Научная визуализация, 2022, том 14, номер 3, страницы 73 - 91, DOI: 10.26583/sv.14.3.06

О визуализации функций в многомерном пространстве с помощью канонического разложения

А.К. Алексеев[1,A,B], А.Е. Бондарев[2,A], Ю.С. Пятакова[3,B]

[A] Институт прикладной математики им. М.В. Келдыша Российской академии наук
[B] ПАО РКК Энергия им. С.П. Королева, Королев, Россия

[1] ORCID: 0000-0001-8317-8688, aleksey.k.alekseev@gmail.com
[2] ORCID: 0000-0003-3681-5212, bond@keldysh.ru
[3] ORCID: 0000-0002-8055-7807, yuliya.pyatakova@rsce.ru

Аннотация

Рассмотрена аппроксимация многомерной функции с помощью тензорных разложений с точки зрения хранения, обработки и визуализации результатов параметрических расчетов в задачах вычислительной аэрогазодинамики. Описан алгоритм расчета канонического разложения с помощью комбинации метода переменных наименьших квадратов и стохастического градиентного спуска. Представлены численные результаты для интерполяции функций в шестимерном пространстве, полученные с использованием канонического разложения, демонстрирующие высокую вычислительную эффективность и качество результатов. Показаны визуальные представления результатов.

Ключевые слова: тензорное разложение, каноническое разложение, вычислительная аэрогазодинамика, визуальное представление результатов.

Введение

Работа с многомерными данными, в том числе их визуализация, представляет собой достаточно сложную задачу из-за “проклятия размерности” (экспоненциального роста требуемой памяти при увеличении размерности задачи). В связи с этим мы рассмотрим возможности, предоставляемые тензорной формой многомерных задач и их аппроксимацией с помощью тензорных разложений, для работы с многомерными данными.

В качестве примеров мы рассмотрим функцию $f(x,y,z,u,v,w)$ определенную в области $\Omega \subset R^6$ и соответствующую плотности распределения Больцмана и набор функций $f_i(x,y,z,\vartheta_1 ... \vartheta_q)$, где $i = 1,...,p$ соответствует переменным течения (плотность, компоненты скорости, энергия), а $(\vartheta_1 ... \vartheta_q) \subset \Omega_q \subset R^q$ соответствует параметрам задачи (число Маха, число Рейнольдса, угол атаки и т.д.).

1. Тензорная форма задач аэрогазодинамики

Не вдаваясь в дискуссии о физическом смысле тензоров, в данной работе мы будем воспринимать тензор просто как многомерный массив [1,2]. В интересующем нас случае он соответствует сеточной функции, определенной на регулярной (это существенно) сетке в многомерном пространстве. Под газодинамическими переменными мы в этой работе всегда подразумеваем их дискретную форму. Таким образом, переменные, соответствующие нестационарному полю течения (n - номер шага по времени, p - номер, присвоенный газодинамической переменной) мы рассматриваем как тензор

$$\theta^n_{p;ijk} = (\rho^n_{ijk}, u^n_{ijk}, v^n_{ijk}, w^n_{ijk}, e^n_{ijk}). \quad (1)$$

Также как тензор мы рассматриваем набор полей течения

$$\theta^n_{p;ijk;m_{\vartheta_1}...m_{\vartheta_q}} = (\rho^n_{ijk;m_{\vartheta_1}...m_{\vartheta_q}}, u^n_{ijk;m_{\vartheta_1}...m_{\vartheta_q}}, v^n_{ijk;m_{\vartheta_1}...m_{\vartheta_q}}, w^n_{ijk;m_{\vartheta_1}...m_{\vartheta_q}}, e^n_{ijk;m_{\vartheta_1}...m_{\vartheta_q}}), \quad (2)$$

получаемый при решении задачи в пространстве параметров $(\vartheta_1...\vartheta_q) \subset \Omega_q \subset R^q$, что соответствует постановкам, типичным для обобщенного вычислительного эксперимента [3].

Соответственно, оператор эволюции решения (пропагатор) тоже является тензором. В простейшем случае (1) пропагатор является тензором порядка 8, действующим на газодинамические переменные

$$\theta^{n+1}_{p;ijk} = A_{ps;ijklmz}\theta^n_{s;lmz}. \quad (3)$$

Здесь мы подразумеваем суммирование по повторяющимся индексам, что не типично для операций с тензорами (многомерными массивами), однако для наших целей иногда бывает удобно.

Таким образом, дискретизации и газодинамических переменных и соответствующих пропагаторов имеют вид тензоров, хотя на это внимание, как правило, не обращается и в практических приложениях такая их форма не используется в основном из-за запредельных требований по оперативной памяти. Пропагаторы неявно реализуются при численном решении систем частных дифференциальных уравнений (ЧДУ). Для одного временного шага (в каждой отдельной точке) их легко можно выписать в явной форме.

Следует отметить, что тензорная запись численных решений потенциально позволяет сжимать и анализировать данные численных расчетов многопараметрических задач (определенных в пространствах большой размерности (больше трех)). Сжатие данных осуществляется с помощью тензорных разложений, которые являются основной темой данной работы.

Важным обстоятельством является то, что тензорная запись позволяет находить нетривиальные внутренние структуры как в решении, так и в пропагаторе. Простейшие примеры касаются тензоров в векторизованной и матризованной формах. Векторизация и матризация дают более привычные и обозримые формы представления тензоров и формально законны. В шестимерном пространстве, которое мы будем использовать, симметричная матризация (естественная для структуры пропагатора) имеет вид $a_{ijklmn} \to a_{\xi,\eta}, \xi = i + (j-1)I + (k-1)IJ; \eta = l + (m-1)L + (n-1)LM$, соответствующая векторизация $a_{\xi\eta} \to a_{\vartheta\xi}, \vartheta = \xi + (\eta - 1)IJK$. Однако надо помнить, что переход от векторов и матриц к тензорам (обратное преобразование, тензоризация) возможен не всегда, например, в случае длины вектора равной простому числу эта операция невозможна.

В результате векторизации и матризации появляются (характерные для алгебры матриц) собственные вектора и собственные числа, которые, строго говоря, не определены в тензорной форме. Тем не менее, они могут отражать некоторую внутреннюю (скрытую) структуру решения и иметь нетривиальный физический смысл.

В частности, в работе [4] по динамике атмосферы такие собственные вектора соответствуют возмущениям течения, максимально растущим на данном временном интервале (сингулярные вектора). Их можно связать с собственными векторами оператора, получаемого из произведения прямого и сопряженного пропагаторов. Для этого газодинамические переменные векторизуются в виде $u \in R^N$ и эволюция течения описывается пропагатором

$$u(t) = Au_0 \quad (4)$$

Норма решения имеет вид

$$\|u(t)\| = (Au_0, Au_0) = (u_0, A^* Au_0) \quad (5)$$

Поиск максимально (в данной норме) растущих линейных возмущений $\|u(t)\| / \|u_0\|$ на временном интервале Δt сводится к поиску собственных векторов задачи $A^* A\eta_{\max} = \sigma^2_{\max}\eta_{\max}$ с максимальным собственным числом $\sigma^2_{\max}$.

В качестве другого примера можно привести разложение по динамическим модам (DMD) [5,6] (нестационарные двумерные уравнения Эйлера), в котором неявно используется векторизация решения и матризация оператора эволюции (пропагатора). В рамках DMD численное решение задачи аэрогазодинамики на временном шаге i векторизуется и записывается в виде $u_i \in R^M$ (срез, snapshot). При этом предполагается, что существует линейный оператор $A(\Delta t) \in R^{M\times M}$, такой, что $u_{i+1} = Au_i$. Тогда срезы представляют последовательность Крылова $Sn_1^{N+1} = \{u_1, Au_1, A^2u_1, ..., A^N u_1\}$, из которой выделяют два набора $X = \{u_1 ... u_N\}$ и $Y = \{u_2 ... u_{N+1}\} = AX$, $X, Y \in R^{M\times N}$. В принципе (в сильно упрощенной версии) эти данные позволяют построить аппроксимацию пропагатора $A = YX^+$ (здесь X^+ псевдообратная матрица (Moore-Penrose pseudoinverse)), который записывается в сжатой форме в виде произведения прямоугольных матриц

$$A = \Omega_R^A \Lambda \Omega_L^A \quad (6)$$

Эта форма позволяет радикально сократить необходимую для хранения оператора A память, что позволяет использовать пропагатор для решения ряда интересных задач, таких как задачи восприимчивости к внешним возмущениям, задачи поиска сингулярных векторов, аппроксимации операторов Купмана и Перрона-Фробениуса [5,6].

Несмотря на естественность тензорных формулировок задач вычислительной аэрогазодинамики, их применимость крайне ограничена проклятием размерности (даже хранение тензора с числом индексов больше трех, как правило, требует нереалистичных объемов памяти).

Однако в настоящее время в преодолении этих трудностей наблюдается заметный прогресс, связанный с использованием тензорных разложений [7,8,9,10].

В этой работе мы рассмотрим возможности аппроксимации шестимерных тензоров с помощью таких тензорных разложений, как каноническое разложение [7,8] и тензорный поезд [9,10] и проиллюстрируем эти возможности численными экспериментами. Выбор используемого порядка тензоров связан с необходимостью аппроксимации уравнения Больцмана в трехмерной постановке, не является принципиальным и не препятствует применению используемых алгоритмов для параметрических задач аэрогазодинамики.

