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

2021, том 10, № 1

SERIES

## COMPUTATIONAL MATHEMATICS AND SOFTWARE ENGINEERING

2021, volume 10, no. 1

1943

# ВЕСТНИК

## ЮЖНО-УРАЛЬСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА

2021
Т. 10, № 1

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

# BULLETIN

## OF THE SOUTH URAL STATE UNIVERSITY

## SERIES

## "COMPUTATIONAL MATHEMATICS AND SOFTWARE ENGINEERING"

2021

Vol. 10, no. 1

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

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


**South Ural State University**

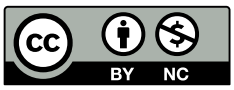
The scope of the journal:

- 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

# Содержание

# Contents





**УДК 519.67** **DOI 10.14529/cmse210101**

# НАПРАВЛЕННЫЕ СПЛАЙНЫ И ИХ ИСПОЛЬЗОВАНИЕ ДЛЯ СГЛАЖИВАНИЯ ВЫБРОСОВ И ИЗЛОМОВ ИНТЕРПОЛЯНТА

**© 2021 В.А. Коднянко**
*Сибирский федеральный университет*
*(660074 Красноярск, ул. Академика Киренского, д. 26А)*
*E-mail: kowlad@rambler.ru*
Поступила в редакцию: 18.07.2020

Сформулирован и предложен метод построения направленного кубического сплайна для набора точек на плоскости. Проведено сравнение сплайна с *B*-сплайном Шёнберга, сплайнами Акимы и Катмулла—Рома. Показано, что для неравноотстоящих точек в сравнении с *B*-сплайном он дает значительно меньшие выбросы и практически лишен сильных изломов, которые свойственны сплайнам Акимы. Сплайн не дает петель и осцилляций, которые являются характерным недостатком параметрических сплайнов, в частности, эрмитовых, к числу которых относится сплайн Катмулла—Рома. Предложен быстрый метод оптимизации направляющего коэффициента сплайна, цель которой состоит в минимизации разрывов второй производной функции в ее промежуточных точках. Приведен пример оптимизации направленного сплайна третьего порядка. Также предложен направленный сплайн четвертого порядка, который лишен изломов. Сформулирован метод оптимизации направленного сплайна четвертого порядка, изложен алгоритм его оптимизации. Критериями оптимизации являются длина сплайна и наименьшее расстояние между его глобальными максимумом и минимумом. Показано, что в сравнении с сплайна Шёнберга направленный сплайн четвертого порядка имеет меньшие выбросы. Предложен метод автоматического притупления острых пиков кривых, который можно применять ко всем типам сплайнов.

*Ключевые слова: сплайн, сплайн Шёнберга, сплайн Акимы, направленный сплайн.*

**ОБРАЗЕЦ ЦИТИРОВАНИЯ**

Коднянко В.А. Направленные сплайны и их использование для сглаживания выбросов и изломов интерполянта // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2021. Т. 10, № 1. С. 5–19. DOI: 10.14529/cmse210101.

## Введение

При проведении вычислительных или натурных экспериментов однофакторную связь между входной и выходной величинами обычно определяют посредством вычислений либо измерений. Результатом являются данные в виде набора точек на плоскости

$$(x_0, y_0), (x_1, y_1), \dots, (x_{n-1}, y_{n-1}), (x_n, y_n), x_{i-1} < x_i, i = 1,2, \dots, n. \quad (1)$$

На его основе зачастую необходимо получить плавную кривую, которая должна проходить через экспериментальные точки.

Задачу можно решить посредством использования интерполяционных методов [1–3]. Для наборов с малым числом точек удовлетворительные результаты дают интерполяционные методы Лагранжа, Ньютона, Стирлинга [2–4]. Однако, с увеличением количества точек набора (1) интерполяционные полиномы в области крайних точек дают осцилляции недопустимо большой амплитуды [4].

Этого недостатка лишены сплайны [5–10, 20–29], которые находят широкое использование в практических приложениях, как-то: при численном решении нелинейных уравнений, сглаживании функций, сжатии и восстановлении графических изображений, ряде других применений. Наиболее известным среди них является *B*-сплайн Шёнберга [5, 17]. Сплайн обеспечивает безупречную гладкость интерполянта для равноотстоящих точек $x_i$ $(i = 0,1, \dots, n)$, для которых

$$h_i = x_i - x_{i-1} = h = const. \quad (2)$$

Если (2) не выполняется, то плавность кривых, как правило, нарушается. В таких случаях $B$-сплайн может давать значительные осцилляции кривой (так называемые «выбросы»), которые имеют место в области отрезков с малыми $h_i$.

Избежать выбросов $B$-сплайна позволяют сплайн Акимы [11, 16] и эрмитовы сплайны, частным случаем которых является применяемый в графической геометрии параметрический сплайн Катмулла—Рома [12, 13]. Однако и эти сплайны имеют свои недостатки. Так, сплайн Акимы часто дает неприемлемые изломы графика функции в узловых точках, которые видны на рис. 1 (кривая 2).

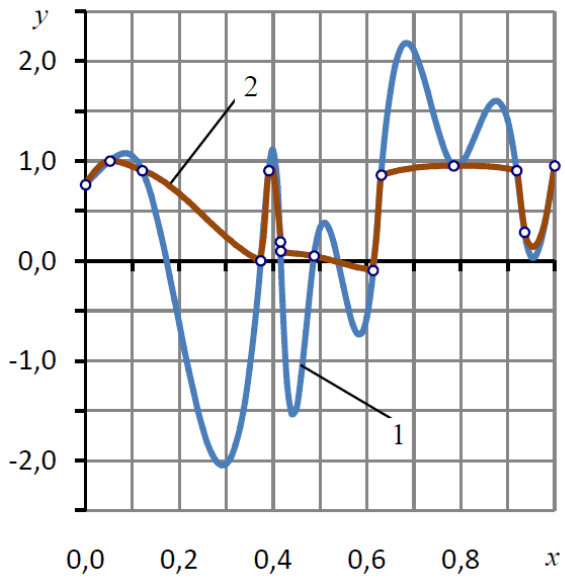



**Рис 1.** Графики $B$-сплайна (1) и сплайна Акимы (2)

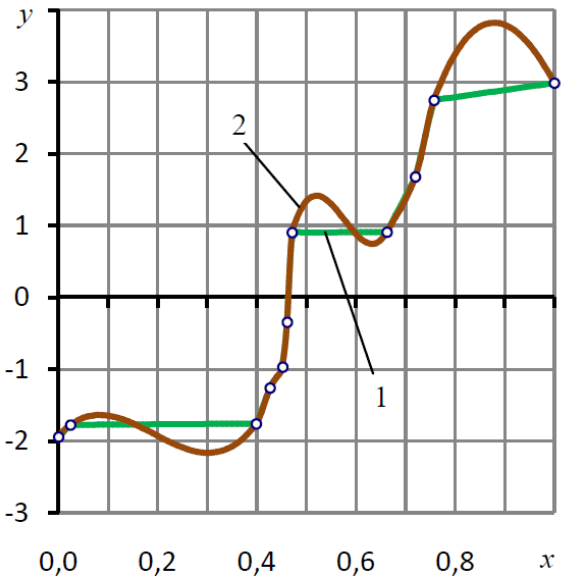



**Рис 2.** Кривые $B$-сплайна (1) и сплайна Катмулла—Рома (2)

Более гладкие переходы между соседними полиномами дают сплайны Катмулла—Рома (рис. 2). Однако при малых $h_i$ они могут образовывать петли в узловых точках. Кроме того, на параметрической зависимости $x = x(t)$ которая в силу (1), очевидно, должна быть монотонно-возрастающей, могут появляться локальные экстремумы, что свидетельствует о явных недостатках алгоритма интерполяции при обработке наборов данных, содержащих точки, находящиеся на относительно малом расстоянии друг от друга.

Таким образом, актуальной является задача разработки сплайна, который характеризуется меньшими выбросами $B$-сплайнов и существенно менее выраженными изломами сплайнов Акимы. Наряду с этим поставлена задача создания такой модели сплайна, которая не уступала бы по скорости вычислений алгоритму сплайна Акимы при изменении одной или нескольких точек, что в сравнении с $B$-сплайном является характерным преимуществом данного сплайна.

Статья организована следующим образом. В разделе 1 рассмотрена методика построения направленного кубического сплайна (НС3-сплайна). В разделе 2 предложен метод и приведен пример оптимизации такого сплайна. Раздел 3 посвящен построению направленного сплайна четвертого порядка (НС4-сплайна). В разделе 4 изложена методика оптимизации направленного сплайна четвертого порядка по критерии минимума длины кривой и минимума расстояния между ее глобальными максимумом и минимумом. В раз-

деле 5 предложен метод притупления острых пиков сплайнов. Методика может быть применена ко всем типам рассмотренных сплайнов. В заключении приводится краткая сводка результатов, полученных в рамках данного исследования, и указаны направления дальнейших исследований.

Ниже рассмотрена методика построения и алгоритм оптимизации такого сплайна.

## 1. Построение направленного кубического сплайна (НС3-сплайна)

На каждом отрезке $[x_{i-1}, x_i]$ будем представлять сплайн $S(x)$ полиномом третьей степени

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3, i = 1,2, \dots, n. \tag{3}$$

На стыках соседних отрезков $[x_{i-1}, x_i]$ для полиномов (3) потребуем выполнения условий непрерывности сплайна и ее первой производной в точках (1)

$$S_i(x_{i-1}) = S_{i-1}(x_{i-1}), \tag{4}$$
$$S_i'(x_{i-1}) = S_{i-1}'(x_{i-1}). \tag{5}$$

Используя (3)–(5), получим

$$\begin{cases} a_i = y_i, \\ a_i - h_i b_i + h_i^2 c_i - h_i^3 d_i = y_{i-1}, \\ b_i - 2h_i c_i + 3h_i^2 d_i = b_{i-1}. \end{cases} \tag{6}$$

Если считать $b_i$ известными, то (6) позволяет получить систему уравнений относительно неизвестных коэффициентов $c_i$, $d_i$

$$\begin{cases} c_i - h_i d_i = \dfrac{b_i - u_i}{h_i}, \\ 2c_i - 3h_i d_i = \dfrac{b_i - b_{i-1}}{h_i}, \end{cases} \tag{7}$$

где

$$u_i = \frac{y_i - y_{i-1}}{h_i}.$$

Решив (6), найдем

$$c_i = \frac{2b_i + b_{i-1} - 3u_i}{h_i}, d_i = \frac{b_i + b_{i-1} - 2u_i}{h_i^2}. \tag{8}$$

В простейшем случае для крайних отрезков $[x_0, x_1]$ и $[x_{n-1}, x_n]$ можно воспользоваться производной линейного интерполяционного сплайна, положив

$$b_0 = u_1, b_n = u_n.$$

Эти же коэффициенты можно определить при помощи построенного по трем точкам $(x_{i-1}, y_{i-1})$, $(x_i, y_i)$, $(x_{i+1}, y_{i+1})$ интерполяционного полинома Лагранжа [2]

$$f_i(x) = y_i + \frac{[u_i h_{i+1} + u_{i+1} h_i + (u_{i+1} - u_i)(x - x_i)](x - x_i)}{h_{i+1} + h_i}, i = 1,2, \dots, n-1,$$

$$b_0 = f_1'(x_0) = u_1 - \frac{h_1(u_2 - u_1)}{h_1 + h_2}, b_n = f_{n-1}'(x_n) = u_n + \frac{h_n(u_n - u_{n-1})}{h_{n-1} + h_n}.$$

Для промежуточных отрезков $[x_{i-1}, x_i]$ будем использовать формулу

$$b_i = \alpha u_i + (1 - \alpha) u_{i+1}, \tag{9}$$

где коэффициент $\alpha \in [0,1]$.

Из (4)–(9) следует, что на отрезке $[x_{i-1}, x_i]$ функция $S_i(x)$ полностью определяется только тремя точками, в то время как локальность сплайна Акимы определятся пятью точками [11], а *B*-сплайн всеми точками набора (1). Это свойство характеризует лучшее сравнительное быстродействие рассматриваемого метода при коррекции коэффициентов (3) на случай изменения отдельных точек набора (1).

Разделенная разность $u_i$ представляет собой угловой коэффициент соответствующего сегмента линейного интерполяционного сплайна, который показан на рис. 3.

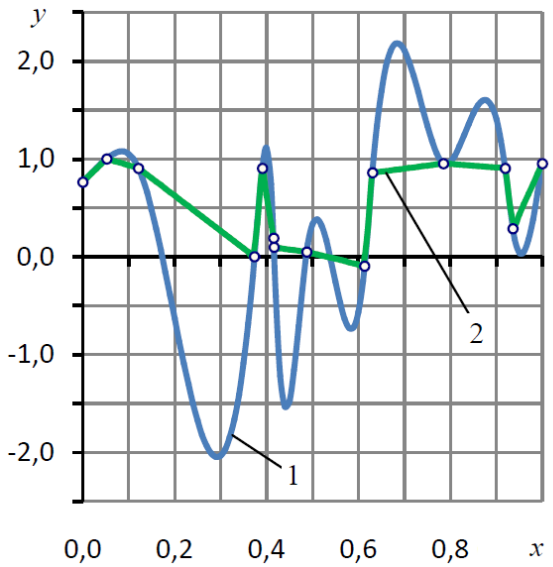



**Рис 3.** *B*-сплайн (1) и линейный интерполяционный сплайн (2)

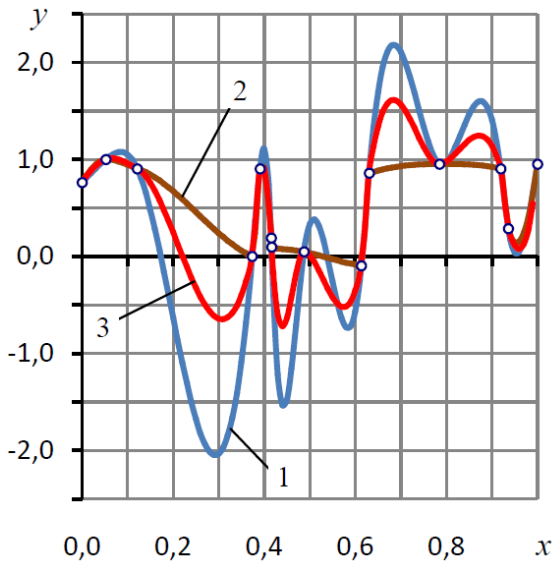



**Рис 4.** *B*-сплайн (1), сплайн Акимы (2) и направленный сплайн (3)

Очевидно, (9) является угловым коэффициентом касательной в точке $(x_i, y_i)$ сопряжения сегментов сплайна, находящихся по обе стороны от нее. Варьируя коэффициент $\alpha$, можно отрегулировать положение касательных в промежуточных точках сплайна, направляя его так, чтобы выбросы были бы минимальными, а изломы не слишком заметными. Назовем такой сплайн *направленным*, а коэффициент $\alpha$ *направляющим*.

В простейшем случае можно положить $\alpha = 0{,}5$, то есть считать, что направляющая касательная сплайна в промежуточных узловых точках должна занимать среднее положение относительно соседних отрезков линейного интерполяционного сплайна.

На рис. 4 показан пример интерполяции набора данных при помощи *B*-сплайна (1), сплайна Акимы (2) и направленного сплайна (3).

Видно, что, в отличие от сплайна Акимы, направленный сплайн не имеет видимых изломов, чего для многих практических применений оказывается достаточным основанием для принятия решения об удовлетворительной аппроксимации исходной табличной функции подобными сплайнами. В сравнении с *B*-сплайном менее выражены и выбросы сплайна.

## 2. Оптимизация НС3-сплайна

Оптимизация преследует цель сделать менее заметными изломы направленного сплайна, которые определяются абсолютной величиной разности значений вторых производных в точке сопряжения соседних полиномов. Величину такого разрыва в промежуточных точках можно определить формулой

$$D_i(\alpha) = \frac{\varepsilon}{2}|S_i''(x_{i-1}) - S_{i-1}''(x_{i-1})| = |c_i - c_{i-1} - 3d_i|, (i = 1,2,\dots,n-1). \qquad (10)$$

Вычислительный эксперимент показал, что в большинстве случаев при крайних значениях направляющего коэффициента $\alpha = 0$ и $\alpha = 1$, когда углы касательной к кривой в промежуточных точках и одного из отрезков линейного сплайна совпадают, направленный сплайн обычно имеет не только выраженные изломы, но и большие выбросы. При промежуточных значениях $\alpha$ эти недостатки менее заметны. Следовательно, существует такие значения $\alpha$, при котором эти недостатки будут наименее выражены.

Суть оптимизации заключается в отыскании такого $\alpha = \alpha_{opt}$, при котором наибольший разрыв

$$D(\alpha) = \max D_i(\alpha), i = 1,2, \dots, n-1 \tag{11}$$

кажется минимальным.

Установлено, что в большинстве случаев $D(\alpha)$ является кусочно-линейной функцией с единственной точкой излома. Однако нередки случаи функций с несколькими изломами.

Найти значение $\alpha_{opt}$ можно одним из методов минимизации унимодальных функций [14, 15, 17]. Однако можно воспользоваться свойством кусочной линейности $D(\alpha)$ и на этом основании предложить более быстрый алгоритм.

