ISSN 2305-9052 (Print)
ISSN 2410-7034 (Online)

ВЕСТНИК ЮЖНО-УРАЛЬСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА

BULLETIN OF THE SOUTH URAL STATE UNIVERSITY

СЕРИЯ

ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА И ИНФОРМАТИКА

2025, том 14, № 3

SERIES

COMPUTATIONAL MATHEMATICS AND SOFTWARE ENGINEERING

2025, volume 14, no. 3

1943

ВЕСТНИК

ЮЖНО-УРАЛЬСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА

2025
Т. 14, № 3

ISSN 2305-9052 (Print)
ISSN 2410-7034 (Online)

СЕРИЯ

«ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА И ИНФОРМАТИКА»

Решением ВАК включен в Перечень научных изданий, в которых должны быть опубликованы результаты диссертаций на соискание ученых степеней кандидата и доктора наук

Учредитель — Федеральное государственное автономное образовательное учреждение высшего образования «Южно-Уральский государственный университет (национальный исследовательский университет)»

Тематика журнала:

- Вычислительная математика и численные методы
- Математическое программирование
- Распознавание образов
- Вычислительные методы линейной алгебры
- Решение обратных и некорректно поставленных задач
- Доказательные вычисления
- Численное решение дифференциальных и интегральных уравнений
- Исследование операций
- Теория игр
- Теория аппроксимации
- Информатика
- Искусственный интеллект и машинное обучение
- Системное программирование
- Перспективные многопроцессорные архитектуры
- Облачные вычисления
- Технология программирования
- Машинная графика
- Интернет-технологии
- Системы электронного обучения
- Технологии обработки баз данных и знаний
- Интеллектуальный анализ данных

Редакционная коллегия
Л.Б. Соколинский, д.ф.-м.н., проф., *гл. редактор*
М.Л. Цымблер, д.ф.-м.н., доц., *зам. гл. редактора*
Я.А. Краева, к.ф.-м.н., *отв. секретарь*
А.И. Гоглачев, *техн. редактор*

Редакционный совет
С.М. Абдуллаев, д.г.н., профессор
А. Андреяк, PhD, профессор (Германия)
В.И. Бердышев, д.ф.-м.н., акад. РАН, *председатель*
В.В. Воеводин, д.ф.-м.н., чл.-кор. РАН
Дж. Донгарра, PhD, профессор (США)
С.В. Зыкин, д.т.н., профессор
И.М. Куликов, д.ф.-м.н.
Д. Маллманн, PhD, профессор (Германия)
А.В. Панюков, д.ф.-м.н., профессор
Р. Продан, PhD, профессор (Австрия)
Г.И. Радченко, к.ф.-м.н., доцент (Австрия)
В.Н. Ушаков, д.ф.-м.н., чл.-кор. РАН
М.Ю. Хачай, д.ф.-м.н., чл.-кор. РАН
А. Черных, PhD, профессор (Мексика)
П. Шумяцкий, PhD, профессор (Бразилия)

BULLETIN
OF THE SOUTH URAL STATE UNIVERSITY
SERIES
"COMPUTATIONAL MATHEMATICS AND SOFTWARE ENGINEERING"

2025
Vol. 14, no. 3

ISSN 2305-9052 (Print)
ISSN 2410-7034 (Online)

Vestnik Yuzhno-Ural'skogo Gosudarstvennogo Universiteta.
Seriya "Vychislitel'naya Matematika i Informatika"

South Ural State University

The scope of the journal:

- Numerical analysis and methods
- Mathematical optimization
- Pattern recognition
- Numerical methods of linear algebra
- Reverse and ill-posed problems solution
- Computer-assisted proofs
- Numerical solutions of differential and integral equations
- Operations research
- Game theory
- Approximation theory
- Computer science
- Artificial intelligence and machine learning
- System software
- Advanced multiprocessor architectures
- Cloud computing
- Software engineering
- Computer graphics
- Internet technologies
- E-learning
- Database processing
- Data mining

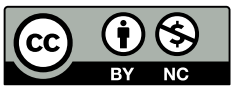
Editorial Board
L.B. Sokolinsky, South Ural State University (Chelyabinsk, Russia)
M.L. Zymbler, South Ural State University (Chelyabinsk, Russia)
Ya.A. Kraeva, South Ural State University (Chelyabinsk, Russia)
A.I. Goglachev, South Ural State University (Chelyabinsk, Russia)

Editorial Council
S.M. Abdullaev, South Ural State University (Chelyabinsk, Russia)
A. Andrzejak, Heidelberg University (Germany)
V.I. Berdyshev, Institute of Mathematics and Mechanics, Ural Branch of the RAS (Yekaterinburg, Russia)
J. Dongarra, University of Tennessee (USA)
M.Yu. Khachay, Institute of Mathematics and Mechanics, Ural Branch of the RAS (Yekaterinburg, Russia)
I.M. Kulikov, Institute of Computational Mathematics and Mathematical Geophysics, Siberian Branch of RAS (Novosibirsk, Russia)
D. Mallmann, Julich Supercomputing Centre (Germany)
A.V. Panyukov, South Ural State University (Chelyabinsk, Russia)
R. Prodan, Alpen-Adria-Universität Klagenfurt (Austria)
G.I. Radchenko, Silicon Austria Labs (Graz, Austria)
P. Shumyatsky, University of Brasilia (Brazil)
A. Tchernykh, CICESE Research Center (Mexico)
V.N. Ushakov, Institute of Mathematics and Mechanics, Ural Branch of the RAS (Yekaterinburg, Russia)
V.V. Voevodin, Lomonosov Moscow State University (Moscow, Russia)
S.V. Zykin, Sobolev Institute of Mathematics, Siberian Branch of the RAS (Omsk, Russia)

Содержание

© Издательский центр ЮУрГУ, 2025

Contents



This issue is distributed under the terms of the Creative Commons Attribution-Non Commercial 4.0 License which permits non-commercial use, reproduction and distribution of the work without further permission provided the original work is properly cited.

УДК 519.852, 514.172.45, 519.168 **DOI: 10.14529/cmse250301**

О ВЫЧИСЛЕНИИ ВЕРШИНЫ МНОГОГРАННИКА ДОПУСТИМЫХ РЕШЕНИЙ СИСТЕМЫ ЛИНЕЙНЫХ ОГРАНИЧЕНИЙ

© 2025 А.Э. Жулев, Л.Б. Соколинский, И.М. Соколинская

Южно-Уральский государственный университет

(454080 Челябинск, пр. им. В.И. Ленина, д. 76)

E-mail: zhulevae@susu.ru, leonid.sokolinsky@susu.ru, irina.sokolinskaya@susu.ru

Поступила в редакцию: 14.06.2025

В статье предлагается новый алгоритм вычисления вершины многогранника, представляющего собой область допустимых решений системы линейных ограничений. Алгоритм, получивший название VeSP, начинает работу в произвольной точке многогранника и, перемещаясь по его граням, завершает свою работу в некоторой его вершине. Для вычисления направления движения по грани используется проекционный метод, суть которого заключается в следующем. Для точки текущего приближения вычисляется аффинное подпространство, являющееся аффинной оболочкой грани, на которой расположена эта точка. К точке текущего приближения прибавляется ненулевой вектор, дающий «внешнюю» точку. Из «внешней» точки по известной аналитической формуле вычисляется ортогональная проекция на текущее аффинное подпространство. Точка проекции определяет направление движения по грани до ее границы, что дает следующее приближение. При каждом таком перемещении размерность текущей грани уменьшается. В конечном итоге приходим к грани нулевой размерности, то есть к вершине многогранника. Дается формальное описание алгоритма VeSP. Доказывается сходимость алгоритма VeSP к вершине многогранника за конечное число итераций, не превышающих размерность пространства. Приводится информация о реализации алгоритма VaSP на языке C++. Описываются результаты вычислительных экспериментов на эталонных задачах из репозитория Netlib-LP. Результаты экспериментов показывают, что алгоритм VeSP применительно ко всем тестовым задачам успешно нашел вершину многогранника за количество итераций, не превышающее размерность пространства. Для большинства задач время нахождения вершины на обычном персональном компьютере потребовало менее одной секунды.

Ключевые слова: линейные ограничения, многогранник допустимых решений, вычисление вершины, проекционный метод, алгоритм VeSP.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Жулев А.Э., Соколинский Л.Б., Соколинская И.М. О вычислении вершины многогранника допустимых решений системы линейных ограничений // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2025. Т. 14, № 3. С. 5–27. DOI: 10.14529/cmse250301.

Введение

Задача нахождения вершины многогранника, задаваемого системой линейных неравенств и уравнений, встречается во всех областях, использующих линейное программирование (ЛП) в качестве основной оптимизационной модели. Так, симплекс-метод [1] для своей работы требует некоторую стартовую вершину многогранника допустимых решений. Проекционный алгоритм AlEM решения задачи ЛП, предложенный в недавней работе [2], также требует в качестве стартовой точки вершину многогранника допустимых решений. Стартовая вершина также необходима в алгоритмах перечисления всех вершин многогранника (см., например, [3, 4]), используемых в таких областях, как теория игр, исследование операций, математическая биология и кристаллография.

Рассмотрим систему линейных ограничений стандартного вида[1]

$$A\boldsymbol{x} \leqslant \boldsymbol{b}, \tag{1}$$

где $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$ — вектор переменных, A — вещественная матрица размера $m \times n$, $\boldsymbol{b}$ — столбец вещественных чисел размера m. Необходимо найти вершину многогранника $M \subset \mathbb{R}^n$, представляющего собой область допустимых решений системы (1). Предполагается, что многогранник M является ограниченным множеством. Наиболее известными методами вычисления вершины многогранника M являются симплекс-метод и метод исключения переменных Фурье—Моцкина.

Идея подхода, основанного на симплекс-методе, заключается в следующем [5]. Переупорядочив строки системы (1), представим ее в виде

$$\begin{cases} A^+\boldsymbol{x} \leqslant \boldsymbol{b}^+ \\ A^-\boldsymbol{x} \geqslant \boldsymbol{b}^-, \end{cases}$$

где $\boldsymbol{b}^+ \geqslant 0$, $\boldsymbol{b}^- > 0$. Рассмотрим задачу ЛП следующего вида[2]:

$$\max\{\langle \mathbf{1}, A^+\boldsymbol{x} - \boldsymbol{z}\rangle \mid \boldsymbol{z} \geqslant 0;\ A^+\boldsymbol{x} \leqslant \boldsymbol{b}^+;\ A^-\boldsymbol{x} - \boldsymbol{z} \leqslant \boldsymbol{b}^-\}, \tag{2}$$

где $\boldsymbol{z} = (z_1, \ldots, z_n) \in \mathbb{R}^n$ — новый вектор переменных, и $\mathbf{1} = (1, \ldots 1) \in \mathbb{R}^n$ — вектор единиц. В этом случае $\boldsymbol{x} = \mathbf{0}$ и $\boldsymbol{z} = \mathbf{0}$ задают вершину многогранника допустимых решений задачи (2). Таким образом, мы можем решить задачу (2) с помощью симплекс-метода. Если максимальное значение задачи (2) равно $\langle \mathbf{1}, \boldsymbol{b}^-\rangle$, и достигается в вершине $(\boldsymbol{x}^*, \boldsymbol{y}^*)$, то $\boldsymbol{x}^*$ будет вершиной многогранника M, представляющего область допустимых решений системы (1). Если максимальное значение задачи (2) меньше $\langle \mathbf{1}, \boldsymbol{b}^-\rangle$, то система (1) не имеет решений. Главным недостатком этого подхода является то, симплекс-метод в общем случае может потребовать экспоненциальное число шагов для получения решения [6].

Метод Фурье—Моцкина [7] основан на решении системы линейных неравенств путем последовательного исключения переменных подобно тому, как это делается при решении системы линейных уравнений. Впервые метод исключения переменных применительно к линейным неравенствам был упомянут Фурье в 1827 году (см. [8]). Моцкин повторно открыл и использовал этот метод в своей докторской диссертации [9]. Следуя [5], рассмотрим метод Фурье—Моцкина применительно к задаче вычисления вершины многогранника допустимых решений системы (1). Обозначим a_i^j — j-тый элемент i-той строки матрицы A. Путем умножения неравенств на соответствующие положительные числа, приведем систему (1) к виду

$$\begin{cases} x_1 + \langle \boldsymbol{a}_i^+, \boldsymbol{x}'\rangle \leqslant b_i^+ & (i = 1, \ldots, m') \\ -x_1 + \langle \boldsymbol{a}_i^-, \boldsymbol{x}'\rangle \leqslant b_i^- & (i = m' + 1, \ldots, m'') \\ \langle \boldsymbol{a}_i^0, \boldsymbol{x}'\rangle \leqslant b_i & (i = m'' + 1, \ldots, m), \end{cases} \tag{3}$$

[1]Заметим, что система линейных ограничений, включающая равенства, может быть легко приведена к стандартному виду путем замены каждого равенства парой противоположных нестрогих неравенств. В свою очередь неравенство «больше, либо равно» может быть преобразовано в неравенство «меньше, либо равно» путем умножения обеих частей на -1. Мы здесь исключаем системы линейных ограничений, включающие строгие неравенства, так как это влечет открытость области допустимых решений.

[2]Здесь $\langle \cdot\ ,\ \cdot \rangle$ обозначает скалярное произведение векторов.

где $\boldsymbol{a}_i^+ = \left(a_i^2/a_i^1, \ldots, a_i^n/a_i^1\right)$, $\boldsymbol{a}_i^- = \left(-a_i^2/a_i^1, \ldots, -a_i^n/a_i^1\right)$, $\boldsymbol{a}_i^0 = \left(a_i^2, \ldots, a_i^n\right)$, $b_i^+ = b_i/a_i^1$, $b_i^- = -b_i/a_i^1$, $\boldsymbol{x}' = (x_2, \ldots, x_n)$. Поскольку первые две строки системы (3) равносильны условию

$$\max_{i\in\{m'+1,\ldots,m''\}} \left(\langle \boldsymbol{a}_i^-, \boldsymbol{x}'\rangle - b_i^-\right) \leqslant x_1 \leqslant \min_{i\in\{1,\ldots,m'\}} \left(b_i^+ - \langle \boldsymbol{a}_i^+, \boldsymbol{x}'\rangle\right), \tag{4}$$

переменная x_1 может быть исключена. Это позволяет перейти к следующей эквивалентной системе:

$$\begin{cases} \langle \boldsymbol{a}_i^-, \boldsymbol{x}'\rangle - b_i^- \leqslant b_1^+ - \langle \boldsymbol{a}_1^+, \boldsymbol{x}'\rangle & (i = m'+1, \ldots, m'') \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots & \ldots\ldots\ldots\ldots\ldots \\ \langle \boldsymbol{a}_i^-, \boldsymbol{x}'\rangle - b_i^- \leqslant b_{m'}^+ - \langle \boldsymbol{a}_{m'}^+, \boldsymbol{x}'\rangle & (i = m'+1, \ldots, m'') \\ \langle \boldsymbol{a}_i^0, \boldsymbol{x}'\rangle \leqslant b_i & (i = m''+1, \ldots, m), \end{cases}$$

которая равносильна системе

$$\begin{cases} \langle \boldsymbol{a}_i^- + \boldsymbol{a}_1^+, \boldsymbol{x}'\rangle \leqslant b_1^+ + b_i^- & (i = m'+1, \ldots, m'') \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots & \ldots\ldots\ldots\ldots\ldots \\ \langle \boldsymbol{a}_i^- + \boldsymbol{a}_{m'}^+, \boldsymbol{x}'\rangle \leqslant b_{m'}^+ + b_i^- & (i = m'+1, \ldots, m'') \\ \langle \boldsymbol{a}_i^0, \boldsymbol{x}'\rangle \leqslant b_i & (i = m''+1, \ldots, m). \end{cases} \tag{5}$$

Эта система включает в себя $m'(m'' - m') + m - m''$ ограничений и $n - 1$ переменных. В действительности, система (5) определяет проекцию многогранника допустимых решений M исходной системы (1) на гиперплоскость $x_1 = 0$. Проекция многогранника является многогранником. Обозначим получившийся многогранник через M_1. Если размерность многогранника M равна n, то размерность многогранника M_1 будет равна $n - 1$:

$$\dim(M_1) = n - 1.$$

Продолжая процесс исключения переменных выше описанным способом, в итоге получим многогранник M_{n-1} размерности 1, являющийся проекцией многогранника M_{n-2} на гиперплоскость $x_n = 0$ и представляющий собой отрезок. Очевидно, что в любой точке этого отрезка переменные $x_1, \ldots, x_{n-1}$ принимают значение 0, а значения переменной x_n ограничиваются условием

$$\beta^- \leqslant x_n \leqslant \beta^+,$$

где β^-, β^+ — скалярные величины, вычисленные в ходе исключения переменных. При этом вершинам многогранника M_{n-1} соответствуют минимальное и максимальное допустимые значения переменной x_n. Для определенности положим $x_n^* = \beta^+$ — n-ная координата вершины многогранника. Двигаясь в обратном порядке, вычислим остальные координаты искомой вершины, используя ограничения вида (4). Например, если мы уже нашли координаты $x_2^*, \ldots, x_n^*$, подставив их в правую часть (4) в качестве $\boldsymbol{x}'$, получим

$$x_1^* = \min_{i\in\{1,\ldots,m'\}} \left(b_i^+ - \langle \boldsymbol{a}_i^+, (x_2^*, \ldots, x_n^*)\rangle\right).$$

Таким образом мы вычислим вершину $\boldsymbol{x}^* = (x_1^*, \ldots, x_n^*)$ многогранника допустимых решений M для исходной системы ограничений (1). Описанный метод имеет тот же недостаток, что и симплекс-метод: в общем случае метод исключения Фурье—Моцкина характеризуется экспоненциальной временной сложностью [10].

В этой статье мы предлагаем новый метод вычисления вершины многогранника допустимых решений системы линейных неравенств, который позволяет вычислить вершину не более, чем за n итераций, где n — размерность пространства. Проекционный алгоритм, реализующий этот метод, получил название VeSP (Vertex Seeking by Projecting). Статья организована следующим образом. Раздел 1 содержит обозначения, определения и утверждения, необходимые для описания алгоритма VeSP и его численной реализации. В разделе 2 дается формализованное описание алгоритма VeSP, и доказывается утверждение о сходимости этого алгоритма. В разделе 3 представлены информация о программной реализации алгоритма VeSP и результаты вычислительных экспериментов. В заключении суммируются полученные результаты и намечаются направления дальнейших исследований. В конце статьи приводится сводка используемых математических обозначений.

1. Определения и обозначения

В этом разделе приводятся основные определения и обозначения, используемые для описания алгоритма VeSP.

В евклидовом пространстве $\mathbb{R}^n$ рассматривается система линейных ограничений общего вида, включающая в себя m линейных неравенств и k линейных уравнений:

$$\begin{cases} \hat{A}\boldsymbol{x} \leqslant \hat{\boldsymbol{b}}; \\ \bar{A}\boldsymbol{x} = \bar{\boldsymbol{b}}, \end{cases} \tag{6}$$

где $\hat{A} \in \mathbb{R}^{m\times n}$, $\bar{A} \in \mathbb{R}^{k\times n}$, $\hat{\boldsymbol{b}} \in \mathbb{R}^m$ и $\bar{\boldsymbol{b}} \in \mathbb{R}^k$. Предполагается, что размерность пространства $n > 1$, количество уравнений $k \geqslant 0$. Матрицу $\hat{A}$ будем называть граничной, а матрицу $\bar{A}$ — опорной. Предполагается, что система (6) включает в себя также неравенства вида

$$\begin{aligned} -x_1 &\leqslant 0; \\ &\cdots\cdots\cdots \\ -x_n &\leqslant 0. \end{aligned} \tag{7}$$

Таким образом, число неравенств m не может быть меньше размерности пространства n:

$$m \geqslant n. \tag{8}$$

Из (7) и (8) следует

$$\operatorname{rank}\left(\hat{A}\right) = n. \tag{9}$$

Подсистема

$$\hat{A}\boldsymbol{x} \leqslant \hat{\boldsymbol{b}}$$

задает m замкнутых полупространств[3]

$$\begin{aligned} P_1 &= \{\boldsymbol{x} \in \mathbb{R}^n | \langle \boldsymbol{a}_1, \boldsymbol{x}\rangle \leqslant b_1\}; \\ &\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ P_m &= \{\boldsymbol{x} \in \mathbb{R}^n | \langle \boldsymbol{a}_m, \boldsymbol{x}\rangle \leqslant b_m\}. \end{aligned} \tag{10}$$

[3]Здесь и далее мы опускаем прилагательное «аффинных» в отношении полупространств, гиперплоскостей и подпространств.

Здесь $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_m$ обозначают строки матрицы $\hat{A}$; $b_1, \ldots, b_m$ — элементы столбца $\hat{\boldsymbol{b}}$. Пересечение полупространств (10) образует замкнутый выпуклый многогранник $\hat{M}$, называемый граничным:

$$\hat{M} = \bigcap_{i=1}^{m} P_i. \tag{11}$$

Полупространства (10) ограничиваются гиперплоскостями, называемыми граничными:

$$\begin{gathered} \hat{H}_1 = \{\boldsymbol{x} \in \mathbb{R}^n | \langle \boldsymbol{a}_1, \boldsymbol{x} \rangle = b_1\}; \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \hat{H}_m = \{\boldsymbol{x} \in \mathbb{R}^n | \langle \boldsymbol{a}_m, \boldsymbol{x} \rangle = b_m\}. \end{gathered} \tag{12}$$

Без ограничения общности мы можем предполагать, что среди граничных гиперплоскостей нет совпадающих.

Подсистема

$$\bar{A}\boldsymbol{x} = \bar{\boldsymbol{b}}$$

задает k гиперплоскостей, называемых опорными:

$$\begin{gathered} \bar{H}_{m+1} = \{\boldsymbol{x} \in \mathbb{R}^n | \langle \boldsymbol{a}_{m+1}, \boldsymbol{x} \rangle = b_{m+1}\}; \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \bar{H}_{m+k} = \{\boldsymbol{x} \in \mathbb{R}^n | \langle \boldsymbol{a}_{m+k}, \boldsymbol{x} \rangle = b_{m+k}\}. \end{gathered} \tag{13}$$

Здесь $\boldsymbol{a}_{m+1}, \ldots, \boldsymbol{a}_{m+k}$ обозначают строки матрицы $\bar{A}$; $b_{m+1}, \ldots, b_{m+k}$ — элементы столбца $\bar{\boldsymbol{b}}$. Пересечение опорных гиперплоскостей образует опорное подпространство $\bar{S}$:

$$\bar{S} = \begin{cases} \bigcap\limits_{i=1}^{k} \bar{H}_{m+i}, & \text{если} \quad k > 0; \\ \mathbb{R}^n, & \text{если} \quad k = 0. \end{cases} \tag{14}$$

Область допустимых решений, определяемая системой ограничений (6), представляет собой замкнутый выпуклый многогранник M, называемый допустимым:

$$M = \bar{S} \cap \hat{M}. \tag{15}$$

В частности, из (15) следуют

$$M \subseteq \bar{S}; \tag{16}$$

$$M \subseteq \hat{M}. \tag{17}$$

Мы будем предполагать, что допустимый многогранник M является непустым ограниченным множеством.

Обозначим через $\hat{I}$ множество индексов строк граничной матрицы $\hat{A}$, и через $\bar{I}$ — множество индексов строк опорной матрицы $\bar{A}$:

$$\hat{I} = \{1, \ldots, m\}; \tag{18}$$

$$\bar{I} = \{m+1, \ldots, m+k\}. \tag{19}$$

В соответствии с (19) формулу (14) можно переписать в виде

$$\bar{S} = \begin{cases} \bigcap\limits_{i \in \bar{I}} \bar{H}_i\,, & \text{если} \quad \bar{I} \neq \emptyset; \\ \mathbb{R}^n\,, & \text{если} \quad \bar{I} = \emptyset. \end{cases} \tag{20}$$

Определение 1. Систему ограничений (6) будем называть регулярной, если выполняются следующие условия:

$$k < n; \tag{21}$$

$$\operatorname{rank}\left(\bar{A}\right) = k; \tag{22}$$

$$\dim(M) = n - k; \tag{23}$$

$$\forall \boldsymbol{v} \in M \;:\; \operatorname{rank}\left(\begin{bmatrix} \bar{A} \\ \hat{A}_{\boldsymbol{v}} \end{bmatrix}\right) = \min(k + l, n), \tag{24}$$

где l — количество граничных гиперплоскостей, проходящих через точку $\boldsymbol{v}$, $\hat{A}_{\boldsymbol{v}}$ — матрица размера $l \times n$, строками которой являются коэффициенты уравнений граничных гиперплоскостей, проходящих через точку $\boldsymbol{v}$.

Условие (21) означает, что количество уравнений в системе ограничений должно быть меньше размерности пространства (в противном случае допустимый многогранник M вырождается в точку). Условие (22) говорит, что опорная матрица $\bar{A}$ должна иметь полный ранг, то есть среди ограничений не должно быть уравнений, задающих различные параллельные гиперплоскости. Условие (23) требует, чтобы допустимый многогранник M имел размерность, равную $n - k$. Это означает, что в системе ограничений не может быть неявных равенств[4]. Согласно условию (24) в системе ограничений не должно быть избыточных равенств, и при этом ранг матрицы коэффициентов неравенств, содержащих граничную точку, должен быть равным минимуму от числа неравенств и размерности пространства. Заметим, что для любой системы линейных ограничений, областью допустимых решений которой является ограниченное множество размерности больше 0, существует ее эквивалентное представление с регулярной системой ограничений в пространстве той же размерности (см. [11], теорема 2.15).

Везде далее будем предполагать, что система ограничений (6) является регулярной. Это обеспечивает корректность всех последующих утверждений и алгоритмов. Следующее утверждение является теоретической основой для построения алгоритма VeSP.

Утверждение 1. Пусть выполняются условия (21)–(24). Пусть через точку $\boldsymbol{v} \in M$ проходит l граничных гиперплоскостей. Обозначим через $\hat{I}_{\boldsymbol{v}}$ множество индексов этих граничных гиперплоскостей:

$$\boldsymbol{v} \in \bigcap_{i \in \hat{I}_{\boldsymbol{v}}} \hat{H}_i. \tag{25}$$

[4]Неравенство $\langle \boldsymbol{a}, \boldsymbol{x} \rangle \leqslant b$ из $A\boldsymbol{x} \leqslant \boldsymbol{b}$ является неявным равенством, если $\langle \boldsymbol{a}, \boldsymbol{x} \rangle = b$ для всех $\boldsymbol{x}$, удовлетворяющих $A\boldsymbol{x} \leqslant \boldsymbol{b}$.

Положим

$$S_{\boldsymbol{v}} = \begin{cases} \left(\bigcap\limits_{i \in \hat{I}_v} \hat{H}_i \right) \cap \bar{S}\,, & \text{если} \quad \hat{I}_{\boldsymbol{v}} \neq \emptyset; \\ \mathbb{R}^n\,, & \text{если} \quad \hat{I}_{\boldsymbol{v}} = \emptyset; \end{cases} \tag{26}$$

$$F = M \cap S_{\boldsymbol{v}}. \tag{27}$$

Тогда множество F является гранью допустимого многогранника M и

$$\dim(F) = n - \min(k + l, n). \tag{28}$$

Доказательство. Сначала предположим, что $l = 0$, то есть через точку $\boldsymbol{v}$ не проходит ни одной граничной гиперплоскости. Следовательно, $\hat{I}_{\boldsymbol{v}} = \emptyset$. Тогда в силу (20) и (26) имеем

$$S_{\boldsymbol{v}} = \bar{S}.$$

С учетом (16) отсюда следует

$$M \subseteq S_{\boldsymbol{v}}.$$

Принимая во внимание (27), таким образом получаем

$$F = M. \tag{29}$$

Сам многогранник M является своей гранью [11]. Так как система ограничений (6) по предположению является регулярной, из (23) и (29) следует $\dim(F) = n - k$. Таким образом, для $l = 0$ утверждение доказано.

Пусть теперь $l > 0$, то есть $\hat{I}_{\boldsymbol{v}} \neq \emptyset$. В силу (11), (17) и (25) для любого $i \in \hat{I}_{\boldsymbol{v}}$ имеем

$$M \subseteq P_i;$$
$$M \cap \hat{H}_i \neq \emptyset.$$

Это — достаточные условия для того, чтобы множество $F_i = M \cap \hat{H}_i$ являлось гранью выпуклого многогранника M (см. [11], определение 2.1). Пересечение граней многогранника M — снова грань многогранника M (см. [11], утверждение 2.3). Поэтому множество

$$G = M \cap \left(\bigcap_{i \in \hat{I}_v} \hat{H}_i \right) \tag{30}$$

является гранью многогранника M. Покажем, что $F = G$. Сначала предположим, что $\bar{I} = \emptyset$. Тогда из (26) следует

$$S_{\boldsymbol{v}} = \bigcap_{i \in \hat{I}_v} \hat{H}_i.$$

С учетом (27) и (30) отсюда получаем

$$F = M \cap S_{\boldsymbol{v}} = M \cap \left(\bigcap_{i \in \hat{I}_v} \hat{H}_i \right) = M \cap S_{\boldsymbol{v}} = G,$$

то есть F является гранью допустимого многогранника M. Пусть теперь $\bar{I} \neq \emptyset$. Тогда с учетом (16), (20), (26) и (30) имеем

$$G = M \cap \left(\bigcap_{i \in \hat{I}_{\boldsymbol{v}}} \hat{H}_i \right) = M \cap \left(\bigcap_{i \in \bar{I}} \bar{H}_i \right) \cap \left(\bigcap_{i \in \hat{I}_{\boldsymbol{v}}} \hat{H}_i \right) = M \cap \left(\bigcap_{i \in \hat{I}_{\boldsymbol{v}}} \hat{H}_i \right) \cap \bar{S} = M \cap S_{\boldsymbol{v}} = F,$$

то есть и в этом случае F является гранью допустимого многогранника M. Вычислим размерность F для случая $l > 0$. Для этого заметим, что из (22) следует

$$\dim(\bar{S}) = n - k.$$

В силу (16) и (23) отсюда получаем, что подпространство $\bar{S}$ является аффинной оболочкой допустимого многогранника M:

$$\operatorname{aff}(M) = \bar{S}. \tag{31}$$

Из (20) и (26) следует

$$\bar{S} \subseteq S_{\boldsymbol{v}}. \tag{32}$$

Обозначим через $\hat{A}_{\boldsymbol{v}}$ матрицу размера $l \times n$, содержащую коэффициенты граничных уравнений с индексами из $\hat{I}_{\boldsymbol{v}}$. Так как система ограничений (6) по предположению является регулярной, то в соответствии с определением 1 выполняется равенство

$$\operatorname{rank}\left(\begin{bmatrix} \bar{A} \\ \hat{A}_{\boldsymbol{v}} \end{bmatrix} \right) = \min(k + l, n).$$

Принимая во внимание (26), отсюда следует

$$\dim(S_{\boldsymbol{v}}) = n - \min(k + l, n). \tag{33}$$

Учитывая, что аффинная оболочка подпространства есть само это подпространство, и принимая во внимание (27), (31), (32) и (33), имеем

$$\dim(F) = \dim(\operatorname{aff}(F)) = \dim(\operatorname{aff}(M) \cap S_{\boldsymbol{v}}) = \dim(\bar{S} \cap S_{\boldsymbol{v}}) = \dim(S_{\boldsymbol{v}}) = n - \min(k + l, n).$$

Утверждение доказано. □

Определение 2. Собственной гранью точки $\boldsymbol{v} \in M$ будем называть грань наименьшей размерности, содержащую точку $\boldsymbol{v}$.