2. Некоторые определения тензорной алгебры

Для дальнейшего изложения нам потребуется использование довольно большого числа редко употребляемых (если не употреблять слово “экзотичный”) обозначений [1,11], которые мы для удобства приведем ниже.

Тензорным пространством в соответствии с [1] мы будем считать тензорное (внешнее) произведение $\otimes_{j=1}^{d} R^{I_j} = R^{I_1} \otimes R^{I_2} ... \otimes R^{I_d}$ векторных пространств R^{I_j}. Здесь символ $\otimes$ означает внешнее (тензорное) умножение векторов $a_i \otimes b_j = a_i b_j$ (иногда внешнее умножение обозначают символом $\circ$, чтобы отличать от Кронекерова умножения).

Тензор $A = [a_{i_1} ... a_{i_d}]$ (массив с d индексами (d-way array)) является элементом тензорного пространства.

Тензор описывается следующими параметрами:

Порядок (***order***) d тензора равен числу его индексов (number of modes, dimensions). При этом по каждой размерности (моде, индексу) i имеется n_i узлов. При моделировании функций порядок тензора соответствует размерности пространства, в котором задана функция.

Размер (***size***) тензора $n_1 \times ... \times n_d$ равен произведению числа узлов по всем размерностям и соответствует количеству чисел, необходимых для хранения тензора и экспоненциально ($\sim n^d$) растет при увеличении порядка тензора.

Ранг тензора $rank(A) = R$ определяется как минимальное число слоев r ядер $a_r^{(i)}$ (при фиксированном r $a_r^{(i)} \in R^N$ - нормированные вектора), которое необходимо для аппроксимации тензора в следующей форме (каноническое разложение):

$$A = \sum_1^R \lambda_r a_r^{(1)} \otimes ... \otimes a_r^{(N)} \qquad (7)$$

Нить тензора (fiber of the tensor) это вектор, получающийся при одном изменяющемся индексе и остальных фиксированных. Для трехмерного тензора можно выделить следующие нити: $x_{:jk}$ (mode-1 fiber, столбец), $x_{i:k}$ (mode-2 fiber, ряд), and $x_{ij:}$ (mode-3 fiber, “туннель”).

Для тензора мы будем использовать или обсуждать следующие операции.

Произведение тензора и вектора по моде n (n –mode product) обозначается с помощью символа $\times_n$ *как* $Y = X \times_n v$. При этом каждая нить по моде ***n*** скалярно умножается на вектор (свертывается) в виде $(X \times_n v)_{i_1 ... i_{n-1} i_{n+1} ... i_N} = \sum_{i_n=1}^{I_n} x_{i_1 ... i_N} v_{i_n}$. В результате получается тензор на единицу меньшего порядка.

Произведение двух тензоров обозначается символом $\times_q^p$ и имеет вид $Z = A \times_q^p B = \sum A_{...n_{q-1} k n_{q+1} ...} \times_q^p B_{...m_{p-1} k m_{p+1} ...}$.

Произведение Кронекера (Kronecker product, tensor product) ($\otimes$) матриц произвольного размера является обобщением внешнего произведения с векторов на матрицы. В ходе умножения элемент первой матрицы умножается на всю вторую матрицу в следующей форме:

$$C = A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & ... & a_{1J}B \\ a_{21}B & a_{22}B & ... & a_{2J}B \\ ... & ... & ... & ... \\ a_{I1}B & a_{I2}B & ... & a_{IJ}B \end{pmatrix}.$$

В индексной форме для матриц $A \in R^{I \times J}$, $B \in R^{K \times L}$, $M = IK; N = JL$ произведение Кронекера $C = A \otimes B \in R^{IK \times JL}$ можно записать так:

$$c_{mn} = a_{ij} b_{kl},\ m = (i-1)K + k,\ n = (j-1)L + l,\ m = 1...M; n = 1...N. \qquad (8)$$

К сожалению, как видно из (8), индексная запись таких операций не добавляет им той прозрачности, к которой мы привыкли в физических приложениях [12].

Произведение Хатри-Рао (Khatri Rao product) обычно обозначается как $\odot$, здесь нам удобнее использовать символ $\bullet$. Используется оно для матриц с совпадающим числом столбцов, при этом каждый элемент столбца первой матрицы умножается на весь столбец второй, результат выстраивается в столбец. Для $A = [A_1 A_2 ... A_R]$, $B = [B_1 B_2 ... B_R]$

$$A \odot B = [A_1 \otimes B_1 ... A_R \otimes B_R]. \quad (9)$$

Произведение Адамара (Hadamard product)

$$C = A * B \ (c_{ij} = a_{ij} b_{ij}). \quad (10)$$

(без суммирования по повторяющимся индексам) обычно обозначается как $*$ и соответствует поэлементному перемножению матриц одного размера.

Очень часто тензоры бывает удобно раскатать в "блин" (матризовать) или вытянуть в нить (векторизовать), (развернуть, unfold), что выполняется следующими операциями.

Матризация по моде n (mode-n matricization) тензора $X \in R^{I_1 \times ... \times I_N}$ обозначается $X_{(n)}$ и располагает нити по моде ***n*** в столбцы матрицы. Матризация тензора в общем случае имеет форму преобразования $X_{i_1,...,i_N} \to m_{i_n, j}$, $j = 1 + \sum_{\substack{k=1 \\ k \neq n}}^{N} ((i_k - 1) \prod_{\substack{m=1 \\ m \neq n}}^{k-1} I_m)$.

Для $X_{i,j,k} \in R^{I \times J \times K}$, $i = 1...I$, $j = i...J$, $k = 1...K$, матризация может выглядеть, например, так $X_{(1)} = X_{i,m}$, $m = 1...M$, $M = J \cdot K$, $m = j + (k-1)J; (j = i...J, k = 1...K)$.

Векторизация разворачивает матрицу $X_{i,m}$ в вектор Y_j, $j = I_2(i-1) + m$.

Матризация и векторизация тензоров очень популярны, так как позволяют использовать весь спектр алгоритмов линейной алгебры, однако на практике их применение ограничивается проклятием размерности при порядке тензора выше трех.

Нас интересуют разложения тензора с помощью тензоров меньшего порядка или размера. К ним относятся

Разложение Таккера (Tucker) $A = B \times_1 G_1 ... \times_N G_N$, в индексной форме

$$a(i_1,...,i_N) = \sum_{\alpha_1=1}^{r_1} ... \sum_{\alpha_N=1}^{r_N} b(i_1,...,i_d) g_1(i_1, \alpha_1) ... g_N(i_N, \alpha_N)$$

Каноническое разложение $A = I_N \times_1 G_1 ... \times_N G_N$, в индексной форме

$$A_{ij...N} = \sum_{r=1}^{R} g_{i,r} g_{j,r} ... g_{N,r}.$$

Тензорный поезд $A = G_1 \times_2^1 G_2 \times_3^1 ... \times_{N-1}^1 G_{N-1} \times_N^1 G_N$, в индексной форме

$$A(i_1,...,i_N) = \sum_{\alpha_0,...,\alpha_N} g_1(\alpha_0, i_1, \alpha_1) g_2(\alpha_1, i_2, \alpha_2) ... g_N(\alpha_{d-1}, i_d, \alpha_N)$$

Приведенные выше обозначения, как правило, громоздки, интуитивно не прозрачны, используются в достаточно узкой области и не знакомы широкому кругу специалистов. Но, к сожалению, описывать текущее положение дел в области тензорных разложений, не используя эти обозначения, невозможно. Однако мы будем стараться, где только возможно, использовать более привычные индексные обозначения, в некоторых случаях дублируя оба подхода.

3. Каноническое разложение

Упоминавшееся выше каноническое разложение $A = \sum_{1}^{R} Q_r^1 \otimes Q_r^2 \otimes ... \otimes Q_r^N$ в индексном виде для не нормированных ядер записывается как

$$A_{ij\ldots k} = \sum_{1}^{R} Q_{i,r} Q_{j,r} \ldots Q_{k,r} \,. \tag{11}$$

Это выражение единственно с точностью до перестановки членов и масштабирования. Задача определения набора ядер $Q^n = \{Q_r^1, \ldots, Q_r^N\}$ в вариационной форме имеет вид

$$Q^n = \arg\min_{Q^n} \left\| A - \sum_{1}^{R} Q_r^1 \otimes Q_r^2 \otimes \ldots \otimes Q_r^N \right\|. \tag{12}$$

Ключевой величиной при использовании канонического разложения является ранг тензора R. По данным [1,13] он не вычислим из-за некорректности постановки задачи

$$R = \arg\min_{R} \left\| A - \sum_{1}^{R} Q_r^1 \otimes Q_r^2 \otimes \ldots \otimes Q_r^N \right\|. \tag{13}$$

Использование канонического разложения страдает от неустойчивостей и нуждается в регуляризации [1,13]. Однако, по данным [14] аппроксимация с помощью канонического разложения положительных функций (типа плотности вероятности) приводит к корректной устойчивой постановке. Осциллирующее поведение ядер наблюдалось и в данной работе. Впрочем, оно не очень принципиально с точки зрения аппроксимации многомерных функций.

Каноническое разложение подразумевает сжатие $N^n \to N \cdot n \cdot R$, N - размерность пространства, n число узлов по одному направлению, R ранг тензора.

В двумерном случае каноническое разложение тождественно DMD [5,6] $A = \Omega_R^A \Lambda \Omega_L^A$.