Методика решения задачи и описание алгоритма нахождения $\alpha_{opt}$ приведены ниже. Использованы величины типа $T = (T.x, T.y, T.z)$, где $T.x$, $T.y$ — абсцисса и ордината точки, $T.z$ — значение производной функции $D(\alpha)$ в этой точке.

*Шаг* 1. Зададим достаточно малое число $\varepsilon$ — точность определения $\alpha_{opt}$, а также границы интервала $A.x = 0$; $B.x = 1$.

*Шаг* 2. Найдем значение функции $A.y = D(A.x)$ и $A.z = [D(A.x + \varepsilon) - A.y]/\varepsilon$ на левом конце интервала поиска. Определим $B.y = D(B.x)$ и $B.z = [B.y - D(B.x - \varepsilon)]/\varepsilon$ — аналогичные им величины на правом конце интервала поиска.

*Шаг* 3. По этим точкам и производным построим прямые, первая из которых проходит через точку $(A.x, A.y)$, вторая через $(B.x, B.y)$. Нетрудно показать, что абсциссу точки пересечения этих прямых можно найти по формуле

$$x = \frac{B.y - A.y - B.zB.x + A.zA.x}{A.z - B.z}. \tag{12}$$

Вычислим абсциссу $C.x = x + \varepsilon/3$, сдвинутую вправо от $x$ на величину меньшую $\varepsilon$. Это необходимо для того чтоб точка $x$ попала в диапазон $[A.x, C.x]$.

*Шаг* 4. Если $|C.x - A.x| < \varepsilon$ или $|C.x - B.x| < \varepsilon$, то точка минимума $\alpha_{opt} = x$ найдена и алгоритм заканчивает работу, иначе находим аналогичные значения $C.y$ и $C.z$. Если $C.z$ и $A.z$ числа одного знака, то полагаем $B = C$, иначе $A = C$ и переходим к шагу 2 для выполнения новой итерации.

Продемонстрируем работу алгоритма на примере оптимизации функции с двумя изломами.

Для этого на шаге 1 зададим точность поиска $\varepsilon = 10^{-3}$. На шаге 2 получим $A.x = 0$; $A.y = 1{,}2$; $A.z = -1{,}7$; $B.x = 0{,}999$; $B.y = 0{,}6$; $B.z = 1{,}7$. Разные знаки производных $A.z$ и $B.z$ указывают на то, что функция $D(\alpha)$ строго унимодальна [18] и, следовательно, ее минимум находится внутри отрезка.

На шаге 3 по формуле (12) найдем $x = 0{,}644$; $C.x = 0{,}645$. Поскольку на шаге 4 ни одно из его условий не выполнилось, то вычислим $C.y$ и $C.z = 0{,}8 > 0$. Это означает, что справа от точки $x$ функция возрастает, следовательно, минимум находится слева от $x$. Положим $B = C$ и поиск минимума функции продолжим на отрезке $[0;\ 0{,}645]$ меньшей длины.

На новой итерации получим $x = 0{,}425$; $C.x = 0{,}426$; $C.z = 0{,}8 > 0$ и новый отрезок $[0;\ 426]$.

На следующей итерации найдем $x = 0{,}425$; $C.x = 0{,}426$. Теперь условие $|C.x - B.x| < \varepsilon$ выполнилось. Это значит, что минимум функции находится в точке $\alpha_{opt} = x = 0{,}425$.

Очевидно, число итераций не превышает $k + 1$, где $k$ — число изломов функции $D(\alpha)$. В частности, данная задача решена за три итерации при двух изломах минимизируемой функции.

Приведем таблицу значений функции $y(x) = \sin x$, использованной в работе [19] для оценки погрешности $B$-сплайна и сплайна Акимы. Сплайны построены на множестве из 14 случайных точек. Вычисления проведены для равноотстоящих узлов. Для сравнения в таблицу добавлены данные по направленному сплайну. В таблице в колонках $\Delta_1$, $\Delta_2$, $\Delta_3$ приведены разности между точным значением функции $y(x)$ и значениями, которые получены с помощью $B$-сплайна ($\Delta_1$), сплайна Акимы ($\Delta_2$) и направленного сплайна ($\Delta_3$), соответственно.

**Таблица**

Сравнительные данные по вычислительной погрешности сплайнов

| $x$ | $y(x)$ | $\Delta_1$ | $\Delta_2$ | $\Delta_3$ |
|---|---|---|---|---|
| 0,00 | 0,0000000 | 0,000E+00 | 0,000E+00 | 0,000E+00 |
| 0,05 | 0,0499792 | 7,819E-08 | –6,100E-05 | 3,529E-05 |
| 0,10 | 0,0998334 | 3,189E-08 | –1,010E-05 | –5,012E-06 |
| 0,15 | 0,1494381 | 1,433E-08 | 7,158E-06 | 9,359E-06 |
| 0,20 | 0,1986693 | 8,864E-09 | 2,348E-05 | –5,406E-06 |
| 0,25 | 0,2474040 | 2,986E-08 | –1,177E-05 | 3,179E-06 |
| 0,30 | 0,2955202 | 8,974E-11 | –1,624E-06 | 6,155E-07 |
| 0,35 | 0,3428978 | 2,180E-08 | 6,243E-06 | –3,199E-06 |
| 0,40 | 0,3894183 | 2,905E-08 | –8,113E-06 | 4,885E-06 |
| 0,45 | 0,4349655 | –2,711E-20 | –2,711E-20 | –5,421E-20 |
| 0,50 | 0,4794255 | 4,830E-10 | 5,141E-06 | –4,438E-06 |
| 0,55 | 0,5226872 | 8,824E-08 | –2,120E-06 | 2,857E-06 |
| 0,60 | 0,5646425 | –6,664E-08 | –6,810E-06 | 2,393E-06 |
| 0,65 | 0,6051864 | –1,044E-07 | 4,300E-06 | –1,165E-06 |
| 0,70 | 0,6442177 | 8,651E-07 | 3,549E-06 | –2,821E-06 |
| 0,75 | 0,6816388 | –2,716E-06 | –1,707E-06 | 1,296E-06 |

Более высокую точность показал $B$-сплайн. Среди двух последних лучшие результаты дал направленный сплайн. Это неочевидный результат, поскольку ожидалось, что якобы имеющий преимущества при интерполяции монотонных функций, а именно таковой в данном примере является функция $y(x)$, сплайн Акимы должен был бы демонстрировать лучшие показатели не только в сравнении с направленным сплайном, но и с $B$-сплайном. Однако в данном случае эти ожидания не оправдались.

Изложенная идея позволяет расширить рамки подхода к построению направленного нового сплайна, который будет лишен изломов, и допускает возможность его оптимизации с целью подавления выбросов. Примером может служить направленный сплайн четвертой степени, методика построения которого изложена ниже.

## 3. Построение направленного сплайна четвертой степени (НС4-сплайна)

На каждом отрезке $[x_{i-1}, x_i]$ будем представлять сплайн $S(x)$ полиномом четвертой степени

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_{i-1})^3 + e_i(x - x_i)^4, i = 1,2, \dots, n. \quad (13)$$

На стыках соседних отрезков $[x_{i-1}, x_i]$ для полиномов (3) потребуем выполнения условий непрерывности сплайна и ее первой и второй производных в точках (1)

$$S_i(x_{i-1}) = S_{i-1}(x_{i-1}), \quad (14)$$

$$S_i'(x_{i-1}) = S_{i-1}'(x_{i-1}), \quad (15)$$

$$S_i''(x_{i-1}) = S_{i-1}''(x_{i-1}). \quad (16)$$

Полагая по-прежнему $b_i$ известными и, используя (14), получим

$$c_i = d_{i-1}h_i - e_i h_i^2 + w_i, \quad (17)$$

$$e_i = \frac{w_i - c_i + d_i h_i}{h_i^2}, \quad (18)$$

где

$$w_i = \frac{b_i - u_i}{h_i}.$$

Используя (15), найдем

$$2c_i + 4e_i h_i^2 = v_i - \frac{3h_{i-1}^2}{h_i} d_{i-1}, \quad (19)$$

где

$$v_i = \frac{b_i - b_{i-1}}{h_i}.$$

Подставив (18) в (19), запишем

$$c_i = 2d_i h_i + \frac{3h_{i-1}^2}{2h_i} d_{i-1} + 2w_i - \frac{v_i}{2}, \quad (20)$$

$$e_i h_i^2 = \frac{v_i}{2} - d_i h_i - \frac{3h_{i-1}^2}{2h_i} d_{i-1} - w_i. \quad (21)$$

Условие (16) дает зависимость

$$c_i + 6e_i h_i^2 = c_{i-1} + 3d_{i-1}h_i, \quad (22)$$

Подставив (20), (21) в (22), найдем

$$8d_i h_i + d_{i-1}\left(15\frac{h_{i-1}^2}{h_i} + 10h_{i-1}\right) + \frac{3h_{i-2}^2}{h_{i-1}} d_{i-2} = 5v_i + v_{i-1} - 4(2w_i + w_{i-1}). \quad (23)$$

Сдвинув в (23) индекс, получим рекуррентные формулы

$$\mu_i d_{i+1} + \eta_i d_i + \lambda_i d_{i-1} = \omega_i, i = 1,2, \dots, n-1, \quad (24)$$

где

$$\mu_i = 8h_{i+1}, \eta_i = 5h_i\left(3\frac{h_i^2}{h_{i+1}} + 2\right), \lambda_i = \frac{3h_{i-1}^2}{h_i}, \omega_i = 5v_{i+1} + v_i - 4(2w_{i+1} + w_i).$$

Формулы (24) представляют трехдиагональную систему линейных алгебраических уравнений относительно неизвестных коэффициентов $d_i$, которая с учетом очевидных краевых условий $d_0 = 0$, $d_n = 0$ может быть решена методом прогонки [12]. Далее коэффициенты $c_i$, $e_i$ можно найти по формулам (17), (18).

## 4. Оптимизация НС4-сплайна

При оптимизации НС3-сплайна использована однопараметрическая процедура. Это объясняется тем, что многопараметрическая оптимизация в пределе дает *B*-сплайн Шёнберга, что влечет утрату достоинств НС3. Сплайн НС4 лишен изломов, поэтому его оп-

тимизация сводится к только максимальному подавлению выбросов. В этом процессе могут быть задействованы все направляющие коэффициенты сплайна $\alpha_i$, при помощи которых вычисляются коэффициенты НС4-сплайна

$$b_i = \alpha_i u_i + (1-\alpha_i)u_{i+1}, i = 1,2,\dots,n-1. \quad (25)$$

В качестве критериев оптимальности данного сплайна использовали

– длину $L$ сплайна,
– разность $R$ между его глобальным максимумом и глобальным минимумом.

Критерий $L$ определяется суммой длины сегментов сплайна и может быть вычислен при помощи известной формулы [23]

$$L = \sum_{i=1}^{n} \int_{x_{i-1}}^{x_i} \sqrt{1 + \left[s_i'(x)\right]^2}\, dx,$$

где

$$S_i'(x) = b_i + 2c_i(x - x_i) + 3d_i(x - x_{i-1})^2 + 4e_i(x - x_i)^3. \quad (26)$$

Для вычисления критерия $R$ использована формула (26), а также формула

$$S_i''(x) = 2c_i + 6d_i(x - x_i) + 12e_i(x - x_i)^2. \quad (27)$$

Используя (26), находили корни уравнения

$$S_i'(x) = 0,$$

их тип контролировали с помощью (27).

Таким образом, как критерий $L$, так и критерий $R$ являются функциями многих переменных

$$K = K(\alpha), \quad (28)$$

где

$$\alpha = (\alpha_1, \alpha_2, \dots, \alpha_{n-1}).$$

Очевидно, $B$-сплайн Шёнберга является частным случаем НС4-сплайна. Следовательно, с позиций минимума указанных критериев оптимальный сплайн не может быть хуже сплайна Шёнберга.

В процессе минимизации критериев придерживались требования поддержания такой формы сплайнов, изогеомертия которых соответствовала бы сплайну Шёнберга [25].

Расчеты НС4-сплайна показали, что без принятия мер сплайн может терять указанную форму.

Среди причин потери формы выделены следующие:

– сплайн может иметь «выпячивания» отдельных фрагментов кривой,
– на кривой могут появляться новые локальные экстремумы,
– могут также появляться новые точки смены кривизны сплайна.

Данные недостатки связаны с появлением новых локальных экстремумов, как самого сплайна, так и его первой и второй производных, а также новых смен знака его третей производной. При оптимизации варианты таких сплайнов отбраковывались.

Численные эксперименты позволили установить тот факт, что функция (28) является многоэкстремальной, то есть имеет множество локальных минимумов, среди которых следует найти глобальный минимум, который соответствует сплайну оптимальной формы.

Так, при $n = 12$, для которого проводилось большинство экспериментов, пришлось бы отыскивать глобальный минимум функции 11 переменных, что представляется задачей чрезвычайной сложности. В общем случае трудности получения решения такой задачи ставят под сомнение ценность практического использования обсуждаемого сплайна. Выход из положения был найден в использовании упомянутых свойств сплайна, которые продиктованы жесткими условиями сохранения его формы.

Методика минимизации критериев состоит в следующем. Изначально устанавливается стартовое состояние, для которого выбирается вектор $\alpha$, все компоненты которого полагаются равными 0,5, и вычисляется стартовый НС-сплайн. Далее назначается шаг $\tau$ малой длины и проводится переборная однопараметрическая минимизация по каждой компоненте вектора $\alpha$, для которой процесс начинается со стартового состояния. Полученные таким образом лучшие покоординатные сплайны дают множество $\alpha$-векторов, число которых $m < n$. Наблюдения показали, что через фильтр жестких требований сохранения формы сплайна обычно проходит не более половины начальных стартов. Так, например, для $n = 12$ обычно $m < 7$. Следующим шагом является аналогичная однопараметрическая оптимизация, где в качестве стартов последовательно используются векторы, прошедшие фильтр первого шага. При этом количество новых векторов, которые прошли фильтр сохранения формы также невелико и оно обычно меньше аналогичного количества предыдущего шага. Рекурсия ведется до тех пор, пока отфильтрованные векторы не дадут новые векторы для продолжения процесса. Результатом оптимизации будет сплайн с наименьшим значением критерия $K$. Расчеты показали, что, например, для $n = 12$ обычно требуется сформировать сплайн и вычислить его характеристики 1000–2000 раз.

На рис. 5 показаны кривые сплайнов, которые дают типичную картину формы оптимального по критерию длины НС4-сплайна. Сплайн обычно занимает среднее положение между сплайном Шёнберга и НС3 ближе к первому.

Приведем характерный пример процесса оптимизации НС4. Длина сплайна Шёнберга 19,70. Исходный НС4 имеет несколько бо́льшую длину — 19,90. Оптимизированный сплайн имеет длину 16,82, что на 14,6% меньше длины сплайна Шёнберга. При проведении оптимизации алгоритм улучшал результат 181 раз. Для решения задачи потребовалось провести вычисление сплайна 1312 раз. Наименьшую длину имел НС3. При длине 11,58 он почти в два раза короче сплайна Шёнберга. Тысячи вычислительных экспериментов, поставленных для $n = 12$, показали, что оптимизированный НС4 короче сплайна Шёнберга на 5–50%, а НС3 — в 1,5–3 раза.

Примеры процессов оптимизации можно наблюдать в динамике на видео по ссылкам [30–32].

Исследования показывают, НС4 позволяет ослабить проявление выбросов сплайна Шёнберга. Однако лучшим в этом отношении следует признать СП3, на котором выбросов нет, а изломы проявляются весьма слабо.

В процессе изучения свойств предложенных сплайнов обнаружено, что кроме выбросов и изломов сплайны могут иметь острые пики локальных максимумов и минимумов, которые в ряде случаев также следует рассматривать как недостатки интерполяции. Ниже предложен и описан способ, который позволять притупить пики экстремумов.

## 5. Методика притупления пиков сплайнов

Пусть необходимо притупить локальный минимум какого-либо из рассмотренных сплайнов и пусть $(x_c, y_c)$ точка такого минимума. Рассмотрим также точки перегиба $(x_a, y_a)$ и $(x_b, y_b)$, расположенные по обе стороны от экстремума.

Для сглаживаемого экстремума сплайна введем добавочную функцию

$$g(x) = G(x - x_a)^k (x_b - x)^m, G \geq 0, k > 2, m \geq 2. \tag{29}$$

Для нее имеют место очевидные краевые условия

$$g'(x_a) = g'(x_b) = 0, g''(x_a) = g''(x_b) = 0.$$

Следовательно, на краях интервала функция не привносит изломов и способствует притуплению сплайна в области его минимума.