Утверждение 2. Точка $\boldsymbol{v} \in M$ имеет только одну собственную грань.

Доказательство. Предположим, имеются две различные грани F' и F'' минимальной размерности, содержащие точку $\boldsymbol{v}$. Поскольку пересечение граней допустимого многогранника M является гранью допустимого многогранника M, то $F' \cap F''$ будет гранью допустимого многогранника M, имеющей размерность меньше минимальной, и содержащей точку $\boldsymbol{v}$. Пришли к противоречию. □

Обозначим через face($\boldsymbol{v}$) собственную грань точки $\boldsymbol{v} \in M$.

Утверждение 3. Пусть $\boldsymbol{v} \in M$. Тогда собственная грань точки $\boldsymbol{v}$ может быть вычислена по формуле

$$\text{face}(\boldsymbol{v}) = \begin{cases} M \cap \left(\bigcap\limits_{i \in \hat{I}_{\boldsymbol{v}}} \hat{H}_i \right) \cap \bar{S}\,, & \text{если} \quad \hat{I}_{\boldsymbol{v}} \neq \emptyset; \\ M\,, & \text{если} \quad \hat{I}_{\boldsymbol{v}} = \emptyset, \end{cases} \tag{34}$$

где $\hat{I}_{\boldsymbol{v}}$ — множество индексов всех граничных гиперплоскостей, содержащих точку $\boldsymbol{v}$.

Доказательство. Положим

$$F = \begin{cases} M \cap \left(\bigcap\limits_{i \in \hat{I}_{\boldsymbol{v}}} \hat{H}_i \right) \cap \bar{S}\,, & \text{если} \quad \hat{I}_{\boldsymbol{v}} \neq \emptyset; \\ M\,, & \text{если} \quad \hat{I}_{\boldsymbol{v}} = \emptyset, \end{cases}$$

где $\hat{I}_{\boldsymbol{v}}$ — множество индексов всех граничных гиперплоскостей, содержащих точку $\boldsymbol{v}$. Множество F является гранью допустимого многогранника M в соответствии с утверждением 1. Грань F имеет минимальную размерность среди всех граней, содержащих точку $\boldsymbol{v}$, так как она является подмножеством пересечения всех граничных гиперплоскостей, проходящих через $\boldsymbol{v}$. □

Следствие 1. Пусть $\boldsymbol{v} \in M$. Тогда

$$\dim(\text{face}(\boldsymbol{v})) = n - \min(k + l, n), \tag{35}$$

где l — количество граничных гиперплоскостей, проходящих через точку $\boldsymbol{v}$.

Доказательство. Следует непосредственно из утверждений 1 и 3. □

Следствие 2. Пусть через точку $\boldsymbol{v} \in M$ проходят $n-k$ или более граничных гиперплоскостей. Тогда $\dim(\text{face}(\boldsymbol{v})) = 0$, то есть точка $\boldsymbol{v}$ является вершиной допустимого многогранника M.

Следствие 3. Пусть через точку $\boldsymbol{v} \in M$ проходят $n - k - 1$ граничных гиперплоскостей. Тогда $\dim(\text{face}(\boldsymbol{v})) = 1$, то есть грань face($\boldsymbol{v}$) является ребром допустимого многогранника M.

Следствие 4. Пусть через точку $\boldsymbol{v} \in M$ проходит 0 граничных гиперплоскостей. Тогда $\dim(\text{face}(\boldsymbol{v})) = n - k$ и face($\boldsymbol{v}$) $= M$, то есть сам допустимый многогранник M является собственной гранью точки $\boldsymbol{v}$.

Доказательство. Следует непосредственно из формул (23), (35) и утверждения (2). □

2. Формализованное описание алгоритма VeSP

Этот раздел посвящен формализованному описанию алгоритма VeSP, вычисляющего вершину допустимого многогранника. Алгоритм VeSP начинает свою работу в произвольной точке $\boldsymbol{v}_0$, принадлежащей некоторой грани F_0 допустимого многогранника M. Обозначим через l_0 размерность этой грани. Двигаясь по прямой в произвольном направлении по грани F_0, алгоритм VeSP достигает ее границы в некоторой точке $\boldsymbol{v}_1$. Эта точка будет

находиться на грани F_1, размерность которой l_1 будет меньше l_0. Продолжая движение по граням допустимого многогранника описанным образом, алгоритм VeSP за конечное число шагов придет к точке $\boldsymbol{v}_k$, лежащей на грани F_k, имеющей размерность 0. Точка $\boldsymbol{v}_k$ и будет искомой вершиной допустимого многогранника.

Основной функцией, используемой алгоритмом VeSP, является функция MoveAlongFace$(\cdot)$, выполняющая прохождение грани допустимого многогранника. Эта функция, в свою очередь, использует следующие две функции: $\boldsymbol{\pi}_J(\cdot)$ — вычисление ортогональной проекции на подпространство и Jump$(\cdot)$ — перемещение по направляющему вектору.

Рассмотрим сначала функцию вычисления ортогональной проекции $\boldsymbol{\pi}_J(\boldsymbol{x})$ точки $\boldsymbol{x}$ на подпространство

$$S_J = \bigcap_{i \in J} H_i \quad (J \subseteq \hat{I} \cup \bar{I}), \tag{36}$$

При построении ортогональной проекции мы не различаем граничные и опорные гиперплоскости. Мы также предполагаем, что $S_J \neq \emptyset$. Обозначим через A_J матрицу коэффициентов уравнений с индексами из множества J, через $\boldsymbol{b}_J$ — столбец правых частей этих уравнений. В случае, когда система уравнений $A_J\boldsymbol{x} = \boldsymbol{b}_J$ является совместной, и матрица $A_J A_J^T$ является невырожденной, ортогональная проекция точки $\boldsymbol{x}$ на подпространство S_J может быть вычислена по следующей формуле [12]:

$$\boldsymbol{\pi}_J(\boldsymbol{x}) = \boldsymbol{x} - A_J^T \left(A_J A_J^T\right)^{-1} (A_J\boldsymbol{x} - \boldsymbol{b}_J)\,. \tag{37}$$

Для того, чтобы гарантировать невырожденность матрицы $A_J A_J^T$ в общем случае, необходимо из множества J удалить индексы всех избыточных уравнений[5]. Получившееся множество J' будет обладать следующими свойствами:

$$\begin{aligned} &S_J = S_{J'};\\ &\operatorname{rank}(A_J) = \operatorname{rank}(A_{J'});\\ &\det\left(A_{J'} A_{J'}^T\right) \neq 0. \end{aligned}$$

Если, тем не менее, не удается построить обратную матрицу к матрице $A_{J'} A_{J'}^T$, например, в следствии потери точности, то в этом случае можно приближенно вычислить ортогональную проекцию точки на подпространство, используя алгоритм Качмажа [13, 14].

Далее рассмотрим функцию Jump$(\boldsymbol{x}, \boldsymbol{d})$, осуществляющую перемещение по направляющему вектору $\boldsymbol{d}$ от допустимой точки $\boldsymbol{x}$ к допустимой точке, максимально удаленной от $\boldsymbol{x}$. Основной элементарной операцией, используемой для перемещения, является d-проекция $\boldsymbol{\gamma}_i^{\boldsymbol{d}}(\boldsymbol{x})$ точки $\boldsymbol{x}$ на граничную гиперплоскость $\hat{H}_i$ по направлению вектора $\boldsymbol{d}$, вычисляемая по формуле

$$\boldsymbol{\gamma}_i^{\boldsymbol{d}}(\boldsymbol{x}) = \begin{cases} \boldsymbol{x} - \frac{\langle \boldsymbol{a}_i, \boldsymbol{x}\rangle - b_i}{\langle \boldsymbol{a}_i, \boldsymbol{d}\rangle}\boldsymbol{d}, & \text{если} \quad \langle \boldsymbol{a}_i, \boldsymbol{d}\rangle \neq 0;\\ \infty, & \text{если} \quad \langle \boldsymbol{a}_i, \boldsymbol{d}\rangle = 0 \end{cases} \tag{38}$$

(см. определение 1 и утверждение 2 в [15]). Реализация функции Jump$(\boldsymbol{x}, \boldsymbol{d})$ представлена в виде алгоритма 1.

[5]Уравнение с индексом $j \in J$ является избыточным, если $\operatorname{rank}(A_J) = \operatorname{rank}(A_{J-\{j\}})$.

Алгоритм 1 Реализация функции Jump($\boldsymbol{x}, \boldsymbol{d}$)

Require: $\boldsymbol{x} \in M$; $\forall i \in \bar{I} : \boldsymbol{x} + \boldsymbol{d} \in \bar{H}_i$

```
1:  function Jump(x, d)
2:      x_min := x
3:      for i ∈ Î do
4:          if ⟨a_i, d⟩ > 0 then
5:              x_γ := γ_i^d(x)
6:              if ‖x_γ − x‖ < ‖x_min − x‖ then
7:                  x_min := x_γ
8:              end if
9:          end if
10:     end for
11:     return x_min
12: end function
```

Для работы алгоритма необходимо, чтобы точка $\boldsymbol{x}$ принадлежала допустимому многограннику M, и выполнялось условие

$$\forall i \in \bar{I} : \boldsymbol{x} + \boldsymbol{d} \in \bar{H}_i, \tag{39}$$

которое означает, что точка, получающаяся путем прибавления вектора $\boldsymbol{d}$ к точке $\boldsymbol{x}$, должна принадлежать всем опорным гиперплоскостям (13). Другими словами, точка $(\boldsymbol{x}+\boldsymbol{d})$ обязана удовлетворять всем уравнениям из системы ограничений (6). Схема работы алгоритма 1 проиллюстрирована на рис. 1.

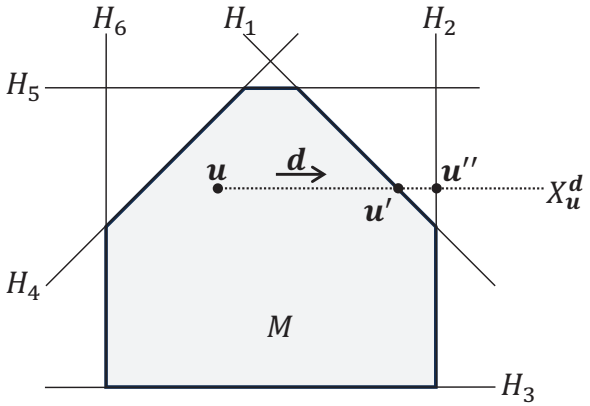


Рис. 1. Действие функции Jump: $\mathrm{Jump}(\boldsymbol{u}, \boldsymbol{d}) = \boldsymbol{u}'$.

В приведенном примере изображен многогранник M, образованный пересечением шести полупространств

$$P_1 = \{\boldsymbol{x} \in \mathbb{R}^n | \langle \boldsymbol{a}_1, \boldsymbol{x} \rangle \leqslant b_1\};$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$P_6 = \{\boldsymbol{x} \in \mathbb{R}^n | \langle \boldsymbol{a}_6, \boldsymbol{x} \rangle \leqslant b_6\}.$$

Эти полупространства ограниченны гиперплоскостями

$$H_1 = \{\boldsymbol{x} \in \mathbb{R}^n | \langle \boldsymbol{a}_1, \boldsymbol{x} \rangle = b_1\};$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$H_6 = \{\boldsymbol{x} \in \mathbb{R}^n | \langle \boldsymbol{a}_6, \boldsymbol{x} \rangle = b_6\}.$$

Заданы точка $\boldsymbol{u} \in M$ и направляющий вектор $\boldsymbol{d}$. Необходимо перенести точку $\boldsymbol{u}$ по направлению вектора $\boldsymbol{d}$ в максимально удаленную точку $\boldsymbol{u}' \in M$. Известно [16], что луч

$$X_{\boldsymbol{u}}^{\boldsymbol{d}} = \{\boldsymbol{x} \in \mathbb{R}^n | \, \boldsymbol{x} = \boldsymbol{u} + \lambda \boldsymbol{d}, \, \lambda \geqslant 0\}$$

пересечет гиперплоскость H_i $(i = 1, \ldots, 6)$ только в том случае, когда

$$\langle \boldsymbol{a}_i, \boldsymbol{d} \rangle > 0. \tag{40}$$

Проверка условия (40) на шаге 4 алгоритма 1 в этом случае покажет, что луч $X_{\boldsymbol{u}}^{\boldsymbol{d}}$ пересечет только гиперплоскости H_1 и H_2. Шаги 5–8 алгоритма 1 вычислят d-проекции $\boldsymbol{\gamma}_1^{\boldsymbol{d}}(\boldsymbol{u})$ и $\boldsymbol{\gamma}_2^{\boldsymbol{d}}(\boldsymbol{u})$ на гиперплоскости H_1, H_2 и выберут ту из них, которая находится ближе всего к точке $\boldsymbol{u}$. Это будет $\boldsymbol{u}' = \boldsymbol{\gamma}_1^{\boldsymbol{d}}(\boldsymbol{u})$.

Реализация функции MoveAlongFace($\boldsymbol{v}$), осуществляющей прохождение грани допустимого многогранника, представлена в виде алгоритма 2. Прокомментируем этот алгоритм. Начальная точка $\boldsymbol{v}$ должна принадлежать допустимому многограннику M и не являться его вершиной. На шагах 2–7 строится множество $I_{\boldsymbol{v}}$, включающее в себя индексы всех опорных гиперплоскостей и индексы граничных гиперплоскостей, проходящих через точку $\boldsymbol{v}$. На шагах 8–13 из множества $I_{\boldsymbol{v}}$ удаляются индексы избыточных уравнений в случае, когда они там присутствуют. Если $I_{\boldsymbol{v}} \neq \emptyset$, то далее выполняются шаги 15–19, в результате чего вычисляется точка $\boldsymbol{w} \in \mathrm{aff}(\mathrm{face}(\boldsymbol{v}))$ такая, что $\boldsymbol{w} \neq \boldsymbol{v}$. Для этого на шаге 16 генерируется случайный вектор $\boldsymbol{c} \in \mathbb{R}^n$. Используя этот вектор, шаг 17 вычисляет точку $\boldsymbol{u}$, удаленную от $\boldsymbol{v}$ на расстояние δ. Здесь δ — «большой» положительный параметр: чем больше δ, тем точнее будет вычислен направляющий вектор $\boldsymbol{d}$, используемый функцией Jump$(\cdot)$. Шаг 18 вычисляет точку $\boldsymbol{w}$, являющуюся ортогональной проекцией точки $\boldsymbol{u}$ на подпространство $\mathrm{aff}(\mathrm{face}(\boldsymbol{v}))$ с помощью формулы (37). Если сгенерированный вектор $\boldsymbol{c}$ случайно оказался перпендикулярным к подпространству $\mathrm{aff}(\mathrm{face}(\boldsymbol{v}))$, то есть

$$\forall i \in I_{\boldsymbol{v}} : \langle \boldsymbol{a}_i, \boldsymbol{c} \rangle = 0,$$

то мы получим $\boldsymbol{v} = \boldsymbol{w}$. В этом случае на шаге 16 генерируется новый случайный вектор $\boldsymbol{c}$ и повторно вычисляются точки $\boldsymbol{u}$ и $\boldsymbol{w}$ (шаги 17, 18). Если на шаге 19 $\boldsymbol{v} \neq \boldsymbol{w}$, то происходит выход из цикла **repeat/until**. При $I_{\boldsymbol{v}} = \emptyset$ вместо шагов 15–19 выполняется шаг 22, который

Алгоритм 2 Прохождение грани

Require: $\boldsymbol{v} \in M$; $\dim(\text{face}(\boldsymbol{v})) > 0$

1: **function** MoveAlongFace($\boldsymbol{v}$)
2: $I_{\boldsymbol{v}} := \bar{I}$
3: **for** $i = 1, \ldots, m$ **do**
4: **if** $\boldsymbol{v} \in \hat{H}_i$ **then**
5: $I_{\boldsymbol{v}} := I_{\boldsymbol{v}} \cup \{i\}$
6: **end if**
7: **end for**
8: $I'_{\boldsymbol{v}} := I_{\boldsymbol{v}}$
9: **for** $i \in I'_{\boldsymbol{v}}$ **do**
10: **if** $\text{rank}(A'_{I_v - \{i\}}) = \text{rank}(A'_{I_v})$ **then**
11: $I_{\boldsymbol{v}} := I_{\boldsymbol{v}} - \{i\}$
12: **end if**
13: **end for**
14: **if** $I_{\boldsymbol{v}} \neq \emptyset$ **then**
15: **repeat**
16: $\boldsymbol{c} := \text{RandomNonZeroVector}()$
17: $\boldsymbol{u} := \boldsymbol{v} + \delta \boldsymbol{c}/\|\boldsymbol{c}\|$
18: $\boldsymbol{w} := \boldsymbol{\pi}_{I_v}(\boldsymbol{u})$
19: **until** $\boldsymbol{v} \neq \boldsymbol{w}$
20: **end if**
21: **if** $I_{\boldsymbol{v}} = \emptyset$ **then**
22: $\boldsymbol{w} := \boldsymbol{\pi}_1(\boldsymbol{v})$
23: **end if**
24: **if** $\langle \boldsymbol{c}, \boldsymbol{w} \rangle > \langle \boldsymbol{c}, \boldsymbol{v} \rangle$ **then**
25: $\boldsymbol{d} := \boldsymbol{w} - \boldsymbol{v}$
26: **else**
27: $\boldsymbol{d} := \boldsymbol{v} - \boldsymbol{w}$
28: **end if**
29: $\boldsymbol{v}_{next} := \text{Jump}(\boldsymbol{v}, \boldsymbol{d})$
30: **return** $\boldsymbol{v}_{next}$
31: **end function**

в качестве $\boldsymbol{w}$ выбирает ортогональную проекцию точки $\boldsymbol{v}$ на граничную гиперплоскость с индексом 1 по следующей формуле [12]:

$$\boldsymbol{\pi}_1(\boldsymbol{v}) = \boldsymbol{v} - \frac{\langle \boldsymbol{a}_1, \boldsymbol{v} \rangle - b_1}{\|\boldsymbol{a}_1\|^2} \boldsymbol{a}_1.$$

Из $I_{\boldsymbol{v}} = \emptyset$ следует, что точка $\boldsymbol{v}$ является внутренней точкой допустимого многогранника M. Поэтому $\boldsymbol{v} \neq \boldsymbol{w}$. Шаги 24–28 вычисляют направляющий вектор $\boldsymbol{d} \neq \boldsymbol{0}$, ведущий к увеличению значения целевой функции. В завершение на шаге 29 с помощью вызова функции $\text{Jump}(\boldsymbol{v}, \boldsymbol{d})$ осуществляется переход по этому направлению к новой граничной точке $\boldsymbol{v}_{next}$, которая возвращается в качестве результата на шаге 30.

Функция MoveAlongFace$(\cdot)$ в качестве аргумента требует точку $\boldsymbol{v}$, принадлежащую допустимому многограннику M. Такую точку можно получить с помощью метода релаксации Агмона—Моцкина—Шёнберга [17, 18]. При этом систему общего вида (6) необходимо будет привести к стандартному виду (1).

Реализация алгоритма VeSP, выполняющего поиск вершины допустимого многогранника M, определяемого системой ограничений (6), представлена в виде алгоритма 3.

Алгоритм 3 VeSP (Vertex Seeking by Projecting)

Require: $\boldsymbol{v}_0 \in M$

1: $t := 0$
2: **while** $\dim(\text{face}(\boldsymbol{v}_t)) > 0$ **do**
3: $\quad \boldsymbol{v}_{t+1} := \text{MoveAlongFace}(\boldsymbol{v}_t)$
4: $\quad t := t + 1$
5: **end while**
6: **output** $\boldsymbol{v}_t$
7: **stop**

Прокомментируем шаги этого алгоритма. Шаг 1 устанавливает счетчик итераций t в значение 0. Шаг 2 открывает цикл **while** вычисления вершины многогранника. Если размерность грани, на которой лежит точка $\boldsymbol{v}_t$, больше нуля, выполняется шаг 3, который с помощью функции MoveAlongFace$(\cdot)$ (см. алгоритм 2) осуществляет переход к точке $\boldsymbol{v}_{t+1}$, лежащей на грани меньшей размерности. После этого выполняется шаг 4, увеличивающий счетчик итераций t на единицу, и выполняется переход на начало цикла **while**. Если условие цикла **while** на шаге 2 не выполняется, то точка $\boldsymbol{v}_t$ является вершиной допустимого многогранника M. В этом случае выполняется переход на шаг 6, который выводит точку $\boldsymbol{v}_t$ в качестве результата. Шаг 7 завершает работу алгоритма.

Сходимость алгоритма 3 к вершине допустимого многогранника за конечное число итераций гарантируется следующей теоремой.

Теорема 1. (Сходимость алгоритма VeSP) Пусть система ограничений (6) является регулярной, то есть выполняются условия определения 1. Обозначим через

$$\boldsymbol{v}_0, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_t, \ldots \tag{41}$$

последовательность точек, генерируемых алгоритмом 3. Тогда эта последовательность сходится к точке, являющейся вершиной допустимого многогранника M, не более, чем за n итераций, где n — размерность пространства:

$$\exists t' \leqslant n : \dim(\text{face}(\boldsymbol{v}_{t'})) = 0.$$

Доказательство. Если $\dim(\text{face}(\boldsymbol{v}_0)) = 0$ то точка $\boldsymbol{v}_0$ является вершиной допустимого многогранника M. В этом случае алгоритм 3 выводит $\boldsymbol{v}_0$ в качестве результата и на этом заканчивает свою работу.

Пусть начальная точка $\boldsymbol{v}_0$ вершиной не является. В этом случае последовательность (41) содержит более одного элемента. Пусть $\boldsymbol{v}_t$ — произвольный не конечный элемент последо-

вательности (41), то есть $\boldsymbol{v}_t$ не является вершиной:

$$\dim(\text{face}(\boldsymbol{v}_t)) > 0. \tag{42}$$

В соответствии с (35) имеем

$$\dim(\text{face}(\boldsymbol{v}_t)) = n - \min(k + l, n),$$

где l — количество граничных гиперплоскостей, проходящих через точку $\boldsymbol{v}_t$. В силу (42) отсюда получаем

$$k + l < n \tag{43}$$

и

$$\dim(\text{face}(\boldsymbol{v}_t)) = n - k - l. \tag{44}$$

Шаг 3 алгоритма 3 с помощью вызова функции MoveAlongFace($\boldsymbol{v}_t$), реализуемой алгоритмом 2, генерирует следующий элемент $\boldsymbol{v}_{t+1}$. Проанализируем работу алгоритма 2. Точке $\boldsymbol{v}_t$ в этом алгоритме соответствует точка $\boldsymbol{v}$:

$$\boldsymbol{v} \equiv \boldsymbol{v}_t. \tag{45}$$

Из (44) и (45) следует

$$\dim(\text{face}(\boldsymbol{v})) = n - k - l. \tag{46}$$

На шагах 2–7 алгоритма 2 формируется множество $I_{\boldsymbol{v}}$, включающее в себя индексы всех опорных и граничных гиперплоскостей, проходящих через точку $\boldsymbol{v}$. Так как по условию теоремы система ограничений является регулярной, то из (43) и (24) следует, что множество $I_{\boldsymbol{v}}$ не содержит индексов избыточных уравнений.

Предположим сначала, что $I_{\boldsymbol{v}} \neq \emptyset$. В этом случае выполняются шаги 15–19. Обозначим через $S_{\boldsymbol{v}}$ подпространство, являющееся аффинной оболочкой грани face($\boldsymbol{v}$):

$$S_{\boldsymbol{v}} = \bigcap_{i \in I_{\boldsymbol{v}}} H_i.$$

На шагах 15–19 вычисляется точка $\boldsymbol{w} \in S_{\boldsymbol{v}}$, отличная от точки $\boldsymbol{v}$. С использованием этих точек шаги 24–28 формируют ненулевой направляющий вектор $\boldsymbol{d} = \boldsymbol{w} - \boldsymbol{v}$, задающий луч

$$\text{ray}(\boldsymbol{v}, \boldsymbol{d}) = \{\, \boldsymbol{x} \in \mathbb{R}^n \,|\, \boldsymbol{x} = \boldsymbol{v} + \lambda \boldsymbol{d},\ \lambda \geqslant 0 \}\,.$$

По построению этот луч принадлежит подпространству $S_{\boldsymbol{v}}$:

$$\text{ray}(\boldsymbol{v}, \boldsymbol{d}) \subseteq S_{\boldsymbol{v}}. \tag{47}$$

На шаге 29 с помощью вызова функции Jump($\boldsymbol{v}, \boldsymbol{d}$) (см. алгоритм 1) вычисляется точка $\boldsymbol{v}_{next} \in M$, являющаяся пересечением луча ray($\boldsymbol{v}, \boldsymbol{d}$) с ближайшей граничной гиперплоскостью $\hat{H}_{i'}$ такой, что $i' \notin I_{\boldsymbol{v}}$. Такая гиперплоскость существует в силу ограниченности допустимого многогранника M. Из (47) следует, что через точку $\boldsymbol{v}_{next}$ проходит не менее $l' = l + 1$ граничных гиперплоскостей. С учетом (35) и (43) отсюда получаем

$$\dim(\text{face}(\boldsymbol{v}_{next})) \leqslant n - k - l - 1. \tag{48}$$

Сопоставляя (46) и (48), имеем

$$\dim(\text{face}(\boldsymbol{v})) > \dim(\text{face}(\boldsymbol{v}_{next})).$$

Теперь предположим, что $I_{\boldsymbol{v}} = \emptyset$, то есть $k = 0$ и $l = 0$. В соответствии со следствием 4 отсюда следует

$$\dim(\text{face}(\boldsymbol{v})) = n. \tag{49}$$

В этом случае выполняется шаг 22, в результате которого мы получаем точку $\boldsymbol{w}$ на граничной гиперплоскости $\hat{H}_1$. Так как через $\boldsymbol{v}$ не проходит ни одной граничной гиперплоскости, то $\boldsymbol{v} \neq \boldsymbol{w}$. Шаги 24–28 формируется ненулевой направляющий вектор $\boldsymbol{d} = \boldsymbol{w} - \boldsymbol{v}$, задающий луч $\text{ray}(\boldsymbol{v}, \boldsymbol{d})$. На шаге 29 с помощью вызова функции $\text{Jump}(\boldsymbol{v}, \boldsymbol{d})$ (см. алгоритм 1) вычисляется точка $\boldsymbol{v}_{next} \in M$, являющаяся пересечением луча $\text{ray}(\boldsymbol{v}, \boldsymbol{d})$ с ближайшей граничной гиперплоскостью. Таким образом, через точку $\boldsymbol{v}_{next}$ проходит не менее одной граничной гиперплоскости. В соответствии с (35) отсюда следует

$$\dim(\text{face}(\boldsymbol{v}_{next})) \leqslant n - 1. \tag{50}$$

Сопоставляя (49) и (50), и в этом случае имеем

$$\dim(\text{face}(\boldsymbol{v})) > \dim(\text{face}(\boldsymbol{v}_{next})). \tag{51}$$

Точка $\boldsymbol{v}_{next}$ на шаге 30 возвращается как результат вызова функции $\text{MoveAlongFace}(\boldsymbol{v}_t)$ в алгоритме 3 на шаге 3. Поэтому

$$\boldsymbol{v}_{next} \equiv \boldsymbol{v}_{t+1}. \tag{52}$$

Из 45, 51 и 52 следует

$$\dim(\text{face}(\boldsymbol{v}_t)) > \dim(\text{face}(\boldsymbol{v}_{t+1})). \tag{53}$$

Таким образом, последовательности (41) соответствует убывающая последовательность неотрицательных целых чисел

$$\dim(\text{face}(\boldsymbol{v}_0)) > \dim(\text{face}(\boldsymbol{v}_1)) > \cdots > \dim(\text{face}(\boldsymbol{v}_t)) > \ldots,$$

ограниченная сверху числом, равным размерности пространства n. Следовательно, последовательность (41) имеет длину не более $n + 1$. Осталось заметить, что в соответствии с условием выхода из цикла **while** (шаг 2), последним элементом последовательности (41) будет некоторая вершина допустимого многогранника M. Теорема доказана. □

3. Реализация и вычислительные эксперименты

Нами была выполнена реализация алгоритма VeSP на языке C++, исходные коды которой свободно доступны на `https://github.com/leonid-sokolinsky/VeSP`. Мы протестировали алгоритм VeSP на эталонных задачах из репозитория Netlib-LP [19, 20], доступного по адресу `https://netlib.org/lp/data`. Тестирование выполнялось на персональном компьютере с процессором Intel Core i7-13620H 2.40 GHz, DDR5 32 ГБ. Результаты тестирования представлены в табл. 1.