Каноническое разложение является некоторым расширением анализа основных компонент (Principal Components Analysis (PCA)) при повышении порядка тензора от двух (перехода от матриц к многоиндексным массивам) и так же позволяет понизить размерность задачи.

4. Тензорный поезд

Каноническое разложение достаточно часто страдает от неустойчивостей [13], по этой причине естественны попытки найти альтернативные разложения, одним из которых является формат тензорного поезда [8] (tensor train, TT). Имеются работы [9], в которых утверждается, что тензорный поезд более устойчив, чем каноническое разложение.

Тензорный поезд (TT) позволяет записать d -мерный $n_1 \times n_2 \times \ldots \times n_d$ тензор A в форме

$$A(i_1, \ldots, i_d) = \sum_{\alpha_0, \ldots, \alpha_d} G_1(\alpha_0, i_1, \alpha_1) G_2(\alpha_1, i_2, \alpha_2) \ldots G_d(\alpha_{d-1}, i_d, \alpha_d) \,, \tag{14}$$

где G_k ядра размера $r_{k-1} \times n_k \times r_k$, $k = 1, \ldots, d$, $r_0 = 1$, $r_d = 1$.

Тензорный поезд является неединственным представлением, так как инвариантен к преобразованию $G_k^{'}(i_k) = G_k(i_k) S$, $G_{k+1}^{'}(i_{k+1}) = S^{-1} G_{k+1}(i_{k+1})$.

Тензорный поезд дает меньшее сжатие nNr^2, чем каноническое разложение.

Тензорный поезд является интересной альтернативой каноническому разложению. Представляет интерес сравнение тензорного поезда с каноническим разложением как с точки зрения устойчивости результатов, так и с точки зрения вычислительной эффективности. Мы надеемся провести соответствующее сравнение в последующих работах.

5. Методы расчета тензорных разложений

Методы расчета тензорных разложений можно с некоторой условностью разделить на два подкласса: методы, основанные на линейной алгебре, например [9] и вариационные методы [6,7].

Методы, основанные на линейной алгебре в существенной степени опираются на матризацию тензоров, сингулярное разложение и содержат много интересных и оригинальных алгоритмов, позволяющих проводить операции прямо над ядрами, не обращаясь к аппроксимируемым функциям.

Вариационные формулировки, как правило, опираются на метод переменных наименьших квадратов (alternating least squares (ALS)) [15,16], однако тоже используют матризацию тензоров.

На наш взгляд слабым местом обоих этих подходов является использование матризованного тензора, хранение которого требует очень большой памяти.

Однако, по нашему мнению, вариационные методы можно избавить от этого недостатка. Поэтому здесь мы используем некоторую комбинацию метода переменных наименьших квадратов и стохастического градиентного спуска (SGD) [17,18,19], которую подробно опишем ниже. Грубо говоря, мы минимизируем невязку на одной случайно выбранной нити (fibres) с помощью ALS, что позволяет нам на каждом шаге решать одномерную задачу с весьма умеренными требованиями к памяти.

5.1 Метод переменных наименьших квадратов (ALS)

Метод переменных наименьших квадратов (alternating least squares (ALS)) [15,16] широко используется при поиске тензорных разложений и позволяет оптимизировать один из параметров при фиксированных остальных. Как правило, для канонического разложения ALS реализуется с помощью произведения Хатри-Рао, чаще всего, для трехмерных задач. В интересующем нас многомерном случае [19,20] ядра Q_k определяются с помощью последовательного решения следующей задачи

$$Q_k = \arg\min_{Q_k} \left\| A_{(k)} - Q_k (Q_1 \bullet \ldots \bullet Q_{k-1} \bullet Q_{k+1} \bullet \ldots \bullet Q_N)^T \right\|^2 . \qquad (15)$$

Здесь $A_{(k)}$ - матризация тензора по моде *k*, $Q_k (Q_1 \bullet \ldots \bullet Q_{k-1} \bullet Q_{k+1} \bullet \ldots \bullet Q_d)^T$ - вспомогательная матрица той же размерности.

Для минимума (15) можно получить элегантное выражение

$$Q_k = A_{(k)} ((Q_1 \bullet \ldots \bullet Q_{k-1} \bullet Q_{k+1} \bullet \ldots \bullet Q_d)^T)^+ , \qquad (16)$$

позволяющее определить искомое ядро методами матричной алгебры.

В общем случае (при вариации всех параметров) выпуклость не гарантирована и градиентный спуск не обязан сходиться. Фиксация основной части параметров и вариация одного ядра позволяет получить выпуклый целевой функционал и успешно его оптимизировать. Ценой относительной универсальности метода переменных наименьших квадратов является низкая скорость сходимости.

Подход к расчету канонического разложения с помощью выражений типа (16) на данный момент доминирует. К сожалению, для наших целей этот подход не пригоден, поскольку использует матризацию тензора, требующую ту же величину памяти, что и сам тензор. Для задач рассматриваемого нами класса необходимая память превосходит все возможности современной вычислительной техники (в данной работе мы оперируем с тензорами, формально содержащими 10^{12} чисел), что исключает использование матризации тензора.

5.2 Стохастический градиентный спуск

Стохастический градиентный спуск (stochastic gradient descent (SGD)) широко используется в задачах высокой размерности [17,18,19]. Парадоксально, но он позволяет найти точку минимума функционала в пространстве управляющих параметров за время меньшее, чем нужно для полного вычисления этого функционала. Это происходит за счет того, что вместо глобального (batch) и трудно вычислимого функционала невязки типа (12) оптимизируется локальный функционал невязки в отдельной случайно выбранной точке или малом наборе случайных точек (minibatch). Достаточно часто он используется в комбинации с ALS, чтобы преодолеть трудности, вызываемые большими потребностями в памяти при использовании произведения Хатри-Рао [17,18].

В нашем случае мы используем функционал, рассчитанный на одной случайно выбранной нити i ($n_i = 1...N_i$ при остальных фиксированных индексах) или небольшом наборе таких нитей. Соответствующий алгоритм подробно описан в следующем разделе.

6. Численный алгоритм, реализующий каноническое разложение

Как мы уже говорили, наш подход соответствует некоторой комбинации стохастического градиентного спуска (в варианте minibatch) и метода переменных наименьших квадратов. Изложим его подробно для случая одной нити (случай использования набора нитей получается простым суммированием невязки и градиента), поскольку описания этого алгоритма в литературе авторам найти не удалось. Для этого рассмотрим шестимерный случай, в котором наша функция аппроксимируется в виде

$$f_{ijklmp} = \sum_{\alpha=1}^{R} Q^x(\alpha,i) \cdot Q^y(\alpha,j) \cdot Q^z(\alpha,k) \cdot Q^u(\alpha,l) \cdot Q^v(\alpha,m) \cdot Q^w(\alpha,p). \qquad (17)$$

Сначала определяем первое ядро $Q^x(\beta,i)$ связанное с координатой x, остальные ядра определяются последовательно, соответствующие выражения можно получить циклической заменой.

Выберем нить (fibre), вдоль x, ($i = 1...N_x$) с помощью случайного равномерно распределенного выбора остальных индексов j,k,l,m,p.

Рассмотрим невязку вдоль этой нити, для этого суммируем локальные (в точке) невязки по i:

$$\varepsilon(Q^x) = \sum_{i}^{N_x} \{\sum_{\alpha} Q^x(\alpha,i) \cdot Q^y(\alpha,j) \cdot Q^z(\alpha,k) \cdot Q^u(\alpha,l) \cdot Q^v(\alpha,m) \cdot Q^w(\alpha,p) - \tilde{f}_{ijklmp}\}^2 / 2 \cdot \qquad (18)$$

Здесь $\tilde{f}_{ijklmp}$ - точное значение функции в точке i,j,k,l,m,p, известное нам заранее (в данной работе из аналитических выражений для тестовых функций). Невязку в соответствии с подходом ALS считаем зависящей только от одного ядра $Q^x(\alpha,i)$. Возмутим это ядро на величину $\Delta Q^x(\beta,i)$. Соответствующее возмущение невязки имеет вид

$$\Delta_x \varepsilon = \sum_{i} \{(\sum_{\alpha} (Q^x(\alpha,i) \cdot Q^y(\alpha,j) \cdot Q^z(\alpha,k) \cdot Q^u(\alpha,l) \cdot Q^v(\alpha,m) \cdot Q^w(\alpha,p) - \tilde{f}_{ijklmp}) \cdot$$
$$(\sum_{\beta} \Delta Q^x(\beta,i) \cdot Q^y(\beta,j) \cdot Q^z(\beta,k) \cdot Q^u(\beta,l) \cdot Q^v(\beta,m) \cdot Q^w(\beta,p))\}. \qquad (19)$$

Выберем возмущение $\Delta Q^x(\beta,i)$ не равное нолю только в одной точке i, что позволит избавиться от суммирования по i в (19). Тогда можно выделить соответствующее значение градиента невязки в форме:

$$\nabla_{x,i,\beta}\varepsilon = (\sum_{\alpha} (Q^x(\alpha,i)\cdot Q^y(\alpha,j)\cdot Q^z(\alpha,k)\cdot Q^u(\alpha,l)\cdot Q^v(\alpha,m)\cdot Q^w(\alpha,p) - \tilde{f}_{ijklmp})\cdot$$
$$(Q^y(\beta,j)\cdot Q^z(\beta,k)\cdot Q^u(\beta,l)\cdot Q^v(\beta,m)\cdot Q^w(\beta,p))\}. \quad (20)$$

Недостатком выражения (20) в сравнении с методами ALS, использующими произведение Хатри-Рао (16), является невозможность прямого использования условия $\nabla_{x,i,\beta}\varepsilon = 0\cdot$(из-за того, что искомая величина $Q^x(\alpha,i)$ суммируется по α) для расчета ядер (фактор-матриц). Достоинством (20) по сравнению с (16) является его экономичность с точки зрения используемой памяти (матризация тензора не используется).