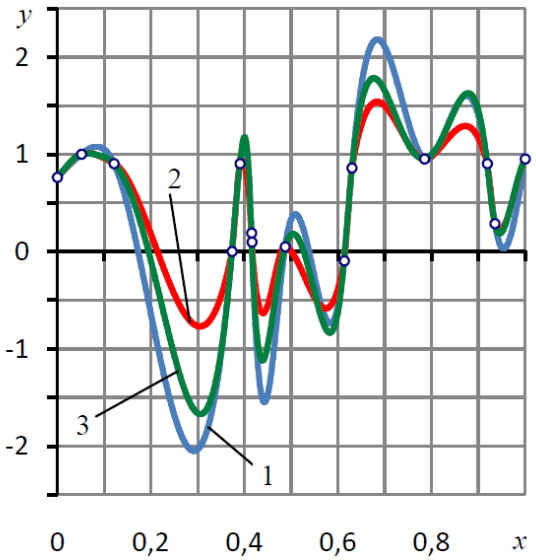



**Рис 5.** *B*-сплайн (1), НС3-сплайн (2), НС4-сплайн (3)

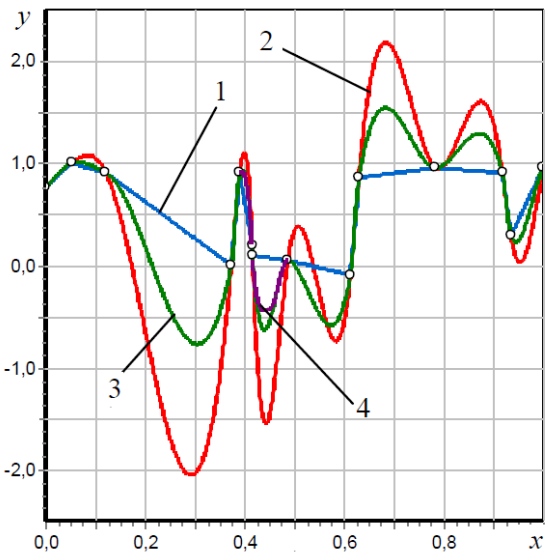



**Рис 6.** Линейный интерполяционный сплайн (1), *B*-сплайн (2), НС3 (3) и НС3 с притупленными экстремумами (4)

Максимум этой функции имеет место при

$$g'(x_c) = G(x_c - x_a)^{k-1}(x_b - x_c)^{m-1}[k(x_b - x_c) - m(x_c - x_a)] = 0, \tag{30}$$

откуда следует

$$m = tk, \quad t = \frac{(x_b - x_c)}{(x_c - x_a)}.$$

Если $t < 1$, то примем $m = 2 + \varepsilon$, где $\varepsilon$ — малое число. Тогда $k = m/t > 2$. Иначе при $t \geq 1$ примем $k = 2 + \varepsilon$. Тогда $m = kt > 2$.

Обратимся к линейному интерполяционному сплайну

$$L(x) = L_a x + L_b, L(x_a) = y_a, L(x_b) = y_b,$$

где

$$L_a = \frac{y_b - y_a}{x_b - x_a}, L_b = \frac{x_b y_a - x_a y_b}{x_b - x_a}.$$

Для точки рассматриваемого минимума сплайна имеем

$$G(x_c - x_a)^k (x_b - x_c)^m = \sigma[L(x_c) - y_c], \tag{31}$$

где $\sigma$ — коэффициент притупления пика.

При $\sigma = 0$ добавка $g(x)$ отсутствует, при $\sigma = 1$ суммарный сплайн $q(x) = s(x) + g(x)$ коснется отрезка прямой $L(x)$, что является предельным случаем, поскольку при $\sigma > 1$ суммарный сплайн получит новый экстремум-максимум, по обе стороны которого будет образовано два новых минимума. Такие режимы не обеспечивают сохранения формы сплайна, потому $0 \leq \sigma \leq 1$.

Для сохранения формы суммарный сплайн кроме того должен иметь кривизну одного знака до $\sigma = \sigma_{max} < 1$. Данная величина может быть определена из условия

$$q''(x_c) = 0.$$

Выполнив дифференцирование функции $q(x)$, найдем

$$s''(x_c) + G(k + m)G(x_c - x_a)^{k-1}(x_b - x_c)^{m-1} = 0.$$

Отсюда

$$G = \frac{s''(x_c)}{(k + m)(x_c - x_a)^{k-1}(x_b - x_c)^{m-1}}. \tag{32}$$

Из (31), (32) вытекает, что

$$\frac{\sigma_{max}[L(x_c) - y_c]}{(x_c - x_a)^{k-1}(x_b - x_c)^{m-1}} = \frac{s''(x_c)}{(k+m)(x_c - x_a)^{k-1}(x_b - x_c)^{m-1}}. \quad (33)$$

Отношение (33) позволяет определить предельно допустимую величину $\sigma$

$$\sigma_{max} = \frac{s''(x_c)(x_c - x_a)(x_b - x_c)}{(k+m)[L(x_c) - y_c]}.$$

Аналогично проводится притупление локальных максимумов.

Решение о подлежащих притуплению экстремумах, может быть принято, как в автоматизированном, так и автоматическом режимах. При автоматическом принятии решения в качестве критерия оценки экстремума можно использовать величину $k_e = s''(x_c)$.

Приведем описание методики оценивания экстремумов, подлежащих автоматическому притуплению. В качестве ориентира будем использовать НСЗ, график которого дан на рис. 4. При визуальной оценке сплайна можно выделить два экстремума, которые необходимо притупить. Один из них является максимумом с абсциссой $x = 0{,}395$, второй соседним минимумом с абсциссой $x = 0{,}440$. Для первого $k_e = -2920$, для второго $k_e = 1871$. Следующий за ними по убыванию абсолютной величины критерия $k_e$ является крайний правый минимум с абсциссой $x = 0{,}946$, для которого $k_e = 1123$. Оценим его как экстремум, не требующий притупления. Данный простой анализ позволяет в первом приближении сформулировать следующую методику автоматической экспертной оценки: экстремумы, которые должны быть притуплены автоматически, должны удовлетворять условию $|k_e| > 1200$, если это не противоречит приведенным выше условиям сохранения формы сплайна.

На рис. 6 в качестве примера притупления экстремумов показаны линейный интерполяционный сплайн, *B*-сплайн, НСЗ-сплайн и два фрагмента НСЗ-сплайна, на котором экстремумы притуплены автоматически в соответствии с описанной выше методикой. При вычислениях принята величина коэффициента притупления $\sigma = 0{,}75\sigma_{max}$. Методика притупления острых пиков кривых может быть применена к сплайнам любого типа.

## Заключение

В статье сформулирован и предложен простой метод построения кубического сплайна для набора точек на плоскости. Проведено сравнение сплайна с *B*-сплайном Шёнберга и сплайнами Акимы и Катммула—Рома. Показано, что для неравноотстоящих точек, при которых обычно проявляются недостатки названных сплайнов, в сравнении с *B*-сплайном он дает значительно меньшие осцилляции. Сплайн с таким набором точек практически лишен сильных изломов, которые свойственны сплайнам Акимы. Он не дает петель и осцилляций, которые являются характерным недостатком параметрических сплайнов, в частности, эрмитовых, к числу которых относится сплайн Катмулла—Рома. Предложен метод оптимизации направляющего коэффициента сплайна, цель которой состоит в минимизации разрывов второй производной функции в ее промежуточных точках. Также предложен направленный сплайн четвертого порядка, который лишен изломов и в сравнении со сплайном Шёнберга имеет меньшие выбросы. Предложен метод притупления острых пиков кривых, который можно применять ко всем типам сплайнов. Приведенные в статье численные и графические результаты получены на основе предложенных алгоритмов, реализованных при помощи разработанного автором программного обеспечения в среде визуального программирования Delphi с использованием приложения MS Office Excel.

В ходе дальнейшей работы планируется улучшить реализацию алгоритма оптимизации направленного сплайна четвертого порядка с целью сокращения времени его работы и повышения точности процедуры минимизации критериев оптимальности сплайна.

## Литература


1. Powell M.J.D. Approximation Theory and Methods. Cambridge University Press, 1981. 352 p. DOI: 10.1017/CBO9781139171502.
2. Atkinson K.A. An Introduction to Numerical Analysis (2nd ed.). John Wiley and Sons, 1988. 615 p.
3. Watson G.A. Approximation Theory and Numerical Methods. John Wiley, 1980. 229 p.
4. Schatzman M. Numerical Analysis: A Mathematical Introduction. Clarendon Press, 2002. 496 p.
5. Schoenberg I.J. Contributions to the problem of approximation of equidistant data by analytic functions // Quart. Appl. Math. 1946. Vol. 4. P. 45–99, 112–141.
6. Ahlberg J.H., Nilson E.N., Walsh J.L. The Theory of Splines and Their Application. Academic Press, 1967. 296 p.
7. David F., Rogers J., Adams A. Mathematical Elements for Computer Graphics. McGraw-Hill Science / Engineering / Math, 2 edition, 1989. 611 p.
8. Cohen D. Incremental Methods for Computer Graphics. PhD Thesis. Harvard University, 1969.
9. Warnock J.E. A Hidden Surface Algorithm for Computer-Generated Halftone Pictures. Computer Science Department, University of Utah, TR 1–15. 1969.
10. Watkins G.S. A Real-Time Visible Surface Algorithm. Computer Science Department, University of Utah, UTECH-CSC-70-101, 1970.
11. Akima H. A New Method of Interpolation and Smooth Curve Fitting Based on Local Procedures // Journal of the ACM. 1970. Vol. 17, no. 4. P. 589–602. DOI: 10.1145/321607.321609.
12. Catmull E., Rom R. A class of local interpolating splines // Computer Aided Geometric Design. 1974. P. 317–326.
13. Barry P.J., Goldman R.N. Recursive evaluation algorithm for a class of Catmull-Rom splines // Computer Graphics. 1988. Vol. 22, no. 4. P. 199–204. DOI: 10.1145/378456.378511.
14. Kiefer J.K. Sequential minimax search for a maximum // P. Am. Math. Soc. 1953. P. 502–506.
15. Brent R.P. Algorithms for Minimization without Derivatives. Dover. 2002. 195 p.
16. Круковец А.С., Горелкин Г.А. Разработка метода интерполяции значений номограммы // Современные научные исследования и инновации. 2015. № 5(2). URL: http://web.snauka.ru/issues/2015/05/53846 (дата обращения: 14.06.2020).
17. Dobson A.J. An Introduction to Statistical Modelling. Chapman and Hall, London, 1983.
18. Коднянко В.А. О вычислительной избыточности метода дихотомии и условной минимизации унимодальных функций методом экономной дихотомии // Системы и средства информатики. 2019. Т. 29, № 1. С. 164–173. DOI: 10.14357/08696527190113.
19. Ruckdeschel F.R. Basic scientific subroutines. Vol. 2. BYTE/McGRAW-HILL, 1981.
20. Яненко Н.Н., Квасов Б.И. Итерационный метод построения поликубических сплайн-функций // Докл. АН СССР. 1970. Т. 195, № 5. С. 1055–1057.
21. Constantini P., Morandi R. An algorithm for computing shape-preserving cubic spline interpolation to data // Calcolo. 1984. Vol. 21. P. 295–305.
22. Рябенький В.С. Локальные формулы гладкого восполнения и гладкой интерполяции функций по их значениям в узлах неравномерной прямоугольной сетки // Препринт. Ин-т прикл. математики АН СССР. 1974. № 21. 35 с.

23. Dietze S., Schmidt J.W. Determination of shape preserving spline interpolants with minimal curvature via dual programs // J. Approxim. Theory. 1988. Vol. 52, no. 1. P. 43–57.
24. Завьялов Ю.С., Квасов Б.И., Мирошниченко В.Л. Методы сплайн-функций. М.: Наука, 1980. 352 с.
25. Мирошниченко В.Л. Изогеометрические свойства и погрешность аппроксимации взвешенных кубических сплайнов // Вычислительные системы. Новосибирск: ИМ СО РАН, Вып. 154: Сплайны и их приложения. 1995. С. 127–154.
26. Корнейчук Н.П., Бабенко В.Ф., Лигун А.А. Экстремальные свойства полиномов и сплайнов. Киев: Наукова думка. 1992. 304 с.
27. Dzyubenko G.A., Gilewicz J., Shevchuk I.A. New phenomena in coconvex approximation // Analysis Mathematica. 2006. Vol. 32, no. 2. P. 113–121. DOI: 10.1007/s10476-006-0005-x.
28. Стечкин С.Б., Субботин Ю.Н. Сплайны в вычислительной математике. М.: Наука, 1976. 248 с.
29. Волков Ю.С. Новый способ построения интерполяционных кубических сплайнов // Журн. вычисл. матем. и матем. физ. 2004. Т. 44, № 2. С. 231–241.
30. Коднянко В.А. Video DS4-spline optimization 1. URL: http://smiuk.sfu-kras.ru/kodnyanko/site/science/Video1.mp4 (дата обращения: 12.07.2020).
31. Коднянко В.А. Video DS4-spline optimization 2. URL: http://smiuk.sfu-kras.ru/kodnyanko/site/science/Video2.mp4 (дата обращения: 12.07.2020).
32. Коднянко В.А. Video DS4-spline optimization 3. URL: http://smiuk.sfu-kras.ru/kodnyanko/site/science/Video3.mp4 (дата обращения: 12.07.2020).

Коднянко Владимир Александрович, д.т.н, профессор, кафедра стандартизации, метрологии и управления качеством, Сибирский федеральный университет (Красноярск, Российская Федерация)

---

**DOI: 10.14529/cmse210101**

# DIRECTIONAL SPLINES AND THEIR USE FOR SMOOTHING EJECTIONS AND FRACTURES OF INTERPOLANT

**© 2021 V.A. Kodnyanko**

*Siberian Federal University (Kirensky 26A, Krasnoyarsk, 660074 Russia)*

*E-mail: kowlad@rambler.ru*

Received: 18.07.2020

A method for constructing a directional cubic spline for a set of points on a plane is formulated and proposed. The spline is compared with the Schoenberg B-spline, Akima and Catmull–Rom splines. It is shown that for unequally spaced points, in comparison with the B-spline, it gives significantly lower overshoots and is practically free of strong kinks, which are characteristic of Akima splines. The spline does not give loops and oscillations, which are a characteristic drawback of parametric splines, in particular, Hermitian ones, which include the Catmull–Rom spline. A fast method for optimizing the spline guiding coefficient is proposed, the purpose of which is to minimize the discontinuities of the second derivative of the function at its intermediate points. An example of optimization of a directional third-order spline is given. A fourth-order directional spline, which is free of kinks, is also proposed. The method of optimization of the directional spline of the fourth order is formulated, the algorithm of its optimization is stated. The optimization criteria are the spline length and the smallest distance between its global maximum and minimum. It is shown that, in comparison with the Schoenberg spline, the fourth-order directional spline has smaller outliers. A method for automatic blunting of sharp peaks of curves is proposed, which can be applied to all types of splines.

*Keywords: spline, Schoenberg spline, Akima spline, directional spline.*

**FOR CITATION**

Kodnyanko V.A. Directional Splines and Their Use for Smoothing Ejections and Fractures of Interpolant. *Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering.* 2021. Vol. 10, no. 1. P. 5–19. (in Russian) DOI: 10.14529/cmse210101.



## References

1. Powell M.J.D. Approximation Theory and Methods. Cambridge University Press, 1981. 352 p. DOI: 10.1017/CBO9781139171502.
2. Atkinson K.A. An Introduction to Numerical Analysis (2nd ed.). John Wiley and Sons, 1988. 615 p.
3. Watson G.A. Approximation Theory and Numerical Methods. John Wiley, 1980. 229 p.
4. Schatzman M. Numerical Analysis: A Mathematical Introduction. Clarendon Press, 2002. 496 p.
5. Schoenberg I.J. Contributions to the problem of approximation of equidistant data by analytic functions. Quart. Appl. Math. 1946. Vol. 4. P. 45–99, 112–141.
6. Ahlberg J.H., Nilson E.N., Walsh J.L. The Theory of Splines and Their Application. Academic Press, 1967. 296 p.
7. David F., Rogers J., Adams A. Mathematical Elements for Computer Graphics. McGraw-Hill Science / Engineering / Math, 2 edition, 1989. 611 p.
8. Cohen D. Incremental Methods for Computer Graphics. PhD Thesis. Harvard University, 1969.
9. Warnock J.E. A Hidden Surface Algorithm for Computer-Generated Halftone Pictures. Computer Science Department, University of Utah, TR 1–15. 1969.
10. Watkins G.S. A Real-Time Visible Surface Algorithm. Computer Science Department, University of Utah, UTECH-CSC-70-101, 1970.
11. Akima H. A New Method of Interpolation and Smooth Curve Fitting Based on Local Procedures. Journal of the ACM. 1970. Vol. 17, no. 4. P. 589–602. DOI: 10.1145/321607.321609.
12. Catmull E., Rom R. A class of local interpolating splines. Computer Aided Geometric Design. 1974. P. 317–326.
13. Barry P.J., Goldman R.N. Recursive evaluation algorithm for a class of Catmull-Rom splines. Computer Graphics. 1988. Vol. 22, no. 4. P. 199–204. DOI: 10.1145/378456.378511.
14. Kiefer J.K. Sequential minimax search for a maximum. P. Am. Math. Soc. 1953. P. 502–506.
15. Brent R.P. Algorithms for Minimization without Derivatives. Dover, 2002. 195 p.
16. Krukovets A.S., Gorelkin G.A. Development of a method for interpolating the values of the nomogram. Modern scientific research and innovations. 2015. No. 5(2). URL: http://web.snauka.ru/issues/2015/05/53846 (accessed: 14.06.2020) (in Russian)
17. Dobson A.J. An Introduction to Statistical Modelling. Chapman and Hall, London, 1983.
18. Kodnyanko V.A. On computational redundancy of the dichotomous search and conditional minimization of unimodal functions by the economical dichotomous search. Systems and