В заголовке таблицы использованы следующие обозначения: в колонке «Задача» указано имя задачи ЛП из репозитория Netlib-LP; m — количество ограничений; n — количество переменных (размерность пространства решений); $\dim(M)$ — размерность многогранника

Таблица 1. Результаты тестирования алгоритма VeSP на задачах Netlib-LP

Задача	m	n	$\dim(M)$	$\langle \boldsymbol{c}, \boldsymbol{u}_0 \rangle$	Итер.	$\langle \boldsymbol{c}, \boldsymbol{v} \rangle$	Время	$\operatorname{dist}(\boldsymbol{v}, M)$
adlittle	56	97	82	0.106230×10^{8}	44	0.716431×10^{6}	0	2.3×10^{-7}
afiro	27	32	24	0.532704×10^{2}	6	-0.309501×10^{2}	0	9.9×10^{-11}
beaconfd	173	262	122	0.338431×10^{5}	35	0.337284×10^{5}	2	1.5×10^{-11}
blend*	74	83	40	-0.508862×10^{1}	18	-0.158981×10^{2}	775	1.1×10^{-12}
fit1d	24	1026	1025	-0.483981×10^{4}	530	-0.899543×10^{4}	614	4.0×10^{-7}
grow7	140	301	161	-0.207633×10^{8}	119	-0.439694×10^{8}	2	9.7×10^{-6}
israel	174	142	142	0.307907×10^{7}	45	-0.102957×10^{6}	0	5.9×10^{-7}
kb2	43	41	25	-0.560444	2	-0.174527×10^{3}	0	2.2×10^{-10}
recipe	91	180	92	-0.230604×10^{3}	9	-0.262820×10^{3}	0	1.5×10^{-12}
sc105	104	103	58	-0.537203×10^{-2}	7	-0.981874×10^{1}	0	3.6×10^{-11}
sc50a	49	48	28	-0.710608×10^{-1}	7	-0.414361×10^{2}	0	9.0×10^{-12}
sc50b	48	48	28	-0.107994	13	-0.478109×10^{2}	0	4.9×10^{-12}
scagr7	129	140	56	-0.174392×10^{7}	22	-0.202156×10^{7}	0	3.9×10^{-8}
share2b	96	79	66	-0.368733×10^{3}	10	-0.389506×10^{3}	0	1.3×10^{-7}
stocfor1	117	111	48	-0.254283×10^{5}	8	-0.254636×10^{5}	0	1.1×10^{-12}

*Применен итерационный алгоритм Качмажа.

допустимых решений, вычисляемая по формуле $\dim(M) = n - \operatorname{rank}(\bar{A})$, где $\bar{A}$ — матрица коэффициентов уравнений, входящих в систему ограничений; $\langle \boldsymbol{c}, \boldsymbol{u}_0 \rangle$ — значение целевой функции в начальной точке $\boldsymbol{u}_0 \in M$; в колонке «Итер.» указано количество итераций, выполненных алгоритмом 3 при поиске вершины многогранника M; $\langle \boldsymbol{c}, \boldsymbol{v} \rangle$ — значение целевой функции в найденной вершине $\boldsymbol{v}$; «Время» — затраченное время в секундах; $\operatorname{dist}(\boldsymbol{v}, M)$ — точность решения, вычисляемая как евклидово расстояние от точки $\boldsymbol{v}$ до многогранника M.

Начальная точка $\boldsymbol{u}_0 \in M$ вычислялась с помощью написанных нами программ FIP и Quest. Программа FIP (Feasible point Iterative Projection) стартует из точки $\boldsymbol{x}_0 = \boldsymbol{0}$ и находит точку $\boldsymbol{z}_0$, принадлежащую допустимому многограннику M. В основе программы FIP лежит метод релаксаций Агмона—Моцкина—Шёнберга [17, 18]. Исходные коды этой программы доступны на `https://github.com/leonid-sokolinsky/FIP`. Полученная точка $\boldsymbol{z}_0 \in M$ использовалась программой Quest для нахождения так называемой точки апекса $\boldsymbol{u}_0$, которая представляла собой «грубое» приближение к решению задачи ЛП [21]. Эта точка использовалась алгоритмом VeSP в качестве начального приближения. Исходные коды программы Quest доступны на `https://github.com/leonid-sokolinsky/Quest`.

Проведенные эксперименты показали, что алгоритм VeSP успешно нашел вершину допустимого многогранника для всех 15 задач, выбранных из репозитория Netlib-LP. Время вычислений для задач с числом переменных $n < 200$, за исключением задачи «blend», составило менее половины секунды. Задачи «beaconfd» и «grow7» с количеством переменных 262 и 301 соответственно потребовали 2 секунды процессорного времени. При решении задачи «fit1d», содержащей 1026 переменных, алгоритм VeSP в процессе поиска вершины выполнил 513 итераций, что потребовало более 10 минут. Таким образом, можно сделать вывод, что время работы алгоритма VeSP существенно зависит от размерности задачи. Что касается задачи «blend» с числом переменных 83, для нее не удалось вычислить обратную матрицу в формуле (37) по причине потери точности при использовании 64-разрядной вещественной арифметики. Поэтому для приближенного вычисления ортогональных проекций (шаги 18, 22 алгоритма 2) пришлось использовать итерационный алгоритм Качмажа. По этой причине 18 итераций, выполненных алгоритмом VeSP для этой задачи, заняли 13 минут.

При этом более 99% процессорного времени было затрачено на вычисление ортогональных проекций. Это объясняется тем, что итерационный алгоритм Качмажа имеет линейную скорость сходимости [22]:

$$\|\boldsymbol{x}_k - \boldsymbol{\pi}_J(\boldsymbol{x}_0)\| \leqslant \theta \, \|\boldsymbol{x}_{k-1} - \boldsymbol{\pi}_J(\boldsymbol{x}_0)\|$$

с вещественной константой $0 < \theta < 1$. Здесь $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{k-1}, \boldsymbol{x}_k, \ldots\}$ — последовательные приближения к ортогональной проекции $\boldsymbol{\pi}_J(\boldsymbol{x}_0)$ точки $\boldsymbol{x}_0$ на подпространство S_J, вычисляемые алгоритмом Качмажа. Константа θ зависит только от углов между гиперплоскостями H_i $(i \in J)$. При малых углах значение θ приближается к единице, и скорость сходимости стремится к нулю. Проведенные эксперименты также показали, что точность вычисления координат вершины допустимого многогранника находится в обратной пропорции относительно количества выполненных итераций.

Заключение

В статье предложен новый проекционный алгоритм VeSP для нахождения вершины многогранника допустимых решений системы линейных ограничений. Под системой линейных ограничений понимается система общего вида, включающая в себя как линейные неравенства, так и линейные уравнения. Подобные системы ограничений характерны для задач линейного программирования. В качестве стартовой точки алгоритм VeSP использует произвольную точку многогранника допустимых решений. Алгоритм VeSP применим к любым системам линейных ограничений, имеющим непустую ограниченную область допустимых решений. Доказано утверждение, позволяющее для заданной допустимой точки вычислить ее собственную грань (грань наименьшей размерности, содержащую эту точку). В качестве следствий получены достаточные условия для того, чтобы собственная грань допустимой точки была вершиной, ребром или многогранником допустимых решений. Представлено формализованное описание алгоритма VeSP на псевдокоде. Также представлены формализованные описания двух подпрограмм-функций, используемых алгоритмом VeSP: подпрограмма-функция, осуществляющая перемещение по направляющему вектору от заданной допустимой точки к максимально удаленной допустимой точке, и подпрограмма-функция прохождения грани многогранника. Доказана теорема сходимости алгоритма VeSP к вершине многогранника допустимых решений за количество итераций, не превышающих размерность пространства. Выполнена реализация алгоритма VeSP на языке программирования C++. Указанная реализация была протестирована на эталонных задачах из репозитория Netlib-LP. Эксперименты показали, что алгоритм VeSP способен эффективно находить базисное решение для реальных задач ЛП. Основным преимуществом алгоритма VeSP является то, что он гарантированно находит вершину многогранника допустимых решений не более, чем за n итераций. Алгоритмам, основанным на симплекс-методе и методе исключения переменных Фурье—Моцкина, для этого может понадобиться 2^n и $2^{n/2}$ итераций соответственно.

Обозначения

n	число переменных в системе ограничений (размерность пространства)
m	количество неравенств в системе ограничений
k	количество уравнений в системе ограничений
$\mathbb{R}^d$	вещественное евклидово пространство размерности d

$\langle \cdot, \cdot \rangle$	скалярное произведение двух векторов
$\hat{A}$	матрица коэффициентов неравенств: $\hat{A} \in \mathbb{R}^{m \times n}$
$\bar{A}$	матрица коэффициентов уравнений: $\bar{A} \in \mathbb{R}^{k \times n}$
$\hat{\boldsymbol{b}}$	столбец правых частей неравенств: $\hat{\boldsymbol{b}} \in \mathbb{R}^m$
$\bar{\boldsymbol{b}}$	столбец правых частей уравнений: $\bar{\boldsymbol{b}} \in \mathbb{R}^k$
$\boldsymbol{a}_i$	i-тая строка матрицы $\begin{bmatrix} \hat{A} \\ \bar{A} \end{bmatrix}$ $(i = 1, \ldots, m + k)$
$\hat{I}$	индексы (номера) строк матрицы $\hat{A}$: $\hat{I} = \{1, \ldots, m\}$
$\bar{I}$	индексы (номера) строк матрицы $\bar{A}$: $\bar{I} = \{m + 1, \ldots, m + k\}$
P_i	полупространство, определяемое формулой $\langle \boldsymbol{a}_i, \boldsymbol{x} \rangle \leqslant b_i$ при $i \in \hat{I}$
$\hat{H}_i$	граничная гиперплоскость полупространства P_i
$\bar{H}_i$	опорная гиперплоскость, определяемая уравнением $\langle \boldsymbol{a}_i, \boldsymbol{x} \rangle = b_i$
$\hat{M}$	граничный многогранник (пересечение всех полупространств P_i)
$\bar{S}$	опорное подпространство (пересечение всех опорных гиперплоскостей $\bar{H}_i$)
M	допустимый многогранник (область допустимых решений): $M = \hat{M} \cap \bar{S}$
$\mathrm{aff}(X)$	аффинная оболочка множества X
$\dim(X)$	размерность множества X: $\dim(X) = \dim(\mathrm{aff}(X))$
$\mathrm{face}(\boldsymbol{v})$	собственная грань точки $\boldsymbol{v} \in M$
$\mathrm{rank}(A)$	ранг матрицы A
$\|\cdot\|$	евклидова норма

Литература

1. Dantzig G.B. Linear programming and extensions. Princeton, N.J.: Princeton university press, 1998. 656 p.
2. Жулев А., Соколинский Л. AlEM: новый параллельный алгоритм линейного программирования для кластерных вычислительных систем // PREPRINTS.RU. 2025. С. 1–19. DOI: 10.24108/preprints-3113529.
3. Avis D. A Revised Implementation of the Reverse Search Vertex Enumeration Algorithm // Polytopes - Combinatorics and Computation. DMV Seminar, vol 29 / ed. by G. Kalai, G. Ziegler. Basel: Birkhauser, 2000. P. 177–198. DOI: 10.1007/978-3-0348-8438-9_9.
4. Assad C.L., Morales G., Arica J. Vertex Enumeration of Polyhedra // Pesquisa Operacional. 2022. Vol. 42: e25457. P. 1–20. DOI: 10.1590/0101-7438.2022.042.00254570.
5. Схрейвер А. Теория линейного и целочисленного программирования: в 2 т. Т. 1. Москва: Мир, 1991. 360 с.
6. Klee V., Minty G.J. How good is the simplex algorithm? // Inequalities - III. Proceedings of the Third Symposium on Inequalities Held at the University of California, Los Angeles, Sept. 1-9, 1969 / ed. by O. Shisha. New York, NY, USA: Academic Press, 1972. P. 159–175.
7. Khachiyan L. Fourier–Motzkin Elimination Method // Encyclopedia of Optimization / ed. by C. Floudas, P. Pardalos. Boston, MA: Springer, 2008. P. 1074–1077. DOI: 10.1007/978-0-387-74759-0_187.
8. Duffin R.J. On fourier's analysis of linear inequality systems // Pivoting and Extensions / ed. by M. Balinski. Berlin, Heidelberg: Springer, 1974. P. 71–95. DOI: 10.1007/BFB0121242.

9. Motzkin T.S. Contributions to the Theory of Linear Inequalities // Theodore S. Motzkin: Selected Papers / ed. by D. Cantor, B. Gordon, B.L. Rothschild. Boston: Birkhauser, 1983. P. 1–80.

10. Gritzmann P., Klee V. Mathematical Programming and Convex Geometry // Handbook of Convex Geometry, Vol. A / ed. by P.M. Gruber, J.M. Wills. Amsterdam: Elsevier, 1993. P. 627–674.

11. Циглер Г.М. Теория многогранников. М.: МЦНМО, 2014. 568 с.

12. Murty K.G. Computational and Algorithmic Linear Algebra and n-Dimensional Geometry. World Scientific, 2011. xxi, 552 p. DOI: 10.1142/8261.

13. Kaczmarz S. Approximate solution of systems of linear equations // International Journal of Control. 1993. Vol. 57, no. 6. P. 1269–1271. DOI: 10.1080/00207179308934446.

14. Chen X. The Kaczmarz algorithm, row action methods, and statistical learning algorithms // Frames and Harmonic Analysis. Contemporary Mathematics, vol. 706 / ed. by Y. Kim, S. Narayan, G. Picioroaga, E. Weber. Providence, Rhode Island: American Mathematical Society, 2018. P. 115–128. DOI: 10.1090/CONM/706.

15. Ольховский Н.А., Соколинский Л.Б. О новом методе линейного программирования // Вычислительные методы и программирование. 2023. Т. 24, № 4. С. 408–429. DOI: 10.26089/NumMet.v24r428.

16. Ольховский Н.А., Соколинский Л.Б. Визуальное представление многомерных задач линейного программирования // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2022. Т. 11, № 1. С. 31–56. DOI: 10.14529/cmse220103.

17. Agmon S. The relaxation method for linear inequalities // Canadian Journal of Mathematics. 1954. Vol. 6. P. 382–392. DOI: 10.4153/CJM-1954-037-2.

18. Motzkin T.S., Schoenberg I.J. The relaxation method for linear inequalities // Canadian Journal of Mathematics. 1954. Vol. 6. P. 393–404. DOI: 10.4153/CJM-1954-038-x.

19. Gay D.M. Electronic mail distribution of linear programming test problems // Mathematical Programming Society COAL Bulletin. 1985. Vol. 13. P. 10–12.

20. Koch T. The final NETLIB-LP results // Operations Research Letters. 2004. Vol. 32, no. 2. P. 138–142. DOI: 10.1016/S0167-6377(03)00094-4.

21. Соколинский Л.Б., Соколинская И.М. О новой версии апекс-метода для решения задач линейного программирования // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2023. Т. 12, № 2. С. 5–46. DOI: 10.14529/cmse230201.

22. Deutsch F. Rate of Convergence of the Method of Alternating Projections // Parametric Optimization and Approximation / ed. by B. Brosowski, F. Deutsch. Basel: Birkhauser Verlag, 1985. P. 96–107. DOI: 10.1007/978-3-0348-6253-0_7.

Жулев Александр Эдуардович, аспирант кафедры системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

Соколинский Леонид Борисович, д.ф.-м.н., профессор, заведующий кафедрой системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

Соколинская Ирина Михайловна, к.ф.-м.н., доцент кафедры прикладной математики и программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

DOI: 10.14529/cmse250301

ON CALCULATING A VERTEX OF FEASIBLE SOLUTIONS POLYTOPE OF LINEAR CONSTRAINT SYSTEM

© 2025 A.E. Zhulev, L.B. Sokolinsky, I.M. Sokolinskaya

South Ural State University (pr. Lenina 76, Chelyabinsk, 454080 Russia)

E-mail: zhulevae@susu.ru, leonid.sokolinsky@susu.ru, irina.sokolinskaya@susu.ru

Received: 14.06.2025

The article is devoted to a new algorithm for calculating a vertex of polytope being the feasible region of linear constraint system. The algorithm called VeSP starts at an arbitrary point of the polytope and, moving along its faces, stops at some vertex. To calculate the movement direction along the face, it uses the projection method. The idea of this method is as follows. For the current approximation point, an affine subspace is calculated, which is the affine hull of the face containing the point. A non-zero vector is added to the current approximation point. This gives an external point relative to the current affine subspace. The orthogonal projection of the external point onto the current affine subspace is calculated using a known analytical formula. The projection point determines the direction of movement along the edge to its boundary, which gives the next approximation point. Each movement reduces the dimension of the current face. Thus, we arrive at a zero-dimensional face, which is the vertex of the polytope. A formal description of the VeSP algorithm is provided. The convergence of the VeSP algorithm to a polytope vertex in a finite number of iterations is proved. This number does not exceed the space dimension. An information about the implementation of the VeSP algorithm in C++ is provided. The results of computational experiments with real problems from the Netlib-LP collection are described. For all test problems, the VeSP algorithm successfully found the vertex of the polytope in a finite number of iterations that did not exceed the space dimension. For most problems, finding the vertex took less than one second on a commodity personal computer.

Keywords: linear constraints, feasible solutions polytope, vertex calculation, projection method, VeSP algorithm.

FOR CITATION

Zhulev A.E., Sokolinsky L.B., Sokolinskaya I.M. On Calculating a Vertex of Feasible Solutions Polytope of Linear Constraints System. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2025. Vol. 14, no. 3. P. 5–27. (in Russian) DOI: 10.14529/cmse250301.

This paper is distributed under the terms of the Creative Commons Attribution-Non Commercial 4.0 License which permits non-commercial use, reproduction and distribution of the work without further permission provided the original work is properly cited.

References

1. Dantzig G.B. Linear programming and extensions. Princeton, N.J.: Princeton university press, 1998. 656 p.
2. Zhulev A., Sokolinsky L. AlEM: a new parallel algorithm for linear programming on cluster computing systems. PREPRINTS.RU. 2025. P. 1–19. (in Russian) DOI: 10.24108/preprints-3113529.

3. Avis D. A Revised Implementation of the Reverse Search Vertex Enumeration Algorithm. Polytopes - Combinatorics and Computation. DMV Seminar, vol 29 / ed. by G. Kalai, G. Ziegler. Basel: Birkhauser, 2000. P. 177–198. DOI: 10.1007/978-3-0348-8438-9_9.
4. Assad C.L., Morales G., Arica J. Vertex Enumeration of Polyhedra. Pesquisa Operacional. 2022. Vol. 42: e25457. P. 1–20. DOI: 10.1590/0101-7438.2022.042.00254570.
5. Schrijver A. Theory of Linear and Integer Programming. Chichester, New York, Brisbane, Tofonfo, Singapore: Wiley, Sons, 1998. 484 p.
6. Klee V., Minty G.J. How good is the simplex algorithm?. Inequalities - III. Proceedings of the Third Symposium on Inequalities Held at the University of California, Los Angeles, Sept. 1-9, 1969 / ed. by O. Shisha. New York, NY, USA: Academic Press, 1972. P. 159–175.
7. Khachiyan L. Fourier–Motzkin Elimination Method. Encyclopedia of Optimization / ed. by C. Floudas, P. Pardalos. Boston, MA: Springer, 2008. P. 1074–1077. DOI: 10.1007/978-0-387-74759-0_187.
8. Duffin R.J. On fourier's analysis of linear inequality systems. Pivoting and Extensions / ed. by M. Balinski. Berlin, Heidelberg: Springer, 1974. P. 71–95. DOI: 10.1007/BFB0121242.
9. Motzkin T.S. Contributions to the Theory of Linear Inequalities. Theodore S. Motzkin: Selected Papers / ed. by D. Cantor, B. Gordon, B.L. Rothschild. Boston: Birkhauser, 1983. P. 1–80.
10. Gritzmann P., Klee V. Mathematical Programming and Convex Geometry. Handbook of Convex Geometry, Vol. A / ed. by P.M. Gruber, J.M. Wills. Amsterdam: Elsevier, 1993. P. 627–674.
11. Ziegler G.M. Lectures on Polytopes. Vol. 152. New York, NY: Springer New York, 1995. XI, 370 p. Graduate Texts in Mathematics. DOI: 10.1007/978-1-4613-8431-1.
12. Murty K.G. Computational and Algorithmic Linear Algebra and n-Dimensional Geometry. World Scientific, 2011. xxi, 552 p. DOI: 10.1142/8261.
13. Kaczmarz S. Approximate solution of systems of linear equations. International Journal of Control. 1993. Vol. 57, no. 6. P. 1269–1271. DOI: 10.1080/00207179308934446.
14. Chen X. The Kaczmarz algorithm, row action methods, and statistical learning algorithms. Frames and Harmonic Analysis. Contemporary Mathematics, vol. 706 / ed. by Y. Kim, S. Narayan, G. Picioroaga, E. Weber. Providence, Rhode Island: American Mathematical Society, 2018. P. 115–128. DOI: 10.1090/CONM/706.
15. Olkhovsky N.A., Sokolinsky L.B. Surface Movement Method for Linear Programming. Lobachevskii Journal of Mathematics. 2024. Vol. 45, no. 10. P. 5061–5079. DOI: 10.1134/S1995080224605745.
16. Olkhovsky N., Sokolinsky L. Visualizing Multidimensional Linear Programming Problems. Parallel Computational Technologies. PCT 2022. Communications in Computer and Information Science, vol. 1618 / ed. by L. Sokolinsky, M. Zymbler. Cham: Springer, 2022. P. 172–196. DOI: 10.1007/978-3-031-11623-0_13.
17. Agmon S. The relaxation method for linear inequalities. Canadian Journal of Mathematics. 1954. Vol. 6. P. 382–392. DOI: 10.4153/CJM-1954-037-2.
18. Motzkin T.S., Schoenberg I.J. The relaxation method for linear inequalities. Canadian Journal of Mathematics. 1954. Vol. 6. P. 393–404. DOI: 10.4153/CJM-1954-038-x.

19. Gay D.M. Electronic mail distribution of linear programming test problems. Mathematical Programming Society COAL Bulletin. 1985. Vol. 13. P. 10–12.

20. Koch T. The final NETLIB-LP results. Operations Research Letters. 2004. Vol. 32, no. 2. P. 138–142. DOI: 10.1016/S0167-6377(03)00094-4.

21. Sokolinsky L.B., Sokolinskaya I.M. Apex Method: A New Scalable Iterative Method for Linear Programming. Mathematics. 2023. Vol. 11, no. 7. P. 1–28. DOI: 10.3390/MATH11071654.

22. Deutsch F. Rate of Convergence of the Method of Alternating Projections. Parametric Optimization and Approximation / ed. by B. Brosowski, F. Deutsch. Basel: Birkhauser Verlag, 1985. P. 96–107. DOI: 10.1007/978-3-0348-6253-0_7.

УДК 004.942 **DOI: 10.14529/cmse250302**

МЕТОД ПРЕДСТАВЛЕНИЯ ТРЕХ WORK-STEALING ДЕКОВ В ДВУХУРОВНЕВОЙ ПАМЯТИ

© 2025 Е.А. Аксёнова[1], Е.А. Барковский, А.В. Соколов[1]

[1] *Институт прикладных математических исследований Карельского научного центра Российской академии наук*

(185910 Петрозаводск, ул. Пушкинская, д. 11)

E-mail: aksenova@krc.karelia.ru, barkevgen@gmail.com, avs@krc.karelia.ru

Поступила в редакцию: 21.07.2025

Эффективность работы распределенных систем во многом зависит от равномерной загруженности потоков, что обеспечивается за счет различных стратегий балансировки задач между ними. Одним из методов децентрализованной динамической стратегии балансировки, где каждый поток участвует в распределении задач, является «work-stealing». Принцип метода в том, что поток не имеющий задач перехватывает («steal») их у других потоков. В основе процесса лежит специальная структура данных, work-stealing дек, в котором находятся указатели на задачи. При работе с деками возникает проблема их эффективного расположения в памяти. В двухуровневой памяти дек можно расположить разделив его на концы и середину: в быстрой памяти (первый уровень) находятся вершина и основание дека; в медленной памяти (второй уровень) — середина. Таким образом, система может быстро обращаться к концам дека, где элементы активно добавляются и удаляются. В статье анализируется новый метод представления трех work-stealing деков в двухуровневой памяти, где концы деков находятся в отдельных участках памяти. Для него строится имитационная модель на основе метода Монте-Карло, с помощью которой исследуются задачи оптимального разделения памяти. В качестве критерия оптимальности используется максимальное среднее время работы системы до перераспределения памяти.

Ключевые слова: двухуровневая память, метод Монте-Карло, структуры данных, work-stealing деки.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Аксёнова Е.А., Барковский Е.А., Соколов А.В. Метод представления трех work-stealing деков в двухуровневой памяти // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2025. Т. 14, № 3. С. 28–41. DOI: 10.14529/cmse250302.

Введение

Эффективность работы распределенных систем во многом зависит от равномерной загруженности потоков. Ярким примером этого являются параллельные [1, 2] и облачные [3, 4] вычисления, где равномерная нагрузка потоков обеспечивается за счет различных стратегий балансировки задач между ними. Для задач, характеристики которых заранее известны, используют стратегию статической балансировки. Если характеристики задачи не известны или могут меняться (например, система работает в реальном времени), используют стратегию динамической балансировки [5]. Такие стратегии бывают централизованными (распределением задач занят специальный поток) и децентрализованными (каждый поток участвует в распределении задач) [6].

Work-stealing является распространенным методом децентрализованной динамической стратегии балансировки задач [7, 8] и реализован в широко используемых балансировщиках: Cilk; dotNET TPL; Intel TBB; X10. Принцип метода в том, что поток без задач перехватывает («steal») их у других потоков [9]. Для этого используется специальная структура данных, work-stealing дек, в котором находятся указатели на задачи.

В целом, work-stealing дек является деком («double ended queue» или «deque») с ограниченным входом [10]: в один конец элементы добавляются и удаляются по принципу LIFO (Last In, First Out) или «последним пришел, первым вышел», из другого конца только удаляются по принципу FIFO (First In, First Out) — «первым пришел, первым вышел». Каждый поток имеет свой work-stealing дек. Таким образом, когда создается новая задача, указатель помещается (операция «push») на вершину дека. Когда поток готов выполнить задачу, он берет (операция «pop») указатель на нее из вершины своего дека. Когда поток готов выполнить задачу, но его дек пуст, он перехватывает (операция «steal») указатель из основания другого дека. На рис. 1 представлена схема работы такого дека.

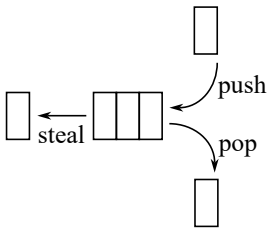


Рис. 1. Принцип работы work-stealing дека

Для эффективного применения стратегии децентрализованной динамической балансировки задач, исследуются как способы реализации метода work-stealing [11, 12], так и способы организации work-stealing деков [13, 14]: в [15] исследуются различные способы представления деков в памяти; в [16, 17] описывается разработка и проводится анализ динамического балансировщика, построенного на основе этого метода.

В этой работе рассматривается следующий способ расположения дека в памяти:

- В быстрой памяти (первый уровень) находятся вершина и основание;
- В медленной памяти (второй уровень) находится середина.

Таким образом, система может быстро обращаться к концам дека, где элементы активно добавляются и удаляются. Для этого способа представления дека возникают задачи разделения и перераспределения памяти.

Модель работы такого work-stealing дека была предложена в [18, 19]. В статьях была решена задача перераспределения двухуровневой памяти для критерия оптимальности: минимизация средних затрат на перераспределение. Модели работы и методы управления двумя work-stealing деками в двухуровневой памяти были рассмотрены в [20]. Модель работы трех work-stealing деков в двухуровневой памяти была предложена в [21]. В предыдущей статье была решена задача разделения памяти для метода представления, где концы деков растут на встречу друг другу в общем участке памяти. В этой статье предлагается модель и исследуется задача разделения двухуровневой памяти для альтернативного метода представления трех work-stealing деков, где концы деков находятся в отдельных участках памяти.

Статья организована следующим образом. В разделе 1 дается описание нового метода и ставятся задачи разделения памяти между тремя деками. В разделе 2 рассматривается имитационная модель процесса и приводится пример ее работы. Результаты исследований, проведенных на основе модели, представлены в разделе 3. В заключении предложены некоторые варианты использования метода и направление дальнейших исследований.

1. Постановка задачи

Пусть в памяти компьютерной системы расположены три work-stealing дека De_1, De_2 и De_3. Память системы разбита на два уровня имеющие разную скорость: память первого уровня быстрая, второго — медленная. Чтобы расположить дек в двухуровневой памяти, он разделен на три части (рис. 2):

1. конец Sk_n, куда указатели на задачи добавляются и удаляются;
2. середина Md_n, где указатели хранятся;
3. конец Qe_n, из которого указатели только удаляются (перехватываются методом work-stealing).

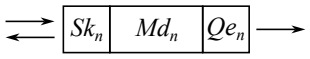


Рис. 2. Разбиение work-stealing дека De_n

Середины трех деков (Md_1, Md_2, Md_3) находятся в памяти второго уровня. В памяти первого уровня находятся концы этих деков: активные (Sk_1, Sk_2, Sk_3) и концы из которых происходит перехват (Qe_1, Qe_2, Qe_3), объединенные в одну очередь Qe. Таким образом, двухуровневая память компьютерной системы, в которой расположены три work-stealing дека, имеет следующий вид (рис. 3):

- m — общий размер быстрой памяти первого уровня;
- s — участок памяти первого уровня, где находятся объединенная очередь Qe и активный конец первого дека Sk_1, представленный в виде стека;
- d и $m - s - d$ — участки памяти, в которых расположены активные концы Sk_2 и Sk_3 соответственно.

Характеристики памяти второго уровня и находящихся в ней частей деков Md_1, Md_2 и Md_3 в этой статье не рассматриваются.

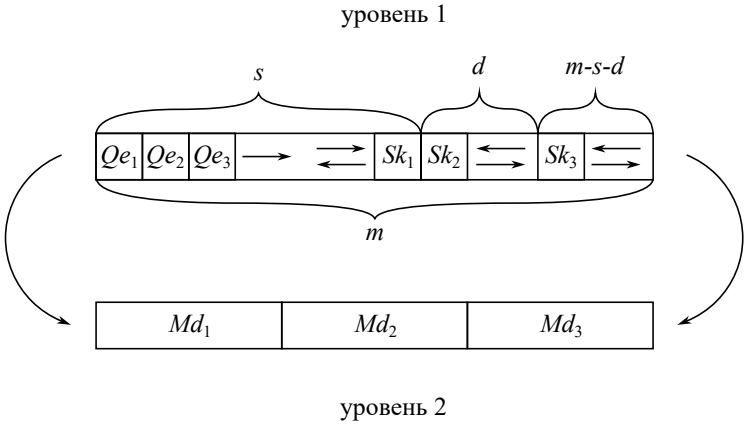


Рис. 3. Расположение деков в двухуровневой памяти

Пусть один указатель на задачу занимает одну единицу памяти. Весь размер быстрой памяти составляет m указателей, размеры концов каждого дека равны между собой $Sk_1 = Qe_1 = Sk_2 = Qe_2 = Sk_3 = Qe_3$. Система работает в дискретном времени, где на каждом шаге происходят операции влияющие на размер деков. Вероятности этих операций для каждого из деков указаны в табл. 1 ($p_n + q_n + w_n + pw_n + qw_n + r_n = 1$).