В случае регуляризации Тихонова нулевого порядка ($\gamma(Q^x(\beta,i))^2$) в выражение (20) нужно добавить регуляризирующий член $-\gamma Q^x(\beta,i)$.

Итерации метода наискорейшего спуска, минимизирующие функционал (18) для элемента ядра Q^x (в точке β,i), имеют вид:

$$\{Q^x(\beta,i)\}^{n+1} = \{Q^x(\beta,i)\}^n - \tau\nabla_{x,i,\beta}\varepsilon \quad (21)$$

где τ - шаг итерации.

Итерации идут по одному ядру на выбранной нити до установления. В расчетах в качестве критерия останова использовалась величина невязки на нити (18). Итерации прекращались при $\varepsilon \le \varepsilon_1 = 10^{-9} \div 10^{-13}$.

После окончания итерации на текущем ядре переходим к следующему ядру, используя новую случайную нить. Форма градиента дается перестановками во втором члене (19), сумма в (18) зависит от выбранной нити.

После того, как мы локально оптимизировали все ядра, мы переходим к следующему шагу глобальной итерации (от новых значений ядер) и снова начинаем поиск оптимальных ядер с Q^x. Формально качество аппроксимации функции каноническим разложением на каждом шаге глобальной итерации может быть оценено с помощью следующей невязки

$$\tilde{\varepsilon}_{total} = \sum_{i,j,k,l,m,p} \{\sum_{\alpha} Q^x(\alpha,i)\cdot Q^y(\alpha,j)\cdot Q^z(\alpha,k)\cdot Q^u(\alpha,l)\cdot Q^v(\alpha,m)\cdot Q^w(\alpha,p) - \tilde{f}_{ijklmp}\}^2 / 2\cdot \quad (22)$$

К сожалению, этот функционал напрямую невычислим (по крайней мере на обычных персональных компьютерах) из-за проблем с размерностью и соответствующими огромными затратами компьютерного времени. Мы его значение численно оценивали с помощью метода Монте-Карло в форме

$$\varepsilon_{total} = \frac{1}{2Mc}\sum_{s=1}^{s=MC} \{\sum_{\alpha} Q^x(\alpha,i)\cdot Q^y(\alpha,j)\cdot Q^z(\alpha,k)\cdot Q^u(\alpha,l)\cdot Q^v(\alpha,m)\cdot Q^w(\alpha,p) - \tilde{f}_{ijklmp}\}^2, \quad (23)$$

где на каждом шаге суммирования s каждый индекс из i,j,k,l,m,p выбирался случайным и равномерно распределенным числом. В результате мы вычисляли среднюю по ансамблю сумму квадратов ошибки аппроксимации. Число попыток в ансамбле находилось в интервале $MC = 1000 \div 100000$, где результаты довольно слабо менялись при изменении MC.

Оптимизация всех ядер (и весь процесс построения канонического разложения) заканчивалась при $\varepsilon_{total} \le \varepsilon_2$, $\varepsilon_2 = 10^{-5} \div 10^{-7}$.

В целом, комбинации ALS и SGD достаточно распространены [17,18], однако, во всех известных авторам случаях, они используют произведение Хатри-Рао. Предлагаемый здесь подход не использует произведение Хатри-Рао ни в какой форме, а опирается на прямое дифференцирование невязки, определенной на одной нити и градиентный спуск. Благодаря этому ни матризация тензора, ни произведение ядер в форме Хатри-Рао не используются, что позволяет радикально сократить потребности в ис-

пользуемой памяти. Ценой этого успеха является увеличение числа итераций, необходимых для решения задачи. К счастью, для рассматриваемых задач это не приводит к ощутимым последствиям, так как время расчета рассмотренных ниже задач на персональном компьютере (Intel I5, 2.66 ГГц) остается в пределах нескольких минут.

7. Результаты численных экспериментов

В результате расчетов мы получаем аппроксимацию шестимерной функции $\tilde{f}$ (точнее говоря тензора $\tilde{f}_{ijklmp}$, соответствующего значениям функции в узлах регулярной сетки) с помощью канонического разложения и соответствующего набора ядер $Q^x(\alpha,i)\cdot Q^y(\alpha,j)\cdot Q^z(\alpha,k)\cdot Q^u(\alpha,l)\cdot Q^v(\alpha,m)\cdot Q^w(\alpha,p)$.

В численных экспериментах использовалась сетка, содержащая 100 узлов по каждой координате. С формальной точки зрения при такой сетке хранение $\tilde{f}_{ijklmp}$ потребует 10^{12} ячеек памяти, что совершенно нереалистично ни с точки зрения хранения, ни с точки зрения визуализации. Отметим, что в этом случае память, занимаемая ядрами с рангом 100, составляет $6\cdot 100\cdot 100 = 60000$ ячеек, что иллюстрирует сверхвысокую степень сжатия информации ($\sim 10^7$) при использовании канонического разложения.

В процессе отладки проводилось сравнение численных (полученных прямым численным дифференцированием) и аналитических градиентов (полученных с помощью выражения (20)) показавшее их практически полное совпадение.

Формально качество аппроксимации функции каноническим разложением может быть оценено с помощью невязки (22), но оценивали ее с помощью метода Монте-Карло (23).

Результаты расчетов дают достаточно устойчивые и воспроизводимые оценки погрешности.

Численные эксперименты осуществлялись с помощью авторского программного обеспечения, написанного на языке Fortran-95 специально под рассматриваемые задачи.

7.1 Тестовые задачи

Проведено тестирование аппроксимации различных функций с помощью канонического разложения. При тестировании представляет интерес качество аппроксимации (17) как с точки визуального представления, так и с точки величины невязки (23). Определение истинного ранга функции и необходимого числа ядер также интересно, как и скорость сходимости. Соответствующие данные приведены в этом разделе. Выбраны шестимерные функции, чье тензорное представление имеет различный ранг. Это позволяет оценивать не только качество аппроксимации, но и возможности численного определения ранга исследуемых функций. Рассмотрены следующие многомерные функции, расположенные в порядке возрастания сложности (ранга канонического разложения).

1. *Произведение векторов*

$$f = x\cdot y \qquad (25)$$

Это простейшая функция, для которой априорный ранг тензора равен единице. Использована одна нить, ранг аппроксимирующего ядра от одного до 10, 30 итераций. Зависимость величины невязки от ранга представлена в Табл. 1.

Таблица 1. Зависимость невязки (23) от ранга в (17) для функции (25).

ранг	1	2	3	4	5	10
невязка (23)	$3.27\cdot10^{-14}$	$2.78\cdot10^{-6}$	$4.61\cdot10^{-5}$	$3.34\cdot10^{-6}$	$2.44\cdot10^{-4}$	$1.63\cdot10^{-4}$

Из этих результатов видно, что величина невязки может служить индикатором истинного ранга тензора. По мере увеличения ранга увеличивается зашумленность результата, по всей видимости, это отражает неустойчивости при определении ранга, возникающие вследствие некорректности для канонического разложения [1,13]. Результаты расчетов ($f(x,y)$) представлены на Рис. 1 (точная функция) и 2 (аппроксимация, ранг 5).

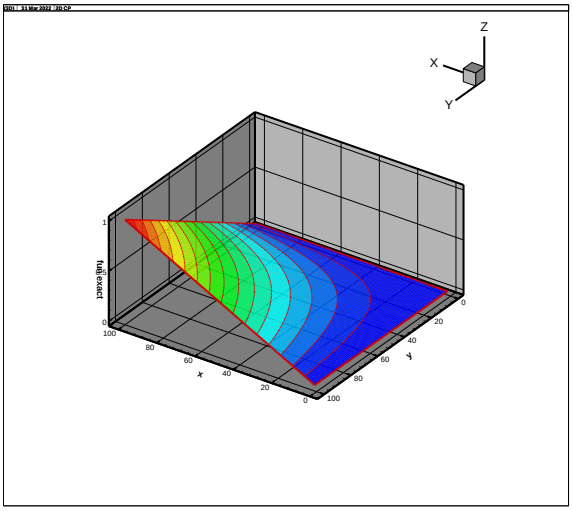

Рис. 1 Точная функция (25)

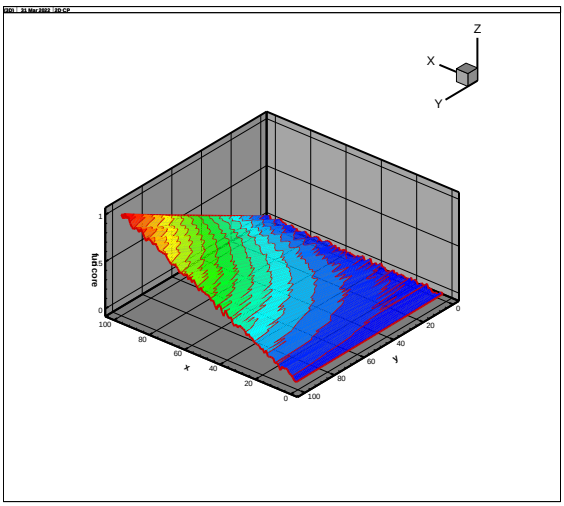

Рис. 2. Аппроксимация функции (25), ранг 5

2. *Сумма векторов*

$$f = x + y \qquad (26)$$

Это тоже предельно простая функция, но ее априорный ранг не известен и было бы желательно определить его в расчете. Использовано 30 итераций, одна нить, величина ранга от одного до 10. Зависимость величины невязки от ранга представлена в Табл. 2.