means of informatics. 2019. Vol. 29, no. 1. P. 164–173. DOI: 10.14357/08696527190113 (in Russian)
19. Ruckdeschel F.R. Basic scientific subroutines. Vol. 2. BYTE/McGRAW-HILL, 1981.
20. Yanenko N.N., Kvasov B.I. An Iterative Method for Constructing Polycubic Spline Functions. Dokl. USSR Academy of Sciences. 1970. Vol. 195, no. 5. P. 1055–1057. (in Russian)
21. Constantini P., Morandi R. An algorithm for computing shape-preserving cubic spline interpolation to data. Calcolo. 1984. Vol. 21. P. 295–305.
22. Ryabenky V.S. Local formulas for smooth completion and smooth interpolation of functions by their values at nodes of an uneven rectangular grid. Preprint, Inst. mathematics of the Academy of Sciences of the USSR. IPM. 1974. No. 21. (in Russian)
23. Dietze S., Schmidt J.W. Determination of shape preserving spline interpolants with minimal curvature via dual programs. J. Approxim. Theory. 1988. Vol. 52, no. 1. P. 43–57.
24. Zavyalov Yu.S., Kvasov B.I., Miroshnichenko V.L. Methods of spline functions. Moscow, Nauka, 1980. (in Russian)
25. Miroshnichenko V.L. Isogeometric properties and approximation error for weighted cubic splines. Computing Systems. Novosibirsk: IM SB RAS, 1995. Vol. 154. P. 127–154. (in Russian)
26. Korneychuk N.P., Babenko V.F., Ligun A.A. Extreme properties of polynomials and splines. Kiev, Naukova Dumka, 1992. 304 p. (in Russian)
27. Dzyubenko G.A., Gilewicz J., Shevchuk I. A. New phenomena in coconvex approximation. Analysis Mathematica. 2006. Vol. 32, no. 2. P. 113–121. DOI: 10.1007/s10476-006-0005-x.
28. Stechkin S.B., Subbotin Yu.N. Splines in computational mathematics. Moscow, Nauka, 1976. 248 p. (in Russian)
29. Volkov Yu.S. A new method for constructing interpolation cubic splines. Computational Mathematics and Mathematical Physics. 2004. Vol. 44, no. 2. P. 231–241. (in Russian)
30. Kodnyanko V.A. Video DS4-spline optimization 1. URL: http://smiuk.sfu-kras.ru/kodnyanko/site/science/Video1.mp4 (accessed: 12.07.2020).
31. Kodnyanko V.A. Video DS4-spline optimization 2. URL: http://smiuk.sfu-kras.ru/kodnyanko/site/science/Video2.mp4 (accessed: 12.07.2020).
32. Kodnyanko V.A. Video DS4-spline optimization 3. URL: http://smiuk.sfu-kras.ru/kodnyanko/site/science/Video3.mp4 (accessed: 12.07.2020).

**УДК 004.4'414** **DOI: 10.14529/cmse210102**

# СТАТИЧЕСКИ-ДЕТЕРМИНИРОВАННЫЙ МЕТОД ПРОГНОЗИРОВАНИЯ ДИНАМИЧЕСКИХ ХАРАКТЕРИСТИК ПАРАЛЛЕЛЬНЫХ ПРОГРАММ*

**© 2021 А.А. Клейменов, Н.Н. Попова**
*Московский государственный университет имени М.В. Ломоносова*
*(119991 Москва, ул. Ленинские Горы, д. 1)*
*E-mail: andreykleimenov@mail.ru, popova@cs.msu.ru*
Поступила в редакцию: 03.07.2020

В статье рассматривается задача прогнозирования характеристик параллельных приложений. Изучаются динамические характеристики, описывающие выполнение параллельных приложений — время выполнения, количество операций с плавающей точкой, потребляемая электроэнергия, количество обращений в память и другие. Прогнозирование динамических характеристик позволяет решать многие проблемы, связанные с проектированием новых архитектур, выбором наиболее подходящих конфигураций многопроцессорных систем для решения конкретных задач, портированием приложений на новые системы, планированием потоков задач и многие другие. Задача прогнозирования характеристик активно исследуется. Возрастающая сложность архитектур современных высокопроизводительных систем требует разработки новых методов решения задачи прогнозирования. В статье дается обзор существующих подходов и программных средств для прогнозирования динамических характеристик и предлагается подход, основанный на статическом анализе исходного кода параллельного приложения. На основе текста параллельной программы, формального описания целевой вычислительной платформы и параметров запуска реализован метод, позволяющий прогнозировать время работы, количество выполненных операций вещественной арифметики, обращения к памяти и другие характеристики параллельного приложения. Применимость предложенного подхода продемонстрирована на примере решения тестовой трехмерной задачи численного моделирования на многопроцессорном кластере на базе процессоров IBM Power8.

*Ключевые слова: параллельные приложения, динамические характеристики, анализ производительности, системы экзафлопсной производительности, модель компьютера, статический анализ.*

**ОБРАЗЕЦ ЦИТИРОВАНИЯ**

Клейменов А.А., Попова Н.Н. Статически-детерминированный метод прогнозирования динамических характеристик параллельных программ // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2021. Т. 10, № 1. С. 20–31. DOI: 10.14529/cmse210102.

## Введение

Одним из ключевых фактором, определяющим рост производительности вычислительных систем, является параллелизм обработки данных, поддерживаемый на всех уровнях организации вычислений. Именно благодаря параллелизму стало возможным преодоление барьеров, определяемых технологическими факторами. Современные суперкомпьютеры, производительность которых неуклонно приближается к экзафлопсам, включают в свой состав до миллиона ядер, обеспечивая тем самым высокую степень параллелизма.

Учитывая сложность суперкомпьютерных архитектур, трудно прогнозировать, насколько быстро приложение, предназначенное для современных суперкомпьютеров петафлопсной производительности, будет выполняться на суперкомпьютерах экзафлопсной производительности, каким будет потребление электроэнергии при этом и какие значения

*Статья рекомендована к публикации программным комитетом Международной конференции «Суперкомпьютерные дни в России – 2020».

будут принимать другие параметры, характеризующие производительность параллельного приложения. Количественные характеристики, описывающие выполнение параллельных приложений, такие как, время выполнения, количество операций с плавающей точкой, потребляемая электроэнергия, объем использованной оперативной памяти и другие, будем называть *динамическими характеристиками*.

Точная экстраполяция характеристик параллельных программ, основанная на интуитивных представлениях и догадках, практически невозможна. Прогнозирование динамических характеристик параллельных программ является актуальной задачей. Прогнозирование необходимо для решения многих задач: разработки и оптимизации алгоритма для определенной архитектуры, оптимизации планирования задач, оптимизации размещения компонентов приложения на гетерогенной архитектуре, выборе архитектуры, наилучшим образом подходящей для конкретного приложения и многих других.

Целью работы является разработка подхода к прогнозированию динамических характеристик параллельных программ, не требующего запуска программы на целевой вычислительной системе, простого в использовании и не требующего много времени для получения прогнозируемых характеристик. В первом разделе статьи приводится классификация подходов к прогнозированию динамических характеристик параллельных программ. Во втором разделе описывается предлагаемый подход к анализу динамических характеристик параллельных программы и проводится его верификация. В заключения представлено краткое описание результатов, полученных в ходе работы, а также указаны дальнейшие направления исследования.

# 1. Подходы к прогнозированию динамических характеристик

Задачу предсказания значения динамической характеристики $h$ параллельной программы $p$ при входных параметрах $v \in V$ на вычислительной системе $c$ можно представить как вычисление функции $\hat{h} = \hat{H}(m, v)$, где $m$ — это модель, которая может быть представлена явно, как объединение двух моделей: модели программы $m_p$ и модель вычислительной системы $m_c$.

Можно выделить три подхода к прогнозированию динамических характеристик (рис. 1): аналитический, симуляционный и гибридный. В аналитических подходах предсказание представляет собой вычисление аналитического выражения. В симуляционных подходах предсказание получается посредством симулирования программы или ее редуцированного представления. Гибридные подходы используют как аналитические выражения, так и симуляцию для предсказания значения динамических характеристик.

## 1.1. Аналитические подходы

Основанием для построения модели прогнозирования могут являться как исходные тексты параллельных программ, так и информация о поведении программы, собранная во время ее выполнения. В зависимости от этого можно выделить три класса подходов к построению модели: статически-детерминированные (модель строится исходя только из исходного кода программы), эмпирически-детерминированные (модель определяется исходя из данных о результате выполнения программы на наборе параметров) и смешанные (используют как исходный код программы, так и данные о ее выполнении).

В зависимости от степени автоматизации получения модели $m$ аналитические подходы можно разделить на ручные, автоматизированные и автоматические.


**Рис. 1.** Классификация подходов к предсказанию характеристик параллельных программ

Ручной подход предполагает, что исследователь создает модель вручную, исходя из своего понимания программы. Можно выделить наиболее общие шаги создания модели, характерные для этого подхода: выделение входных параметров, влияющих на значение исследуемой характеристики, выделение ядер, определение коммуникационного шаблона и возможности перекрытия выполнения передачи данных с вычислениями. У ручного подхода есть два основных недостатка: необходимость глубокого понимания работы программы исследователем и большая трудоемкость.

Автоматизированные подходы более строго формализованы, что позволяет автоматизировать построение модели. С другой стороны, эти подходы зачастую требуют наличия исходного кода программы. В качестве примера автоматизированных подходов можно привести Modeling assertion framework (MA framework) [1] и Performance and architecture lab modeling tool (Palm) [2].

Автоматические подходы требуют минимального вмешательства со стороны исследователя, позволяя существенно облегчить разработку модели. Исследуемая программа в этом случае рассматривается как черный ящик. Автоматические подходы, в свою очередь, можно разделить на две группы: эмпирически-детерминированные и статически-детерминированные подходы. В эмпирически-детерминированных подходах модель строится исходя из данных, полученных при запуске параллельной программы. Статически-детерминированные подходы предполагают построение модель исходя из статического анализа исходного кода программы.

Эмпирически-детерминированные подходы хорошо изучены, что объясняется простотой их реализации. Наиболее часто встречающийся алгоритм построения модели в рамках данного подхода сводится к двум шагам: сбор значений исследуемой характеристики программы на подмножестве всевозможных значений входных параметров и обучение модели с помощью методов машинного обучения. В статье [3] проводится сравнение двенадцати различных алгоритмов машинного обучения на тестовом наборе мини-приложений Mantevo [4]. Исходя из результатов сравнения авторы делают вывод, что в случае плохо подобранных признаков сложные методы машинного обучения, например, глубокие нейронные сети требуют больший размер обучающей выборки и больше времени на обучение, но при этом имеют большую точность, чем такие простые методы, как МНК (метод наименьших квадратов).

Статически-детерминированные подходы сложны в своей реализации, но в отличие от эмпирически-детерминированных подходов они не требуют запуска целевой программы и зачастую не требуют наличия целевой вычислительной системы, что позволяет

использовать их как инструмент суперкомпьютерного кодизайна. Для оценки сложных динамических характеристик рассматриваемые системы используют довольно простые модели, например, Roofline model [5]. В качестве немногочисленных представителей данной группы подходов можно назвать системы COMPASS [6] и ExaSAT [7]. COMPASS генерирует модель программы, используя расширение предметно-ориентированного языка для моделирования производительности Aspen [8], а ExaSAT использует специально разработанное представление модели в формате XML. Отметим, что свободный доступ к обеим упомянутым системам не предоставляется.

### 1.2. Симуляционные подходы

В зависимости от источника информации о симулируемых действиях данную группу подходов можно разделить на программно-детерминированные подходы и трассо-детерминированные подходы. Программно-детерминированные подходы симулируют программу или ее редуцированное представление, а трассо-детерминированные подходы симулируют трассу событий. Трассо-детерминированные подходы зачастую работают быстрее. Они проще в реализации, чем подходы, симулирующие исполнение кода программы, но требуют получения исходной трассы путем исполнения программы или симуляции ее исполнения. Заметим, что трассы могут занимать много памяти. Существует множество симуляторов, предназначенных для работы с последовательными программами. Однако, такие симуляторы непригодны для анализа параллельных приложений.

Примером симулятора, предназначенного для работы с приложениями для больших вычислительных систем, является симулятор BigSim [9]. Получение прогнозируемых характеристик с использованием BigSim можно описать следующим образом. Анализируемое параллельное приложение сначала эмулируется на малом количестве узлов. При этом обеспечивается сбор трассы приложения. Затем проводится симуляция полученной трассы. BigSim поддерживает до 100000 виртуальных MPI-процессов, распределенных по 2000 узлам.

Возможность рассматривать целевую программу в качестве черного ящика и отсутствие необходимости в наличие целевой машины являются главными преимуществами симуляционного подхода. Основным недостатком данной группы подходов является их времязатратность.

### 1.3. Гибридные подходы

Гибридные подходы пытаются уменьшить времязатратность симуляционных подходов и увеличить точность аналитических подходов, комбинируя элементы обоих подходов.

В качестве примера инструмента, реализующего гибридный подход, можно привести программный пакет PSINS [10], нацеленный на предсказание времени работы параллельных MPI-приложений. Модель приложения в рамках данного инструмента создается для каждого набора входных параметров. Модель включает в себя информацию о количестве обращений в память и их характере, количестве операций с плавающей точкой и трассы вызовов MPI-функций. Модель целевой вычислительной системы включает информацию о производительности (flop/s), пропускной способности памяти и характеристиках коммуникационной сети. Для получения времени работы приложения сначала определяется время выполнения последовательных участков трасс на целевой вычислительной системе. После этого трассы модифицируются в соответствии с полученными временами и симулируются.

# 2. Предлагаемый статически-детерминированный подход к прогнозированию динамических характеристик

## 2.1. Описание подхода

Разработка подхода для прогнозирования характеристик проводилась исходя из следующих требований: подход не должен требовать запуска программы на целевой машине, подход не должен быть трудоемок для пользователя, предсказание характеристик не должно требовать много времени.

Исходя из представленных требований, предложенный подход можно классифицировать как статически-детерминированный. Существующие реализации статически-детерминированных подходов (COMPASS, ExaSAT) не имеют свободного доступа, что мотивировало разработку предлагаемого подхода.

Построение модели происходит на основе статического анализа исходного кода программы (рис. 2). Модель ВС и параметры запуска задаются пользователем, а затем подаются на вход инструментов предсказания динамических характеристик.

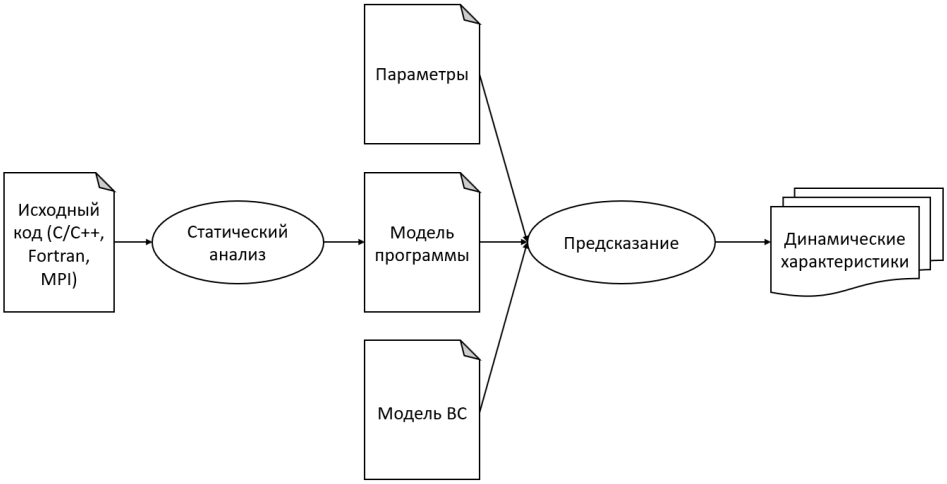



**Рис. 2.** Схема предлагаемого подхода

Модель программы описывается следующим образом:

$$m_p = < f_1, f_2, \dots, f_n >, \tag{1}$$

где $f_i$ — это модель $i$-ой функции программы. Функции упорядочены в соответствии с последовательностью их объявления в коде.

$$f_i = < flops_i,\ loadBytes_i,\ storeBytes_i,\ mpiCalls_i,\ sentBytes_i,\ recvBytes_i,\ loadAccess_i,\ storeAccess_i,\ mathCalls_i >, \tag{2}$$

где $flops_i$ = <$flopsTotal_i$, $flopsSingle_i$, $flopsDouble_i$> — кортеж из функций, возвращающих общее количество операций с плавающей точкой, количество операций с числами одинарной точности и количество операций с числами двойной точности выполняемых функцией,

$loadBytes_i$ — количество считанных из памяти байт, $storeBytes_i$ — количество записанных в память байт,

$mpiCalls_i$ = <$send_i$, $recv_i$, $sendrecv_i$, $bcast_i$,...> — кортеж, содержащий информацию о количестве вызовов MPI-функций,

$sentBytes_i$ — количество отправленных по сети байт данных, $recvBytes_i$ — количество принятых по сети байт данных,

$loadAccess_i$ — количество запросов на чтение из памяти, $storeAccess_i$ — количество запросов на запись в память,
$mathCalls_i = <sin_i, cos_i, tan_i,...>$ — кортеж, содержащий информацию о количестве вызовов математических функций.