Таблица 1. Вероятности операций с деком De_n

Обозначение	Описание вероятности
p_n	Добавление одного указателя в рабочий конец Sk_n дека De_n
q_n	Удаление одного указателя из рабочего конца Sk_n дека De_n
w_n	Перехват одного указателя из общей work-stealing очереди Qe, в то время как дек De_n обрабатывает задачу
pw_n	Параллельное добавление указателя в Sk_n и перехват указателя из Qe
qw_n	Параллельное удаление указателя из Sk_n и перехват указателя из Qe
r_n	Операция не изменяющая размеры концов дека De_n (обработка задачи)

Время работы системы заранее не ограничено, но она прекращает работу если возникает одно из условий перераспределения памяти: дек De_n пытается забрать указатель из уже пустого конца Sk_n ($Sk_n < 0$); общая очередь work-stealing концов деков опустошена $Qe < 0$; один из участков памяти первого уровня переполнен $Qe + Sk_1 > s$, $Sk_2 > d$ или $Sk_3 > m - s - d$.

Для вышеописанной системы рассматриваются следующие задачи:

1. нахождение оптимального разделения s памяти первого уровня m, если участкам d и $m - s - d$ выделена половина оставшейся памяти ($d = m - s - d = (m - s)/2$);
2. нахождение оптимальных значений s и d разделения памяти между тремя деками.

Критерий оптимальности: максимальное среднее время работы системы (среднее количество операций) до перераспределения памяти.

2. Имитационная модель

Поставленные задачи были решены с помощью имитационного моделирования. Для корректной работы этой модели, основанной на методе Монте-Карло, некоторые из характеристик системы должны быть заранее известны. Так, значения вероятностей операций с деками должны быть получены в ходе предварительных статистических исследований. Размер общей памяти m фиксирован, в то время как значение разделения s всей памяти m и значение разделения d участка памяти $m - s$ вычисляются в ходе работы модели. На каждом шаге дискретного времени для каждого дека генерируется случайное значение от нуля до единицы, соответствующее одной из вероятности операций с ним. Среднее количество шагов, которые модель проделывает до перераспределения памяти, является средним временем работы t такой системы. Комбинации s и d при которых t принимает наибольшее значение, считаются оптимальными значениями разделения памяти первого уровня m.

Для примера, рассмотрим систему с характеристиками $m = 100$, $s = 50$, $d = 25$, $m - s - d = 25$ и значениями вероятностей $p_n = q_n = w_n = pw_n = qw_n = 0.16$ и $r_n = 0.20$ (рис. 4).

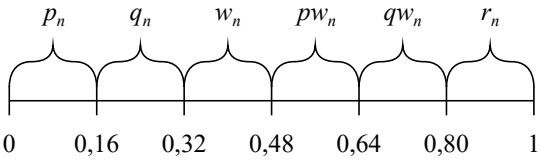


Рис. 4. Пример разбиения отрезка от нуля до одного между вероятностями операций с деками

Допустим, на шаге моделирования n эта система имеет следующий вид:

- $Qe = 1$;
- $Sk_1 = 10$;
- $Sk_2 = 0$;
- $Sk_3 = 25$.

Затем генерируется случайные значения, на основе которых совершаются операции с деками. Например: для дека De_1 было получено число 0.24, соответствующие вероятности q_1 удаления элемента из Sk_1; для дека De_2 число 0.08, что соответствует вероятности p_2 добавления элемента в Sk_2; и для дека De_3 число 0.72 и вероятность qw_3 — операция одновременного удаления элемента из Sk_3 и перехвата элемента из Qe.

Таким образом, на шаге $n + 1$ рабочие концы деков принимают следующие размеры:

- $Qe = 1 - 1 = 0$;
- $Sk_1 = 10 - 1 = 9$;
- $Sk_2 = 0 + 1 = 1$;
- $Sk_3 = 25 - 1 = 24$.

Так как переполнение памяти ($Qe + Sk_1 \not> 50$, $Sk_2 \not> 25$, $Sk_3 \not> 25$) или опустошение концов деков ($Qe \not< 0$, $Sk_1 \not< 0$, $Sk_2 \not< 0$, $Sk_3 \not< 0$) не произошло, то система продолжает свою работу. Полученные размеры рабочих концов становятся стартовыми размерами для следующего шага моделирования.

Рассмотрим ситуации, при которых такая система прекратила бы свою работу на шаге моделирования n:

1. Для дека De_2 сгенерировано случайное значение, соответствующее вероятности q_2, что приводит к опустошению рабочего конца дека ($Sk_2 = 0 - 1 = -1$, $Sk_2 < 0$);
2. Для дека De_3 получено число, соответствующие вероятности p_3 — память, отведенная рабочему концу дека, переполняется ($Sk_3 = 25 + 1 = 26$, $Sk_3 > 25$);
3. Произошло две подряд операции перехвата элемента (w_n, pw_n и/или qw_n) из общей очереди work-stealing концов деков, что приводит к ее опустошению ($Qe = 1-1-1 = -1$, $Qe < 0$).

В этой статье моделируется система, где размер общей памяти первого уровня равен $m = 100$ единиц и начальные размеры концов деков составляют $Sk_1 = Qe_1 = Sk_2 = Qe_2 = Sk_3 = Qe_3 = 10$ единиц памяти. Таким образом, участки памяти могут иметь следующие размеры: участок, в котором расположены конец Sk_1 и очередь work-stealing концов Qe — $s = 40 \dots 80$ единиц; участок, в котором расположен конец Sk_2 — $d = 10 \dots 50$ единиц и, соответственно, участок для конца Sk_3, $m - s - d = 10 \dots 50$ единиц.

3. Численный анализ

В результате работы имитационной модели были получены оптимальные значения разделения памяти (s, d) и среднее время работы системы (t) для различный вероятностей операций с деками. Вероятности имеют теоретические значения для более легкого сравнения результатов.

В следующих таблицах приведены результаты для задачи нахождения оптимального разделения s памяти m, где участки памяти d и $m - s - d$ имеют одинаковый размер ($d = m - s - d = (m - s)/2$). В табл. 2 рассматривается система, где указатели чаще добавляются в первый дек ($p_1 > p_2$, $p_1 > p_3$); в табл. 3 — чаще во второй ($p_2 > p_1$, $p_2 > p_3$). Система, в которой указатели с большей вероятностью добавляются в два дека ($p_1 > p_3$,

$p_2 > p_3$ и $p_2 > p_1$, $p_3 > p_1$), анализируется в табл. 4 и 5. Система, где указатели добавляются равномерно в три дека ($p_1 = p_2 = p_3$) рассматривается в табл. 6.

Таблица 2. Среднее время работы системы при $d = (m - s)/2$, $p_1 > p_2$ и $p_1 > p_3$

Вероятности операций	Метод разделения памяти	
	Разделение пополам ($s = 50$, $d = 25$)	Оптимальное разделение ($s = 66$, $d = 17$)
$p_1 = 0.50$, $p_2 = p_3 = 0.26$, $q_1 = 0.02$, $q_2 = q_3 = 0.26$, $w_1 = w_2 = w_3 = 0.01$, $pw_1 = pw_2 = pw_3 = 0.01$, $qw_1 = qw_2 = qw_3 = 0.01$, $r_1 = r_2 = r_3 = 0.45$	27.82	56.59
$p_1 = 0.60$, $p_2 = p_3 = 0.31$	22.23	46.11
$p_1 = 0.70$, $p_2 = p_3 = 0.36$	18.50	38.91
$p_1 = 0.80$, $p_2 = p_3 = 0.41$	15.83	33.66
$p_1 = 0.90$, $p_2 = p_3 = 0.46$	13.83	29.69

Таблица 3. Среднее время работы системы при $d = (m - s)/2$, $p_2 > p_1$ и $p_2 > p_3$

Вероятности операций	Метод разделения памяти	
	Разделение пополам ($s = 50$, $d = 25$)	Оптимальное разделение ($s = 46$, $d = 27$)
$p_2 = 0.50$, $p_1 = p_3 = 0.26$, $q_2 = 0.02$, $q_1 = q_3 = 0.26$, $r_1 = r_2 = r_3 = 0.45$	32.78	36.73
$p_2 = 0.60$, $p_1 = p_3 = 0.31$	27.18	30.31
$p_2 = 0.70$, $p_1 = p_3 = 0.36$	23.23	25.85
$p_2 = 0.80$, $p_1 = p_3 = 0.41$	20.31	22.54
$p_2 = 0.90$, $p_1 = p_3 = 0.46$	18.05	19.98

В целом, оптимальные значения разделения памяти s для каждого набора вероятностей совпадают. Для примера, опишем первый результат из табл. 2. Вероятность добавления указателя в активный конец Sk_1 дека De_1 составляет $p_1 = 0.50$. Вероятность того, что указатель будет добавлен в конец (Sk_2, Sk_3) другого дека (De_2, De_3) равна $p_2 = p_3 = 0.26$. Если память первого уровня ($m = 100$) разделена пополам, то участкам памяти выделены следующие размеры: участок, в котором расположены Sk_1 и Qe, $s = 50$ единиц; участок, в котором расположен Sk_2, $d = 25$ единиц; участок, в котором расположен Sk_3, $m - s - d = 25$ единиц. Если память разделена оптимально, то размеры участков равны $s = 66$, $d = 17$ и $m - s - d = 17$. Среднее время работы системы до перераспределения памяти первого уровня, где память разделена пополам — $t = 27.82$ операций; системы, где память разделена оптимально — $t = 56.59$ операций. Таким образом, система с оптимально разделенной памятью совершает на 28.77 операций больше, чем система у которой память разделена пополам.

Таблица 4. Среднее время работы системы при $d = (m - s)/2$, $p_1 > p_3$ и $p_2 > p_3$

Вероятности операций	Метод разделения памяти	
	Разделение пополам ($s = 50$, $d = 25$)	Оптимальное разделение ($s = 54$, $d = 23$)
$p_1 = p_2 = 0.50$, $p_3 = 0.26$, $q_1 = q_2 = 0.02$, $q_3 = 0.26$, $r_1 = r_2 = r_3 = 0.45$	25.78	27.89
$p_1 = p_2 = 0.60$, $p_3 = 0.31$	21.08	23.17
$p_1 = p_2 = 0.70$, $p_3 = 0.36$	17.84	19.89
$p_1 = p_2 = 0.80$, $p_3 = 0.41$	15.48	17.48
$p_1 = p_2 = 0.90$, $p_3 = 0.46$	13.66	15.66

Таблица 5. Среднее время работы системы при $d = (m - s)/2$, $p_2 > p_1$ и $p_3 > p_1$

Вероятности операций	Метод разделения памяти	
	Разделение пополам ($s = 50$, $d = 25$)	Оптимальное разделение ($s = 45$, $d = 27$)
$p_2 = p_3 = 0.50$, $p_1 = 0.26$, $q_2 = q_3 = 0.02$, $q_1 = 0.26$, $r_1 = r_2 = r_3 = 0.45$	29.44	33.43
$p_2 = p_3 = 0.60$, $p_1 = 0.31$	24.68	27.91
$p_2 = p_3 = 0.70$, $p_1 = 0.36$	21.34	24.04
$p_2 = p_3 = 0.80$, $p_1 = 0.41$	18.90	21.19
$p_2 = p_3 = 0.90$, $p_1 = 0.46$	17.07	19.04

Таблица 6. Среднее время работы системы при $d = (m - s)/2$, $p_1 = p_2 = p_3$

Вероятности операций	Метод разделения памяти	
	Разделение пополам ($s = 50$, $d = 25$)	Оптимальное разделение ($s = 52$, $d = 24$)
$p_1 = p_2 = p_3 = 0.50$, $q_1 = q_2 = q_3 = 0.02$, $r_1 = r_2 = r_3 = 0.45$	24.87	25.97
$p_1 = p_2 = p_3 = 0.60$	20.55	21.74
$p_1 = p_2 = p_3 = 0.70$	17.54	18.81
$p_1 = p_2 = p_3 = 0.80$	15.32	16.69
$p_1 = p_2 = p_3 = 0.90$	13.60	15.12

В табл. 7 и 8 приведены результаты для задачи нахождения оптимальных значений разделения памяти s и d. В этих таблицах рассматриваются только системы с большой вероятность добавления указателя во второй дек ($p_2 > p_1$, $p_2 > p_3$ и $p_1 > p_3$, $p_2 > p_3$): оптимальное разделение s, разделение d и время работы t для других наборов вероятностей

совпадает с результатами в предыдущих таблицах. Это можно объяснить тем, что значение разделения памяти d зависит от вероятностных характеристик дека De_2 и не имеет влияния, если вероятность добавления указателя во второй дек маленькая.

Таблица 7. Среднее время работы системы при $d = max$, $p_2 > p_1$ и $p_2 > p_3$

Вероятности операций	Размер участка памяти $m - s$ ($s = 46$)	
	Половина участка ($d = 27$)	Оптимальный размер ($d = 37$)
$p_2 = 0.50$, $p_1 = p_3 = 0.26$, $q_2 = 0.02$, $q_1 = q_3 = 0.26$, $w_1 = w_2 = w_3 = 0.01$, $pw_1 = pw_2 = pw_3 = 0.01$, $qw_1 = qw_2 = qw_3 = 0.01$, $r_1 = r_2 = r_3 = 0.45$	36.73	51.92
$p_2 = 0.60$, $p_1 = p_3 = 0.31$	30.31	43.02
$p_2 = 0.70$, $p_1 = p_3 = 0.36$	25.85	36.72
$p_2 = 0.80$, $p_1 = p_3 = 0.41$	22.54	32.03
$p_2 = 0.90$, $p_1 = p_3 = 0.46$	19.98	28.42

Таблица 8. Среднее время работы системы при $d = max$, $p_1 > p_3$ и $p_2 > p_3$

Вероятности операций	Размер участка памяти $m - s$ ($s = 54$)	
	Половина участка ($d = 23$)	Оптимальный размер ($d = 29$)
$p_1 = p_2 = 0.50$, $p_3 = 0.26$, $q_1 = q_2 = 0.02$, $q_3 = 0.26$, $r_1 = r_2 = r_3 = 0.45$	27.89	34.03
$p_1 = p_2 = 0.60$, $p_3 = 0.31$	23.17	27.95
$p_1 = p_2 = 0.70$, $p_3 = 0.36$	19.89	23.74
$p_1 = p_2 = 0.80$, $p_3 = 0.41$	17.48	20.67
$p_1 = p_2 = 0.90$, $p_3 = 0.46$	15.66	18.31

Аналогично с предыдущей задачей, система с оптимально разделенной памятью работает дольше. Например, проанализируем первый результат из табл. 7. Если участок памяти первого уровня $m - s = 54$ разделен пополам ($d = m - s - d = 27$), то система совершает в среднем $t = 36.73$ операций до перераспределения памяти. Если память разделена оптимально: участку, в котором расположены Sk_1 и Qe выделяется $s = 46$ единиц; участку, в котором расположен Sk_2 — $d = 37$ единиц; участку, в котором находится Sk_3 выделяется $m - s - d = 17$ единиц. Среднее время работы такой системы составляет $t = 51.92$ операций, т.е. она работает в среднем на 15.19 операций дольше. Зависимость среднего времени работы этой системы t от разделения d участка памяти $m - s$ представлена в виде графика на рис. 5.

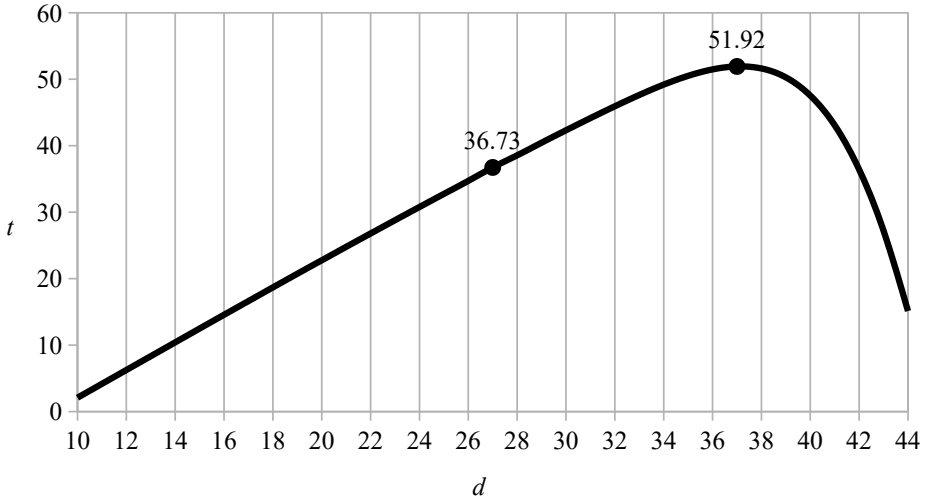


Рис. 5. Зависимость времени работы t от разделения памяти d

Заключение

В статье описывается один из методов представления трех work-stealing деков в двухуровневой памяти, где концы деков расположены в отдельных участках памяти. На основе этого метода была построена имитационная модель и решены задачи оптимального разделения памяти первого уровня между тремя деками для различных теоретических компьютерных систем. В качестве критерия оптимальности используется максимальное среднее время работы системы до перераспределения памяти.

Системы, где участки памяти разделены оптимально, работают дольше систем, память которых разделена поровну. Так, в зависимости от вероятностных характеристик, среднее время работы может увеличится на треть или в два раза.

Задача разделения памяти между деками может возникнуть во время разработки системного программного обеспечения. В таком случае, на основе предложенной модели и предварительного статистического исследования системы, можно найти оптимальные значения разделения быстрой памяти первого уровня.

В будущем можно провести новые исследования не только для других наборов входных данных, но и для других критериев оптимальности, например, минимизация затрат на перераспределение памяти. Также стоит заметить, что в этом методе концы деков расположены в отдельных участках памяти — возможно увеличить число структур данных не изменяя количества участков: концы новых деков следует располагать в участках памяти вместе с уже существующими, пустив их на встречу друг другу.

Литература

1. Dinu S., Raicu G. Load Balancing in Parallel Computing: An Evolutionary Approach // Advanced Topics in Optoelectronics, Microelectronics, and Nanotechnologies XI. Vol. 12493 / ed. by M. Vladescu, R.D. Tamas, I. Cristea. International Society for Optics, Photonics. SPIE, 2023. P. 124931M. DOI: 10.1117/12.2643056.

2. Alam M., Varshney A.K. A New Approach of Dynamic Load Balancing Scheduling Algorithm for Homogeneous Multiprocessor System // International Journal of Applied Evolutionary Computation. 2016. Vol. 7, no. 2. P. 61–75. DOI: 10.4018/IJAEC.2016040104.

3. Shafiq D., Jhanjhi N., Abdullah A. Load Balancing Techniques in Cloud Computing Environment: A Review // Journal of King Saud University – Computer and Information Sciences. 2021. Vol. 34. DOI: 10.1016/j.jksuci.2021.02.007.

4. Амелина Н.О., Корнивец А.Д., Иванский Ю.В., Тюшев К.И. Применение консенсусного протокола для балансировки загрузки стохастической децентрализованной сети при передаче данных // XII Всероссийское совещание по проблемам управления: Труды научной конференции, Москва, 16–19 июня 2014. Москва: ИПУ РАН, 2014. С. 8902–8911.

5. Alakeel A.M. A Guide to Dynamic Load Balancing in Distributed Computer Systems // International Journal of Computer Science and Network Security. 2010. Vol. 10, no. 6. P. 153–160.

6. Xia Y., Prasanna V.K., Li J. Hierarchical Scheduling of DAG Structured Computations on Manycore Processors with Dynamic Thread Grouping // Job Scheduling Strategies for Parallel Processing / ed. by E. Frachtenberg, U. Schwiegelshohn. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. P. 154–174. DOI: 10.1007/978-3-642-16505-4_9.

7. Yang J., He Q. Scheduling Parallel Computations by Work Stealing: A Survey // International Journal of Parallel Programming. 2018. Vol. 46, no. 2. P. 173–197. DOI: 10.1007/s10766-016-0484-8.

8. Hwu W.-m. GPU Computing Gems, Jade Edition. 2012. DOI: 10.1016/C2010-0-68654-8.

9. Arora N.S., Blumofe R.D., Plaxton C.G. Thread Scheduling for Multiprogrammed Multiprocessors // Proceedings of the Tenth Annual ACM Symposium on Parallel Algorithms and Architectures. Puerto Vallarta, Mexico: Association for Computing Machinery, 1998. P. 119–129. SPAA '98. DOI: 10.1145/277651.277678.

10. Knuth D.E. The Art of Computer Programming, Volume 1 (3rd ed.): Fundamental Algorithms. USA: Addison Wesley Longman Publishing Co., Inc., 1997. DOI: 10.5555/260999.

11. Lin C.-X., Huang T.-W., Wong M.D.F. An Efficient Work-Stealing Scheduler for Task Dependency Graph // 2020 IEEE 26th International Conference on Parallel and Distributed Systems (ICPADS). 2020. P. 64–71. DOI: 10.1109/ICPADS51040.2020.00018.

12. Li J., Agrawal K., Elnikety S., *et al.* Work Stealing for Interactive Services to Meet Target Latency // SIGPLAN Notices. New York, NY, USA, 2016. Vol. 51, no. 8. Article 14. DOI: 10.1145/3016078.2851151.

13. Gmys J., Leroy R., Mezmaz M., *et al.* Work Stealing with Private Integer-Vector-Matrix Data Structure for Multi-Core Branch-and-Bound Algorithms // Concurrency and Computation: Practice and Experience. 2016. Vol. 28, no. 18. P. 4463–4484. DOI: 10.1002/cpe.3771.

14. Wimmer M., Versaci F., Traff J.L., *et al.* Data Structures for Task-Based Priority Scheduling // SIGPLAN Notices. New York, NY, USA, 2014. Vol. 49, no. 8. P. 379–380. DOI: 10.1145/2692916.2555278.

15. Аксенова Е.А., Соколов А.В. Методы управления work-stealing деками в динамических планировщиках многопроцессорных параллельных вычислений // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2023. Т. 12, № 4. С. 76–93. DOI: 10.14529/cmse230403.

16. Kuchumov R., Sokolov A., Korkhov V. Staccato: Cache-Aware Work-Stealing Task Scheduler for Shared-Memory Systems // Computational Science and Its Applications – ICCSA 2018 / ed. by O. Gervasi, B. Murgante, S. Misra, *et al.* Cham: Springer International Publishing, 2018. P. 91–102. DOI: 10.1007/978-3-319-95171-3_8.

17. Kuchumov R., Sokolov A., Korkhov V. Staccato: Shared-Memory Work-Stealing Task Scheduler with Cache-Aware Memory Management // International Journal of Web and Grid Services. 2019. Vol. 15, no. 4. P. 394–407. DOI: 10.1504/IJWGS.2019.103233.

18. Аксенова Е.А., Лазутина А.А., Соколов А.В. Минимизация средних затрат на перераспределение при работе с work-stealing деком в двухуровневой памяти // Программные системы: теория и приложения. 2021. Т. 12, № 2. С. 53–71. DOI: 10.25209/2079-3316-2021-12-2-53-71.

19. Лазутина А.А., Соколов А.В. Об оптимальном управлении Work-Stealing деками в двухуровневой памяти // Вестник компьютерных и информационных технологий. 2020. Т. 17, № 4. С. 51–60. DOI: 10.14489/vkit.2020.04.pp.051-060.

20. Aksenova E.A., Lazutina A.A., Sokolov A.V. About Optimal Management of Work-Stealing Deques in Two-Level Memory // Lobachevskii Journal of Mathematics. 2021. Vol. 42, no. 7. P. 1475–1482. DOI: 10.1134/S1995080221070027.

21. Аксенова Е.А., Барковский Е.А., Соколов А.В. Оптимальное управление тремя work-stealing деками в двухуровневой памяти // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2024. Т. 13, № 3. С. 47–60. DOI: 10.14529/cmse240303.

Аксёнова Елена Алексеевна, к.ф.-м.н., лаборатория информационных компьютерных технологий, Институт прикладных математических исследований Карельского научного центра Российской академии наук (Петрозаводск, Российская Федерация)

Барковский Евгений Александрович, независимый исследователь (Петрозаводск, Российская Федерация)

Соколов Андрей Владимирович, д.ф.-м.н., профессор, лаборатория информационных компьютерных технологий, Институт прикладных математических исследований Карельского научного центра Российской академии наук (Петрозаводск, Российская Федерация)

DOI: 10.14529/cmse250302

A REPRESENTATION METHOD OF THREE WORK-STEALING DEQUES IN TWO-LEVEL MEMORY

© 2025 E.A. Aksenova[1], E.A. Barkovsky, A.V. Sokolov[1]

[1] *Institute of Applied Mathematical Research of the Karelian Research Centre of the Russian Academy of Sciences (str. Pushkinskaya 11, Petrozavodsk, 185910 Russia)*

E-mail: aksenova@krc.karelia.ru, barkevgen@gmail.com, avs@krc.karelia.ru

Received: 21.07.2025

The efficiency of distributed systems largely depends on the uniformity of thread workload. To achieve this, various strategies for balancing tasks between threads are utilized. One of the methods of dynamic distributed load balancing, where each thread participates in task distribution, is called "work-stealing." In this method, a thread without tasks steals them from other threads. This process is based on a special data structure, a work-stealing deque, where pointers to tasks are stored. Thus, the problem of efficient placement of deques in memory arises. A deque can be represented in two-level memory in a divided form: the often-accessed ends of a deque are placed in the fast memory of the first level, while its middle is left in the slow memory of the second level. This way, the system can quickly access the ends of the deque where elements are being actively added and removed. In this paper, a new representation method of three work-stealing deques in two-level memory where the ends of the deques are placed in separate memory areas is described. We propose a simulation model of this approach based on the Monte Carlo method and solve several problems of optimal partitioning of memory. The optimality criterion is the maximum mean system operating time to memory reallocation.

Keywords: data structures, Monte Carlo method, two-level memory, work-stealing deques.

FOR CITATION

Aksenova E.A., Barkovsky E.A., Sokolov A.V. A Representation Method of Three Work-Stealing Deques in Two-Level Memory. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2025. Vol. 14, no. 3. P. 28–41. (in Russian) DOI: 10.14529/cmse250302.

This paper is distributed under the terms of the Creative Commons Attribution-Non Commercial 4.0 License which permits non-commercial use, reproduction and distribution of the work without further permission provided the original work is properly cited.

References

1. Dinu S., Raicu G. Load Balancing in Parallel Computing: An Evolutionary Approach. Advanced Topics in Optoelectronics, Microelectronics, and Nanotechnologies XI. Vol. 12493 / ed. by M. Vladescu, R.D. Tamas, I. Cristea. International Society for Optics, Photonics. SPIE, 2023. P. 124931M. DOI: 10.1117/12.2643056.
2. Alam M., Varshney A.K. A New Approach of Dynamic Load Balancing Scheduling Algorithm for Homogeneous Multiprocessor System. International Journal of Applied Evolutionary Computation. 2016. Vol. 7, no. 2. P. 61–75. DOI: 10.4018/IJAEC.2016040104.
3. Shafiq D., Jhanjhi N., Abdullah A. Load Balancing Techniques in Cloud Computing Environment: A Review. Journal of King Saud University – Computer and Information Sciences. 2021. Vol. 34. DOI: 10.1016/j.jksuci.2021.02.007.
4. Amelina N.O., Kornivec A.D., Ivanskij J.V., Tjushev K.I. Primenenie konsensusnogo protokola dlja balansirovki zagruzki stohasticheskoj decentralizovannoj seti pri peredache dan-

nyh. XII Vserossijskoe soveshhanie po problemam upravlenija: Scientific Conference Proceedings, Moscow, Russia, June 16–19, 2014. Moscow: ICS of RAS, 2014. P. 8902–8911. (in Russian).

5. Alakeel A.M. A Guide to Dynamic Load Balancing in Distributed Computer Systems. International Journal of Computer Science and Network Security. 2010. Vol. 10, no. 6. P. 153–160.
6. Xia Y., Prasanna V.K., Li J. Hierarchical Scheduling of DAG Structured Computations on Manycore Processors with Dynamic Thread Grouping. Job Scheduling Strategies for Parallel Processing / ed. by E. Frachtenberg, U. Schwiegelshohn. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. P. 154–174. DOI: 10.1007/978-3-642-16505-4_9.
7. Yang J., He Q. Scheduling Parallel Computations by Work Stealing: A Survey. International Journal of Parallel Programming. 2018. Vol. 46, no. 2. P. 173–197. DOI: 10.1007/s10766-016-0484-8.
8. Hwu W.-m. GPU Computing Gems, Jade Edition. 2012. DOI: 10.1016/C2010-0-68654-8.
9. Arora N.S., Blumofe R.D., Plaxton C.G. Thread Scheduling for Multiprogrammed Multiprocessors. Proceedings of the Tenth Annual ACM Symposium on Parallel Algorithms and Architectures. Puerto Vallarta, Mexico: Association for Computing Machinery, 1998. P. 119–129. SPAA '98. DOI: 10.1145/277651.277678.
10. Knuth D.E. The Art of Computer Programming, Volume 1 (3rd ed.): Fundamental Algorithms. USA: Addison Wesley Longman Publishing Co., Inc., 1997. DOI: 10.5555/260999.
11. Lin C.-X., Huang T.-W., Wong M.D.F. An Efficient Work-Stealing Scheduler for Task Dependency Graph. 2020 IEEE 26th International Conference on Parallel and Distributed Systems (ICPADS). 2020. P. 64–71. DOI: 10.1109/ICPADS51040.2020.00018.
12. Li J., Agrawal K., Elnikety S., *et al.* Work Stealing for Interactive Services to Meet Target Latency. SIGPLAN Notices. New York, NY, USA, 2016. Vol. 51, no. 8. Article 14. DOI: 10.1145/3016078.2851151.
13. Gmys J., Leroy R., Mezmaz M., *et al.* Work Stealing with Private Integer-Vector-Matrix Data Structure for Multi-Core Branch-and-Bound Algorithms. Concurrency and Computation: Practice and Experience. 2016. Vol. 28, no. 18. P. 4463–4484. DOI: 10.1002/cpe.3771.
14. Wimmer M., Versaci F., Traff J.L., *et al.* Data Structures for Task-Based Priority Scheduling. SIGPLAN Notices. New York, NY, USA, 2014. Vol. 49, no. 8. P. 379–380. DOI: 10.1145/2692916.2555278.
15. Aksenova E.A., Sokolov A.V. Control Methods of Work-Stealing Deques in Dynamic Schedulers of Multiprocessor Parallel Computations. Bulletin of the SUSU. Series: Computational Mathematics and Software Engineering. 2023. Vol. 12, no. 4. P. 76–93. (in Russian) DOI: 10.14529/cmse230403.
16. Kuchumov R., Sokolov A., Korkhov V. Staccato: Cache-Aware Work-Stealing Task Scheduler for Shared-Memory Systems. Computational Science and Its Applications – ICCSA 2018 / ed. by O. Gervasi, B. Murgante, S. Misra, *et al.* Cham: Springer International Publishing, 2018. P. 91–102. DOI: 10.1007/978-3-319-95171-3_8.