Таблица 2. Зависимость невязки (23) от ранга в (17) для функции (26).

ранг	1	2	3	4	5	10
невязка (23)	$4.87\cdot10^{-2}$	$1.12\cdot10^{-2}$	$8.67\cdot10^{-5}$	$8.02\cdot10^{-5}$	$1.5\cdot10^{-4}$	$2.03\cdot10^{-4}$

Если судить по минимуму невязки, эта функция имеет ранг 3÷4. На Рис. 3 представлена точная функция, на Рис. 4- ее аппроксимация (ранг 5), совпадение вполне достаточное.

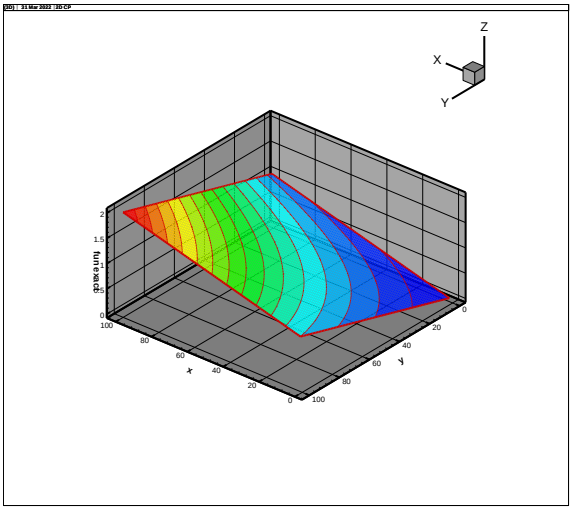

Рис. 3 Точная функция (26)

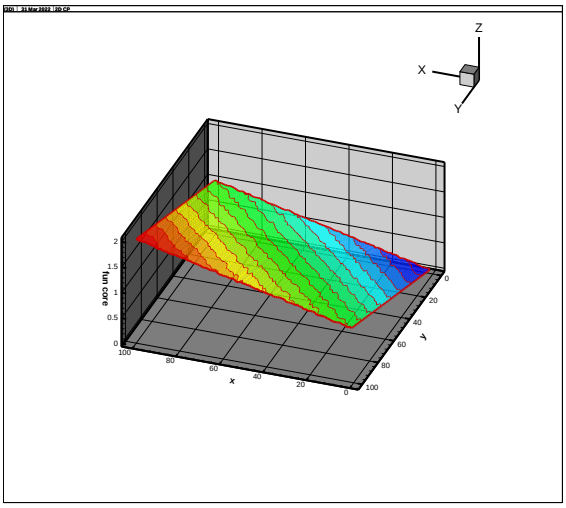

Рис. 4. Аппроксимация функции (26)

3. *Сумма синусов*

$$f = \sin(x/20) + \sin(y/20) \qquad (27)$$

Это двумерная функция в шестимерном пространстве, визуально существенно более сложная, чем функции (25) и (26). Расчеты показали, что ранг этой функции около 10. Результаты расчетов представлены на Рис. 5 (точная функция) и Рис. 6 (аппроксимация, ранг 10, 1 нить, $\varepsilon = 5 \cdot 10^{-4}$, 13 итераций).

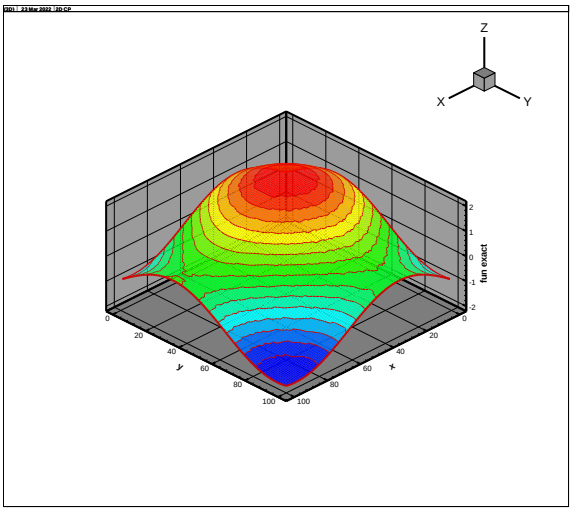

Рис. 5. Точная функция (27)

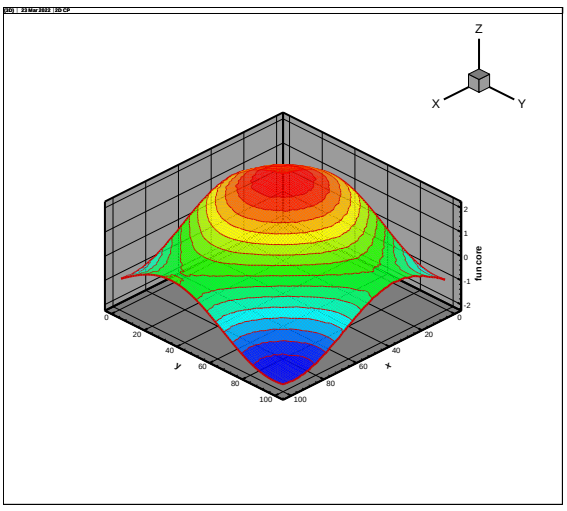

Рис. 6. Аппроксимация функции (27)

4. *Двумерное произведение синусов*

$$f = \sin(x/20) \cdot \sin(y/20) \qquad (28)$$

Это тоже двумерная функция в шестимерном пространстве, мультипликативный аналог (27). Ранг этой функции около 10. Результаты расчетов представлены на Рис. 7 (точная функция) и Рис. 8 (аппроксимация, ранг 10, 1 нить, $\varepsilon = 7 \cdot 10^{-5}$, 30 итераций).

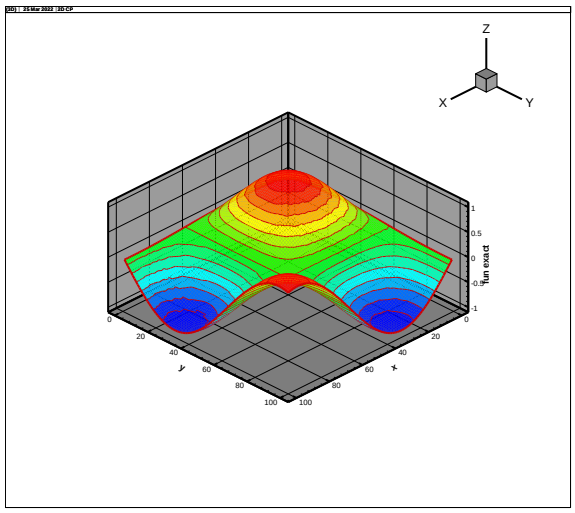

Рис. 7. Точная функция (28)

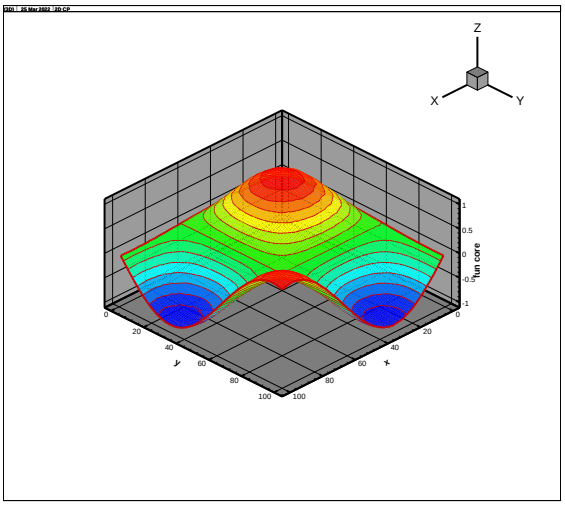

Рис. 8. Аппроксимация функции (28)

Предыдущие тесты проводились в шестимерном пространстве над двумерными функциями и продемонстрировали достаточно быструю сходимость (за 10-30 итераций) и низкие значения невязки. Далее мы рассмотрим истинно многомерные задачи, которые требуют большего числа итераций и дают большие невязки.

5. *Гауссиана в многомерном пространстве*

Рассмотрим функцию в многомерном пространстве, описываемую следующим уравнением

$$f = \exp(-rad^2)$$

$$rad = 0.001 \cdot ((ix - 50) + (iy - 50) + (iz - 50) + (iu - 50) + (iv - 50) + (iw - 50)) \quad (29)$$

Формально эта функция определена в шестимерном пространстве, но, по сути, эта функция одномерна (зависит только от радиуса) и определяется произведением векторов, поэтому ее ранг равен единице. В Табл. 3. представлена зависимость невязки от ранга для функции (29).

Таблица 3. Зависимость невязки (23) от ранга в (17) для функции (29).