На практике представленная модель программы генерируется статическим анализатором исходя из исходного кода программы. Модель представляется совокупностью python-классов, каждый из которых является моделью функции. Пример python-класса, представляющего фрагмент модели функции, приведен на рис. 3.

```
class calculate:

    @staticmethod
    def flops(gv_NT, gv_lNX, gv_lNY, gv_lNZ, **kwargs):
        return 12 + (-2 + max(2, gv_NT)) * updateValues.flops(gv_lNX, gv_lNY, gv_lNZ)

    @staticmethod
    def flops4(**kwargs):
        return 0

    @staticmethod
    def flops8(gv_NT, gv_lNX, gv_lNY, gv_lNZ, **kwargs):
        return 12 + (-2 + max(2, gv_NT)) * updateValues.flops8(gv_lNX, gv_lNY, gv_lNZ)

    @staticmethod
    def load_access(gv_NT, gv_lNX, gv_lNY, gv_lNZ, **kwargs):
        return 28 + (-2 + max(2, gv_NT)) * (updateValues.load_access(gv_lNX, gv_lNY, gv_lNZ) +
            exchangeShadow.load_access(gv_lNX, gv_lNY, gv_lNZ) + 4)

    @staticmethod
    def store_access(gv_NT, gv_lNX, gv_lNY, gv_lNZ, **kwargs):
        return 24 + (-2 + max(2, gv_NT)) * (updateValues.store_access(gv_lNX, gv_lNY, gv_lNZ) +
                exchangeShadow.store_access(gv_lNX, gv_lNY, gv_lNZ) + 2)
```

**Рис. 3.** Пример фрагмента модели функции

Модель ВС описывается следующим образом:

$$m_c =< netLatency, netBandwidth, n_1, \dots, n_k >, \tag{3}$$

где $netLatency$ — латентность сети, $netBandwidth$ — пропускная способность сети, $n_1$, ..., $n_k$ — модели узлов вычислительной системы.

$$n_i =< coreCount, performance, memBandwidth, mathFlops >, \tag{4}$$

где $coreCount$ — кол-во ядер на узле, $performance$ — производительность узла (в Gflop/s), $memBandwidth = <l1Band, l2Band, l3Band, dramBand>$ — пропускные способности различных уровней памяти, $mathFlops$ — кортеж с оценками количества операций с плавающей точкой, выполняемых математическими функциями.

Модель ВС также реализуется как python класс. Пример модели вычислительной системы, состоящей из пяти 20-ядерных узлов, приведен на рис. 4.

Прогнозирование времени работы MPI-приложения, исходя из модели программы и модели вычислительной системы, проводится следующим образом:

$$T_{mpi} = \max(T_1, T_2, \dots, T_n), \tag{5}$$

где $T_i$ — время работы $i$-го MPI-процесса.

$$T_i = T_{comm_i} + T_{comp_i}, \tag{6}$$

где $T_{comp_i}$ — время, затрачиваемое на вычисления, $T_{comm_i}$ — время, затрачиваемое на межсетевые коммуникации.

$$T_{comp_i} = \max\left(\frac{flops_i}{perf_i}, \frac{mem_i}{bandwidth_i}\right), \tag{7}$$

```
class Polus:
    @staticmethod
    def isMultiNode():
        return True

    @staticmethod
    def nodeCount():
        return 4

    @staticmethod
    def getNode(index):
        return PolusNode

    @staticmethod
    def networkLatency():
        return 60

    @staticmethod
    def networkBandwidth():
        return 10000
```

```
class PolusNode:
    @staticmethod
    def isMultiNode():
        return False

    @staticmethod
    def coreCount():
        return 20

    @staticmethod
    def l1Bandwidth():
        return 13000

    @staticmethod
    def dramBandwidth():
        return 9000

    @staticmethod
    def performance():
        return 5.3

    @staticmethod
    def flopsPerSin8():
        return 39

    @staticmethod
    def flopsPerCos8():
        return 39

    @staticmethod
    def flopsPerTan8():
        return 39

    @staticmethod
    def flopsPerSqrt8():
        return 1

    @staticmethod
    def flopsPerLog8():
        return 39

    @staticmethod
    def flopsPerLog10_8():
        return 39
```

**Рис. 4.** Пример модели вычислительной системы

где $flops_i$ — количество операций с плавающей точкой, выполненных $i$-ым MPI процессом, $perf_i$ — производительность ядра, на котором выполняется $i$-ый MPI процесс, $mem_i$ — объем записанных и прочитанных данных, $bandwidth_i$ — пропускная способность памяти. Подставляя пропускные способности различных уровней памяти, получаем времена работы при условии, что все запросы в память попадают в этот уровень памяти.

$$T_{comm_i} = callCount_i * netLatency + \frac{netSize_i}{netBandwidth}, \tag{8}$$

где $callCount_i$ — количество вызовов MPI функций, $netLatency$ — латентность сети, $netSize_i$ — размер передаваемых и получаемых данных, $netBandwidth$ — пропускная способность сети. Отметим, что такой способ оценки времени коммуникационного взаимодействия не очень точен. Однако, он требует минимум информации о вычислительной системе. В отличие от симуляции он является простым и быстрым в применении.

## 2.2. Верификация подхода

Верификация предложенного подхода проводилась на вычислительном кластере Polus [11]. Исследовались динамические характеристики программы, моделирующей распространение волны в трехмерном пространстве. Вычислительный кластер Polus состоит из пяти узлов, один из которых является головным. На каждом узле установлено два процессора IBM Power8 с графическими процессорами NVIDIA Tesla P100. Для получения пропускных способностей уровней памяти и производительности ядер использовался Empirical Roofline Tool [12]. Для определения латентности и пропускной способности сети использовался тестовый пакет OSU Micro-Benchmarks [13].

Программа, моделирующая распространение волны в трехмерном пространстве, основана на методе конечных разностей с использованием регулярной трехмерной сетки. Сетка разбивается на трехмерные блоки и равномерно распределяется по MPI-процессам. В каждой точке сетки итерационно рассчитываются значения искомых функций согласно заданному семиточечному разностному оператору. Для расчета значения переменной в каждой точке сетки требуется значения шести переменных в соседних точках сетки. На каждой итерации по времени процессы рассчитывают свою часть сетки и обмениваются боковыми гранями с шестью соседними процессами.

Запуски программы проводились на кубических сетках с размерами: 8x8x8, 16x16x16, 32x32x32, 64x64x64, 128x128x128, 256x256x256, 512x512x512. Для сбора информации о количестве операций с плавающей точкой отслеживалось аппаратное событие PM_FLOP.

На рис. 5 представлена относительная ошибка предсказания количества операций с плавающей точкой, рассчитываемая как $E_{rel} = \left|\left(h - \hat{h}\right)/h\right|$, где $h$ — реальное значение характеристики, $\hat{h}$ — предсказанное значение характеристики. Из рисунка видно, что максимальное значение относительной ошибки достигается при минимальных размерах сетки и значение ошибки существенно уменьшается с увеличением размера сетки. Этот эффект объясняется тем, что доля математических функций (sin, cos), используемых для инициализации сетки, больше на маленьких сетках. Количество же операций с плавающей точкой, выполняемых этими функциями, зависит от вычисляемого значения, из-за чего погрешность предсказания для таких функций больше.

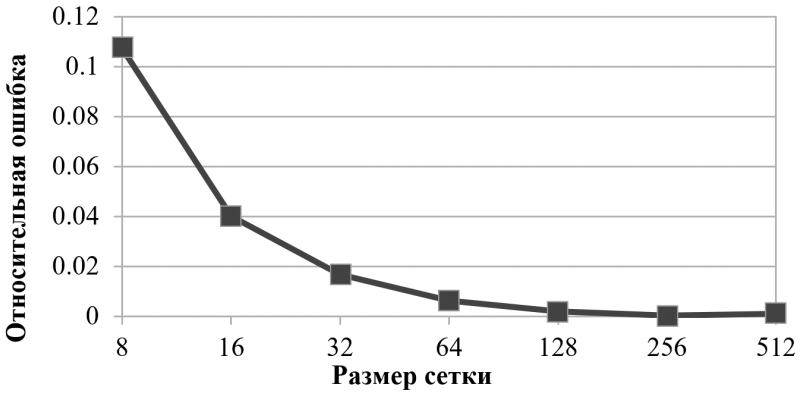
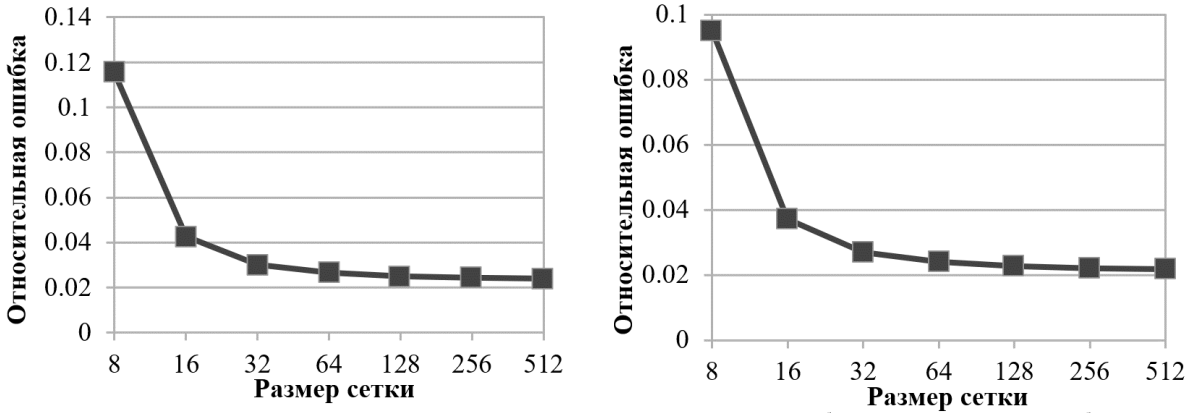



**Рис. 5.** Относительная ошибка предсказания количества операций с плавающей точкой

На рис. 6 представлены графики относительной ошибки предсказания количества операций доступа к памяти на чтение и запись. Для измерения количества запросов на чтение и запись использовались аппаратные события PM_LD_CMPL и PM_ST_CMPL соответственно. Погрешность при маленьких размерах сетки объясняется большей долей математических функций, а также обращениями к памяти, совершаемыми в процессе динамической линковки, количество которых не зависит от размера сетки.



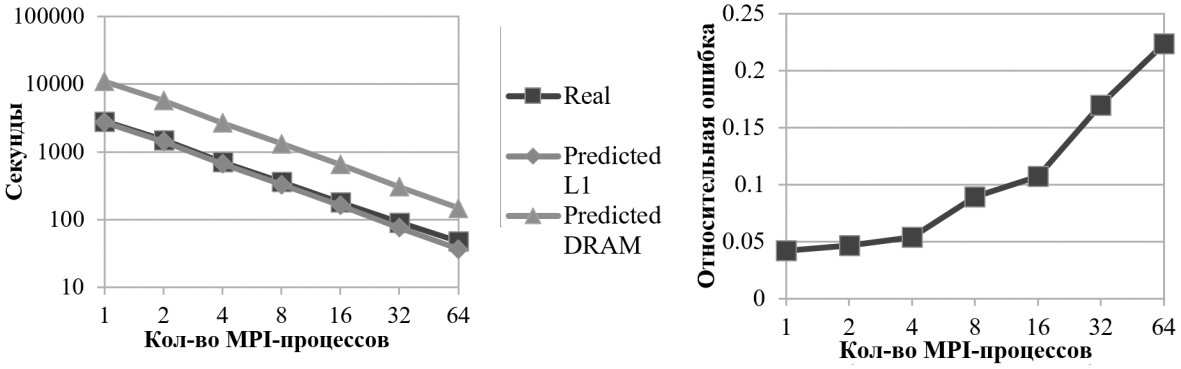

а) Относительная ошибка предсказания количества запросов на чтение

б) Относительная ошибка предсказания количества запросов на запись

**Рис. 6.** Относительная ошибка предсказания количества запросов на чтение и запись

Предсказание времени работы проводилось для сетки размером 512x512x512. На рис. 7а представлены графики времени выполнения реального приложения, а также предсказанного времени выполнения при условии, что все обращения в память попадают в кэш L1 или в оперативную память. На рис. 7б представлена относительная ошибка предсказания времени выполнения при условии, что все обращения попадают в кэш L1. Из рисунка видно, что реальное время выполнения очень близко к предсказанному времени

работы при попадании всех обращений в кэш L1. Исходя из этого, можно сделать вывод о том, что программа обладает хорошей локальностью обращений в память. Рост ошибки предсказания времени работы с увеличением количества процессов можно объяснить ростом погрешности в предсказании времени коммуникаций.




а) Реальное и предсказанное время работы

б) Относительная ошибка предсказания времени работы L1

**Рис. 7.** Измеренное и предсказанное время работы, а также относительная ошибка предсказания

## Заключение

В статье представлено описание разрабатываемого подхода для прогнозирования динамических характеристик параллельных программ, основанного на статическом анализе исходного кода. Предлагаемый подход позволяет предсказывать значение таких характеристик параллельных программ, как число обращений на чтение и запись в память, число считанных и записанных в память байт, количество операций с плавающей точкой, размер отправленных и принятых по сети данных, а также время работы программы. Верификация предложенного подхода продемонстрирована на параллельной программе, моделирующей распространение волны в трехмерном пространстве. Относительная ошибка прогнозирования динамических характеристик составила менее 25%.

Дальнейшее направление исследований предполагает верификацию приведенного подхода на ВС с процессорами Intel и AMD, расширение подхода для предсказания энергопотребления, повышения точности предсказания коммуникационных расходов и работу с приложениями, использующими графические ускорители.

*Работа выполнена при поддержке Российского фонда фундаментальных исследований (проект № 20-07-01053).*

## Литература

1. Alam S.R., Vetter J.S. A framework to develop symbolic performance models of parallel applications // Proceedings 20th IEEE International Parallel & Distributed Processing Symposium. IEEE, 2006. Vol. 2006. P. 1–8. DOI: 10.1109/IPDPS.2006.1639625.
2. Tallent N.R., Hoisie A. Palm: Easing the Burden of Analytical Performance Modeling // Proceedings of the 28th ACM International Conference Supercomput. 2014. P. 221–230. DOI: 10.1145/2597652.2597683.
3. Malakar P., Balaprakash P., Vishwanath V., et al. Benchmarking Machine Learning Methods for Performance Modeling of Scientific Applications // 2018 IEEE/ACM Performance

Modeling, Benchmarking and Simulation of High Performance Computer Systems, PMBS. IEEE, 2018. P. 33–44. DOI: 10.1109/PMBS.2018.8641686.
4. Mantevo Project. URL: https://mantevo.github.io (дата обращения: 03.03.2020).
5. Williams S., Waterman A., Patterson D. Roofline: an insightful visual performance model for multicore architectures // Commun. ACM. 2009. Vol. 52, no. 4. P. 65–76. DOI: 10.1145/1498765.1498785.
6. Lee S., Meredith J.S., Vetter J.S. COMPASS: A Framework for Automated Performance Modeling and Prediction // Proceedings of the 29th ACM on International Conference on Supercomputing. ACM Press, 2015. P. 405–414. DOI: 10.1145/2751205.2751220.
7. Unat D., Chan C., Zhang W., et al. ExaSAT: An exascale co-design tool for performance modeling // Int. J. High Perform. Comput. Appl. 2015. Vol. 29, no. 2. P. 209–232. DOI: 10.1177/1094342014568690.
8. Spafford K.L., Vetter J.S. Aspen: A domain specific language for performance modeling // 2012 International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE, 2012. P. 1–11. DOI: 10.1109/SC.2012.20.
9. Zheng G., Kakulapati G., Kale L.V. BigSim: a parallel simulator for performance prediction of extremely large parallel machines // Proceedings of the 18th International Parallel and Distributed Processing Symposium. IEEE, 2004. P. 78–87. DOI: 10.1109/IPDPS.2004.1303013.
10. Tikir M.M., Laurenzano M.A., Carrington L., et al. PSINS: An open source event tracer and execution simulator // Dep. Def. Proc. High Perform. Comput. Mod. Progr. - Users Gr. Conf. HPCMP-UGC 2009. 2009. P. 444–449. DOI: 10.1109/HPCMP-UGC.2009.73.
11. Polus. URL: http://hpc.cmc.msu.ru/polus (дата обращения: 03.03.2020).
12. Empirical Roofline Tool. URL: https://crd.lbl.gov/departments/computer-science/PAR/research/roofline/software/ert/ (дата обращения: 03.03.2020).
13. OSU Micro-Benchmarks. URL: http://mvapich.cse.ohio-state.edu/benchmarks/ (дата обращения: 03.03.2020).

