolgov S.* Solution of linear systems and matrix inversion in the TT-format. SIAM Journal on Scientific Computing. 2012. 34(5):A2718–A2739.

15. *Kazeev V., Schwab Ch.* Quantized tensor-structured finite elements for second-order elliptic PDEs in two dimensions. Numerische Mathematik. 2018. 128(1):133–190.

16. *Oseledets I.* Tensor-train decomposition. SIAM Journal on Scientific Computing. 2011 Sep 22. 33(5):2295–317.

17. *Markeeva L., Tsybulin I.* Constructing z-permuted matrix in QTT-format. Computational Mathematics and Mathematical Physics special issue for Matrix Methods in Mathematics 2019. 2020. (in Russian).

18. *Oseledets I.V.* TTPY. 2013. https://github.com/oseledets/ttpy;

19. *Logg A., Mardal K.-A., Wells G.* Automated solution of differential equations by the finite element method: The FEniCS book. Springer Science & Business Media, 2012. 84 p.

20. *Bachmayr M., Kazeev V.* Stability of low-rank tensor representations and structured multilevel preconditioning for elliptic PDEs. 1–62, Foundations of Computational Mathematics. Springer, 2020.

Поступила в редакцию 28.04.2020