ранг	1	2	3	4	5	10
невязка (23)	$5.31 \cdot 10^{-13}$	$6.92 \cdot 10^{-7}$	$2.78 \cdot 10^{-6}$	$6.74 \cdot 10^{-4}$	$3.91 \cdot 10^{-5}$	$5.13 \cdot 10^{-3}$

Использовано 30 итераций, одна нить. Для ранга 1 наблюдается полное совпадение функции и ее аппроксимации. По мере увеличения ранга увеличивается зашумленность результата и падает точность (увеличивается невязка).

Таким образом, определение точного ранга функции необходимо и расчеты для ранга “с запасом” могут не дать необходимого качества.

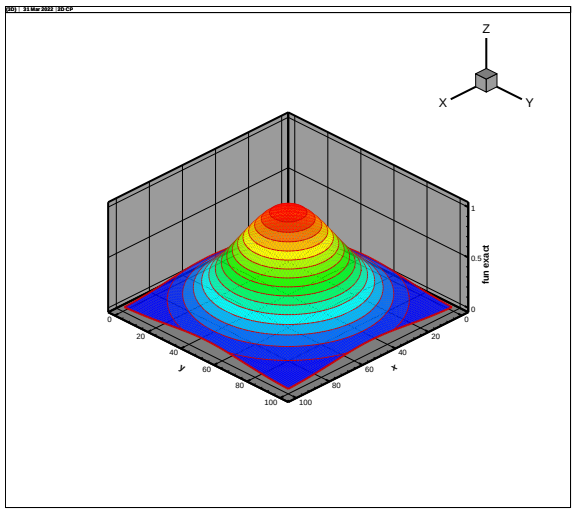

Рис. 9. Точная функция (29)

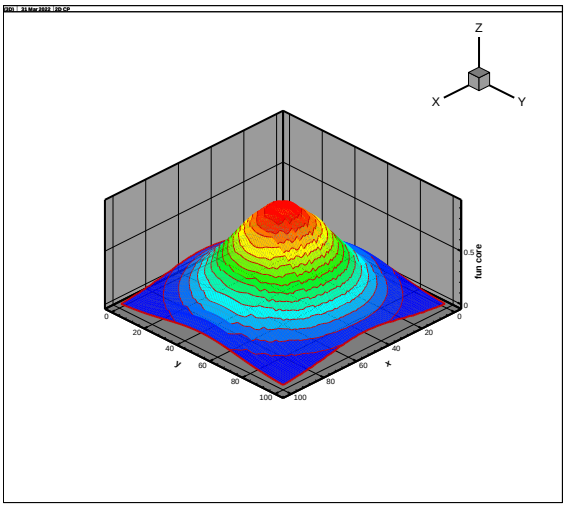

Рис. 10. Аппроксимация функции (29), ранг 5.

6. *Сумма синусов*.

$$f = \sin(x/20) + \sin(y/20) + \sin(z/20) + \sin(u/20) + \sin(v/20) + \sin(w/20) \qquad (30)$$

Это наиболее сложный для расчетов вариант, истинно шестимерный. В следующем разделе будет показано, что ранг этой функции около 200. Расчеты для этого варианта сходятся достаточно медленно (около 300 итераций) и требуют достаточно большой ранг. На Рис. 11 и 12 представлены результаты для 10 нитей и ранга 200 в плоскости x, y, невязка $\varepsilon = 3 \cdot 10^{-4}$. Остальные переменные соответствуют серединам интервалов на сетке 100 ($iz = iu = iv = iw = 50$).

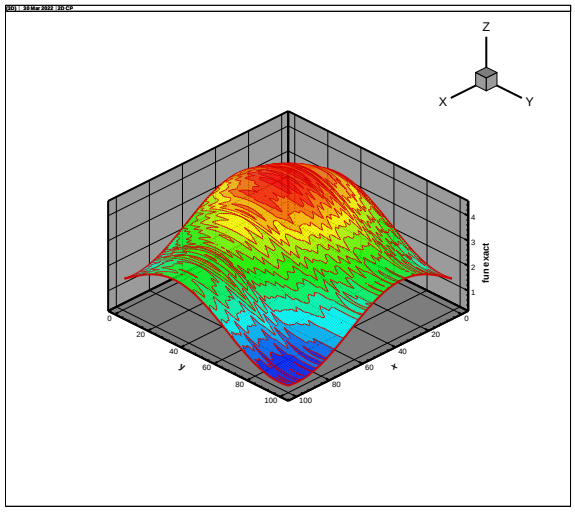

Рис. 11. Точная функция (30)

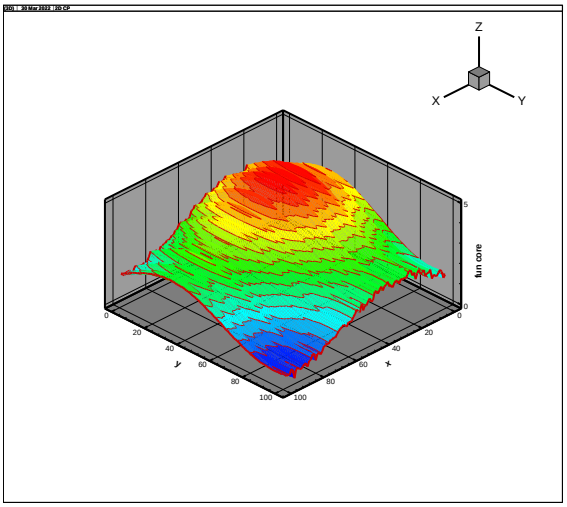

Рис. 12. Аппроксимация функции (30), ранг 200

7.2. Критерии сходимости итераций.

На Рис.13 представлено поведение разных критериев сходимости в зависимости от числа итераций для функции (27). Рассмотрены невязки на нити (eps_fibre), глобальная невязка, оцененная с помощью Монте-Карло (eps_MC) и норма градиента невязки (grad norm). В варианте оптимизации, проиллюстрированном Рис. 13, было разрешено увеличение невязки при переходе к следующему шагу глобальной итерации (обновлении ранга).

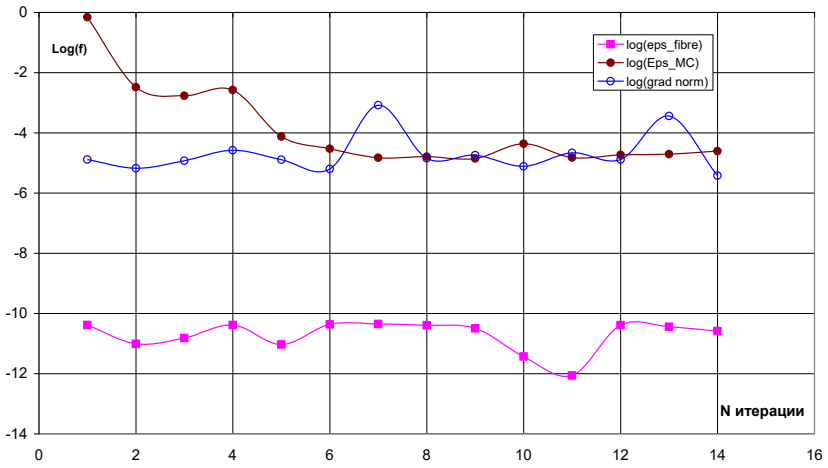

Рис. 13 Разные критерии сходимости в зависимости от числа итераций.

Данный вариант не дает монотонной сходимости, хотя на достаточно простых функциях минимизация осуществляется. Локальная сходимость (сумма по нити) достигается на каждом шаге глобальной итерации. Сходимость по норме градиента отсутствует. Наблюдается медленная и немонотонная сходимость глобальной погрешности (разницы точного и приближенного решений в норме L_2, оцененной с помощью метода Монте-Карло (23)).

Для более сложных функций использовался вариант минимизации с запретом возрастания невязки при переходе на следующий (случайный) набор нитей (при следующем шаге глобальной итерации). Фактически для этой версии алгоритма в какой-то момент градиентная оптимизация меняется на стохастический поиск по нитям.

С интуитивной точки зрения кажется, что использование нескольких нитей вместо одной (minibatch) должно улучшить монотонность сходимости. Однако численные эксперименты показали, что с какого-то момента это ухудшает скорость сходимости и достижимую величину невязки. На Рис. 14 представлены результаты сходимости при использовании 1 и 5 нитей для функции (27).

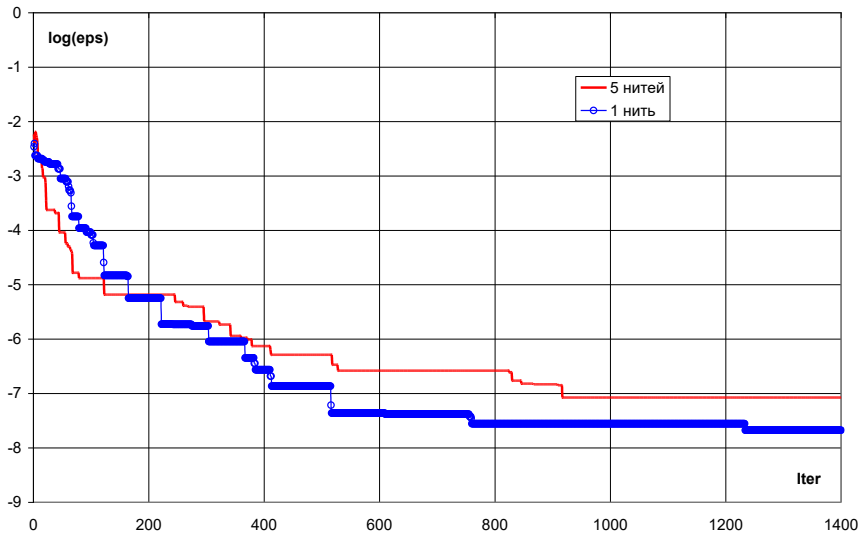

Рис. 14. Логарифм невязки (23) в зависимости от числа итераций для одной и пяти нитей.