Клейменов Андрей Анатольевич, аспирант, кафедра суперкомпьютеров и квантовой информатики, Московский государственный университет имени М.В. Ломоносова (Москва, Российская Федерация)

Попова Нина Николаевна, к.ф.-м.н., доцент, кафедра суперкомпьютеров и квантовой информатики, Московский государственный университет имени М.В. Ломоносова (Москва, Российская Федерация)

DOI: 10.14529/cmse210102

# A METHOD FOR PREDICTION DYNAMIC CHARACTERISTICS OF PARALLEL PROGRAMS BASED ON STATIC ANALYSIS

© 2021 A.A. Kleymenov, N.N. Popova
*Lomonosov Moscow State University*
*(GSP-1, Leninskie Gory 1, Moscow, 119991 Russia)*
*E-mail: andreykleimenov@mail.ru, popova@cs.msu.ru*
Received: 03.07.2020

In this paper, we consider the problem of prediction of parallel program dynamic characteristics, like execution time, count of floating-point operations, energy consumption, count of memory accesses and others. Prediction of dynamic characteristics allows solving many problems, related to design of new architectures, selection of the most suitable configurations of multiprocessor systems for solving specific problems, porting applications to new systems, task flow planning and more. The task of predicting characteristics is being actively investigated. Increasing complexity of the architectures of modern high-performance systems requires the development of new methods for solving the prediction problem. The article provides an overview of the existing approaches and software for predicting dynamic characteristics and proposes an approach based on a static analysis of the source code of a parallel application. Based on the text of the parallel program, the formal description of the target computing platform and the launch parameters, a method is implemented that allows predicting the operating time, the number of floating-point operations, number of memory accesses, and other characteristics of the parallel application. The applicability of the proposed approach is demonstrated by solving the test 3-dimensional numerical simulation problem on a multiprocessor cluster based on IBM Power8 processors.

*Keywords: parallel applications, dynamic characteristics, performance analysis, exaflop systems, computer model, static analysis.*

**FOR CITATION**

Kleymenov A.A., Popova N.N. A Method for Prediction Dynamic Characteristics of Parallel Programs Based on Static Analysis. *Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering.* 2021. Vol. 10, no. 1. P. 20–31. (in Russian) DOI: 10.14529/cmse210102.



## References

1. Alam S.R., Vetter J.S. A framework to develop symbolic performance models of parallel applications. Proceedings 20th IEEE International Parallel & Distributed Processing Symposium. IEEE, 2006. Vol. 2006. P. 1–8. DOI:10.1109/IPDPS.2006.1639625.
2. Tallent N.R., Hoisie A. Palm: Easing the Burden of Analytical Performance Modeling. Proceedings of the 28th ACM International Conference Supercomput. 2014. P. 221–230. DOI: 10.1145/2597652.2597683.
3. Malakar P., Balaprakash P., Vishwanath V., et al. Benchmarking Machine Learning Methods for Performance Modeling of Scientific Applications. 2018 IEEE/ACM Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems, PMBS. IEEE, 2018. P. 33–44. DOI: 10.1109/PMBS.2018.8641686.
4. Mantevo Project. URL: https://mantevo.github.io (accessed: 03.03.2020).

5. Williams S., Waterman A., Patterson D. Roofline: an insightful visual performance model for multicore architectures. Commun. ACM. 2009. Vol. 52, no. 4. P. 65–76. DOI: 10.1145/1498765.1498785.
6. Lee S., Meredith J.S., Vetter J.S. COMPASS: A Framework for Automated Performance Modeling and Prediction. Proceedings of the 29th ACM on International Conference on Supercomputing. ACM Press, 2015. P. 405–414. DOI: 10.1145/2751205.2751220.
7. Unat D., Chan C., Zhang W., et al. ExaSAT: An exascale co-design tool for performance modeling. Int. J. High Perform. Comput. Appl. 2015. Vol. 29, no. 2. P. 209–232. DOI: 10.1177/1094342014568690.
8. Spafford K.L., Vetter J.S. Aspen: A domain specific language for performance modeling. 2012 International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE, 2012. P. 1–11. DOI: 10.1109/SC.2012.20.
9. Zheng G., Kakulapati G., Kale L.V. BigSim: a parallel simulator for performance prediction of extremely large parallel machines. Proceedings of the 18th International Parallel and Distributed Processing Symposium. IEEE, 2004. P. 78–87. DOI: 10.1109/IPDPS.2004.1303013.
10. Tikir M.M., Laurenzano M.A., Carrington L., et al. PSINS: An open source event tracer and execution simulator. Dep. Def. Proc. High Perform. Comput. Mod. Progr. - Users Gr. Conf. HPCMP-UGC 2009. 2009. P. 444–449. DOI: 10.1109/HPCMP-UGC.2009.73.
11. Polus. URL: http://hpc.cmc.msu.ru/polus (accessed: 03.03.2020).
12. Empirical Roofline Tool. URL: https://crd.lbl.gov/departments/computer-science/PAR/research/roofline/software/ert/ (accessed: 03.03.2020).
13. OSU Micro-Benchmarks. URL: http://mvapich.cse.ohio-state.edu/benchmarks/ (accessed: 03.03.2020).

**УДК 004.942, 004.93'11** **DOI: 10.14529/cmse210103**

# МЕТОД УСКОРЕННОЙ ИДЕНТИФИКАЦИИ ОТПЕЧАТКОВ ПАЛЬЦЕВ

**© 2021 В.Ю. Гудков**
*Южно-Уральский государственный университет*
*(454080 Челябинск, пр. им. В.И. Ленина, д. 76)*
*E-mail: diana@sonda.ru*
Поступила в редакцию: 11.08.2020

В статье излагается метод ускоренной идентификации изображений отпечатков пальцев на основе шаблонов, которые формируются в результате автоматической обработки изображений. Метод опирается на свойства ближайших окрестностей контрольных точек в виде окончаний и разветвлений линий узоров пальцев и состоит из двух этапов. На первом этапе каждая контрольная точка запросного шаблона сравнивается с каждой контрольной точкой ссылочного шаблона из базы данных и оценивается степень похожести таких пар контрольных точек. Для ускорения вычислительных операций вводятся классы, которые позволяют быстро аккумулировать степень похожести контрольных точек из этих двух шаблонов в гистограмме. Оценивается качество такой гистограммы. Гистограммы строятся для всех ссылочных шаблонов из базы данных и одного запросного шаблона. На втором этапе на основе оценок гистограмм отбираются наиболее похожие шаблоны, число которых значительно меньше объема базы данных. Эти шаблоны сравниваются дополнительно с учетом консолидации контрольных точек и оценивается компактность расположения соответствующих пар контрольных точек из двух сравниваемых шаблонов. Значительное ускорение алгоритма идентификации достигается за счет отбрасывания непохожих пар контрольных точек на первом этапе и пар шаблонов с плохими оценками гистограмм на втором этапе. Приводятся результаты экспериментов, опубликованные в интернете.

*Ключевые слова: отпечаток пальца, идентификация, контрольная точка, гистограмма.*

**ОБРАЗЕЦ ЦИТИРОВАНИЯ**

Гудков В.Ю. Метод ускоренной идентификации отпечатков пальцев // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2021. Т. 10, № 1. С. 32–48. DOI: 10.14529/cmse210103.

## Введение

Для биометрической идентификации личности человека успешно применяются отпечатки пальцев [1]. Задача идентификации является основной для биометрических систем, таких как автоматизированные дактилоскопические системы, системы контроля доступа, паспортно-визовые системы и др. Работа этих систем опирается на специфические физиологические характеристики человека [2], которые сохраняют в базе данных (БД). В ее состав входят изображения отпечатков пальцев и их шаблоны — модели изображений [3]. Для ускорения вычислительных операций идентифицируют шаблоны, а не оригинальные изображения [3]. Применение шаблонов позволяет обеспечить приемлемый уровень качества распознавания [4, 5], однако производительность процедуры идентификации при этом остается ограниченной [6]. Это явление особенно становится заметным с ростом объема БД. Компенсировать его можно за счет повышения стоимости материальной части биометрической системы. Однако такой подход не всегда приемлем [5, 7]. Как существенно сократить время задачи идентификации без увеличения стоимости системы, обеспечивая при этом приемлемый уровень снижения качества распознавания? Эта задача чрезвычайно актуальна и может быть решена с помощью новых математических моделей изображений, новых парадигм, методов и алгоритмов распознавания изображений [8, 9].

Целью исследования является разработка метода и алгоритма ускоренной идентификации изображений отпечатков пальцев по их шаблонам.

Статья состоит из трех разделов и заключения. В разделе 1 коротко освещаются предшествующие работы, указываются недостатки опубликованных методов и нерешенные проблемы. В разделе 2 излагается основная идея по улучшению качества и повышению производительности задачи идентификации, предлагаются классификаторы на основе топологических и геометрических свойств изображений, описывается способ сравнения контрольных точек (КТ) [3] для запросного и ссылочного шаблонов, а также способ построения гистограммы и оценки ее качества. Здесь же освещается новый алгоритм сравнения шаблонов отпечатков пальцев для больших БД и раскрываются особенности ускоренной идентификации. В разделе 3 представлены результаты экспериментов, опубликованные в интернете. В заключении приводится резюме, выводы и направление дальнейших исследований.

## 1. Обзор предшествующих работ

Для улучшения качества и повышения производительности задачи идентификации предложено много методов [1, 2, 4–9]. Корректный выбор пары КТ (одна из запросного, а другая из ссылочного шаблонов), образующих пару предполагаемых похожих признаков, в основном определяет качество и производительность идентификации [5, 6]. Однако множество КТ, а также отношений между ними, может быть сохранено в шаблоне различными способами. Наиболее часто для сравнения нескольких пар КТ применяется гребневый счет, разность расстояний между ними и разность их векторов направлений [7–10]. Под гребневым счетом понимают число линий на изображении, которые пересекаются отрезком, проведенным от одной КТ до другой [3, 11].

Локальная топологическая структура для КТ (LMTS — local minutiae topological structure) предложена в работе [12]. Структура устойчива к появлению шумов на изображении и деформации узора, потому что отражает его топологические свойства. Применение LMTS снижает ошибки идентификации. Несколько позже был предложен метод, объединяющий структуры LMTS в пары [13], что позволило авторам улучшить предшествующие результаты исследований за счет увеличения числа сравниваемых геометрических и топологических характеристик изображений. Следующим этапом развития этого подхода стало объединение LMTS в созвездия по 3 и более элементов [14]. Интересно, что аналогичный LMTS, но несколько упрощенный подход для оценки подобия локальных структур был предложен ранее в [15], однако он не обеспечил сравнимые с созвездием LMTS результаты. Авторам указанных работ удалось повысить производительность и улучшить результаты идентификации отпечатков пальцев.

В работе [16] специальные структуры, называемые $k$-plet, формируются для каждой КТ изображения. На основе этих структур предложен метод двойного поиска в ширину (CBFS — Coupled Breadth First Search). Многошаговый алгоритм CBFS обрабатывает изображение, фильтрует, строит скелет линий [1, 2, 7] и, прослеживая скелет линий, детектирует структуру $k$-plet для каждой КТ. По сути эти структуры являются небольшими подграфами, описывающими скелет узора в некоторой области КТ. Подграфы объединяются в граф. При идентификации изображений такие графы сравниваются и вычисляется степень их подобия.

В работе [17] метод CBFS был развит с помощью преобразования графа в ориентированный граф. Каждая КТ снабжается ориентированным графом смежности (MAG — Minutiae Adjacency Graph) на основе выбора другой ближайшей КТ и заданных ограничений. Тогда исследуемая КТ как вершина графа представляется в виде $\left\{i, j, d_{ij}, r_{ij}, \varphi_{ij}\right\}$,

где $i, j$ — номера КТ, а значения $d_{ij}$, $r_{ij}$ и $\varphi_{ij}$ — расстояние, гребневый счет и разность векторов направлений между этими КТ.

В работе [18] идея графа MAG приобретает интересную модификацию. В ней предлагаются графы для петель, дельт и завитков узора так же, как и для КТ, а граф рассматривается как иерархически организованное дерево. При сравнении шаблонов все графы используются консолидировано с учетом признаков различного уровня [3].

В работе [19] предлагается использовать хэш-коды для ускорения идентификации в больших БД. Для каждой КТ строится вектор цилиндрического кода (MCC — Minutia Cylinder Code), инвариантный к положению и ориентации КТ. Новизна заключается в оригинальном численном расчете степени похожести векторов MCC на основе классификации и взвешенного сглаживания элементов векторов. Показана их хорошая эффективность. Развитие идеи MCC предложено в работе [20] на основе компактного представления признаков в векторе сферического кода и модификации метрического пространства (MSCC — Minutia Spherical Coordinate Code). В ней авторы вводят представление признаков на сбалансированной решетке, что позволяет лучше отбирать идентичные КТ.

В целом, эти методы и алгоритмы демонстрируют развитие задачи идентификации. Но они ориентированы на относительно хорошие изображения. Корректный отбор идентичных КТ нарушается на изображениях с дефектами. Качество изображения можно оценить по критерию NIST (National Institute of Standards and Technology) [21]. Идентификация состоит из ряда подзадач классификации, которые выполняются при решении самой задачи идентификации, а при обработке изображений — лишь частично. Ясно, что предварительная классификация разнообразных структур и подграфов, а также их упорядочение, снижает объем вычислений при выполнении основной задачи.

Подведем итог. При идентификации отбор наиболее похожих пар КТ необходим, но недостаточен. Проблемы это не решает. Решение видится в том, чтобы дополнительно накопить оценки похожести отобранных пар КТ заранее и аккумулировать их. Такие оценки позволяют отфильтровать непохожие шаблоны при минимуме вычислений и выполнить более детальное их сравнение для небольшой доли БД. Такое решение непросто реализовать в алгоритме.

Сформулированное направление исследований оказывается не без подводных камней. Они скрываются в изменчивости и широкой вариабельности изображений отпечатков пальцев, в фрагментарности изображений и поражающих воображение деформациях узоров, в неочевидных искажениях узоров в процессе следообразования отпечатков на воспринимающей поверхности. Это влияет на состав шаблонов и классификацию структур, подграфов и графов. Для компенсации нежелательного эффекта необходимо разработать устойчивые классификаторы различного уровня и механизм классификации накопленных оценок похожести КТ.

## 2. Метод ускоренной идентификации

### 2.1. Классификаторы первого уровня

Рассмотрим поэтапно метод и алгоритм идентификации отпечатков пальцев. На первом этапе определяются классификаторы. Их большая часть рассчитывается при обработке изображений. Основной состав шаблона изображения — это список КТ. Каждая КТ $m_i$ определяется следующим образом:

$$m_i = \{(x_i, y_i), \alpha_i, t_i\},\ i \in 1..n, \tag{1}$$

где $(x_i, y_i)$ — координаты $m_i$; $\alpha_i$ — вектор направления $m_i$ в диапазоне $[0; 2\pi)$; $t_i$ — тип $m_i$, $t_i \in \{0,1\}$, $t_i = 0$ для КТ типа разветвления и $t_i = 1$ для КТ типа окончания [7]; $n$ — количество КТ на изображении. Вектор направления $\alpha_i$ указывает в сторону увеличения

числа линий по касательной к линии, на которой располагается КТ $m_i$ [22]. Дополнительно каждая КТ снабжается описанием некоторой компактной окрестности, которое назовем гнездом.

Рассмотрим следующую последовательность операций.

Выполним обработку и скелетизацию изображения [22]. Этот шаг можно выполнить по-разному [1, 2], но от качества его выполнения зависят ошибки идентификации. На скелете изображения детектируем КТ, часть из которых отмечена точками (см. рис. 1).

**Рис. 1.** Скелет изображения с группой КТ, детектированных в гнезде, которое построено на основе сечения, показанного толстой линией

От каждой КТ отметим проекции на соседних скелетных линиях. Этих линий две. Проекции перпендикулярны вектору направления КТ (см. рис. 2).

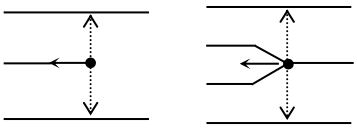


**Рис. 2.** Проекции от окончания и разветвления

От каждой КТ проведем сечение на несколько линий в обе стороны. Для обеспечения устойчивости сечения оно проводится перпендикулярно линиям (см. рис. 1). Сечением каждая линия разрезается на две части, которые назовем связями. Связи можно нумеровать по линейному порядку, но лучше нумеровать их по часовой стрелке, как показано на рис. 3. Множество связей и признаки, детектированные на связях, образуют гнездо. Таким образом, каждая КТ снабжается гнездом. Два гнезда представлены на рис. 3.

Для КТ типа окончания номер 19 (показано слева на рис. 3) нумерация связей начинается с линии, заканчивающейся окончанием (связь номер 0). Для КТ типа разветвления номер 19 (показано справа) нумерация связей начинается с раздваивающейся линии (связь номер 0). Сечение для КТ типа окончания генерирует 17 связей (0..16), для КТ типа разветвления — тоже 17 связей (0..16). Глубину сечения задают от 2 до 8 линий.

Указанный метод нумерации связей по спирали позволяет легко менять глубину сечения без нарушения структуры гнезда. Это важное свойство.

При конструировании гнезда проследим ход каждой связи, начиная от сечения. Это задает ориентацию связи. Прослеживая связь, детектируем КТ, располагаемую на связи, или отмеченную проекцию от другой КТ (КТ номер 20..27 на рис. 3).