17. Kuchumov R., Sokolov A., Korkhov V. Staccato: Shared-Memory Work-Stealing Task Scheduler with Cache-Aware Memory Management. International Journal of Web and Grid Services. 2019. Vol. 15, no. 4. P. 394–407. DOI: 10.1504/IJWGS.2019.103233.

18. Aksenova E.A., Lazutina A.A., Sokolov A.V. About Optimal Managment of Work-Stealing Deques in Two-Level Memory. Program Systems: Theory and Applications. 2021. Vol. 12, no. 2. P. 53–71. (in Russian) DOI: 10.25209/2079-3316-2021-12-2-53-71.

19. Lazutina A.A., Sokolov A.V. About Optimal Management of Work-Stealing Deques in Two-Level Memory. Herald of Computer and Information Technologies. 2020. Vol. 17, no. 4. P. 51–60. (in Russian) DOI: 10.14489/vkit.2020.04.pp.051-060.

20. Aksenova E.A., Lazutina A.A., Sokolov A.V. About Optimal Management of Work-Stealing Deques in Two-Level Memory. Lobachevskii Journal of Mathematics. 2021. Vol. 42, no. 7. P. 1475–1482. DOI: 10.1134/S1995080221070027.

21. Aksenova E.A., Barkovsky E.A., Sokolov A.V. Optimal Control of Three Work-Stealing Deques Located in Two-Level Memory. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2024. Vol. 13, no. 3. P. 47–60. (in Russian) DOI: 10.14529/cmse240303.

УДК 681.125 **DOI: 10.14529/cmse250303**

ПОВЫШЕНИЕ ТОЧНОСТИ ПОКАЗАНИЙ ВИХРЕАКУСТИЧЕСКИХ РАСХОДОМЕРОВ ЗА СЧЕТ ВЫСОКОТОЧНОЙ ОЦЕНКИ ЧАСТОТЫ ВИХРЕОБРАЗОВАНИЯ

© 2025 О.Л. Ибряева, А.Д. Яковенко, В.В. Синицин, А.Л. Шестаков

Южно-Уральский государственный университет

(454080 Челябинск, пр. им. В.И. Ленина, д. 76)

E-mail: ibriaevaol@susu.ru, iakovenkoad@susu.ru, sinitcinvv@susu.ru, a.l.shestakov@susu.ru

Поступила в редакцию: 22.08.2025

В статье рассматривается задача повышения точности вихреакустических расходомеров за счет высокоточной оценки частоты вихреобразования в условиях коротких временных окон и зашумленных сигналов. Традиционные методы, основанные на быстром преобразовании Фурье (БПФ), сталкиваются с фундаментальным ограничением разрешения при анализе коротких интервалов, что снижает их эффективность в динамических режимах измерений. В качестве альтернативы предложен модифицированный метод матричных пучков (ММП), относящийся к параметрическим методам высокого разрешения. Метод позволяет моделировать сигнал как сумму комплексных экспонент и обеспечивает устойчивую оценку частоты даже при низком отношении сигнал/шум. Проведено сравнение ММП и БПФ на модельных и экспериментальных сигналах с вихреакустического расходомера. Показано, что ММП обеспечивает более стабильные оценки частоты: стандартное отклонение уменьшается в среднем в 1.5 раза. При этом вычислительная сложность метода оказывается сопоставимой или даже ниже за счет малой длины анализируемых окон. Полученные результаты демонстрируют потенциал ММП для создания алгоритмов автоматического контроля достоверности показаний средств измерений. Метод может быть положен в основу систем самодиагностики и коррекции погрешностей, вызванных нестационарностью потока, вибрациями или наличием двухфазного течения.

Ключевые слова: вихреакустический расходомер, оценка частоты, метод матричных пучков, высокое разрешение, обработка сигналов, достоверность измерений, двухфазный поток, параметрические методы.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Ибряева О.Л., Яковенко А.Д., Синицин В.В., Шестаков А.Л. Повышение точности показаний вихреакустических расходомеров за счет высокоточной оценки частоты вихреобразования // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2025. Т. 14, № 3. С. 42–58. DOI: 10.14529/cmse250303.

Введение

Измерение расхода жидкостей и газов является одной из ключевых задач в нефтегазовой, энергетической, химической и пищевой промышленности. Среди разнообразия средств измерений вихреакустические расходомеры занимают особое место благодаря своей простоте, надежности и отсутствию движущихся частей. Принцип их работы основан на явлении периодического срыва вихрей Кармана за телом обтекания, частота которых прямо пропорциональна скорости потока. Точное определение этой частоты является основой высокой метрологической точности прибора.

Однако в реальных условиях измерительный сигнал подвержен шумам, вибрациям и нестационарности, что затрудняет точную оценку частоты, особенно при необходимости высокого временного разрешения. Традиционные методы спектрального анализа, такие как

быстрое преобразование Фурье (БПФ), обладают фундаментальным ограничением: разрешение по частоте обратно пропорционально длительности анализируемого интервала. Это означает, что при коротких окнах анализа, необходимых для отслеживания быстрых изменений расхода, точность оценки частоты резко падает.

Для преодоления этого ограничения все большее внимание привлекают параметрические методы высокого разрешения, способные оценивать частоту сигнала с высокой точностью даже на малых интервалах наблюдения. К ним относится метод матричных пучков (Matrix Pencil Method, MPM, ММП), позволяющий моделировать сигнал как сумму комплексных экспонент и находить его параметры с помощью разложения структурированных матриц. В работе [1] предложена его модификация, повышающая устойчивость к шуму за счет критерия согласованности полюсов и обратных к ним.

Актуальность данной работы обусловлена задачей обеспечения достоверности показаний средств измерений в условиях цифровой промышленности. Поскольку расход вычисляется на основе измеренной частоты вихреобразования, любая погрешность в ее оценке напрямую влияет на точность конечного результата. Более того, в условиях нестационарных процессов, пульсирующих потоков или внешних помех, традиционные методы могут давать систематические ошибки, что ставит под сомнение достоверность измерений. В этом контексте разработка методов высокоточной обработки сигнала становится ключевым элементом автоматического контроля достоверности и коррекции результатов измерений — одной из центральных задач современных цифровых измерительных систем.

В работе проведено исследование модифицированного метода матричных пучков как инструмента повышения точности и устойчивости оценки частоты. Показано, что применение ММП позволяет существенно снизить разброс оценок частоты по сравнению с БПФ, особенно при коротких временных окнах, что напрямую способствует повышению метрологической надежности расходомера. Полученные результаты могут быть использованы для построения алгоритмов самодиагностики и автоматической коррекции показаний в «умных» измерительных системах.

Цель данной работы — исследование эффективности модифицированного метода матричных пучков для обработки сигналов вихреакустического расходомера в сравнении с классическим подходом на основе БПФ. Проведено сопоставление методов на модельных и экспериментальных данных, включая оценку точности, стабильности и вычислительной сложности в условиях коротких окон анализа.

Статья организована следующим образом. В разделе 1 представлен обзор современных подходов к обработке сигналов в вихревых расходомерах. В разделе 2 описан принцип работы вихреакустического расходомера и поставлена задача оценки частоты вихреобразования. Раздел 3 посвящен изложению модифицированного метода матричных пучков. В разделе 4 представлены результаты моделирования и обработки реальных сигналов. В заключении подведены итоги и обозначены направления дальнейших исследований.

1. Обзор литературы

Вихреакустические расходомеры занимают важное место среди средств измерений благодаря своей простоте, надежности и отсутствию движущихся частей [2]. Однако их метрологические характеристики могут существенно ухудшаться под влиянием внешних и внутренних факторов, таких как шум, вибрации, нестационарность потока и, что особенно важ-

но, нарушение однородности среды — например, при наличии газовых включений в жидком потоке.

В условиях однофазного потока и при стабильных гидродинамических режимах вихревые расходомеры демонстрируют высокую точность [3]. Однако в реальных промышленных системах часто возникают двухфазные режимы течения: пузырьковый, пробковый, туманный и др., которые нарушают структуру вихревого следа и приводят к систематическим погрешностям измерений [4]. Как показано в работе [4], даже при низком объемном содержании газа (1–8%) происходит изменение частоты вихреобразования и снижение энергии вихревого сигнала, что обусловлено как искажением формы вихрей, так и снижением амплитуды пульсаций давления. Эти эффекты делают классическую калибровку прибора неадекватной и требуют введения поправок. Данный вывод подтверждается результатами моделирования, показывающими, что наличие вихря вблизи приемопередатчика может вызывать ошибку в измерении инструментального коэффициента до 17% [5].

Для компенсации погрешностей предлагаются различные подходы [4, 6, 7], не требующие изменения конструкции прибора, но предъявляющие повышенные требования к достоверности оценки частоты вихреобразования, поскольку любая ошибка в ее определении будет напрямую влиять на точность коррекции. В частности, для низких расходов (0.5–50 $м^3/ч$) предлагается использование адаптивных алгоритмов на основе скользящего среднего и экспоненциального сглаживания [2, 8], а также методов, основанных на анализе стабильности частотной последовательности, а не мгновенных значений [9]. Последнее исследование [9] показывает, что коэффициент вариации частоты вихреобразования может достигать 3–9% в зависимости от режима течения, что подтверждает необходимость статистической обработки данных и проверки их на нормальность.

Традиционно для анализа сигнала в вихревых расходомерах применяется быстрое преобразование Фурье, обеспечивающее устойчивую работу в стационарных режимах [10]. Однако его применение в условиях коротких временных окон, шумов или нестационарности ограничено из-за фундаментального компромисса между временным и частотным разрешением. Для повышения точности в ряде работ используются методы интерполяции спектральных пиков [11, 12], однако их эффективность резко падает при низком отношении сигнал/шум. Альтернативой является адаптивное использование автокорреляции для низких частот ($<$200 Гц) и БПФ для высоких частот ($>$200 Гц) в сочетании с перестраиваемым цифровым полосовым фильтром, что позволяет достигать ошибки измерения менее 0.3% на средних и высоких расходах и около 1.1% на самых низких расходах, хотя и с задержкой оценки около 0.512 с при резких изменениях потока [13].

В этих условиях все большее внимание привлекают параметрические методы высокого разрешения, способные оценивать частоту сигнала с высокой точностью даже на коротких интервалах. К ним относятся метод Прони [14], MUSIC [15], ESPRIT [16] и метод матричных пучков [17]. Эти методы моделируют сигнал как сумму затухающих синусоид и позволяют достигать сверхразрешения за счет использования структуры сигнального подпространства. В частности, ММП отличается относительной простотой реализации и хорошей устойчивостью к шуму при корректном выборе параметров. Также применяются такие методы, как периодограмма и оценка спектральной плотности Уэлча [18], преобразование Гильберта-Хуанга (Hilbert-Huang Transform) и эмпирическая модовая декомпозиция для выделения слабых вихревых сигналов на низких расходах [19].

Однако в условиях сильного шума часть оцененных полюсов может быть ложной. Для повышения надежности в [1] предложена модификация ММП, основанная на критерии согласованности оценок для полюсов и обратных к ним. Этот подход позволяет эффективно подавлять шумовые компоненты и повышать достоверность оценки частоты, что особенно важно в условиях ослабленного сигнала, как в случае двухфазного потока [4].

Кроме того, в работах [20–22] показано, что сигнал вихреобразования может использоваться не только для измерения расхода, но и для распознавания режимов течения с помощью вейвлет-анализа, эмпирической модовой декомпозиции и нейронных сетей. Это открывает путь к созданию интеллектуальных расходомеров, способных не только измерять, но и диагностировать состояние потока.

Важным аспектом является также аппаратное улучшение метрологических характеристик. Показано, что оптимизация геометрии обтекаемого тела в ультразвуковых вихревых расходомерах может привести к 12-кратному повышению чувствительности [23], а использование аналогового предварительного фильтра для пьезоэлектрических датчиков — к значительному подавлению шумов [24].

Таким образом, существует четкая тенденция к переходу от простых измерительных приборов к системам с автоматическим контролем достоверности показаний, и в этом контексте высокоточная оценка частоты вихреобразования с помощью модифицированного ММП может служить основой для алгоритмов самодиагностики и коррекции показаний.

2. Принцип работы вихреакустического расходомера

Вихреакустические расходомеры широко применяются для измерения расхода жидких и газообразных сред благодаря простоте конструкции, отсутствию движущихся частей и высокой надежности. Принцип их работы основан на регистрации колебаний, возникающих вследствие генерации вихрей Кармана за обтекателем в потоке. Типичный расходомер представляет собой моноблочную конструкцию (рис. 1), состоящую из проточной части и электронного блока [25].

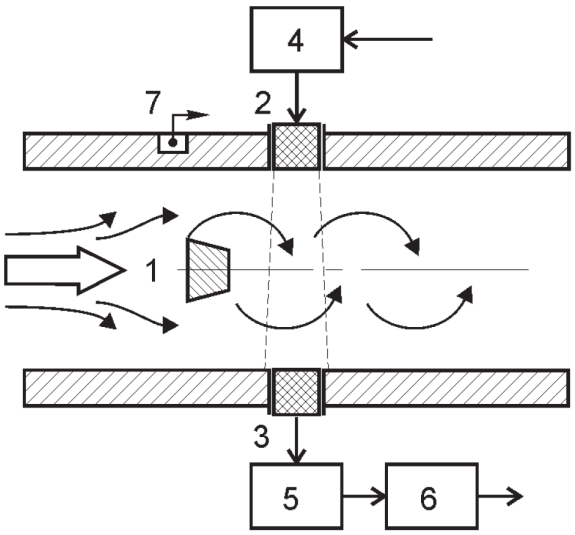


Рис. 1. Устройство расходомера: 1 — тело обтекания, 2 — пьезоизлучатель, 3 — пьезоприемник, 4 — генератор, 5 — фазовый детектор, 6 — блок формирования выходных сигналов, 7 — термодатчик

При обтекании потоком жидкости тела обтекания, за ним формируется вихревая дорожка, причем частота f пульсаций давления газа или жидкости в результате периодического срыва вихрей прямо пропорциональна скорости потока v, а, следовательно и расходу. Справедлива формула [3]:

$$f = Sh \cdot \frac{v}{d},$$

где Sh — число Струхаля (безразмерный коэффициент, зависящий от формы тела), d — характерный размер тела обтекания.

От генератора на пьезоизлучатель подается переменное напряжение, которое преобразуется в ультразвуковые колебания. При прохождении через поток, в результате взаимодействия с вихрями, ультразвуковые колебания модулируются по фазе. На пьезоприемнике модулированные ультразвуковые колебания вновь преобразуются в напряжение, которое подается на фазовый детектор.

На фазовом детекторе определяется разность фаз между сигналами с пьезоприемника и опорного генератора для однолучевых расходомеров (или сигналами с пьезоприемников первой и второй пары пьезоэлементов для двухлучевых расходомеров). Напряжение на выходе фазового детектора является практически синусоидальным сигналом с частотой, прямо пропорциональной скорости потока. Таким образом, задача точного определения частоты синусоидального вихреакустического сигнала является ключевым этапом измерительного процесса.

Традиционные методы спектрального анализа, основанные на быстром преобразовании Фурье, обладают рядом ограничений при решении этой задачи. В частности, существует фундаментальный компромисс между длительностью анализируемого окна и разрешением по частоте: сокращение окна, необходимое для повышения быстродействия измерений, неизбежно приводит к снижению точности определения частоты. Дополнительно на результаты влияют шум и возможные дрейфы частоты сигнала в пределах измерительного интервала.

Для преодоления этих ограничений целесообразно использовать методы высокоточного спектрального анализа, способные обеспечивать устойчивую оценку частоты на коротких временных интервалах. Перейдем к описанию одного из таких методов.

3. Модифицированный метод матричных пучков

Метод матричных пучков (Matrix Pencil Method, MPM, ММП) относится к параметрическим методам спектрального анализа, моделирующим сигнал как сумму комплексных экспонент:

$$x(n) = \sum_{k=1}^{M} R_k z_k^n,$$

где $R_k = A_k e^{j\varphi_k}$ — комплексные амплитуды, $z_k = e^{(\alpha_k + j2\pi f_k)T}$ — полюсы сигнала, T — период дискретизации, α_k — коэффициенты затухания, f_k — частоты гармоник.

В случае вихреакустического расходомера полезный сигнал на коротком интервале можно аппроксимировать одной доминирующей затухающей синусоидой, соответствующей основной частоте вихрей Кармана.

Классический ММП находит полюса z_k как собственные значения пучка матриц, сформированного из отсчетов сигнала. При отсутствии шума эти полюса дают точные оценки частоты и затухания синусоид. Однако в присутствии шума часть найденных полюсов будет

не связана с полезным сигналом. Чтобы отделить истинные полюса от ложных, в работе [1] был предложен модифицированный ММП, в котором дополнительно вычисляются оценки для обратных полюсов $1/z_k$.

Приведем основные этапы алгоритма.

1. Из N отсчетов сигнала $x(n)$ формируются две матрицы:

$$Y_0 = \begin{bmatrix} x(L-1) & x(L-2) & \dots & x(0) \\ \vdots & \vdots & \ddots & \vdots \\ x(N-2) & x(N-3) & \dots & x(N-L-1) \end{bmatrix},$$

$$Y_1 = \begin{bmatrix} x(L) & x(L-1) & \dots & x(1) \\ \vdots & \vdots & \ddots & \vdots \\ x(N-1) & x(N-2) & \dots & x(N-L) \end{bmatrix}.$$

   Параметр L выбирается в диапазоне $N/3 \leq L \leq 2N/3$. Показано [17], что при таком выборе дисперсия оценки полюсов z_k будет минимальна, т.е. ММП будет наименее чувствителен к шуму. Оказывается [1], что собственными числами пучков матриц $Y_0 - \lambda Y_1$, $Y_1 - \lambda Y_0$ будут z_k, $1/z_k$, соответственно. На этом факте и основан описываемый метод.
2. Находим усеченное до ранга M сингулярное разложение $Y_0 = U_0 S_0 V_0^H$. Число M определяется по числу ненулевых сингулярных чисел. В случае зашумленного сигнала ненулевых сингулярных чисел не будет, однако между первыми M и последующими сингулярными числами будет наблюдаться ярко выраженный скачок, который и позволит определить число комплексных экспонент в сигнале. Аналогично находим $Y_1 = U_1 S_1 V_1^H$.
3. Составляем матрицы:

$$Z_E = S_0^{-1} U_0^H Y_1 V_0,$$
$$Z_I = S_1^{-1} U_1^H Y_0 V_1.$$

   для оценки полюсов z_k и $1/z_k$, соответственно.
4. Находим собственные числа p_k, q_m матриц Z_E, Z_I и получаем тем самым оценки для полюсов z_k и $1/z_k$.
5. Определяем истинные полюсы сигнала следующим образом.
   Для каждого p_k выполняется поиск q_m, такое что:

$$\left| p_k - \frac{1}{q_m} \right| \leqslant \varepsilon.$$

   Если условие выполняется — полюс p_k признается истинным.
6. Находим частоту: $f_k = \frac{\arg(p_k)}{2\pi T}$.

Далее применим описанный метод к анализу модельного и реального сигнала с вихреакустического расходомера.

4. Экспериментальная часть

4.1. Анализ модельного сигнала

В этом параграфе мы сравним точность оценки частоты синусоидального сигнала классическим спектральным анализом на основе быстрого преобразования Фурье (БПФ) и модифицированным методом матричных пучков (ММП). В качестве тестового сигнала рас-

смотрим синусоиду с известной частотой $f_0 = 115$ Гц, дискретизированную с частотой $Fs = 2$ кГц и зашумленную аддитивным белым шумом с дисперсией, соответствующей отношению сигнал/шум SNR (Signal to Noise Ratio) около 7.5 дБ:

$$\mathrm{SNR} = 10\log_{10}\left(\frac{P_\mathrm{s}}{P_\mathrm{n}}\right).$$

Здесь $P_\mathrm{s} = \frac{1}{L}\sum_{n=1}^{L} s^2[n]$ — мощность полезного сигнала, $P_\mathrm{n} = \frac{1}{L}\sum_{n=1}^{L} w^2[n]$ — мощность шума, $s[n]$ — отсчеты полезного сигнала, $w[n]$ — отсчеты шума, L — число отсчетов сигнала. В качестве шума брались значения нормальной случайной величины с нулевым средним.

Длительность сигнала составляет 10 секунд. На рис. 2 приведен фрагмент в 0.1 секунды рассматриваемого сигнала.

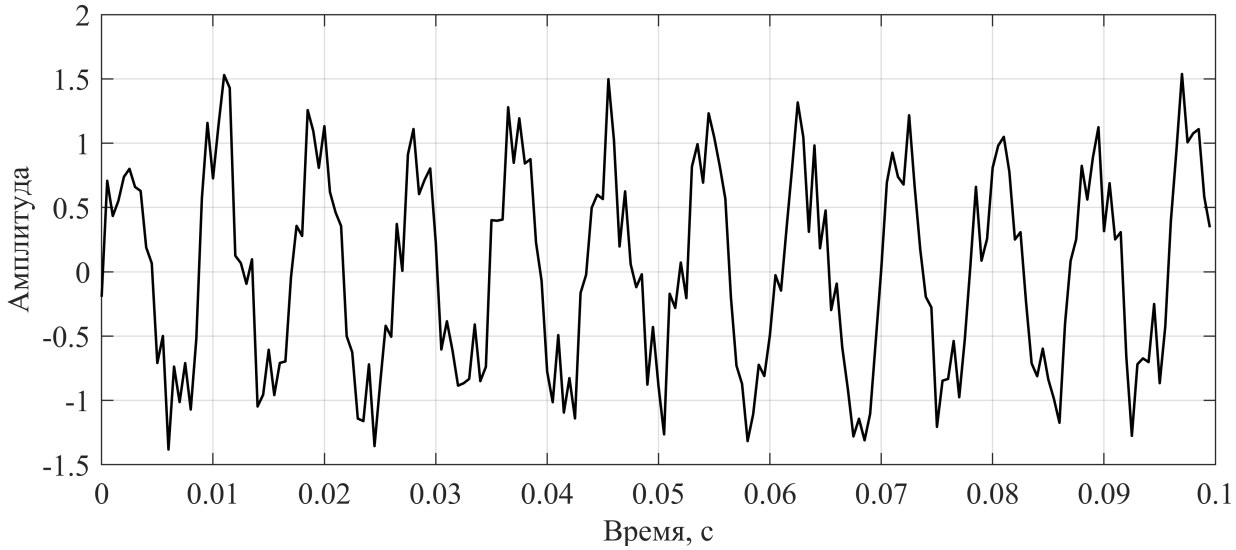


Рис. 2. Фрагмент модельного сигнала

Сравнение методов выполнялось в режиме скользящего окна для различных длительностей окна в диапазоне от 0.025 до 0.3 с. Для каждого значения длительности окна вычислялись среднее значение оценки частоты:

$$\overline{f} = \frac{1}{N}\sum_{i=1}^{N} f_i,$$

и среднеквадратичная ошибка $RMSE$ (Root Mean Squared Error) относительно истинной частоты $f_0 = 115$ Гц:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(f_i - f_0\right)^2}.$$

Здесь N — количество окон, f_i — оценка частоты в i-м окне.

Чтобы методу БПФ было легче «конкурировать» с ММП, для уточнения оценки частоты между дискретными бинами БПФ применялась параболическая интерполяция положения пика спектра. Если α, β, γ – амплитуды спектра в соседних частотных бинах (левом, максимальном и правом), то смещение пика относительно центрального бина оценивается как:

$$p = \frac{1}{2}\cdot\frac{\alpha - \gamma}{\alpha - 2\beta + \gamma}$$

и уточненная оценка частоты $\hat{f}$ определяется по формуле:

$$\hat{f} = f_{bin} + p \cdot \frac{Fs}{N_{FFT}},$$

где f_{bin} — частота бина с максимумом, N_{FFT} – число точек, используемое в вычислении БПФ и определяющее частотное разрешение спектра: $\Delta f = \frac{Fs}{N_{\text{FFT}}}$. Чем больше N_{FFT}, тем мельче частотный шаг и тем точнее можно определить частоту пика, но тем выше вычислительные затраты. В нашем эксперименте $N_{FFT} = 8192$.

На рис. 3 приведены отклонения среднего значения частоты $\overline{f} - f_0$ и $RMSE$ в зависимости от длины скользящего окна.

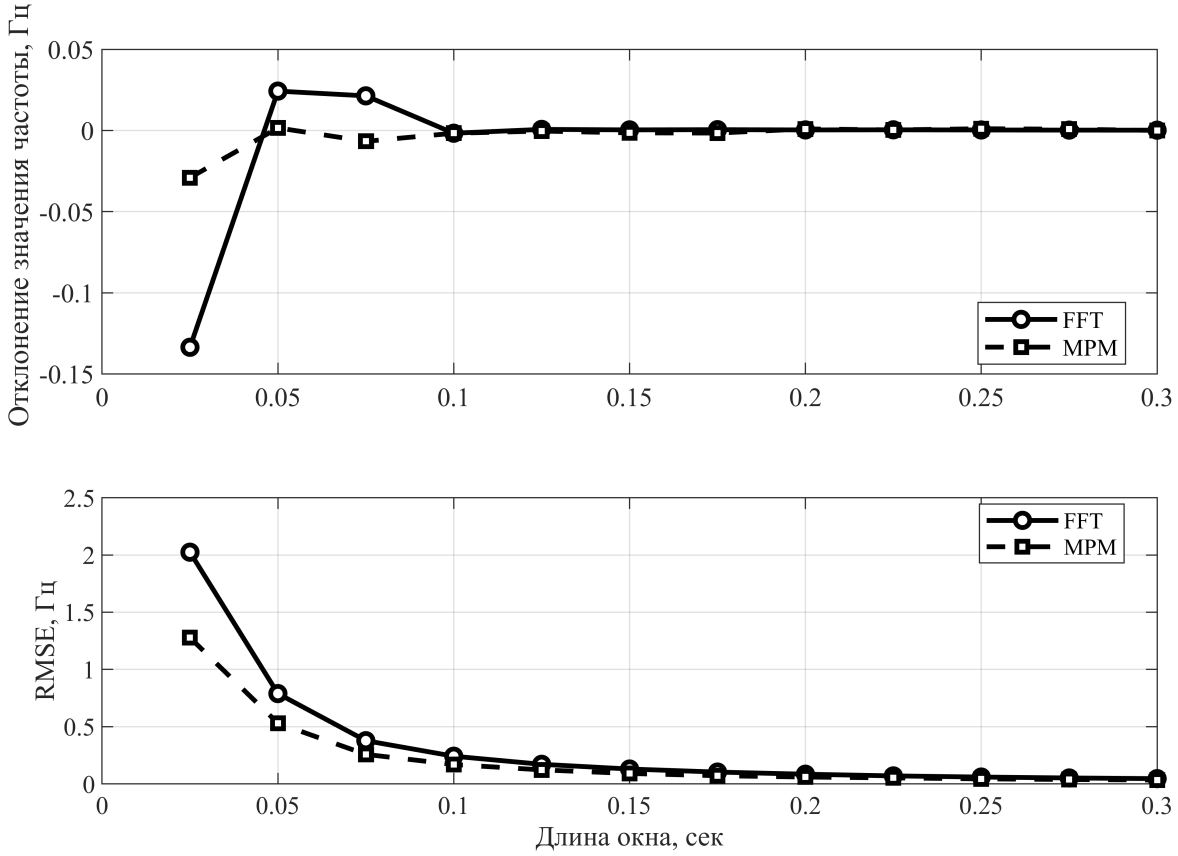


Рис. 3. К сравнению методов Фурье и ММП на модельном сигнале

Как можно видеть, при длительности окна менее 0.1 с модифицированный метод матричных пучков обеспечивает существенно более высокую точность оценки частоты по сравнению с классическим методом БПФ. Для коротких временных интервалов ММП демонстрирует меньшую систематическую погрешность и заметно более низкое значение $RMSE$, что указывает на его лучшую устойчивость к влиянию шума и малого времени наблюдения. В контексте вихреакустических расходомеров такая особенность особенно важна при необходимости отслеживания быстрых изменений расхода, когда доступные интервалы анализа ограничены.

4.2. Описание экспериментальной установки

Экспериментальные исследования вихреакустического расходомера проводились на поверочной установке, предназначенной для высокоточного измерения расхода жидкостей (рис. 4). Установка реализует два независимых метода измерения: весовой метод и метод непосредственного сличения с эталонными расходомерами, что позволяет охватить диапазон расходов от 0.0025 до 750 м^3/ч. Рабочей средой является вода с температурой от 10 до 30 °C, соответствующая нормированным показателям по мутности, общей жесткости и солесодержанию.

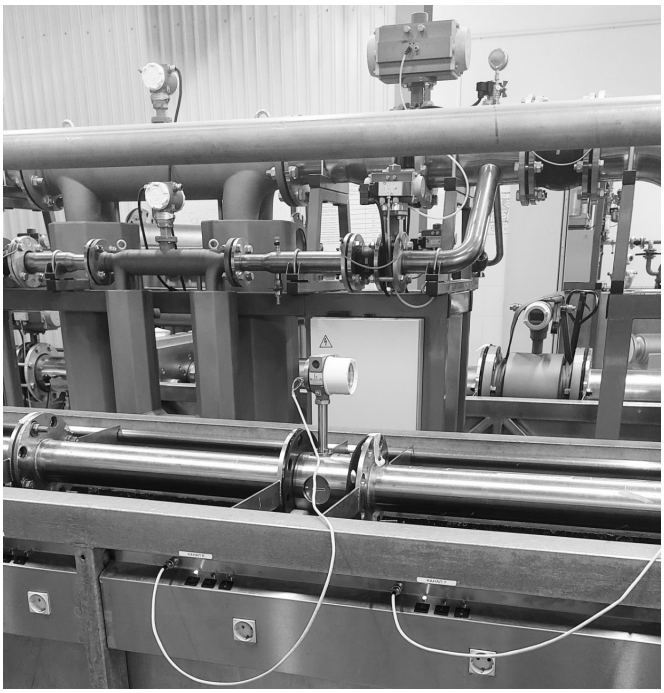


Рис. 4. Экспериментальная установка

Конструктивно установка включает систему хранения и подачи воды с накопительной емкостью 15 м^3, систему создания и стабилизации расхода на основе центробежных насосов с частотным регулированием, а также комплекс эталонных средств измерений. В качестве первичных эталонов используются четыре весоизмерительных устройства с пределами взвешивания 2, 20, 200 и 4500 кг и девять массовых корнолисовых расходомеров с диаметрами условных проходов от 5 до 150 мм. Точность измерения массы весовым методом составляет $\pm 0.05\%$, а при использовании эталонных расходомеров — $\pm 0.15\%$.