7.3 Ранг разложения

Численная оценка ранга тензора получается перебором по величине ранга при решении вариационной задачи (23). Предполагалось, что с увеличением ранга невязка должна убывать, что обычно и наблюдается в расчетах. Однако для некоторых простых функций поведение имеет противоположный характер. Например, для многомерной гауссианы (29) минимум невязки получается при ранге 1. Увеличение ранга эту невязку несколько увеличивает.

Для функции, описываемой уравнением (30), зависимость логарифма невязки от ранга представлена на Рис. 15 (1 нить, 300 итераций).

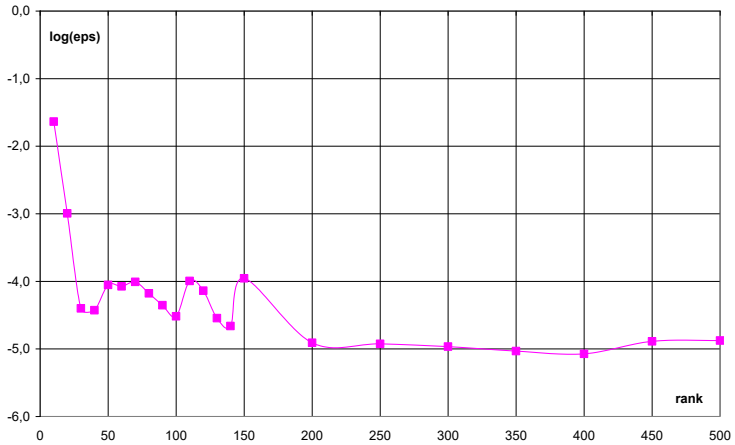

Рис. 15. Зависимость невязки от величины ранга для функции (30)

Для этой функции зависимость невязки от ранга достаточно немонотонна. Хотя в целом увеличение ранга невязку уменьшает. Поэтому при расчетах нужно адаптировать ранг для уменьшения невязки. Здесь использован простейший вариант – перебор увеличивающихся рангов. Но решать задачи в последовательно раздувающемся пространстве достаточно затратно, поэтому поиск более совершенного алгоритма для определения ранга целесообразен, несмотря на имеющиеся принципиальные трудности [13].

На Рис. 16 представлено ядро $Q^x(\alpha,i)$ для функции (30), ранг 100. На Рис. 17 представлено ядро $Q^x(\alpha,i)$ для функции (30), ранг 250. Наблюдается осциллирующий характер ядра, затухания ядра при увеличении ранга нет, хотя точность аппроксимации функции растет.

Трудности с определением ранга в каноническом разложении стимулируют исследование других тензорных разложений, в частности тензорного поезда.

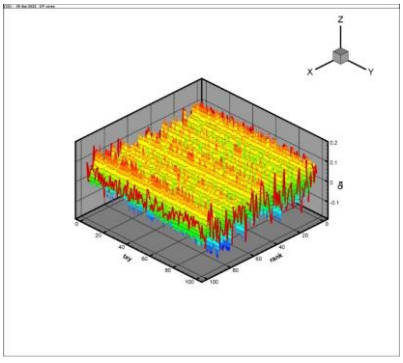

Рис. 16 $Q^x(\alpha,i)$, ранг 100

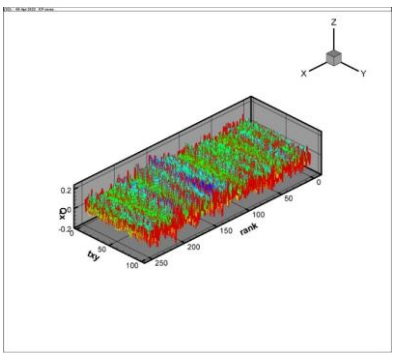

Рис. 17 $Q^x(\alpha,i)$, ранг 250

8. Обсуждение

Если выбрать начальное приближение ядер постоянным, например $Q^m(\alpha,i)=1$, все значения градиентов для разных α автоматически совпадут (19) и процесс оптимизации запустить не получится. Поэтому в качестве начального приближения ядер использовалось выражение $Q^m(\alpha,i)=1+N(\sigma)$ (использовалась случайная нормально распределенная величина, дисперсия которой равна σ). Естественно, что при $\sigma=0$ сходимости нет, при больших $\sigma\approx10$ наблюдались неустойчивости, оптимальным значением было $\sigma=0.1$, которое и использовалось при всех вышеописанных расчетах.

Полученные результаты зависят от случайного выбора нитей и от шума в начальном приближении ядер, что несколько затрудняет их сравнение при отладке и поиске оптимальных параметров настройки алгоритма. Стабилизировать их получается фиксацией стартовой точки генератора случайных чисел и запоминанием последовательностей координат нитей, используемых при оптимизации.

В связи с некорректностью задачи [13] при определении ядер канонического разложения предусмотрена возможность квадратичной регуляризации нулевого порядка по Тихонову [21]. Однако в численных экспериментах подавление осцилляций при расчете ядер приводило к нарушению аппроксимации функции, поэтому положительного влияния регуляризации на расчеты не наблюдалось и вышеприведенные численные результаты соответствуют отсутствию регуляризации. В рамках задачи по аппроксимации функции некорректность канонического разложения существенного влияния не оказывает. В рамках задач по решению эволюционных задач в частных дифференциальных уравнениях [10] устойчивость тензорного разложения представляется полезной, поскольку позволяет часть эволюции проводить в пространстве ядер, не возвращаясь в пространство функций. В рамках тензорного поезда [9] возможность эволюции в пространстве ядер предусмотрена и существует набор операций, включающий специальную операцию округления (TT-rounding) позволяющую понижать ранг аппроксимации после нескольких шагов. Это объясняет существенный интерес к формату тензорного поезда. Таким образом, неустойчивость на уровне определения ядер не является непреодолимым препятствием при моделировании ЧДУ в пространствах высокой размерности, однако переход к более устойчивым постановкам (тензорный поезд) может позволить получить более быстрые алгоритмы.

В расчетах наблюдалось совпадение численных и аналитических градиентов, но аналитический расчет на рассматриваемой задаче быстрее примерно на два порядка. Это связано с тем, что при численном расчете градиента приходится для каждого элемента ядра рассчитывать возмущение невязки (18), при аналитическом (20) потребность в этом (и в суммировании по координате) отсутствует.

Правдоподобная идея, что переход от одной нити к нескольким сделает оптимизацию устойчивей, в расчетах не подтвердилась, этот переход замедляет скорость сходимости и качество оптимизации (увеличивает величину достижимой невязки).

Численные эксперименты показали, что ранг тензора очень сильно зависит от типа аппроксимируемой функции.

По мере увеличения ранга увеличивается зашумленность результата, поэтому выбор величины ранга “с запасом”, позволяющий несколько упростить алгоритм, может привести к ухудшению качества результатов.

Тензорные разложения достаточно активно используются для целей визуализации, поскольку позволяют построить легко вычислимую модель, аппроксимирующую труднообрабатываемый набор данных в параметрическом пространстве. Например, в [22,23] для этих целей используется формат тензорного поезда и кросс-аппроксимация [24].

Тензорные разложения (каноническое разложение, тензорный поезд, иерархический Таккер) используются для экономичного решения многомерных задач типа уравнения Больцмана [7,8,10].

Каноническое разложение позволяет эффективно аппроксимировать и хранить многомерные функции. Затраты компьютерного времени на работу с функциями в шестимерном пространстве (при использовании 100 узлов по каждой координате, что формально требует хранение и работу с 10^{12} чисел) составляют 2-3 минуты на ПК (процессор Intel I5, 2.66 ГГц) при затратах памяти на хранение ядер в максимальном случае около 10^5 чисел.

Заключение

Применимость тензорных формулировок задач вычислительной аэрогазодинамики ограничена "проклятием размерности". К счастью, использование тензорных разложений дает определенную надежду на его преодоление.

Предложен алгоритм, объединяющий метод переменных наименьших квадратов и стохастический градиентный спуск, требования по используемой памяти которого, существенно меньше, чем в методах, использующих произведение Хатри-Рао.

Численные эксперименты показывают, что применение данного алгоритма для канонического разложения позволяет хранить и визуализировать функции в многомерном пространстве с очень умеренными затратами памяти и времени счета. Приведены результаты визуализации проведенных численных экспериментов.