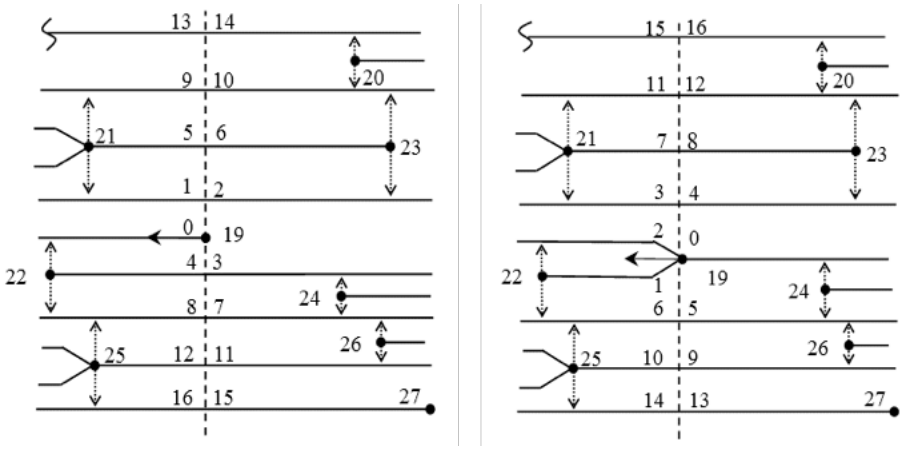



**Рис. 3.** Сечения для окончания и разветвления с нумерованными связями

Объем шаблона в основном зависит от количества связей, определяемых по формуле

$$w_i = 4x + 2 + (-1)^{t_i}, \tag{2}$$

где $t_i$ — тип $m_i$ (1); $x$ — число разрезаемых линий. Поэтому число линий, разрезаемых сечением, необходимо задавать аккуратно. С одной стороны, с ростом числа $x$ появляется возможность обогащения гнезда данными. С другой стороны, увеличивается требуемый объем памяти для представления гнезда в алгоритме идентификации. При $x = 7$ объем памяти для размещения одного шаблона прокатного отпечатка пальца колеблется около 30 Кб. Тогда для работы с 1 млн. дактилокарт, а каждая дактилокарта содержит 10 отпечатков пальцев, требуется около 300 Гб памяти. А объем БД современных дактилоскопических систем иногда достигает 100 млн. дактилокарт.

При построении гнезда и прослеживании связей на скелете встречаются КТ или проекции от них, которые формируют события (см. рис. 4). Каждое событие ассоциируется с номером КТ, сформировавшей это событие, и с номером связи [7, 22]. В составе события отражается совпадение или несовпадение ориентации вектора направления КТ с ориентацией связи с точностью до $\pi/2$, наличие на связи КТ или проекции от нее, причем справа или слева от связи, обрыв связи. События 1100 и 0000 не снабжаются номером КТ, потому что связь замыкается, не встречая событие, или обрывается на краю узора. Совокупность событий и ассоциированных с ними номеров КТ, упорядоченных по номерам связей, образует гнездо. Его можно трактовать как ориентированных граф. Достоинства конструкций в виде CBFS и MAG в гнезде учитываются в полной мере.

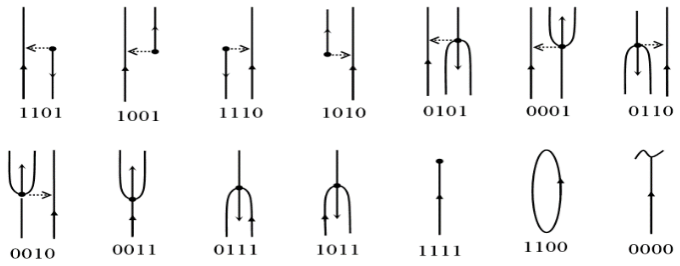



**Рис. 4.** События, детектируемые на связях при прослеживании скелетных линий

Математически множество связей в гнезде для КТ $m_i$ записывается в виде

$$N_i = \{m_k, d_{ik}, r_{ik}, \gamma_{ik}, v_{ik}\},\ m_k = \{(x_k, y_k), \alpha_k, t_k\},\ k \in K, \tag{3}$$

где $d_{ik}$, — расстояние между КТ $m_i$ и $m_k$; $r_{ik}$ — гребневый счет между ними; $\gamma_{ik}$ — азимут как угол поворота вектора направления $\alpha_i$ до луча, ориентированного от $m_i$ на

$m_k$ (см. рис. 5), $\alpha_k$ — вектор направления (1); $v_{ik}$ — событие, сформированное $m_k$, в гнезде для $m_i$ (см. рис. 4); $K$ — количество связей в гнезде. Ясно, что ассоциированные со связями данные устойчивы: они не зависят от ориентации и положения системы координат. Они же инвариантны к масштабу изображения, кроме $d_{ik}$.

Рассмотрим КТ $m_i$. Она и КТ на связи $m_k$, $k \in K$, образуют диполь (см. рис. 5). Диполь снабжается длиной $d_{ik}$, азимутом $\gamma_{ik}$, гребневым счетом $r_{ik}$ (3), а также углом $\varphi_{ik}$ как разностью направлений векторов $\alpha_i$ и $\alpha_k$ (1). Диполь как вектор выражается в виде

$$D_{ik} = (\gamma_{ik}, d_{ik}, r_{ik}, \varphi_{ik}). \tag{4}$$

Для КТ $m_i$ на основе гнезда можно построить множество диполей. Назовем это множеством диполей гнезда (MND — Multiplicity of Nest Dipoles). На рис. 5 показаны два диполя из запросного и ссылочного шаблонов. В ссылочном шаблоне рассматривается гнездо для КТ $m_j$.

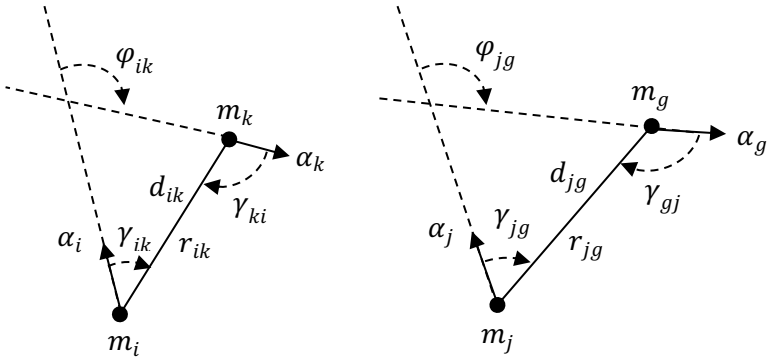



**Рис. 5.** Параметры диполей для запросного и ссылочного шаблонов

Все построенные диполи MND сохраняются в шаблоне. Сравним каждый диполь $\omega(m_i, m_k)$ из запросного шаблона с каждым диполем $\omega(m_j, m_g)$ из ссылочного шаблона. Оценим степень их похожести по формуле:

$$\psi\left(\omega(m_i, m_k), \omega(m_j, m_g)\right) = f(\Delta d, \Delta\gamma_1, \Delta\gamma_2, \Delta\gamma_3). \tag{5}$$

Здесь $\Delta d = \|d_{ik} - d_{jg}\|$ — норма разности длин диполей (см. рис. 5); $\Delta\gamma_1$ — норма разности углов азимутов $\gamma_{ik}$ и $\gamma_{jg}$; $\Delta\gamma_2$ — норма разности углов азимутов $\gamma_{ki}$ и $\gamma_{gj}$; $\Delta\gamma_3$ — норма разности углов $\varphi_{ik}$ и $\varphi_{jg}$, см. (4); функция $f$ является полиномом первого порядка. Нормируем значение функции $0 \le \psi \le 1$ [23]. Если $\psi = 1$, то диполи идентичны.

Степень похожести диполей по топологии зависит от события $v_{ik}$ запросного шаблона и события $v_{jg}$ ссылочного шаблона (3). Зависимость определяется составом бит в событии, см. рис. 4. Рассматривая состав бит, определим числа $\tau_{ik}$ и $\tau_{jg}$ как состояние связей. Состояние связи равно нулю, если вектор направления КТ ориентирован вдоль связи, иначе единица [7, 22]. Оно кодируется соответствующим событием с помощью функции как таблицы $T_1 = (\sim,0,0,0,\sim,1,1,1,\sim,0,0,1,\sim,1,1,1)$ в виде:

$$\tau_{ik} = T_1(v_{ik}),$$
$$\tau_{jg} = T_1(v_{jg}).$$

В таблице $T_1$ символ $\sim$ означает отсутствие состояния связи. Такая связь не используется.

Для оценки степени похожести диполей по топологии применим операцию «исключающее или»:

$$\lambda\left(\omega(m_i, m_k), \omega(m_j, m_g)\right) = \tau_{ik} \oplus \tau_{jg}. \tag{6}$$

Ясно, что $\lambda \in \{0,1\}$.

Для расчета финальной степени похожести двух диполей умножим оценки (5) и (6):

$$s_{ik,jg} = \psi\left(\omega(m_i, m_k), \omega(m_j, m_g)\right) * \lambda\left(\omega(m_i, m_k), \omega(m_j, m_g)\right). \quad (7)$$

Очевидно, что $0 \le s_{ik,jg} \le 1$. Для ускорения операций по вычислению различных норм необходимо задать длину диполей и величин углов в целочисленном диапазоне, например, с помощью квантования, а затем применить сложение, вычитание и вычисление абсолютных значений многобайтовых величин [24].

Для каждой КТ в ее гнезде выберем по крайней мере две ближайшие КТ. Например, в гнезде КТ $m_i$ на рис. 6 выбраны две КТ $m_l$ и $m_r$. Чем богаче выбор, тем выше надежность идентификации и ниже производительность алгоритма. Соответствующий гребневый счет равен $r_{il}$ и $r_{ir}$. Угол азимута $\gamma_{ik} \in [0; 2\pi), k \in \{l, r\}$, с помощью квантования преобразуется к целочисленному углу $\gamma_{ik} \in 0..63$, который рассматривается как один из 64 возможных классов.

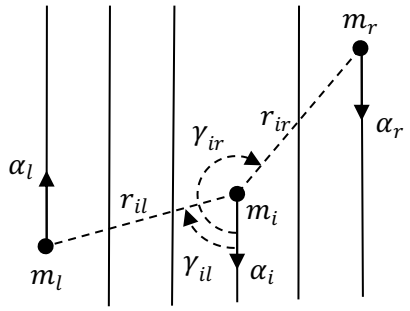



**Рис. 6.** Два диполя для образующей гнездо КТ $m_i$

Такого числа классов недостаточно: высока вероятность ложного сравнения двух различных диполей. Дополнительный бит в описании диполя позволяет удвоить число классов. Это бит состояния связи $\tau_{ik} = T_1(v_{ik})$, где $k \in \{l, r\}$, $\tau_{ik} \in \{0,1\}$. Таким образом, число классов становится равным 128.

Введем еще один реверсивный бит состояния связи $\tau_{ki}$, который вычисляется в гнезде для КТ $m_k$, $k \in \{l, r\}$ аналогичным способом. Тогда классы $\mathrm{c}_{ik}$, $k \in \{l, r\}$, для диполей на рис. 6, определяются в виде одного байта (см. рис. 7).

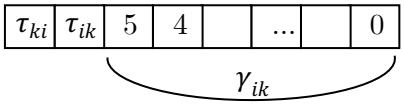



**Рис. 7.** Структура класса $\mathrm{c}_{ik}$

Таким образом, число классов расширяется до 256. Перепишем структуру диполей MND как вектора (4), считая класс диполя (см. рис. 7) его первым компонентом:

$$D_{ik} = \{\mathrm{c}_{ik}, d_{ik}, r_{ik}, \varphi_{ik}\}, k \in \{l, r\}. \quad (8)$$

При обработке изображения упорядочим диполи по номеру классов и сформируем список диполей для каждого шаблона. Эта предварительная операция упорядочения существенна. Вычислим пересечение списков запросного и ссылочного шаблонов [23]. Если классы совпадают, вычислим разность расстояний $d_{ik}$ и $d_{jg}$, разность гребневого счета $r_{ik}$ и $r_{jg}$, разность углов $\varphi_{ik}$ и $\varphi_{jg}$ (см. рис. 5). Если разности в пределах заданных допусков, то КТ $m_i$ и $m_j$ рассматриваются как похожие и сохраняются в списке $L_0$. Список $L_0$ оформляется в виде пар индексов

$$L_0 = \{(i, j)\}. \quad (9)$$

Величины допусков определяются на этапе обучения алгоритма. Сравнение классов диполей выполняется столь же быстро, как и сравнение целых значений. Остальные вычисления выполняются строго после сравнения классов. Таким образом, список $L_0$ формируется очень быстро. Для узора, содержащего 30..100 КТ, обычно строят 60..200 диполей. С учетом сравнения классов как пересечения списков [23], контроля допусков для разностей параметров диполей и алгоритмических трюков [24] при сопоставлении двух шаблонов на основе классификации всех возможных пар КТ на практике требуется менее $10^3$ вычислительных операций.

## 2.2. Классификаторы второго уровня

В списке $L_0$ согласно (9) сохраняется много предполагаемых похожих пар КТ, причем одной и той же КТ из запросного шаблона может быть поставлено в соответствие несколько КТ из ссылочного шаблона. Ясно, что избирательность классификаторов первого уровня недостаточна. Поэтому введем классификаторы на основе топологии гнезда.

В результате воздействия помех тип КТ может мутировать. На рис. 3 КТ типа окончания 19 мутирует влево и замыкается в КТ типа разветвления 19. Видно, что связи номер 0 для КТ типа окончания соответствует связь номер 2 для КТ типа разветвления, связи номер 1 — связь номер 3 и т.д. Мутации $\delta$ распознают по несовпадающим типам КТ согласно (1).

Если мутации нет, то связи не перенумеровывают, а используют как есть по формуле

$$H_0(j) = \{0,1,2,\ 3,4,5,6,\ 7,8,9,10,\ 11,12,13,14\}, \qquad \delta = 0.$$

При замыкании КТ типа окончания влево связи перенумеровывают по формуле

$$H_1(j) = \{2,3,4,\ 0,1,7,8,\ (5,6,11,12),(9,10,15,16)\}, \qquad \delta = 1.$$

При замыкании КТ типа окончания вправо связи перенумеровывают по формуле

$$H_2(j) = \{1,2,0,\ 5,6,3,4,\ (9,10,7,8),(13,14,11,12)\}, \qquad \delta = 2.$$

При разрыве КТ типа разветвления влево связи перенумеровывают по формуле

$$H_3(j) = \{2,0,1,\ 5,6,3,4,\ (9,10,7,8),(13,14,11,12)\}, \qquad \delta = 3.$$

При разрыве КТ типа разветвления вправо связи перенумеровывают по формуле

$$H_4(j) = \{3,4,0,\ 1,2,7,8,\ (5,6,11,12),(9,10,15,16)\}, \qquad \delta = 4.$$

Каждая формула — вектор, в котором индекс элемента (счет от 0) есть номер связи для КТ из запросного шаблона. Значение элемента — номер связи для КТ из ссылочного шаблона. Так связи номер 7 (элемент номер 7) КТ типа окончания соответствует связь номер 5 КТ типа разветвления, см. $H_1(j)$. А связи номер 11 (элемент номер 11) — связь номер 9 так же по $H_1(j)$. Круглыми скобками выделены элементы, значения которых подчиняются закону периодичности. Последующие значения вычисляются добавлением числа 4 (9=5+4) (см. рис. 3). Поэтому легко строить произвольную глубину сечения.

Выберем пару КТ $m_i$ и $m_j$ из списка $L_0$ (см. рис. 5). Гнезда каждой КТ образуют набор состояний связей $\sigma_i = \{\tau_{ik1}, \tau_{ik2}, \ldots\}$ и $\sigma_j = \{\tau_{jg1}, \tau_{jg2} \ldots\}$ согласно формулам (3) и (6). Для 32 связей образуется два 32-битных слова $\sigma_i$ и $\sigma_j$. Номер связи (см. рис. 3) определяет позицию соответствующего бита состояния этой связи. Таким образом, 32-битное слово отражает топологические свойства некоторой окрестности изображения. Сравним слова $\sigma_{ij} = \sigma_i \oplus \sigma_j$ побитовой операцией «исключающее или» и инвертируем значения бит в слове $\sigma_{ij}$, чтобы биты для совпадающих по состоянию связей стали равны единице. Введем

таблицу $T_2 = (0,1,1,1,0,1,1,1,0,1,1,1,0,1,1,1)$ для маскирования событий 1100 и 0000 (см. рис. 4) и на основе событий согласно (3) определим маскирующие биты в виде

$$w_{ik} = T_2(v_{ik}),$$
$$w_{jg} = T_2(v_{jg}).$$

Два набора маскирующих бит $\mu_i = \{w_{ik1}, w_{ik2}, \ldots\}$ и $\mu_j = \{w_{jg1}, w_{jg2} \ldots\}$ формируют две 32-битные маски. Вычислим обобщенную маску. $\mu_{ij} = \mu_i \& \mu_j$, применяя операцию «побитовое и», и функцией $f$ подсчитаем число единичных бит $b_{ij} = f\left(\sigma_{ij} \& \mu_{ij}\right)$ в слове $\sigma_{ij}$ [24].