Принцип работы установки основан на объемно-временном методе измерения расхода. При весовом методе масса жидкости, прошедшей через поверяемый прибор, измеряется с помощью электронных весов, после чего пересчитывается в объем с учетом температуры и плотности воды. При методе сличения показания поверяемого прибора сравниваются в реальном времени с данными эталонного расходомера, при этом сигнал от эталона используется в цепи обратной связи частотного преобразователя для стабилизации расхода с точностью $\pm 0.5\%$.

На экспериментальной установке были проведены измерения для двух стабильных режимов течения: сигнал 1 был получен при расходе 110 м^3/ч, а сигнал 2 — при расходе

4 м^3/ч. Стабильность расходов обеспечивалась системой частотного регулирования насосов и контролировалась комплектом эталонных кориолисовых расходомеров.

4.3. Анализ экспериментальных сигналов

Сигналы, полученные с вихреакустического расходомера, имели частоту дискретизации $Fs = 200$ кГц, что избыточно для целевого анализа частотного диапазона вихреобразования, который обычно составляет около 5–150 Гц, и существенно увеличивает объем вычислений.

Поэтому сигналы были предварительно подвергнуты следующим этапам предобработки, направленным на снижение вычислительной нагрузки и повышение точности последующего анализа.

1. Поэтапная децимация в два шага: с 200 кГц до 20 кГц и с 20 кГц до 2 кГц. В обоих случаях использовалась функция ‘decimate‘ Matlab со встроенным фильтром для подавления частот выше половины новой частоты дискретизации. Такой подход позволил избежать прямого масштабного понижения частоты дискретизации (с 200 кГц до 2 кГц за один шаг), что могло бы привести к ухудшению качества фильтрации из-за ограниченной крутизны фильтра.
2. После децимации был применен полосовой фильтр, пропускающий частоты в диапазоне 90–140 Гц для сигнала 1 с частотой вихреобразования около 115 Гц и фильтр с полосой 3–7 Гц для сигнала 2 с частотой около 5 Гц.

После предобработки были получены сигналы с частотой дискретизации $Fs = 2$ кГц и выделенной узкой полосой частот, содержащей основную гармонику вихреобразования. Такие сигналы используются как вход для методов БПФ и ММП в задаче трекинга частоты. Длительность сигналов составляет 12 секунд.

Дальнейшая обработка велась в режиме скользящего окна с фиксированным шагом и длиной окна, одинаковыми для обоих алгоритмов, что обеспечивало одинаковое количество частотных оценок. Для сигналов с частотами вихреобразования около 115 и 5 Гц были выбраны окна длительностью 0.02 и 0.4 секунды, соответственно, что составило около двух периодов синусоид.

После обработки всех окон для каждого метода вычислялись: среднее значение частоты $\overline{f} = \frac{1}{N}\sum\limits_{i=1}^{N} f_i$, стандартное отклонение $\sigma_f = \sqrt{\frac{1}{N}\sum\limits_{i=1}^{N}(f_i - \overline{f})^2}$, а также суммарное время работы алгоритма. Полученные значения для двух экспериментальных сигналов приведены в таблице.

Таблица. Сравнение методов оценки частоты сигналов

Метод	**Сигнал 1 (115 Гц)**			**Сигнал 2 (5 Гц)**		
	$\overline{f}$**, Гц**	σ_f**, Гц**	**t, с**	$\overline{f}$**, Гц**	σ_f**, Гц**	**t, с**
БПФ	114.9697	3.4290	0.0497	4.9033	1.3515	0.0482
ММП	114.8548	2.7061	0.0201	4.8664	0.8937	0.0313

Анализ результатов, представленных в таблице, показывает, что ММП обеспечивает более стабильные оценки частоты по сравнению с БПФ: стандартное отклонение σ_f для ММП ниже примерно в 1.5 раза. Это указывает на более высокую устойчивость метода к шумам и нестационарности в наблюдаемых данных.

Примечательно, что ММП демонстрирует выигрыш по точности при сопоставимой или даже меньшей вычислительной трудоемкости, несмотря на необходимость выполнения сингулярного разложения. Причиной является малый объем данных в пределах одного окна анализа, что делает матричные операции относительно дешевыми.

На рис. 5, 6 приведены временные реализации предобработанных экспериментальных сигналов и оценки частоты, полученные с помощью БПФ и ММП.

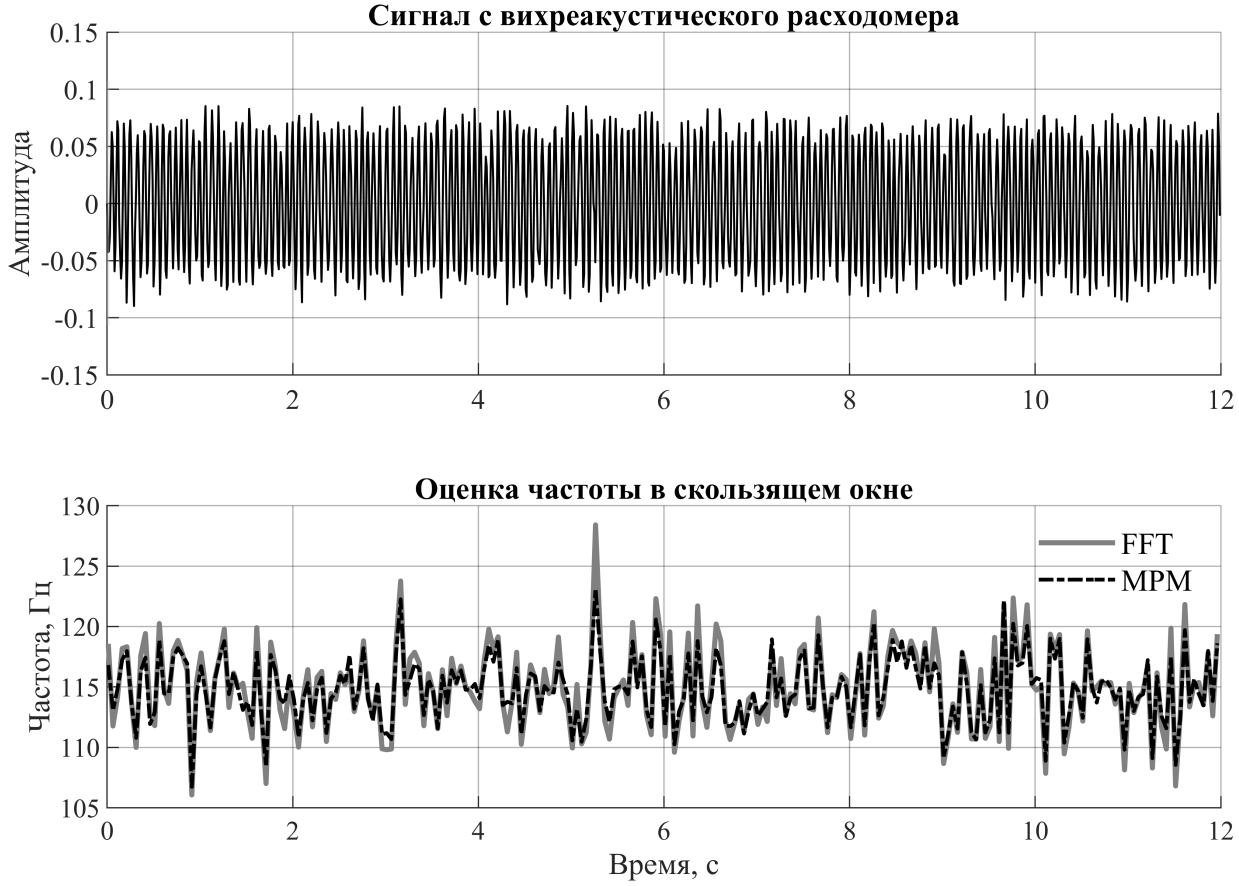


Рис. 5. Предобработанный сигнал 1 с вихреакустического расходомера и найденные значения частоты

Графический анализ подтверждает численные результаты: оба метода воспроизводят динамику изменения частоты вихреобразования, однако оценки ММП характеризуются меньшей разбросанностью, что особенно заметно при низкочастотных колебаниях (сигнал 2, рис. 6). Кроме того, временные зависимости подтверждают, что процесс вихреобразования носит нестационарный характер, а частота вихревого следа изменяется во времени.

Таким образом, в условиях ограниченной длины окон и умеренных вычислительных ресурсов ММП представляется предпочтительным методом для задач отслеживания точного значения частоты вихревых сигналов.

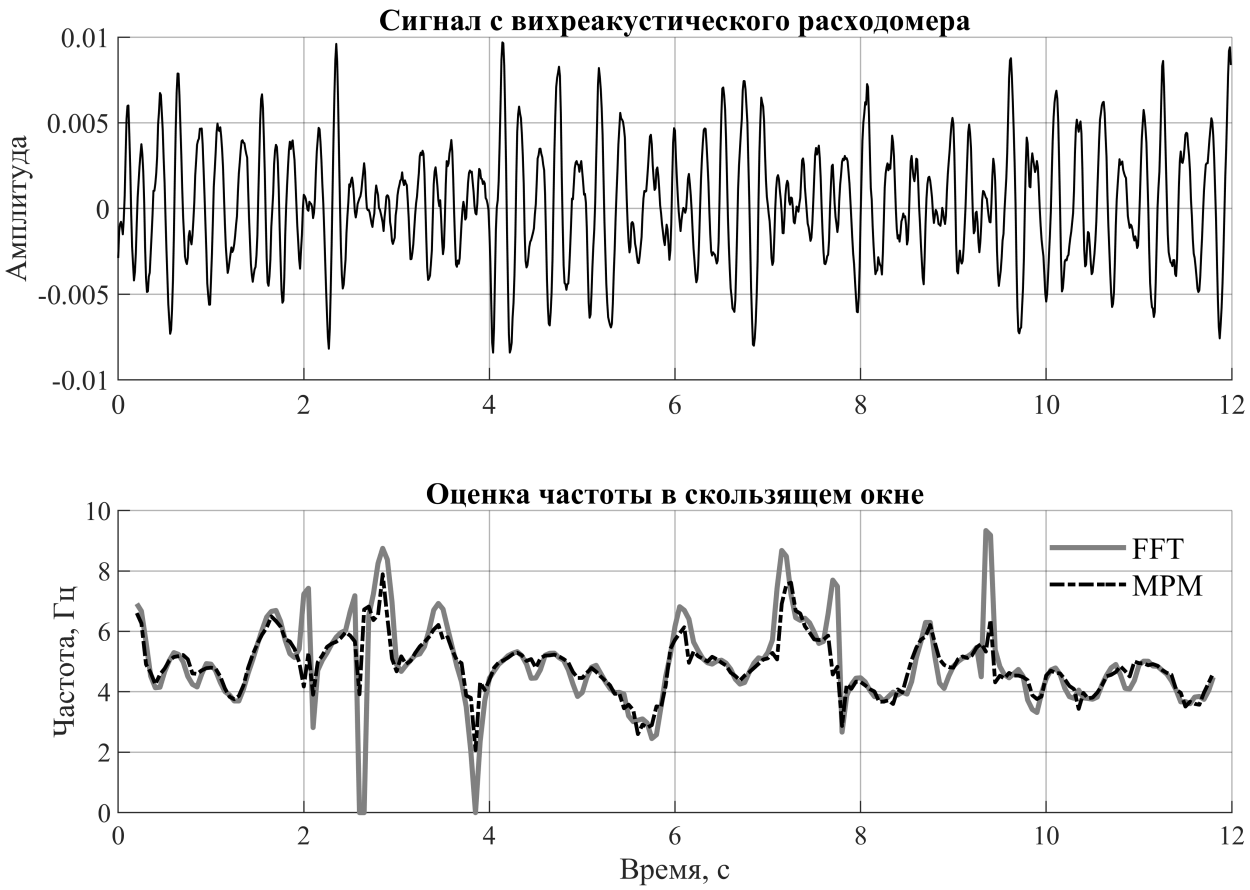


Рис. 6. Предобработанный сигнал 2 с вихреакустического расходомера и найденные значения частоты

Заключение

В данной работе рассмотрен принцип действия вихреакустических расходомеров, основанный на регистрации частоты вихрей Кармана, возникающих за телом обтекания в потоке. Показано, что точное и быстрое определение частоты вихреобразования является ключевой задачей, от решения которой зависит точность измерения расхода. Традиционные методы спектрального анализа, такие как быстрое преобразование Фурье, сталкиваются с фундаментальными ограничениями при работе на коротких временных интервалах, особенно в условиях шума и нестационарности сигнала.

В качестве альтернативы предложен модифицированный метод матричных пучков (ММП), относящийся к параметрическим методам высокого разрешения. Данный метод позволяет с высокой точностью оценивать частоту синусоидального сигнала даже при малой длительности аналитического окна, что особенно важно для отслеживания динамических изменений расхода в реальном времени.

Результаты моделирования и обработки реальных сигналов с вихреакустического расходомера показали, что модифицированный ММП превосходит классический БПФ по точности и устойчивости к шуму. При этом вычислительная сложность метода оказывается сопоставимой или даже ниже, благодаря эффективной обработке коротких выборок. Эксперименты подтвердили, что стандартное отклонение оценок частоты при использовании ММП снижается в среднем в 1.5 раза по сравнению с БПФ, что свидетельствует о более стабильной и надежной работе алгоритма в реальных условиях.

Таким образом, модифицированный метод матричных пучков представляет собой перспективный инструмент для цифровой обработки сигналов в системах измерения расхода, особенно в тех случаях, когда требуется высокая динамическая точность и малое время реакции. Его применение позволяет повысить метрологические характеристики вихреакустических расходомеров и расширить их функциональные возможности в составе современных систем автоматизации и управления технологическими процессами.

В дальнейшем представляется перспективным развитие алгоритма, включая его адаптацию к многокомпонентным сигналам, учет температурных и гидродинамических дрейфов, а также реализацию в виде встраиваемого программного обеспечения для микроконтроллеров, что сделает метод еще более пригодным для промышленного применения.

Исследование выполнено при финансовой поддержке Министерства науки и высшего образования Российской Федерации (государственное задание на выполнение фундаментальных научных исследований №FENU-2023-0010 (2023010ГЗ)).

Литература

1. Ибряева О.Л., Салов Д.Д. Модификация метода матричных пучков, использующая совместное оценивание полюсов сигнала и обратных к ним // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2017. Т. 6, № 1. С. 26–37. DOI: 10.14529/cmse170102.
2. Venugopal A., Agrawal A., Prabhu S.V. Review on vortex flowmeter - designer perspective // Sensors and Actuators A: Physical. 2011. Vol. 170. P. 8–23. DOI: 10.1016/j.sna.2011.05.034.
3. Кремлевский П.П. Расходомеры и счётчики количества веществ. Справочник: Кн. 2. Спб: Политехника, 2004. 412 с.
4. Wang X., Lin S., Kang Y., *et al.* Simulation analysis of the influence of two-phase flow on the accuracy of the low-temperature vortex flowmeter // Journal of the Brazilian Society of Mechanical Sciences and Engineering. 2025. Vol. 47. P. 365. DOI: 10.1007/s40430-025-05679-7.
5. Sun Y., Zhang T., Zheng D. New analysis scheme of flow-acoustic coupling for gas ultrasonic flowmeter with vortex near the transducer // Sensors. 2018. Vol. 18, no. 4. P. 1151. DOI: 10.3390/s18041151.
6. Sun H.-j., Huo C., Wang H.-x. The experimental research of vortex flowmeter in vertical upward gas-liquid two-phase flow. 2009 IEEE Instrumentation and Measurement Technology Conference (IMTC). 2009. P. 167–170. DOI: 10.1109/IMTC.2009.5168437.
7. Sun Z. Mass flow measurement of gas-liquid bubble flow with the combined use of a Venturi tube and a vortex flowmeter // Measurement Science and Technology. 2010. Vol. 21, no. 5. P. 055403. DOI: 10.1088/0957-0233/21/5/055403.
8. Shen H., Fu X., Chen J.-D., Ye P. development of a vortex flowmeter with good performance at low-flowrate. 15th International Flow Measurement Conference (FLOMEKO 2010). 2010. Vol. 1. P. 524–530.
9. Miau J., Hu C., Chou J. Response of a vortex flowmeter to impulsive vibrations // Flow Measurement and Instrumentation. 2000. Vol. 11. P. 41–49. DOI: 10.1016/S0955-5986(99)00018-7.

10. Proakis J.G., Manolakis D.G. Digital dignal processing: principles, algorithms, and applications. 4th ed. Upper Saddle River, NJ: Prentice Hall, 2006. 1019 p.

11. Jacobsen E., Kootsookos P. Fast, accurate frequency estimators [DSP tips & tricks] // IEEE Signal Processing Magazine. 2007. Vol. 24, no. 3. P. 123–125. DOI: 10.1109/MSP.2007.361611.

12. Candan Ç. A method for fine resolution frequency estimation from three DFT samples // IEEE Signal Processing Letters. 2011. Vol. 18, no. 6. P. 351–354. DOI: 10.1109/LSP.2011.2136378.

13. Hans V., Windorfer H. Comparison of pressure and ultrasound measurements in vortex flow meters // Measurement. 2003. Vol. 33. P. 121–133. DOI: 10.1016/S0263-2241(02)00057-X.

14. Prony G.R. Essai expérimental et analytique sur les lois de la dilatabilité des fluides élastiques et sur celles de la force expansive de la vapeur de l'eau et de la vapeur de l'alcool, à différentes températures // Journal de l'École Polytechnique. 1795. Vol. 1, no. 22. P. 24–76.

15. Schmidt R. Multiple emitter location and signal parameter estimation // IEEE Transactions on Antennas and Propagation. 1986. Vol. 34, no. 3. P. 276–280. DOI: 10.1109/TAP.1986.1143830.

16. Roy R., Kailath T. ESPRIT-estimation of signal parameters via rotational invariance techniques // IEEE Transactions on Acoustics, Speech, and Signal Processing. 1989. Vol. 37, no. 7. P. 984–995. DOI: 10.1109/29.32276.

17. Hua Y., Sarkar T.K. On the total least squares linear prediction method for frequency estimation // IEEE Transaction on Acoustics, Speech and Signal Processing. 1990. Vol. 38, no. 12. P. 2186–2189. DOI: 10.1109/29.61547.

18. Sun Z., Chen H., Chen Y. Application of periodogram and welch based spectral estimation to vortex frequency extraction // Proceedings - 2012 International Conference on Intelligent Systems Design and Engineering Applications, ISDEA 2012. 2012. DOI: 10.1109/ISdea.2012.689.

19. Zheng D., Zhang T., Xing J., Jianqiang M. Improvement of the HHT method and application in weak vortex signal detection // Measurement Science and Technology. 2007. Vol. 18. P. 2769. DOI: 10.1088/0957-0233/18/9/005.

20. Huang S., Yin J., Sun Z., *et al.* Characterization of gas–liquid two-phase flow by correlation dimension of vortex-induced pressure fluctuation // IEEE Access. 2017. Vol. 5. P. 10307–10314. DOI: 10.1109/ACCESS.2017.2713458.

21. Huang S., Sun Z., Zhou T., *et al.* Application of time-frequency entropy from wake oscillation to gas-liquid flow pattern identification // Journal of Central South University. 2018. Vol. 25, no. 7. P. 1690–1700. DOI: 10.1007/s11771-018-3860-2.

22. Sun Z., Chen Y., Gong H. Classification of gas-liquid flow patterns by the norm entropy of wavelet decomposed pressure fluctuations across a bluff body // Measurement Science and Technology. 2012. Vol. 23, no. 12. P. 125301. DOI: 10.1088/0957-0233/23/12/125301.

23. Hans V. New aspects of the arrangement and geometry of gluff godies in ultrasonic vortex flow meters. Proceedings of the 19th IEEE Instrumentation and Measurement Technology Conference. 2002. Vol. 2. P. 1661–1664. DOI: 10.1109/IMTC.2002.1007209.

24. Akresh M., Reindl L., Walker W. Improved vortex flow meter using pre-filter. 2008 2nd International Conference on Signals, Circuits and Systems. 2008. P. 1–5. DOI: 10.1109/ICSCS.2008.4746877.

25. Группа компаний Метран. Каталог СИ. Том 2. Датчики температуры. Расходомеры. Уровнемеры. URL: https://metran.ru/journal/ (дата обращения: 12.08.2025).

Ибряева Ольга Леонидовна, к.ф.-м.н., старший научный сотрудник НИЛ технической самодиагностики и самоконтроля приборов и систем, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

Яковенко Артем Дмитриевич, аспирант, кафедра информационно-измерительной техники, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

Синицин Владимир Владимирович, к.т.н., доцент, заместитель заведующего НИЛ технической самодиагностики и самоконтроля приборов и систем, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

Шестаков Александр Леонидович, профессор, д.т.н., президент Южно-Уральского государственного университета (национальный исследовательский университет) (Челябинск, Российская Федерация)

DOI: 10.14529/cmse250303

IMPROVING THE ACCURACY OF VORTEX FLOWMETERS THROUGH HIGH-RESOLUTION ESTIMATION OF VORTEX SHEDDING FREQUENCY

© 2025 O.L. Ibryaeva, A.D. Yakovenko, V.V. Sinitcin, A.L. Shestakov

South Ural State University (pr. Lenina 76, Chelyabinsk, 454080 Russia)

E-mail: ibriaevaol@susu.ru, iakovenkoad@susu.ru, sinitcinvv@susu.ru, a.l.shestakov@susu.ru

Received: 22.08.2025

The paper addresses the problem of improving the accuracy of vortex flowmeters by means of high-resolution frequency estimation of vortex shedding under conditions of short observation intervals and noisy signals. Traditional methods based on the Fast Fourier Transform (FFT) face fundamental resolution limitations when analyzing short data segments, which reduces their effectiveness in dynamic measurement regimes. As an alternative, a modified Matrix Pencil Method (MPM) is proposed, belonging to the class of high-resolution parametric techniques. The method models the signal as a sum of complex exponentials and provides robust frequency estimation even at low signal-to-noise ratios. A comparative analysis of MPM and FFT was carried out using both simulated and experimental signals from a vortex flowmeter. It is shown that MPM provides more stable frequency estimates, with the standard deviation reduced by an average of 1.5 times. At the same time, the computational complexity of the method is comparable to or even lower than that of FFT due to the small size of the analysis windows. The results demonstrate the potential of MPM for developing algorithms of automatic metrological reliability control. The method can serve as a basis for self-diagnostic systems and correction of measurement errors caused by flow nonstationarity, vibrations, or two-phase flow conditions.

Keywords: vortex flowmeter, frequency estimation, matrix pencil method, high resolution, signal processing, measurement reliability, two-phase flow, parametric methods.

FOR CITATION

Ibryaeva O.L., Yakovenko A.D., Sinitcin V.V., Shestakov A.L. Improving the Accuracy of Vortex Flowmeters through High-Resolution Estimation of Vortex Shedding Frequency. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2025. Vol. 14, no. 3. P. 42–58. (in Russian) DOI: 10.14529/cmse250303.

This paper is distributed under the terms of the Creative Commons Attribution-Non Commercial 4.0 License which permits non-commercial use, reproduction and distribution of the work without further permission provided the original work is properly cited.

References

1. Ibryaeva O.L., Salov D.D. Modification of the matrix pencil method using joint estimation of signal poles and their inverses. Bulletin of South Ural State University. Series: Computational Mathematics and Informatics. 2017. Vol. 6, no. 1. P. 26–37. DOI: 10.14529/cmse170102.
2. Venugopal A., Agrawal A., Prabhu S.V. Review on vortex flowmeter - designer perspective. Sensors and Actuators A: Physical. 2011. Vol. 170. P. 8–23. DOI: 10.1016/j.sna.2011.05.034.
3. Kremlevskii P.P. Flowmeters and counters of quantity of substances. Reference book: Vol. 2. St. Petersburg: Politekhnika, 2004. 412 p.
4. Wang X., Lin S., Kang Y., *et al.* Simulation analysis of the influence of two-phase flow on the accuracy of the low-temperature vortex flowmeter. Journal of the Brazilian Society of Mechanical Sciences and Engineering. 2025. Vol. 47. P. 365. DOI: 10.1007/s40430-025-05679-7.
5. Sun Y., Zhang T., Zheng D. New analysis scheme of flow-acoustic coupling for gas ultrasonic flowmeter with vortex near the transducer. Sensors. 2018. Vol. 18, no. 4. P. 1151. DOI: 10.3390/s18041151.
6. Sun H.-j., Huo C., Wang H.-x. The experimental research of vortex flowmeter in vertical upward gas-liquid two-phase flow. 2009 IEEE Instrumentation and Measurement Technology Conference (IMTC). 2009. P. 167–170. DOI: 10.1109/IMTC.2009.5168437.
7. Sun Z. Mass flow measurement of gas-liquid bubble flow with the combined use of a Venturi tube and a vortex flowmeter. Measurement Science and Technology. 2010. Vol. 21, no. 5. P. 055403. DOI: 10.1088/0957-0233/21/5/055403.
8. Shen H., Fu X., Chen J.-D., Ye P. Development of a vortex flowmeter with good performance at low-flowrate. 15th International Flow Measurement Conference (FLOMEKO 2010). 2010. Vol. 1. P. 524–530.
9. Miau J., Hu C., Chou J. Response of a vortex flowmeter to impulsive vibrations. Flow Measurement and Instrumentation. 2000. Vol. 11. P. 41–49. DOI: 10.1016/S0955-5986(99)00018-7.
10. Proakis J.G., Manolakis D.G. Digital signal processing: principles, algorithms, and applications. 4th ed. Upper Saddle River, NJ: Prentice Hall, 2006. 1019 p.
11. Jacobsen E., Kootsookos P. Fast, accurate frequency estimators [DSP tips & tricks]. IEEE Signal Processing Magazine. 2007. Vol. 24, no. 3. P. 123–125. DOI: 10.1109/MSP.2007.361611.

12. Candan C. A method for fine resolution frequency estimation from three DFT samples. IEEE Signal Processing Letters. 2011. Vol. 18, no. 6. P. 351–354. DOI: 10.1109/LSP.2011.2136378.

13. Hans V., Windorfer H. Comparison of pressure and ultrasound measurements in vortex flow meters. Measurement. 2003. Vol. 33. P. 121–133. DOI: 10.1016/S0263-2241(02)00057-X.

14. Prony G.R. Experimental and analytical essay on the laws of the dilatability of elastic fluids and on the expansive force of water vapor and alcohol vapor at different temperatures. Journal de l'École Polytechnique. 1795. Vol. 1, no. 22. P. 24–76.

15. Schmidt R. Multiple emitter location and signal parameter estimation. IEEE Transactions on Antennas and Propagation. 1986. Vol. 34, no. 3. P. 276–280. DOI: 10.1109/TAP.1986.1143830.

16. Roy R., Kailath T. ESPRIT-estimation of signal parameters via rotational invariance techniques. IEEE Transactions on Acoustics, Speech, and Signal Processing. 1989. Vol. 37, no. 7. P. 984–995. DOI: 10.1109/29.32276.

17. Hua Y., Sarkar T.K. On the total least squares linear prediction method for frequency estimation. IEEE Transactions on Acoustics, Speech, and Signal Processing. 1990. Vol. 38, no. 12. P. 2186–2189. DOI: 10.1109/29.61547.

18. Sun Z., Chen H., Chen Y. Application of periodogram and welch based spectral estimation to vortex frequency extraction. Proceedings - 2012 International Conference on Intelligent Systems Design and Engineering Applications, ISDEA 2012. 2012. DOI: 10.1109/ISdea.2012.689.

19. Zheng D., Zhang T., Xing J., Jianqiang M. Improvement of the HHT method and application in weak vortex signal detection. Measurement Science and Technology. 2007. Vol. 18. P. 2769. DOI: 10.1088/0957-0233/18/9/005.

20. Huang S., Yin J., Sun Z., *et al.* Characterization of gas–liquid two-phase flow by correlation dimension of vortex-Induced pressure fluctuation. IEEE Access. 2017. Vol. 5. P. 10307–10314. DOI: 10.1109/ACCESS.2017.2713458.

21. Huang S., Sun Z., Zhou T., *et al.* Application of time-frequency entropy from wake oscillation to gas-liquid flow pattern identification. Journal of Central South University. 2018. Vol. 25, no. 7. P. 1690–1700. DOI: 10.1007/s11771-018-3860-2.

22. Sun Z., Chen Y., Gong H. Classification of gas-liquid flow patterns by the norm entropy of wavelet decomposed pressure fluctuations across a bluff body. Measurement Science and Technology. 2012. Vol. 23, no. 12. P. 125301. DOI: 10.1088/0957-0233/23/12/125301.

23. Hans V. New aspects of the arrangement and geometry of bluff bodies in ultrasonic vortex flow meters. Proceedings of the 19th IEEE Instrumentation and Measurement Technology Conference. 2002. Vol. 2. P. 1661–1664. DOI: 10.1109/IMTC.2002.1007209.

24. Akresh M., Reindl L., Walker W. Improved vortex flow meter using pre-filter. 2008 2nd International Conference on Signals, Circuits and Systems. 2008. P. 1–5. DOI: 10.1109/ICSCS.2008.4746877.

25. Metran group of vompanies. Catalog of measuring instruments. Volume 2. Temperature sensors. Flowmeters. Level gauges. URL: https://metran.ru/journal/ (accessed: 12.08.2025).