Библиографический список

1. W. Hackbusch. Tensor Spaces and Numerical Tensor Calculus. Springer, 2012.
2. H. Yorick, S. Willi-Hans, Matrix Calculus, Kronecker Product and Tensor Product: A Practical Approach to Linear Algebra, Multilinear Algebra and Tensor Calculus with Software Implementation, Singapore : World Scientific Publishing Co. Pte. Ltd., 2019.
3. А. К. Алексеев, А. Е. Бондарев, В. А. Галактионов, А. Е. Кувшинников, Обобщенный Вычислительный Эксперимент и Задачи Верификации, Программирование, 2021, № 3, стр. 30-38
4. Farrell B.F. and A.M. Moore, An adjoint method for obtaining the most rapidly growing perturbation to oceanic flows, J. Phys. Oceanogr, 22 338-349, 1992
5. Алексеев А.К., Бондарев А.Е., О применении разложения по динамическим модам в задачах вычислительной газовой динамики, Препринты ИПМ им. М.В.Келдыша. 2018, N 154. с. 30
6. A.K. Alekseev, D.A. Bistrian, A.E. Bondarev, I.M. Navon, On Linear and Nonlinear Aspects of Dynamic Mode Decomposition, Int. J. Numer. Meth. Fluids, 2016, V. 82, Issue 6, p. 348–371
7. A. M. P. Boelens, D. Venturi, D. M. Tartakovsky, Parallel tensor methods for high-dimensional linear PDEs, J. Computat. Phys. 375 (2018) 519-539.
8. Arnout M. P. Boelens, Daniele Venturi, Daniel M. Tartakovsky, Tensor methods for the Boltzmann-BGK equation, arXiv:1911.04904v2 2020
9. Oseledets I. V., Tensor- train decomposition, SIAM J. Sci. Comput. , 33 (2011), pp. 2295–2317
10. A.V. Chikitkin, E.K. Kornev, V.A. Titarev, Numerical solution of the Boltzmann equation with S-model collision integral using tensor decompositions, arXiv:1912.04582v1 2019,
11. Kolda T. G. and Bader B. W., Tensor Decompositions and Applications, *SIAM Review, 51(3):455–500, 2009*.
12. Коренев Г.В., Тензорное исчисление, М. МФТИ, 1995
13. V. D. Silva and L -H. Lim, Tensor rank and the ill-posedness of the best low rank approximation problem, SIAM J. Matrix Anal. Appl., 30(3) 1084-1127, 2008.

14. L-H. Lim and Pierre Comon, Nonnegative approximations of nonnegative tensors, Chemometrics 2009; 23: 432–441.

15. P. Comon, X. Luciani, and A. L.F. De Almeida, Tensor decompositions, alternating least squares and other tales, Journal of Chemometrics, vol. 23, no. 7-8, pp. 393–405, 2009.

16. A. Uschmajew, Local convergence of the alternating least squares algorithm for canonical tensor approximation, SIAM J. Matrix Anal. Appl. 33 (2) (2012) 639–652.

17. C. Battaglino, G. Ballard, T.G. Kolda, A practical randomized CP tensor decomposition, arXiv:1701.06600, 2017.

18. X. Fu, S. Ibrahim, H.-T. Wai, C. Gao, and K. Huang, Block-randomized stochastic proximal gradient for low-rank tensor factorization, *IEEE Transactions on Signal Processing*, 2020.

19. Ioanna Siaminou, Athanasios P. Liavas, An Accelerated Stochastic Gradient for Canonical Polyadic Decomposition, arXiv:2109.13964v1 2021.

20. Guoxu Zhou, Andrzej Cichocki, and Shengli Xie, Accelerated Canonical Polyadic Decomposition by Using Mode Reduction, arXiv:1211.3500v2, 2013

21. Тихонов А.Н., Арсенин В.Я., Методы решения некорректных задач, М., Наука, 1979.

22. R. Ballester-Ripoll, E. G. Paredes, and R. Pajarola, A Surrogate Visualization Model Using the Tensor Train Format, SA '16: SIGGRAPH ASIA 2016 Symposium on Visualization November 2016 Article No. 13 Pages 1–8 https://doi.org/10.1145/3002151.3002167

23. Susanne K. Suter, Tensor Approximation in Visualization and Graphics: Background Theory, PhD thesis, University of Zurich, Switzerland, April 2013.

24. Oseledets I., Tyrtyshnikov E., TT-cross approximation for multidimensional arrays, Linear Algebra Appl., 432 (2010), pp. 70–88.

Scientific Visualization, 2022, volume 14, number 3, pages 73 - 91, DOI: 10.26583/sv.14.3.06

On the Visualization of Multidimensional Functions using Canonical Decomposition

A.K. Alekseev[1,A,B], A.E. Bondarev [2,A], Yu.S. Pyatakova[3,B]

[A] Keldysh Institute of Applied Mathematics RAS
[B] Korolev Rocket and Space Corporation Energia, Korolev, Russia

[1] ORCID: 0000-0001-8317-8688, aleksey.k.alekseev@gmail.com
[2] ORCID: 0000-0003-3681-5212, bond@keldysh.ru
[3] ORCID: 0000-0002-8055-7807, yuliya.pyatakova@rsce.ru

Abstract

The approximation of a multidimensional function by means of tensor decompositions is considered in terms of storage, processing and visualization of the results of parametric calculations in computational aerogasdynamics problems. An algorithm for calculating the canonical decomposition using a combination of the alternative least squares method and stochastic gradient descent is described. Numerical results for interpolation of functions in six-dimensional space obtained using the canonical decomposition are presented, demonstrating the high computational efficiency and quality of results. Visual representations of the results are provided.

Keywords: tensor decomposition, canonical decomposition, computational fluid dynamics, visual representation of results.

References

1. W. Hackbusch. Tensor Spaces and Numerical Tensor Calculus. Springer, 2012.
2. H. Yorick, S. Willi-Hans, Matrix Calculus, Kronecker Product and Tensor Product: A Practical Approach to Linear Algebra, Multilinear Algebra and Tensor Calculus with Software Implementation, Singapore: World Scientific Publishing Co. Pte. Ltd., 2019.
3. Alekseev, A.K., Bondarev, A.E., Galaktionov, V.A., Kuvshinnikov A.E. Generalized Computational Experiment and Verification Problems. Program Comput Soft 47, 177–184 (2021). https://doi.org/10.1134/S0361768821030026
4. Farrell B.F. and A.M. Moore, An adjoint method for obtaining the most rapidly growing perturbation to oceanic flows, J. Phys. Oceanogr, 22 338-349, 1992
5. Alekseev A.K., Bondarev A.E., On the application of the dynamic mode decomposition in problems of computational fluid dynamics, KIAM Preprints. 2018, N 154. p. 30
6. A.K. Alekseev, D.A. Bistrian, A.E. Bondarev, I.M. Navon, On Linear and Nonlinear Aspects of Dynamic Mode Decomposition, Int. J. Numer. Meth. Fluids, 2016, V. 82, Issue 6, p. 348–371
7. A. M. P. Boelens, D. Venturi, D. M. Tartakovsky, Parallel tensor methods for high-dimensional linear PDEs, J. Computat. Phys. 375 (2018) 519-539.
8. Arnout M. P. Boelens, Daniele Venturi, Daniel M. Tartakovsky, Tensor methods for the Boltzmann-BGK equation, arXiv:1911.04904v2 2020
9. Oseledets I. V., Tensor- train decomposition, SIAM J. Sci. Comput. , 33 (2011), pp. 2295–2317
10. A.V. Chikitkin, E.K. Kornev, V.A. Titarev, Numerical solution of the Boltzmann equation with S-model collision integral using tensor decompositions, arXiv:1912.04582v1 2019
11. Kolda T. G. and Bader B. W., Tensor Decompositions and Applications, *SIAM Review, 51(3):455–500, 2009.*
12. Korenev G.V., Tensor calculus, M. MIPT, 1995

13. V. D. Silva and L -H. Lim, Tensor rank and the ill-posedness of the best low rank approximation problem, SIAM J. Matrix Anal. Appl., 30(3) 1084-1127, 2008.

14. L-H. Lim and Pierre Comon, Nonnegative approximations of nonnegative tensors, Chemometrics 2009; 23: 432–441.

15. P. Comon, X. Luciani, and A. L.F. De Almeida, Tensor decompositions, alternating least squares and other tales, Journal of Chemometrics, vol. 23, no. 7-8, pp. 393–405, 2009.

16. A. Uschmajew, Local convergence of the alternating least squares algorithm for canonical tensor approximation, SIAM J. Matrix Anal. Appl. 33 (2) (2012) 639–652.

17. C. Battaglino, G. Ballard, T.G. Kolda, A practical randomized CP tensor decomposition, arXiv:1701.06600, 2017.

18. X. Fu, S. Ibrahim, H.-T. Wai, C. Gao, and K. Huang, Block-randomized stochastic proximal gradient for low-rank tensor factorization, *IEEE Transactions on Signal Processing*, 2020.

19. Ioanna Siaminou, Athanasios P. Liavas, An Accelerated Stochastic Gradient for Canonical Polyadic Decomposition, arXiv:2109.13964v1 2021.

20. Guoxu Zhou, Andrzej Cichocki, and Shengli Xie, Accelerated Canonical Polyadic Decomposition by Using Mode Reduction, arXiv:1211.3500v2, 2013

21. Tikhonov, A.N., Arsenin, V.Y.: Solutions of Ill-Posed Problems. Winston and Sons, Washington DC (1977).

22. R. Ballester-Ripoll, E. G. Paredes, and R. Pajarola, A Surrogate Visualization Model Using the Tensor Train Format, SA '16: SIGGRAPH ASIA 2016 Symposium on Visualization November 2016 Article No. 13 Pages 1–8 https://doi.org/10.1145/3002151.3002167

23. Susanne K. Suter, Tensor Approximation in Visualization and Graphics: Background Theory, PhD thesis, University of Zurich, Switzerland, April 2013

24. Oseledets I., Tyrtyshnikov E., TT-cross approximation for multidimensional arrays, Linear Algebra Appl., 432 (2010), pp. 70–88.