УДК 519.63, 519.688 DOI: 10.14529/cmse250304

ДИНАМИЧЕСКАЯ БАЛАНСИРОВКА ВЫЧИСЛИТЕЛЬНОЙ НАГРУЗКИ ПРИ МОДЕЛИРОВАНИИ НЕУСТОЙЧИВЫХ ТЕЧЕНИЙ*

© 2025 А.М. Титова, Н.А. Михайлов, Ю.Ф. Юсупов

Федеральное государственное унитарное предприятие «Российский Федеральный Ядерный Центр – Всероссийский научно-исследовательский институт технической физики имени академика Е.И. Забабахина»

(ФГУП «РФЯЦ–ВНИИТФ им. академ. Е.И. Забабахина»)

(456770 Челябинская обл., Снежинск, ул. Васильева, д. 13)

E-mail: a.m.titova@vniitf.ru, n.a.mikhaylov@vniitf.ru, yusupovyuf@vniitf.ru

Поступила в редакцию: 19.04.2025

В работе описан алгоритм динамической балансировки вычислительной нагрузки при моделировании неустойчивых течений на динамически адаптивных расчетных сетках с помощью трехмерной эйлеровой газодинамической программы. Выравнивание вычислительной нагрузки между MPI-процессами производится независимо вдоль линий (наборов MPI-фрагментов, расположенных вдоль выделенного направления), и может выполняться в трех режимах: по числу ячеек, по времени счета, и в смешанном (или автоматическом) режиме. Для минимизации накладных расходов на адаптацию и балансировку используется адаптация с запасом, при этом строится последовательность из этапов адаптации, балансировки и счета в течение заданного количества шагов — цикл Адаптация — Балансировка — Счет (цикл А-Б-С). Оптимальное число шагов в цикле А-Б-С определяется путем минимизации времени счета задачи, которое зависит от общего числа ячеек. Рассматривается тестовая задача о сферическом сжатии легкого центрального вещества тяжелой оболочкой (тест Янгса). Источником развития неустойчивостей является начальное гармоническое возмущение на контактной границе веществ. Моделирование проводится с применением динамической сеточной адаптации второго уровня в области развития неустойчивостей. Оценивается ускорение времени счета задачи при включении балансировки вычислительной нагрузки в разных режимах относительно расчета без балансировки. В результате проведенных исследований показана эффективность применения адаптации с запасом в рамках цикла А-Б-С. При этом автоматический режим выравнивания нагрузки является наиболее практичным.

Ключевые слова: газовая динамика, неустойчивые течения, сферическое сжатие, динамическая адаптация сетки, динамическая балансировка, вычислительная нагрузка.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Титова А.М., Михайлов Н.А., Юсупов Ю.Ф. Динамическая балансировка вычислительной нагрузки при моделировании неустойчивых течений // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2025. Т. 14, № 3. С. 59–76. DOI: 10.14529/cmse250304.

Введение

На контактной границе между веществами могут развиваться неустойчивые течения по разным причинам [1]. Например, неустойчивость Рэлея—Тейлора (РТ–неустойчивость, или НРТ) возникает при ускоренном движении двух разноплотных жидкостей (имеющих плоскую границу раздела) в направлении от легкого вещества к тяжелому. Частным случаем РТ-неустойчивости является неустойчивость Рихтмайера—Мешкова (НРМ), которая воз-

*Статья рекомендована к публикации программным комитетом Всероссийской научной конференции с международным участием «Параллельные вычислительные технологии (ПаВТ) 2025».

никает при прохождении ударной волны через контактную границу двух различных газов, на которой имеются малые возмущения границы. В случае, когда на контактной границе имеется разрыв тангенсальной компоненты скорости, возникает неустойчивость Кельвина—Гельмгольца (неустойчивость КГ, или НКГ).

В работе [2] проводится моделирование турбулентного перемешивания в сферическом секторе. Авторы описывают расчет сжатия легкого вещества тяжелой оболочкой, при этом на контактной границе веществ задаются гармонические возмущения (тест Янгса). В результате исследований показано, что степень подробности разностной сетки влияет на ширину зоны пермешивания, и для моделирования турбулентного перемешивания необходимо использовать достаточно подробную сетку, которая сможет разрешать основные масштабы течения.

Радиус моделируемого шара в тесте Янгса — 15 см. Прицельные расчеты с помощью трехмерной эйлеровой газодинамичской программы [3] показали, что сетки с размером ячеек 0.05 мм достаточно для описания зоны перемешивания с хорошей точностью. Если использовать статическую сетку и проводить расчет в 1/8 шара, то число ячеек достигает 14 миллиардов. Такая сетка требует огромного количества вычислительных ресурсов.

В целях экономии моделирование необходимо проводить не на статической, а на динамически адаптивной расчетной сетке. Она позволяет сохранять требуемый размер ячеек лишь в зоне перемешивания, где это необходимо. В остальной области сетка может быть в несколько раз грубее. При этом, по мере продвижения зоны, при необходимости ячейки измельчаются или загрубляются. Тестовые оценки показали, что пиковое значение количества ячеек на адаптивной сетке на порядок меньше, чем на статической.

Однако, использование динамической адаптации сетки приводит к дисбалансу вычислительной нагрузки на MPI-фрагментах, ведь, например, в случае адаптации второго уровня, число ячеек на параллельных областях (параобластях) может отличаться до 16 раз в двумерном случае и до 64 раз в трехмерном. В следствие этого менее загруженные MPI-процессы будут простаивать в ожидании других, более загруженных процессов. Поэтому, необходимо иметь функционал, который будет выравнивать вычислительную нагрузку между MPI-процессами, тем самым обеспечивая более равномерное время работы процессов и уменьшение простоев.

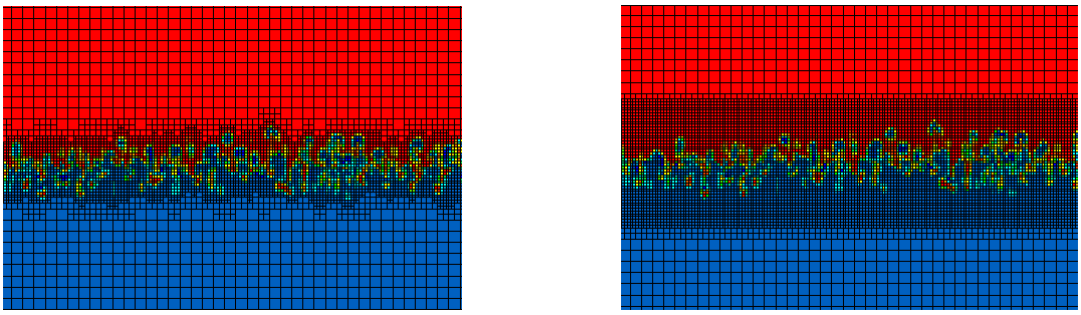
Существуют откртытые библиотеки, в которых реализована динамическая балансировка, например SCOTCH, PT-SCOTCH [4], ParMETIS [5]. Однако, тестирование данных библиотек выявило несколько недостатков. Во-первых, миграция большого количества ячеек между MPI-процессами, вплоть до глобальной передекомпозиции, значительно увеличивает временные затраты на сетках с числом ячеек больше 10 млн. Во-вторых, работа алгоритма оказалась нестабильной при включении адаптации второго уровня. Именно это и стало поводом для создания собственного модуля балансировки вычислительной нагрузки.

Статья организована следующим образом. В разделе 1 приведены основные особенности газодинамической методики, расчетной сетки и декомпозиции на параллельные области. В разделе 2 описан алгоритм реализации модуля балансировки вычислительной нагрузки в трех режимах: балансировка по числу ячеек, по фактическому времени счета, смешанный режим. В разделе 3 приведены результаты серий тестовых расчетов в двумерной и трехмерной постановках. В заключении проведен анализ результатов тестирования и сделан вывод о том, какой режим балансировки вычислительной нагрузки наиболее оптимален при проведении расчетов задач моделирования неустойчивых течений на динамически адаптивной расчетной сетке.

1. Газодинамическая программа

Модуль балансировки вычислительной нагрузки внедряется в трехмерную программу, которая решает уравнения радиационной газовой динамики на неструктурированных эйлеровых сетках, тем самым, моделируя произвольные вихревые течения. В качестве численной схемы применяется схема годуновского типа [6], которая позволяет избежать использования искусственной вязкости. Для того, чтобы обеспечить монотонность численной схемы, на гранях ячеек производится кусочно-линейная TVD-реконструкция величин [7]. Отсутствие численной диффузии обеспечивается наличием механизма восстановления контактной границы (геометрический метод Volume of Fluid). Верификация данной методики пройдена на задачах моделирования различных режимов развития неустойчивостей [8]. Результаты расчетов сравнивались с открытыми газодинамическими кодами [3].

Для моделирования задач сферического сжатия используются квазисферические динамически адаптивные расчетные сетки [9]. Существует несколько подходов для динамической адаптации сеток: перемещение узлов сетки без изменения ее топологии [10] (лагранжево-эйлеровый подход), покрытие необходимой области вложенной сеткой нужного разрешения [11] (блочно-структурированный подход) или, как в нашем случае, измельчение или укрупнение любой ячейки независимо от других в соответствии с критериями. При произвольной форме зоны перемешивания, используется локальная адаптация ячеек — режим, при котором крупные ячейки, в случае необходимости, измельчаются, а измельченные, когда это не противоречит критерию адаптации, укрупняются. Однако, при моделировании задач, в которых зона перемешивания локализована и двигается слоем в течение расчета, можно производить атаптацию сетки слоями (см. рис. 1). Такой подход лучше описывает зону перемешивания, так как разрешение ячеек в поперечном к направлению роста струй направлении остается одинаковым. При этом, он позволяет адаптировать ячейки с запасом, прогнозируя положение зоны перемешивания через заданное количество счетных шагов [12].

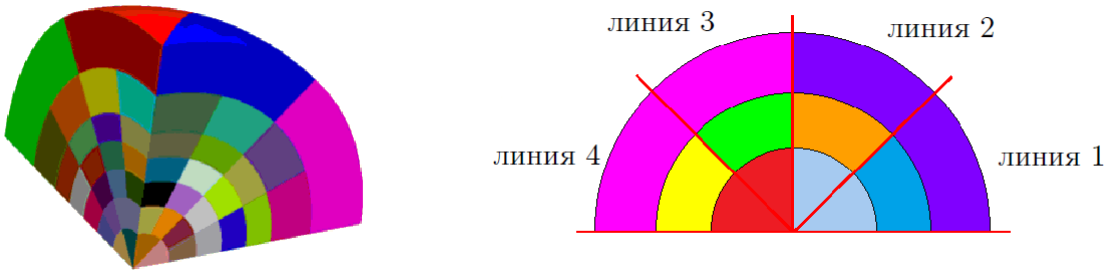


а) локальная адаптация б) послойная адаптация

Рис. 1. Динамическая адаптация сетки 2-ого уровня

Распараллеливание программного кода на распределенной памяти выполнено в рамках стандартного подхода пространственной декомпозиции сетка и обмена данными с помощью библиотеки MPI [13]. Расчетная сетка декомпозируется по углу и по радиусу на сферические сектора, как показано на рис. 2. Набор MPI-фрагментов, расположенных вдоль радиуса, называется линией. Число таких линий в каждом угловом направлении совпадает с максимальным количеством параобластей по углу. Так как MPI-фрагмент может лежать в

нескольких линиях, то вводится его вес — коэффициент, который определяет вклад этого процесса в вычислительную нагрузку на всей линии.



а) декомпозиция на MPI-фрагменты б) распределение по линиям

Рис. 2. Сферическая декомпозиция расчетной сетки

В начальный момент времени число ячеек в параобластях одинаково. Однако, измельчение сетки в зоне перемешивания при адаптации приводит к тому, что число ячеек на некоторых MPI-фрагментах становится больше, чем на других, и время на работу таких фрагментов увеличивается. Таким образом, возникает необходимость иметь функционал, который будет балансировать вычислительную нагрузку между MPI-процессами и обеспечивать выровненное время работы всех MPI-процессов.

2. Реализация алгоритма

2.1. Общая схема работы

Так как зона адаптации является квазисферической, и геометрия параобластей также обладает сферичностью, балансировка вычислительной нагрузки производится вдоль линий. При этом, каждый процесс может передавать нагрузку на процессы, которые находятся выше или ниже по радиусу, и не передает нагрузку процессам слева или справа по углу. Балансировка вычислительной нагрузки производится после адаптации расчетной сетки. Алгоритм балансировки включает в себя следующие этапы:

1. **Формирование линий.**
   Так как в каждой линии по углу располагается не более одного процесса, то число линий в системе определяетя произведением максимальных количеств MPI-фрагментов по полярному и азимутальному углам. В случае процессного каскада (случая, когда процесс располагается в нескольких линиях), определяется вес процесса, который учитывается при вычислении целевой нагрузки на MPI-фрагменте в линии. Этап формирования линий запускается один раз в начале счета задачи.
2. **Определение вычислительной нагрузки на каждом MPI-процессе.**
   В случае, если балансировка вычислительной нагрузки производится по ячейкам, то вычислительная нагрузка определяетя как число ячеек на MPI-фрагменте. При балансировке по времени вычислительная нагрузка есть фактическое время счета шага для MPI-процесса.
3. **Вычисление коэффициента разбалансированности системы.**
   Коэффициент разбалансированности определяется следующим образом:

$$K = \frac{\text{макс. нагрузка}}{\text{средняя нагрузка}}. \qquad (1)$$

Максимальная нагрузка определяется с учетом всех MPI-фрагментов в линии. Средняя нагрузка определяется как сумма вычислительных нагрузок на MPI-процессах с учетом их весов в линии, деленная на количество MPI-процессов в линии. Если K оказывается больше максимально разрешенного коэффициента разбалансированности, заданного пользователем, то запускается алгоритм балансировки.

4. **Построение списка количества нагрузки, которую должен отдать или принять каждый MPI-процесс.**
   Данный этап является единственным последовательным алгоритмом во всем модуле балансировки и производится мастер-процессом. Он собирает информацию о дисбалансе вычислительной нагрузки (разнице между текущей вычислительной нагрузкой и средней) на каждом MPI-процессе и определяет, сколько ячеек принять или отдать каждому MPI-процессу. В цикле по всем линиям, начиная с самого близкого к центру MPI-фрагмента в линии, происходит вычисление количества нагрузки, которое необходимо отдать с текущего фрагмента на следующий или принять от него. Если соседний MPI-фрагмент не имеет достаточного количества нагрузки, то аналогичная процедура проводится для MPI-фрагмента, следующего за соседом. Так происходит до тех пор, пока для все MPI-процессов не будут определены списки приема-передачи вычислиельной нагрузки. После этого, построенный на мастер-процессе двумерный массив рассылается всем слейв-процессам.
5. **Маркировка ячеек на каждом MPI-фрагменте с учетом построенного списка приема-передач ячеек.**
   Получив информацию о том, каким количеством нагрузки необходимо обменяться, каждый MPI-процесс определяет, какие именно ячейки будут передаваться. Ячейки маркируются по очереди, начиная с ближайшего к процессу-приемнику слоя еще неотмаркированных ячеек. Таким образом, если процесс-приемник находится выше по радиусу, чем текущий MPI-фрагмент, то маркировака ячеек начинается со слоя ячеек сверху, причем для самого удаленного фрагмента ячейки маркируются первыми. Схематично этап маркировки ячеек изображен на рис. 3.

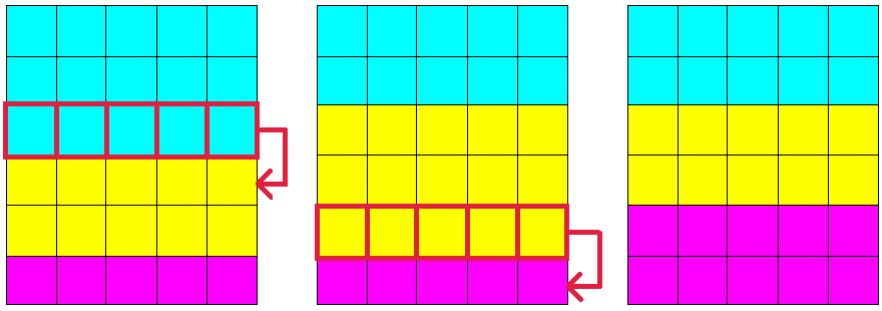


Рис. 3. Схематическое изображение работы этапа маркировки ячеек для трех MPI-фрагментов (ячейки с одного фрагмента окрашены одним цветом)

Стоит отметить, что все поколения адаптированной ячейки передаются вместе с родительской ячейкой, что может привести к небольшому отличию нового значения вычислительной нагрузки от целевого.

2.2. Балансировка по числу ячеек

Динамическая адаптация расчетной сетки приводит к тому, что число ячеек на параобластях становится разным. Это влечет за собой замедление времени счета шага, так как некоторые из MPI-процессов простаивают в ожидании окончания работы загруженных процессов. Поэтому балансировка числа ячеек между фрагментами является необходимым инструментом при проведении расчетов на динамически адаптивных сетках, который позволяет ускорить общее время счета задачи.

При балансировке по числу ячеек вычислительная нагрузка равняется числу ячеек на MPI-фрагменте. Целевое число ячеек на MPI-фрагменте P в линии L после балансировки по числу ячеек определяется следующим образом:

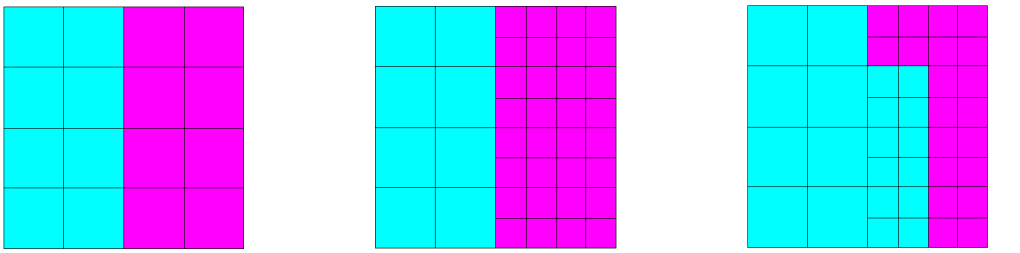
$$< N_P^L >= \frac{\sum_{p \in L} N_p \cdot W_P^L}{\sum_{p \in L} W_P^L} \cdot W_P^L, \tag{2}$$

где W_P^L — вес MPI-фрагмента P в линии L, который равен:

$$W_P^L = \frac{N_P^L}{N_P}. \tag{3}$$

Здесь N_P — число ячеек на MPI-фрагменте P, N_P^L — число ячеек на MPI-фрагменте P, которые попали в линию L.

На рис. 4 схематично представлен пример работы алгоритма балансировки по числу ячеек при адаптации первого уровня. После измельчения сетки число ячеек на параобластях стало отличаться в 4 раза. После балансировки число ячеек вновь стало одинаковым, при этом дочерние ячейки находятся на одном MPI-фрагменте с родительскими.



а) исходная сетка б) после измельчения в) после балансировки

Рис. 4. Пример распределения ячеек между MPI-фрагментами

2.3. Балансировка по времени счета

Несмотря на то, что число ячеек на MPI-фрагментах одинаковое, длительность работы каждого MPI-процесса может отличаться. Время счета шага определяется как сумма времен счета основных этапов для каждой ячейки с MPI-фрагмента. К основным этапам относятся: вычисление годуновских потоков, восстановление контактной границы, обновление величин в ячейках, расчет давления и температуры из уравнения состояния, при этом время работы для каждой ячейки на некоторых этапах зависит от количества веществ в ней. Как видно из рис. 5, это приводит к тому, что счет смешанных ячеек дольше, чем чистых.

Таким образом, при балансировке вычислительной нагрузи необходимо выравнивать не количество ячеек на MPI-фрагментах, а фактическое время их работы, которое определя-

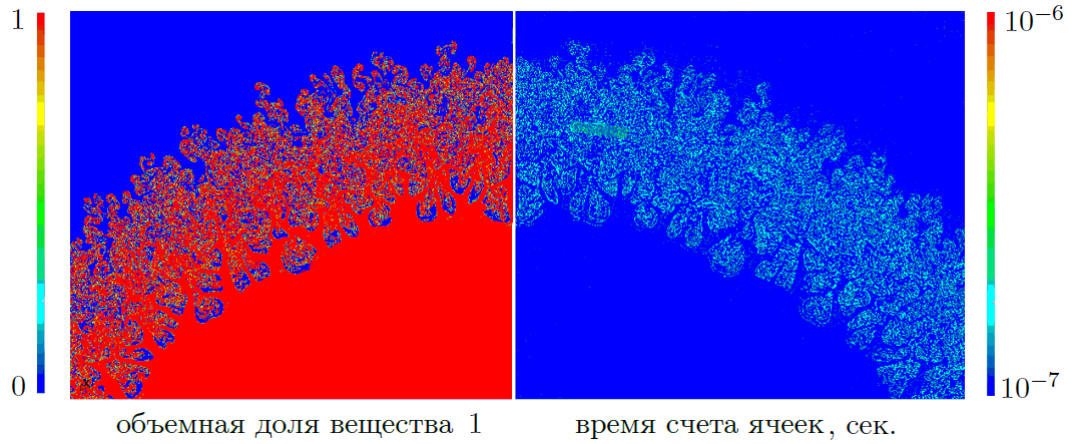


Рис. 5. Объемная доля вещества и время обработки ячеек для тестовой задачи Янгса

ется как:

$$t_{cell} = t_{cell}^{flux} + t_{cell}^{\text{КГ}} + t_{cell}^{EOS} + t_{cell}^{update}, \tag{4}$$

где t_{cell}^{flux} — время счета годуновских потоков для ячейки, $t_{cell}^{\text{КГ}}$ — время работы механизма восстановления контактной границы для ячейки, t_{cell}^{EOS} — время работы с уравнением состояния, t_{cell}^{update} — время обновления величин в ячейке. Вес MPI-фрагмента в линии равен:

$$W_P^L = \frac{t_P^L}{t_P}, \tag{5}$$

где t_P^L — суммарное время счета ячеек фрагмента P, лежащих в линии L, t_P — суммарное время счета всех ячеек на MPI-фрагменте. В таком случае, целевое время работы MPI-процесса P в линии L после балансировки по времени есть:

$$< t_P^L >= \frac{\sum_{p \in L} t_P \cdot W_P^L}{\sum_{p \in L} W_P^L} \cdot W_P^L. \tag{6}$$

Таким образом, реализовано два режима балансировки вычислительной нагрузки: по числу ячеек и по фактическому времени счета MPI-процесса. Балансировка по числу ячеек обеспечивает одинаковое заполнение памяти MPI-фрагментов и позволяет избежать ошибок переполнения на MPI-процессах при измельчении сетки и появлении новых ячеек на этапах адаптации. При этом из-за того, что смешанные и чистые ячейки считаются разное количество времени, время работы MPI-процессов отличается. Балансировка по фактическому времени счета позволяет сделать время работы всех MPI-процессов одинаковым и уменьшить простои. Однако, при таком режиме число ячеек на фрагментах может отличаться значительно. При этом, если на уже заполненном ячейками MPI-фрагменте произведется этап адаптации, то имеется риск выхода за пределы оперативной памяти. Комбинирование двух режимов балансировки в процессе счета задачи позволит наиболее эффективно использовать сильные стороны реализованных алгоритмов и обойти слабые.

2.4. Автоматический режим

Тестовые расчеты показали, что накладные расходы на адаптацию и балансировку могут составлять до 50% от общего времени счета задачи. Для того, чтобы их минимизиро-

вать, рассматривается подход адаптации с запасом на заданное количество шагов. Прогнозирование положения зоны перемешивания делается в предположении постоянной скорости движения контактной границы в течение этих шагов. Скорость движения определяется по изменению пложения границ зоны на предыдущем интервале [12].

Измельчение сетки с запасом приводит к росту количества ячеек, что увеличивает время счета. Для того, чтобы минимизировать общее итоговое время счета, необходимо решить задачу оптимизации количества временных шагов, на которые делается прогноз. Общее время счета задачи в зависимости от количества запасных шагов $N_{\text{АБС}}$ в приближении роста ширины зоны перемешивания оценивается как:

$$T(N_{\text{АБС}}) = \sum_{j=1}^{N_T/N_{\text{АБС}}} \left(t_{cell} \cdot (N^0_{cells} + j \cdot N_R \cdot N_{\text{АБС}}) \cdot N_{\text{АБС}} + t^{cell}_{\text{АБ}} \cdot (N^0_{cells} + j \cdot N_R \cdot N_{\text{АБС}}) \right), \tag{7}$$

где T — время счета задачи из N_T временных шагов, $N_{\text{АБС}}$ — количество шагов прогнозирования, t_{cell} — время счета одной ячейки на одном временном шаге, N^0_{cells} — начальное количество ячеек в задаче, N_R — количество ячеек, добавляемое при прогнозировании на один временной шаг, $t^{cell}_{\text{АБ}}$ — накладные расходы на одну ячейку. Минимум данной функции дает $N^*_{\text{АБС}}$, получающаяся из следующего квадратного уравнения:

$$\frac{dT}{dN_S}(N^*_{\text{АБС}}) = 0, \tag{8}$$

или

$$t_{cell} \cdot N_R \cdot N^2_{\text{АБС}} + \frac{N_R t^{cell}_{\text{АБ}}}{2} N_{\text{АБС}} - \left(\frac{N_R t^{cell}_{\text{АБ}}}{2} \cdot N_T + t^{cell}_{\text{АБ}} \cdot N^0_{cells} \right) = 0. \tag{9}$$

Из расчетов тестовой задачи были получены следующие значения параметров построенной модели: $t_{cell} = 1.7 \cdot 10^{-7}$, $N_R = 2 \cdot 10^3$, $t^{cell}_{\text{АБ}} = 4 \cdot 10^{-7}$.

Для двумерного расчета с начальным числом ячеек $N^0_{cells} = 10^6$ ячеек и числом временных шагов $N_T = 5 \cdot 10^4$ оптимальное количество шагов в цикле А-Б-С равно $N^*_{\text{АБС}} = 244$. Данное значение дает примерно 2% накладных расходов.

Таким образом, строится цикл Адаптация — Балансировка — Счет (цикл А-Б-С) — последовательность из этапов адаптации, балансировки и счета на заданном количестве $N_{\text{АБС}}$ счетных шагов.

Цикл строится таким образом, что сначала производятся все этапы измельчений, количество которых определяется максимальным уровнем адаптации. При этом после каждого из этих этапов производится балансировка вычислительной нагрузки по числу ячеек. Далее укрупняются все ячейки, которые можно укрупнить. После укрупнения производится балансировка по числу ячеек и счетный шаг, на котором замеряются времена счета ячеек. Далее делается балансировка по времени счета и счет N_S числа шагов (N_S — количество счетных шагов в цикле А-Б-С). Общее число шагов в цикле А-Б-С есть сумма счетных шагов, шагов измельчения и укрупнения, и шагов балансировки. Число шагов в цикле задается пользователем. Пример порядка шагов при адаптации 2-го уровня представлен на рис. 6.

Здесь И — этап измельчения ячеек, У — этап укрупнения ячеек, Б — этап балансировки вычислительной нагрузки (зеленым — балансировка по числу ячеек, розовым — балансировка по времени счета), С — этап счета.

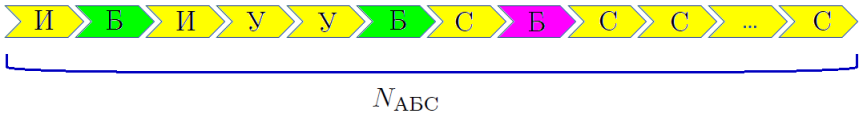


Рис. 6. Порядок расположения шагов в цикле А-Б-С при адаптации 2-го уровня

Таким образом, построен численный алгоритм, который позволяет балансировать вычислительную нагрузку в автоматическом режиме, используя комбинацию из балансировок по ячейкам и по фактическому времени счета. При этом накладные расходы на адаптацию и балансировку минимизируются за счет использования цикла Адаптация — Балансировка — Счет с прогнозированием положения зоны перемешивания.

3. Тестирование алгоритма

3.1. Постановка тестовой задачи

Рассматривается тестовая задача Янгса о сжатии легкого центрального вещества плотной оболочкой [14]. При наличии шероховатости на контактной границе между первым и вторым веществом развивается неустойчивость Рихтмайера—Мешкова, поэтому данная задача часто используется для отработки технологии проведения расчетов перемешенивания. Для моделирования граничных условий по эйлеровой программе используется буферная область (вещество 3). Распределение веществ и начальные характеристики представлены на рис. 7 и в табл. 1.

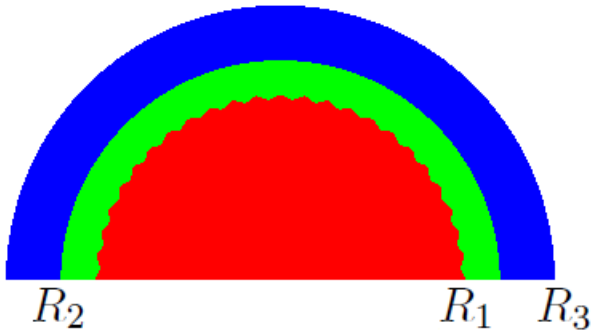


Рис. 7. Геометрия тестовой задачи

Таблица 1. Параметры веществ в тестовой задаче

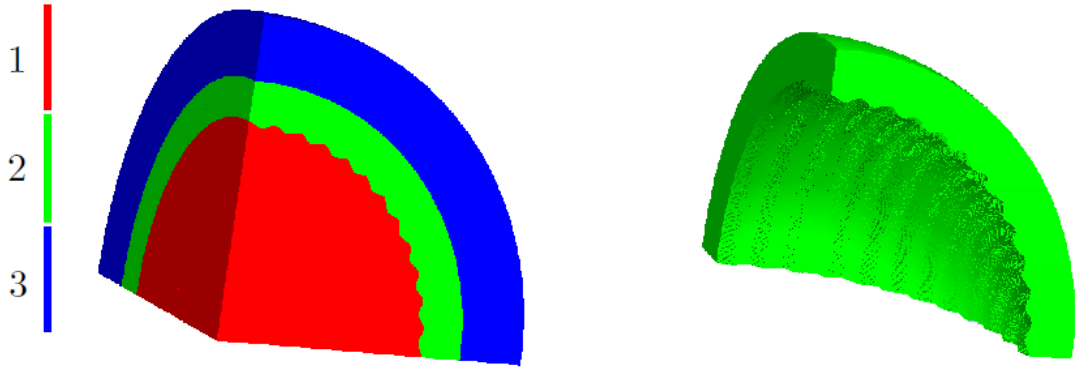
вещество	R, см	ρ, г см^3	C_v	e, эрг
1	10	0.05	2	3
2	12	1	0.1	0.15
3	15	0.1	100	150

Уравнение состояния веществ — идеальный газ с показателем адиабаты 5/3. Внутренняя удельная энергия на внешней границе оболочки (вещества 2) до $t = 0.5$ сек поддерживалась постоянной и равнялась 150 эрг. Далее она снижалась по линейному закону до

1.5 эрг к моменту времени $t = 3$ сек. Затравкой для развития неустойчивостей на контактной границе между веществами служит гармоническое возмущение:

$$P(\theta) = A \cdot \cos(m \cdot \theta), \tag{10}$$

здесь амплитуда $A = 1.25$ см, номер моды $m = 48$. Конечное время счета $t_k = 2.5$ сек — момент максмального сжатия центрального вещества. В трехмерной постановке начальное возмущение контактной границы «центральное вещество — оболочка» задавалось квазидвумерным: гармонические возмущения задавались только по одному угловому направлению. На рис. 8 представлено распределение веществ и слой оболочки на начальный момент времени.



а) распределение веществ б) вещество 2

Рис. 8. Распределение веществ в задаче Янгса в 3D постановке

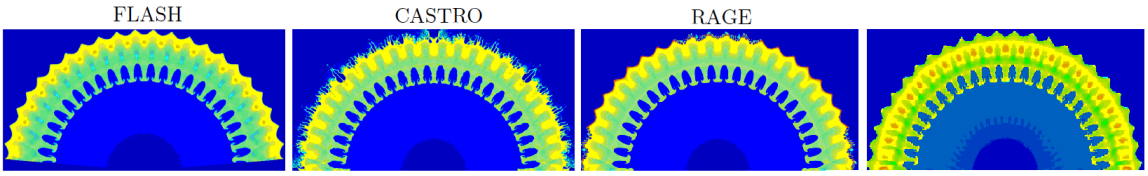
3.2. Характеристики расчетов

В двумерном случае расчеты проводились в осесимметричной постановке в секторе $0 \le \theta \le 180^o$. Использовалась квазисферическая сетка с базовым размером ячеек 0.02 см. При этом в области третьего вещества размер ячеек увеличивался до 0.04 см. Режим адаптации — послойная, 2-ого уровня, что обеспечивает размер ячеек в зоне перемешивания 0.05 мм.

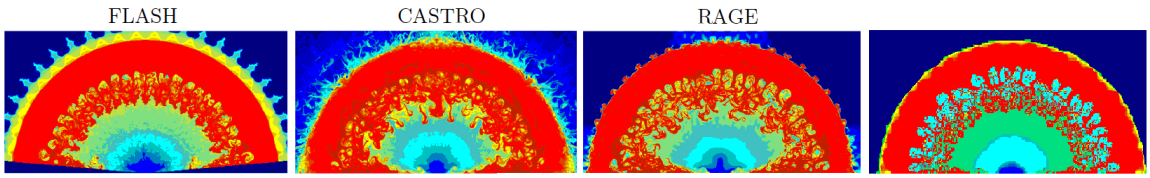
В трехмерной постановке рассматривался сектор $0 \le \theta \le 90^o$ и $0 \le \phi \le 90^o$. Аналогично двумерному случаю, использовалась квазисферическая сетка с базовым размером ячеек 0.02 см и 0.04 см на переферии системы. В зоне перемешивания была включена адаптация 2-го уровня с прогнозированием положения зоны перемешивания.

Коэффициент разбалансированности в расчетах $K_{max} = 1.2$. Число шагов в цикле $N_{\text{АБС}} = 300$.

На рис. 9 представлено распределение плотности в системе до и в момент максимального сжатия центрального вещества ($t = 1.5$ сек и $t = 2.5$ сек). Сравнение с открытыми CFD-программами (FLASH, RAGE, CASTRO) [14] показывает хорошее сходство.



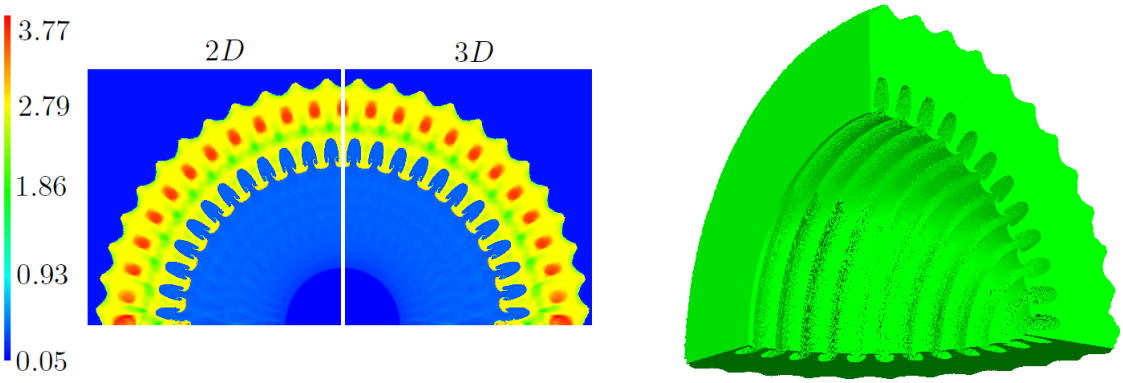
а) профиль плотности при $t = 1.5$ сек



б) профиль плотности при $t = 2.5$ сек

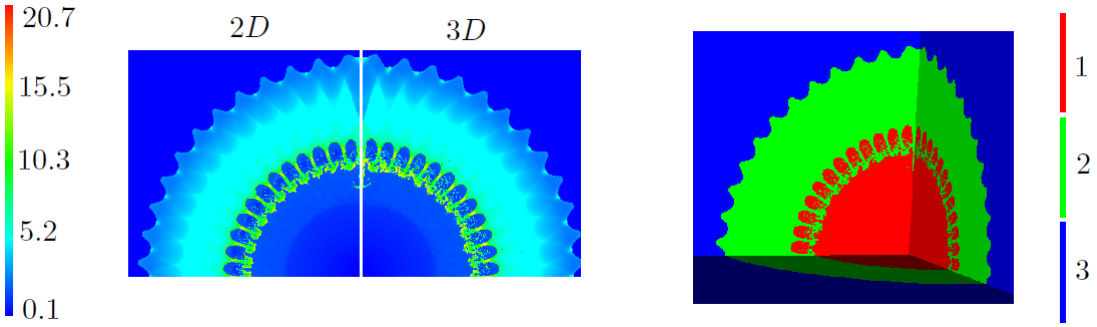
Рис. 9. Профиль плотности на для разных программ. Правый столбец — расчет программы, рассматриваемой в рамках данной статьи

На рис. 10 и 11 представлен результат моделирования в трехмерной постановке. Как видно, результат трехмерного расчета согласуется с двумерным.



а) профиль плотности б) вещество 2

Рис. 10. Сравнение решений в 2D и 3D постановках на момент времени $t = 1.5$ сек

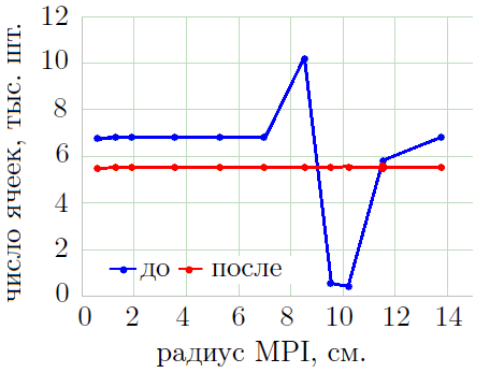


а) профиль плотности б) распределение веществ

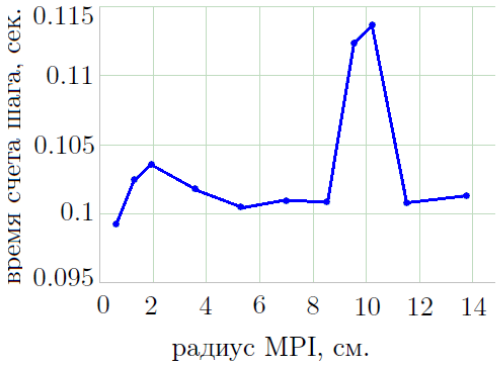
Рис. 11. Сравнение решений в 2D и 3D постановках на момент времени $t = 2$ сек

3.3. Результаты тестирования

Как видно из рис. 12, количество ячеек на MPI-фрагментах вдоль выделенной линии после балансировки по числу ячеек становится одинаковым. При этом время счета шага этих MPI-процессов отличается значительно.



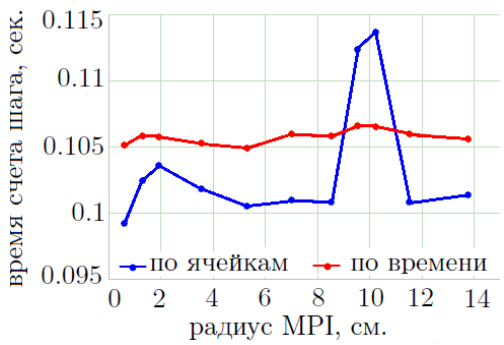
а) число ячеек на MPI-фрагментах



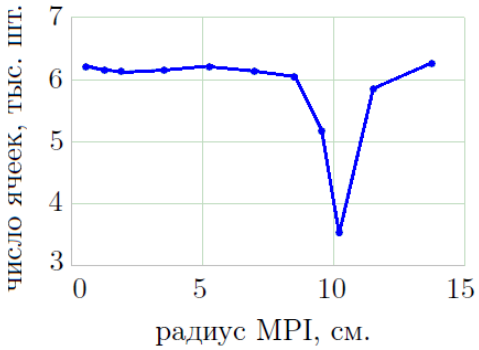
б) время счета шага

Рис. 12. Число ячеек на MPI-фрагментах вдоль выделенного направления и время счета шага при балансировке по числу ячеек

На рис. 13 слева представлено распределение времени счет шага для MPI-процессов, расположенных вдоль выделенного направления, при балансировке вычисительной нагрузки по фактическому времени счета. При этом число ячеек на этих MPI-фрагментах отличается (см. рис. 13 справа).



а) время счета шага



б) число ячеек на MPI-фрагментах

Рис. 13. Время счета шага и число ячеек на MPI-фрагментах при балансировке по времени

По представленным графикам можно сделать вывод о том, что реализованные алгоритмы балансировки вычислительной нагрузки по ячейкам и по времени работают корректно, выравнивая число ячеек на MPI-фргаменте или время их счета.

Для анализа эффективности работы реализованного алгоритма балансировки вычислительной нагрузки было проведено несколько серий сравнительных расчетов тестовой задачи в двумерной постановке.

В первой серии расчетов сравнивалось полное время счета тестовой задачи без прогнозирования положения зоны перемешивания для двух режимов балансировки: по числу ячеек и по времени счета. Оценивалось ускорение времени счета задачи относительно рас-

чета без балансировки (время счета без балансировки — 17.7 ч.). Результаты исследования приведены в табл. 2.

Таблица 2. Время счета тестовой задачи в 2D постановке при разных режимах балансровки вычислительной нагрузки без прогнозирования

режим	время счета, час	ускорение, раз
по ячейкам	10.4	1.7
по времени	9.2	1.9

При проведении расчета без прогнозирования положения зоны перемешивания балансировка вычислительной нагрузки сокращает время счета почти в 2 раза. При этом расчет с балансировкой по фактическому времени счета прошел на 11.5% быстрее, чем с балансировкой по числу ячеек.

Во второй серии расчетов сравнивалось время счета задачи для разных режимов балансировки при включении прогнозирования положения зоны перемешивания. Расчет без балансировки прошел за 16.3 ч.

Таблица 3. Временные характеристики расчета тестовой задачи в 2D постановке при разных режимах балансровки вычислительной нагрузки с прогнозированием

режим	время счета, час	ускорение, раз
по ячейкам	8.9	1.8
по времени	5	3.3
автоматический	6.5	2.5

Как видно из табл. 3, расчет с адаптацией с запасом прошел на 10–15% быстрее, чем расчет без прогнозирования. При этом наибольшее ускорение в расчете относительно расчета без балансировки вычислительной нагрузки было получено при балансировке по времени (в 3.3 раза). Балансировка по числу ячеек дает наименьшее ускорение (1.8 раз).

В третьей серии расчетов оценивалось практическое значение наиболее оптимального количества шагов в цикле А-Б-С. Для этого в расчете тестовой задачи число шагов N_{ABC} варьировалось от $N_{ABC} = 50$ до $N_{ABC} = 900$. Из графика 14 видно, что минимум времени счета достигается при $N_{ABC} = (200; 400)$, что согласуется с теоретическим значением $N_{ABC}^{\text{теор.}} = 244$.

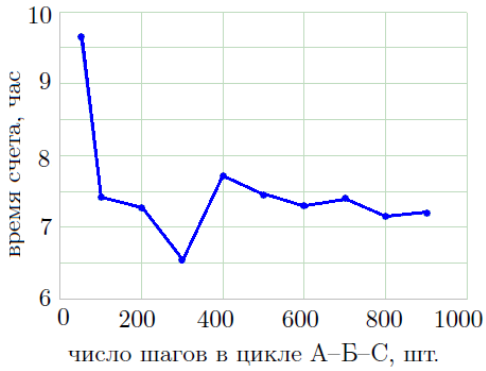


Рис. 14. Зависимость времени счета задачи от числа шагов в цикле А-Б-С

На первой и второй диаграммах на рис. 15 показано, какую долю общего времени счета задачи занимают расходы на адаптацию и балансировку.

а) 2D расчет без прогнозирования

б) 2D расчет с прогнозированием

в) 3D расчет с прогнозированием

Рис. 15. Накладные расходы на адаптацию и балансировку

В случае, когда не используется адаптация с запасом, накладные расходы на составляют до 45% времени счета задачи. При использовании прогнозирования в рамках цикла А-Б-С накладные расходы снижаются и не превышают 5%.

Также была проведена серия трехмерных расчетов с использованием прогнозирования положения зоны, в которой оценивалось время счета задачи на характерном временном интервале $t = 1$ сек до $t = 2$ сек при балансировке по числу ячеек, по времени счета и в автоматическом режиме. В табл. 4 представлены времена счета задачи.

Таблица 4. Время расчета интервала $t \in (1, 2)$ сек в 3D постановке при разных режимах балансровки вычислительной нагрузки

режим	время счета, дни	накладные расходы на АБ, %
по ячейкам	4.5	7.6
по времени	–	–
автоматический	3.9	9.1

При проведении расчета с использованием балансировки по времени наличие сильно неравномерного количества данных на MPI-процессах привело к выходу за пределы оперативной памяти на вычислительном узле. Действительно, дисбаланс числа ячеек на MPI-фрагментах при балансировке по времни в 3D расчете достигает ∼10, а при адапатции 2-го уровня число ячеек на параобласти может увеличиться еще до 64 раз. Именно в таком случае и происходит выход за пределы памяти. Проведение расчета с балансировкой по времени возможно лишь с увеличением вычислительных ресурсов. Именно для того, чтобы минимизировать риски переполнения памяти, в автоматическом режиме между этапами измельчения и укрупнения сетки производятся балансировки по числу ячеек, и лишь перед счетными шагами цикла А-Б-С делается балансировка по времени счета.

Результаты трехмерных расчетов показали, что расчет характерного интервала прошел на 15% быстрее при балансировке в автоматическом режиме. Накладные расходы на адаптацию и балансировку в трехмерных расчетах отражены в табл. 4 и на рис. 15в.

Заключение

В трехмерной газодинамической эйлеровой программе реализован функционал динамической балансировки вычислительной нагрузки вдоль линий, который может работать в трех режимах: выравнивать нагрузку по числу ячеек, по фактическому времени счета, или в автоматическом режиме с построением цикла Адаптация—Балансировка—Счет. Для анализа работоспособности аглоритма проводилось моделирование неустойчивого течения, возникающего в задаче о сферическом сжатии легкого центрального вещества тяжелой оболочкой (задача Янгса). Было проведено несколько серий тестовых расчетов на квазисферической динамически адаптивной сетке, которая имела сферическую декомпозицию на MPI-фрагменты. Адаптация проводилась в послойном режиме.

Для двумерной постановки было проведено несколько серий тестовых расчетов. В первой серии сравнивалось полное время счета задачи при разных режимах балансировки вычислительной нагрузки без использования прогнозирования положения зоны перемешивания. Анализ результатов показал, что балансировка вычислтельной нагрузки по времени счета на 10% сокращает время счета задачи. При этом наклыдные расходы на адаптацию и балансировку составляют значительную часть от общего времени (до 45% для балансировки по времени).

Во второй серии расчетов оценивалось полное время счета тестовой задачи при активации прогнозирования положения зоны перемешивания на $N_{\text{АБС}} = 300$ счетных шагов. Анализ результатов показал, что внедрение адаптации с запасом в рамках цикла А-Б-С позволяет снизить накладные расходы на адаптацию и балансировку до 5%. При этом общее время счета тестовой задачи сократилось на 30% относительно расчета без прогнозирования.

В третьей серии расчетов варьировалось число шагов в цикле А-Б-С, на которое делался запас. Результат показал, что теоретическая оценка оптимального количества шагов $N_{\text{АБС}}^{\text{теор.}} = 244$ в цикле А-Б-С совпадает с практической.

В трехмерной постановке была проведена одна серия расчетов, которая показала, что расчеты практичнее проводить либо с балансировкой по ячейкам, либо с использованием автоматического режима. При балансировке по времени счета дисбаланс количества ячеек может приводить к выходу за пределы оперативной памяти. При использовании автоматического режима расчет замеряемого характреного интервала прошел на 13 % быстрее, чем балансировка по времени. При этому накладные расходы на балансировку и адаптацию менее 9%.

В результате проведенных исследований можно сделать вывод о том, что режим балансировки по фактическому времени счета наиболее эффективен для двумерных расчетов. В трехмерных же расчетах практичнее выбирать автоматический режим балансировки вычислительной нагрузки при моделировании неустойчивых течений.

Литература

1. Янилкин Ю.В., Стаценко В.П., Козлов В.И. Математическое моделирование турбулентного перемешивания в сжимаемых средах. Том 2: учебное пособие. Саров: Российский федеральный ядерный центр – Всероссийский НИИ экспериментальной физики, 2020. 407 с.
2. Youngs D.L., Williams R.J. Turbulent Mixing in Spherical Implosions // Int. J. Numer. Methods Fluids. 2008. Vol. 56, no. 8. P. 1597–1603. DOI: 10.1002/fld.1594.

3. Глазырин И.В., Михайлов Н.А. Метод укручения контактных границ для моделирования трехмерных многофазных сжимаемых течений в эйлеровых переменных // Забабахиский научные чтения: Сборник тезисов XIII международной конференции, Снежинск, 20–24 марта 2017. Снежинск: Из-во РФЯЦ-ВНИИТФ, 2017. С. 326.
4. Chevalier C., Pellegrini F. PT-Scotch: A Tool for Efficient Parallel Graph Ordering // Parallel Computing. 2008. Vol. 34, no. 6. P. 318–331. DOI: 10.1016/j.parco.2007.12.001.
5. Karypis G. METIS and ParMETIS // Encyclopedia of Parallel Computing. Springer US, 2011. P. 1117–1124. DOI: 10.1007/978-0-387-09766-4_500.
6. Toro E. Rieman Solvers and Numerical Methods for Fluid Dynamics: a Practical Introduction. Berlin, Heidelberg: Springer Verlag, 2009. 721 p. DOI: 10.1007/b79761.
7. Darwish M., Moukalled F. TVD-schemes for Unstructured Grids // International Journal of Heat and Mass Transfer. 2003. Vol. 46, no. 8. P. 599–611. DOI: 10.1016/S0017-9310(02)00330-7.
8. Глазырин И.В., Михайлов Н.А. Конечно-объемная схема для многокомпонентных сжимаемых течений на неструктурированной сетке в трехмерной программе Фокус // ЖФММФ. 2021. Т. 61, № 6. С. 1019–1033. DOI: 10.31857/S0044466921060041.
9. Балашов Н.М., Глазырин И.В., Михайлов Н.А. Двумерная динамическая адаптация сетки в газодинамической программе // Параллельные вычислительные технологии – XIV международная конференция, ПаВТ'2020, Пермь, 31 марта – 2 апреля 2020. Короткие статьи и описания плакатов. Челябинск: Издательский центр ЮУрГУ, 2020. С. 325.
10. Jasak H., Tukovic Z. Automatic Mesh Motion for the Unstructed Finite Volume Method // Transactions of FAMENA. 2006. Vol. 30, no. 2. P. 1–20.
11. Colella P., Berger M. Local Adaptive Mesh Refinement for Shock Hydrodynamics // Journal of Computational Physics. 1989. Vol. 82, no. 1. P. 64–84. DOI: 10.1016/0021-9991(89)90035-1.
12. Глазырин И.В., Глазырина Н.В., Михайлов Н.А., Писклова М.А. Использование динамической адаптации сетки для расчетов неустойчивостей Рэлея—Тейлора // Забабахинские научные чтения: сборник материалов XVI Международной конференции 29 мая – 2 июня 2023. Снежинск: Изд-во РФЯЦ-ВНИИТФ, 2023. С. 364.
13. Оленев Н.Н. Основы параллельного программирования в системе MPI. Москва: Вычислительный центр А.А. Дородницына, 2005. 79 с.
14. Joggerst C.C., Nelson A., Woodward P., *et al.* Cross-code Comparison of Mixing During the Implosion of Dense Cylindrical and Spherical Shells // Journal of Computational Physics. 2014. Vol. 82, no. 1. P. 154–173. DOI: 10.1016/j.jcp.2014.06.037.

Титова Александра Михайловна, научный сотрудник, научно-теоретическое отделение 2, ФГУП «Российский Федеральный Ядерный Центр – Всероссийский научно-исследовательский институт технической физики имени академика Е.И. Забабахина» (Снежинск, Российская Федерация)

Михайлов Никита Анатольевич, к.ф.-м.н., начальник лаборатории, научно-теоретическое отделение 2, ФГУП «Российский Федеральный Ядерный Центр – Всероссийский научно-исследовательский институт технической физики имени академика Е.И. Забабахина» (Снежинск, Российская Федерация)

Юсупов Юлиус Фаритович, научный сотрудник, научно-теоретическое отделение 2, ФГУП «Российский Федеральный Ядерный Центр – Всероссийский научно-исследовательский институт технической физики имени академика Е.И. Забабахина» (Снежинск, Российская Федерация)

DOI: 10.14529/cmse250304

DYNAMIC COMPUTATIONAL LOAD BALANCING DURING SIMULATIONS OF INSTABILITY FLOWS

© 2025 A.M. Titova, N.A. Mikhaylov, Y.Y. Yusupov

Federal State Unitary Enterprise "Russian Federal Nuclear Center – All-russian Research Nuclear Institute of Tecknical Physics named after Academician E.I. Zababakhin", (FSUE "RFNC-VNIITF named after Academ. E.I. Zababakhin") (Vasiliev st. 13, Snezhinsk, Chelyabinsk region, 456770 Russian Federation) E-mail: a.m.titova@vniitf.ru, n.a.mikhaylov@vniitf.ru, yusupovyuf@vniitf.ru

Received: 19.04.2025

In this work we describe the algorithm of dynamic computational load balancing during modeling of instability flows over dynamically refined mesh using three-dimensional eulerian gas dynamic program. Balancing of computational load between MPI-fragments is executed along lines (set of MPI-fragments that are situated along preferential direction) independently and may be carried out in three regimes: accounting number of cells, accounting real calculation time, and automatic (or composite) regime. We use adaptation with reservation to minimize overheads for adaptation and balancing: we construct a sequence of adaptation, balancing and calculation stages during fixed count of steps – "Adaptation — Balancing — Calculation" cycle (A–B–C cycle). The optimal count of steps in A–B–C cycle is determined by minimization of task calculation time that depends on count of cells. For testing of realized algorithm we considered a spherical implosion of a light material by a dense shell (Youngs' problem). A source of instabilities was the initial harmonic perturbation on the contact boundary. The calculations are made using second-level dynamically adaptive mesh refinement in the region of instability growth. We analyze the calculation time of the problem using different regimes of load balancing algorithm. The result of the research proves the efficiency of using adaptive mesh refinement with reservation within A–B–C cycle. The automatic regime of load balancing is the most stable for three-dimensional modelling.

Keywords: gas dynamics, instability flows, spherical implosion, adaptation mesh refinement, load balancing, computational load.

FOR CITATION

Titova A.M., Mikhaylov N.A., Yusupov Y.Y. Dynamic Computational Load Balancing During Simulations of Instability Flows. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2025. Vol. 14, no. 3. P. 59–76. (in Russian) DOI: 10.14529/cmse250304.

This paper is distributed under the terms of the Creative Commons Attribution-Non Commercial 4.0 License which permits non-commercial use, reproduction and distribution of the work without further permission provided the original work is properly cited.

References

1. Yanilkin Y.V., Stacenko V.P., Kozlov V.I. Mathematical Modeling of Turbulent Mixing in Compressible Medium. Vol. 2: Training manual. Sarov: Russia Research Nuclear Center – All-Russian SRI of Experimental Physics, 2020. 407 p. (in Russian).

2. Youngs D.L., Williams R.J. Turbulent Mixing in Spherical Implosions. Int. J. Numer. Methods Fluids. 2008. Vol. 56, no. 8. P. 1597–1603. DOI: 10.1002/fld.1594.

3. Glazyrin I.V., Mikhaylov N.A. The Torsion Method for Contact Bounds for Modeling of Three-dimensional Multi-component Compressible Flows in Eulerian Variables. Zababakhin Scientific Readings: Collection of Abstracts XIII International Conference, Snezhinsk, 20–24 March 2017. Snezhinsk: Publishing of RFNC VNIITF, 2017. P. 326. (in Russian).

4. Chevalier C., Pellegrini F. PT-Scotch: A Tool for Efficient Parallel Graph Ordering. Parallel Computing. 2008. Vol. 34, no. 6. P. 318–331. DOI: 10.1016/j.parco.2007.12.001.

5. Karypis G. METIS and ParMETIS. Encyclopedia of Parallel Computing. Springer US, 2011. P. 1117–1124. DOI: 10.1007/978-0-387-09766-4_500.

6. Toro E. Rieman Solvers and Numerical Methods for Fluid Dynamics: a Practical Introduction. Berlin, Heidelberg: Springer Verlag, 2009. 721 p. DOI: 10.1007/b79761.

7. Darwish M., Moukalled F. TVD-schemes for Unstructured Grids. International Journal of Heat and Mass Transfer. 2003. Vol. 46, no. 8. P. 599–611. DOI: 10.1016/S0017-9310(02)00330-7.

8. Glazyrin I.V., Mikhailov N.A. Finite-Volume Scheme for Multicomponent Compressible Flow on Unstructed Meshes in the Focus 3D Code. JCMMP. 2021. Vol. 61, no. 6. P. 1019–1033. (in Russian) DOI: 10.31857/S0044466921060041.

9. Balashov N.M., Glazyrin I.V., Mikhaylov N.A. Two-dimensional Adaptive Mesh Refinement in Gas Dynamics Program. Parallel Computational Technologies – XIV International Conference, PCT'2020, Perm, 31 March – 2 April 2020. Short articles and posters' description. Chelyabinsk: Publishing of the South Ural University, 2020. P. 325. (in Russian).

10. Jasak H., Tukovic Z. Automatic Mesh Motion for the Unstructed Finite Volume Method. Transactions of FAMENA. 2006. Vol. 30, no. 2. P. 1–20.

11. Colella P., Berger M. Local Adaptive Mesh Refinement for Shock Hydrodynamics. Journal of Computational Physics. 1989. Vol. 82, no. 1. P. 64–84. DOI: 10.1016/0021-9991(89)90035-1.

12. Glazyrin I.V., Glazyrina N.V., Mikhaylov N.A., Pisklova M.A. Using Dynamic Mesh Adaptation for Rayleigh-Taylor Instability Calculations. Zababakhin Scientific Readings: Collection of Materials XVI International Conference, 29 May – 2 June 2023. Snezhinsk: Publishing of RFNC VNIITF, 2023. P. 364. (in Russian).

13. Olenev N.N. Basics of MPI Parallel Programming. Moscow: Calculating center A.A. Dorodnicina, 2005. 79 p. (in Russian).

14. Joggerst C.C., Nelson A., Woodward P., *et al.* Cross-code Comparison of Mixing During the Implosion of Dense Cylindrical and Spherical Shells. Journal of Computational Physics. 2014. Vol. 82, no. 1. P. 154–173. DOI: 10.1016/j.jcp.2014.06.037.

СВЕДЕНИЯ ОБ ИЗДАНИИ

Научный журнал «Вестник ЮУрГУ. Серия «Вычислительная математика и информатика» основан в 2012 году.

Учредитель — Федеральное государственное автономное образовательное учреждение высшего образования «Южно-Уральский государственный университет» (национальный исследовательский университет).

Главный редактор — Л.Б. Соколинский.

Свидетельство о регистрации ПИ ФС77-57377 выдано 24 марта 2014 г. Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций.

Журнал включен в Реферативный журнал и Базы данных ВИНИТИ; индексируется в библиографической базе данных РИНЦ. Журнал размещен в открытом доступе на Всероссийском математическом портале MathNet. Сведения о журнале ежегодно публикуются в международной справочной системе по периодическим и продолжающимся изданиям «Ulrich's Periodicals Directory».

Решением Президиума Высшей аттестационной комиссии Министерства образования и науки Российской Федерации журнал включен в «Перечень рецензируемых научных изданий, в которых должны быть опубликованы основные научные результаты на соискание ученой степени кандидата наук, на соискание ученой степени доктора наук» по научным специальностям и соответствующим им отраслям науки: 1.2.3 – Теоретическая информатика, кибернетика (физико-математические науки), 2.3.5 – Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей (физико-математические науки).

Подписной индекс научного журнала «Вестник ЮУрГУ», серия «Вычислительная математика и информатика»: 10244, каталог «Пресса России». Периодичность выхода — 4 выпуска в год.

Адрес редакции: 454080, г. Челябинск, ул. С. Кривой, 79, Издательский центр ЮУрГУ, каб. 2.

Адрес издателя: 454080, г. Челябинск, проспект Ленина, 76.

ПРАВИЛА ДЛЯ АВТОРОВ

1. Правила подготовки рукописей и пример оформления статей можно загрузить с сайта серии `https://vestnikvmi.susu.ru`. **Статьи, оформленные без соблюдения правил, к рассмотрению не принимаются.**
2. Адрес редакционной коллегии научного журнала «Вестник ЮУрГУ», серия «Вычислительная математика и информатика»:

   Россия 454080, г. Челябинск, пр. им. В.И. Ленина, 76, ЮУрГУ, НОЦ ИИКТ,
   зам. главного редактора Цымблеру М.Л.
3. Адрес электронной почты редакции: `vestnikvmi@susu.ru`.
4. **Плата с авторов за публикацию рукописей не взимается, гонорары авторам не выплачиваются.**

ВЕСТНИК
ЮЖНО-УРАЛЬСКОГО
ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА
Серия
«ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА И ИНФОРМАТИКА»
2025 Том 14, № 3

16+

Техн. редактор *А.В. Миних*

Издательский центр Южно-Уральского государственного университета

Подписано в печать 30.09.2025. Дата выхода в свет 30.10.2025. Формат 60×84 1/8. Печать цифровая.
Усл. печ. л. 9,30. Тираж 500 экз. Заказ 272/289. Цена свободная.

Отпечатано в типографии Издательского центра ЮУрГУ.
454080, г. Челябинск, проспект Ленина, 76.