Для учета возможных мутаций $\delta$ типов КТ воспользуемся таблицами перенумерации связей $H_\delta(j)$, определим максимум числа подобных связей в гнездах по топологии в виде

$$b_{ij}^{\delta} = \max_{\delta} b_{iH_\delta(j)} \tag{10}$$

и сформируем список $L_1$ для лучших оценок $b_{ij}^{\delta}$ пар КТ $m_i$ и $m_j$:

$$L_1 = \left\{\left(i, j, b_{ij}^{\delta}\right)\right\}, i \in \{1, n_1\}, j \in \{1, n_2\}, \tag{11}$$

где $n_1$ — количество КТ на запросном шаблоне, $n_2$ — на ссылочном. Упорядочим список $L_1$ по оценкам $b_{ij}^{\delta}$. Список $L_1$, построенный на классификаторах второго уровня, отражает значительно лучшую избирательность похожих КТ и формируется очень быстро, так как слова и маски рассчитываются на этапе обработки узора. Для оценки классификаторов второго уровня (вызов функции $f$ [24]) при совпадении типов двух КТ требуется менее 25 однотактовых команд, а при несовпадении — менее 50. Классификаторы первого уровня пропускают лишь долю от числа $n_1 n_2$ сравниваемых КТ, равную, как правило, 20..200. Поэтому классификаторы второго уровня оцениваются менее чем за $50 * 200 = 10000$ вычислительных операций, а с учетом накладных расходов число операций удваивается.

## 2.3. Классификаторы третьего уровня

Классификаторы второго уровня отражают подобие окрестностей гнезд КТ по топологии. Следующий шаг в повышении избирательности шаблонов заключается в аккумулировании наиболее подобных по топологии гнезд. Для этого из упорядоченного списка $L_1$ по (11) выберем $M$ лучших пар КТ по оценке (10), причем $M \le \min(n_1, n_2)$, и оценим степень похожести двух шаблонов в виде

$$s = \sum_{m=1}^{M} b_{i_m, j_m}^{\delta}, \tag{12}$$

где индексы $i_m$, $j_m$ уникальны и не дублированы. Значение $M$ определяется при обучении алгоритма. Его увеличение улучшает отбор шаблонов, но замедляет алгоритм.

Сравнивая запросный шаблон с каждым ссылочным шаблоном из БД, построим гистограмму похожести шаблонов $H(s)$ для запросного шаблона, где $s$ определяется по критерию (12). Таким образом, классификаторы третьего уровня аккумулируются в гистограмме. Опираясь на уже вычисленные оценки (10), оценка по (12) выполняется за $M$ команд суммирования.

На этом первый этап идентификации шаблонов отпечатков пальцев заканчивается. Он обеспечивает основное ускорение метода идентификации.

## 2.4. Алгоритм идентификации шаблонов

Второй этап идентификации опирается на гистограммные оценки.

Построим функцию распределения вероятностей $F(s)$ для гистограммы $H(s)$ и определим квантиль уровня $p$ для лучших оценок по критерию (12). Он определит пороговую величину для доли ссылочных шаблонов БД, которую обозначим в виде $F^{-1}(p)$. Для ускорения идентификации список $L_1$ можно сохранить на первом этапе.

Для выбранной пары запросного и одного ссылочного шаблона с оценкой $b_{ij}^{\delta} > F^{-1}(p)$ рассчитаем финальную степень подобия.

Из списка $L_1$ выберем $K = \sqrt{n_1 n_2}$ наиболее похожих пар КТ по оценке (10). Рассмотрим пару КТ $m_i$ и $m_j$ (см. рис. 3). Обозначим числами $K_i$ и $K_j$ количество связей в гнезде для КТ $m_i$ и $m_j$ соответственно. Воспользуемся геометрическими и топологическими характеристиками изображения [25, 26] и оценим подобие гнезд по формуле

$$Q_{ij} = \sum_{k,g=1}^{\min(K_i,K_j)} \max_{\delta} \psi\left(\omega(m_i, m_k), \omega\left(m_j, m_{H_\delta(g)}\right)\right), \tag{13}$$

где $\psi$ вычисляется по оценке (5) с учетом возможных мутаций $\delta$ для КТ из ссылочного шаблона. Диполи сравниваются на идентичных или перенумерованных связях.

На основе списка $L_1$ по (11) и оценки подобия по критерию (13) сконструируем список $L_2$ в виде

$$L_2 = \{(i, j, Q_{ij})\}, i \in \{1, n_1\}, j \in \{1, n_2\}, \tag{14}$$

где $n_1$ и $n_2$ — количество КТ как в (11). Упорядочим список $L_2$ по подобию (13).

Список $L_2$ сохраняет множество индексов пар КТ. Выберем лучшую пару КТ $m_{i_1}$ и $m_{j_1}$ из списка $L_2$ по (14), $i_1 \in \{1, n_1\}, j_1 \in \{1, n_2\}$. Положим число $t \leftarrow 1$ пар КТ выбранных из списка $L_2$ и зададим пороговое значение $thr$. Построим стек $L_3$ и очистим его $L_3 \leftarrow \{\emptyset\}$. Обнулим переменные $V_1 \leftarrow 0$, $V_2 \leftarrow 0$, $N \leftarrow 0$. Зададим длину маршрута $len$, функцию сортировки $sort$, функцию чтения вершины стека $pop$.

Алгоритм развивается от начальной пары КТ $m_{i_1}$ и $m_{j_1}$. Он представлен на рис. 8 и состоит из следующих шагов.

```
1:  L3 ← L3 + (i1, j1);  (i1, j1) ∈ L2;  Q_{i1 j1} ← 0
2:  V1 ← 0;  N ← 0
3:  for all (i, j) ∈ L3 do
      if Q_ij = 0 then оценим Q_ij по (13)
4:  L3 ← sort L3; на вершине стека располагается лучшая пара КТ
5:  Читаем пару КТ (i, j) ← pop L3; обновляем N ← N + 1
6:  Оценим Q_ij по (13) и аккумулируем оценки V1 ← V1 + Q_ij
7:  Удалим пары КТ с теми же индексами i или j из L3
      for all (a, b) ∈ L3 do
          if a = i or b = j then
              L3 ← L3 − (a, b)
8:  Сравним диполи D_ik и D_jg и рассчитаем степень похожести s_{ik,jg}
    согласно формулам (4)-(7) в зависимости от равенства классов,
    разностей углов φ_ik и φ_jg, углов азимутов, расстояний d_ik и d_jg
    и гребневого счета r_ik и r_jg (см. рис. 5);
    если разности в пределах заданных допусков, то L3 ← L3 + (k, g)
    и Q_kg ← 0;
    пара КТ m_k и m_g рассматривается как канидат идентичных КТ;
    согласно формуле (8) может сформироваться множество пар
    (m_{k1}, m_{g1}), (m_{k2}, m_{g2}) и т.д.;
    все они сохраняются в L3
9:  if L3 ≠ {∅} and N < len then go to 3
10: if V2 < V1 then V2 ← V1
11: Очистим стек L3 ← {∅}
      if t < thr and L2 ≠ {∅} then
          L3 ← L3 + (i_{t+1}, j_{t+1}), (i_{t+1}, j_{t+1}) ∈ L2;  Q_{i_{t+1} j_{t+1}} ← 0
12: if L3 ← {∅} then go to 2
13: return V2
```

**Рис. 8.** Алгоритм идентификации шаблонов отпечатков пальцев

От начальных пар КТ маршруты проходят по другим КТ как по вершинам графов. Лучшая оценка одного из маршрутов есть степень подобия шаблонов.

Согласно алгоритму, маршруты тяготеют к вершинам графов с наиболее похожими гнездами, которые консолидируются. Это близко к работам [16, 19]. Финальная оценка шаблонов находится в виде

$$MS = V_2 / \min(n_1, n_2, len), \tag{15}$$

где $n_1$ — количество КТ запросного шаблона, $n_2$ — ссылочного шаблона, $len$ — заданная длина маршрута. Для шаблонов вне доли БД, определенной величиной $F^{-1}(p)$, положим $MS = 0$. На этом заканчивается второй этап идентификации.

## 3. Вычислительные эксперименты

Для тестирования алгоритма было набрано 8788 изображений из архивов FVC 2000, 2002, 2004 и 2006 годов (только оптические сенсоры) [27]. Из этих БД дубли изображений были обнаружены и удалены. В полученной тестовой БД сохранены изображения хорошего, среднего, плохого качества и с малым пересечением фрагментов изображений.

На рис. 9 показана зависимость вероятности ошибок распознавания $p$ (вероятность ошибки первого рода, FRR — false rejected rate) от числа $n$ (млн. сравнений пар шаблонов в секунду) на процессоре Intel i7-4770@3,4 ГГц на тестовой БД. Ошибка $p$ измеряется в зависимости от вероятности ошибки второго рода (FMR — false match rate) [2, 13, 19]. Верхний график соответствует ошибке первого рода при FMR равной $10^{-5}$ (далее $\mathrm{FMR}_{10000}$), средний — при FMR равной $10^{-3}$ (далее $\mathrm{FMR}_{1000}$), нижний — равным ошибкам первого и второго рода (EER — equal error rate).

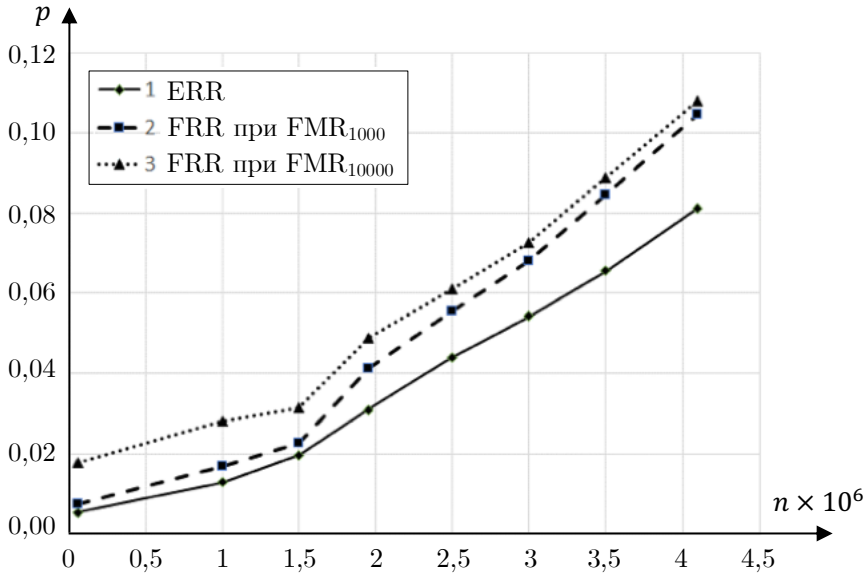



**Рис. 9.** Графики ошибок первого рода при идентификации

Данные результатов измерений ошибок распознавания, по которым построены графики на рис. 9, приведены в табл. 1.

Производительность алгоритма регулировалась величинами допусков, определяющих работу классификаторов всех трех уровней, а также долей ссылочных шаблонов. Минимальные ошибки распознавания получены без применения ускоряющих классификаторов и по всей тестовой БД (всего 55 тыс. сравнений пар шаблонов в секунду, см. табл. 1). Следует обратить внимание на наклон ломанных, который показывает, во сколько раз рост производительности алгоритма опережает рост ошибок распознавания. В экспериментах это отношение варьируется от 4 до 8.

**Таблица 1**

Экспериментальные данные результатов идентификации

| **Сравнений млн/с** | **EER** | **$FMR_{1000}$** | **$FMR_{10000}$** |
|---|---|---|---|
| 0,055 | 0,0053 | 0,0073 | 0,0177 |
| 1 | 0,0128 | 0,0167 | 0,0279 |
| 1,5 | 0,0196 | 0,0225 | 0,0315 |
| 1,95 | 0,0309 | 0,0412 | 0,0491 |
| 2,5 | 0,0441 | 0,0555 | 0,0611 |
| 3 | 0,0543 | 0,0681 | 0,0727 |
| 3,5 | 0,0655 | 0,0845 | 0,0887 |
| 4,1 | 0,0813 | 0,1045 | 0,1078 |

Для сравнения минимальных ошибок распознавания (без ускорителей) с ошибками распознавания других разработчиков из различных стран метод, реализованный в алгоритме FPM, был проверен международным тестом верификации отпечатков пальцев на сайте FVC-ongoing [25]. Наиболее интересные результаты, опубликованные для базы изображений FV-HARD-1.0, представлены в табл. 2. Обработка изображений, на которую опирается алгоритм, изложена в [26]. Таким образом, минимальные ошибки распознавания на рис. 9 соответствуют лидирующим результатам независимого международного тестирования. Необходимо заметить, что для малых ошибок значимый наклон ломанных линий обеспечить очень трудно.

**Таблица 2**

Результаты независимого международного тестирования на сайте FVC-ongoing

| **Алгоритм** | **Время обработки, мс** | **Время сравнения, мс** | **$FMR_{1000}$, %** | **$FMR_{10000}$, %** |
|---|---|---|---|---|
| MM_FV | 855 | 51 | 0,342 | 0,626 |
| FPM | 323 | 26 | 0,952 | 1,227 |
| EMB9300 | 82 | 10 | 1,092 | 1,542 |
| TigerAFIS | 464 | 20 | 1,077 | 1,781 |
| HXKJ | 1413 | 33 | 0,797 | 1,879 |

## Заключение

В статье предлагается метод и алгоритм ускорения идентификации изображений отпечатков пальцев. Он состоит из двух этапов. На первом этапе применяются классификаторы трех уровней, позволяющие ускорить отбор идентичных пар КТ и сократить долю ссылочных шаблонов для второго этапа идентификации. Для этого используются геометрические и топологические характеристики узоров, а также гистограммные оценки. На втором этапе выполняется детальное сравнение шаблонов, но только для малой доли шаблонов из БД. Применение многоуровневых классификаторов и гистограммных оценок позволяет значительно увеличить отношение величины роста производительности алгоритма к величинам роста ошибок идентификации и создать программу коммерческого уровня.

Предложенный метод опирается на топологические события и состояния связей, определяемые для каждой КТ на этапе обработки. Эти признаки сохраняются в шаблоне.

Размер шаблона увеличивается незначительно. Большая часть классификаторов рассчитывается на этапе обработки изображения и используется при идентификации шаблонов в готовом виде. Это позволяет применить однотактовые вычислительные операции при построении списков $L_0$ и $L_1$ и значительно уменьшить время идентификации.

Дальнейшее направление исследований видится в развитии числа уровней классификаторов, проверке алгоритма на отдельной БД FVC-2004-DB1 [27], участии в тестировании на сайте NIST USA, применении набора команд микроархитектуры Intel Core SSE4, во внедрении нейротехнологий в состав действующего алгоритма.

## Литература


1. Bolle R.M., Connel J.Y., Pankanti S., et al. Guide to Biometrics. New York, Springer-Verlag, 2004. 364 p. DOI: 10.1007/978-1-4757-4036-3.
2. Maltoni D., Maio D., Jain A.K., et al. Handbook of Fingerprint Recognition. London, Springer-Verlag, 2009. 494 p. DOI: 10.1007/978-1-84882-254-2.
3. ISO/IEC 19794-2:2011. Information technology – Biometric data interchange formats – Part 2: Finger minutiae data (дата обращения: 23.07.2020).
4. Bae G., Lee H., Hwang S.D., et al. Secure and Robust User Authentication Using Partial Fingerprint Matching // Proceedings of 2018 IEEE International Conference on Consumer Electronics, ICCE. 2018. P. 1–6. DOI: 10.1109/icce.2018.8326078.
5. Hidayat R., Souvanlit K., Bejo A. An Improvement of Minutiae-based Fingerprint Matching: Two Level of Scoring System // Proceedings of 2016 International Symposium on Electronics and Smart Devices, ISESD. 2016. P. 264–267. DOI: 10.1109/ISESD.2016.7886730.
6. Singh P., Kaur L. Fingerprint Feature Extraction Using Morphological Operations // Proceedings of 2015 International Conference on Advances in Computer Engineering and Applications. 2015. P. 764–767. DOI: 10.1109/ICACEA.2015.7164805.
7. Гудков В.Ю. Способ математического описания и идентификации отпечатков пальцев // под ред. член-корр. РАН В.Л. Арлазарова и д.т.н. проф. Н.Е. Емельянова // Обработка изображений и анализ данных: Труды ИСА РАН. М.: ЛИБРОКОМ. 2008. Т. 38. С. 336–356.
8. Liao C.C., Chiu C.T. Fingerprint Recognition with Ridge Features and Minutiae on Distortion // Proceedings of 2016 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP. 2016. P. 2109–2113. DOI: 10.1109/ICASSP.2016.7472049.
9. Barman S., Chattopadhyay S., Samanta D., et al. An Efficient Fingerprint Matching Approach Based on Minutiae to Minutiae Distance Using Indexing With Effectively Lower Time Complexity // Proceedings of 2014 International Conference on Information Technology. 2014. P. 179–183. DOI: 10.1109/ICIT.2014.46.
10. Tran M.H., Duong T.N., Nguyen D.M., et al. A Local Feature Vector for an Adaptive Hybrid Fingerprint Matcher // 2017 International Conference on Information and Communications, ICIC. 2017. P. 249–253. DOI: 10.1109/INFOC.2017.8001668.
11. Гудков В.Ю. Модель гребневого счета на основе топологии дактилоскопическогоизображения // Вестник ЧелГУ. 2011. № 13. С. 99–108.
12. Jiang X., Yau W.Y., Fingerprint Minutiae Matching Based on the Local and Global Structures // Proceedings of the 15th International Conference on Pattern Recognition, ICPR-2000. 2000. Vol. 2. P. 1038–1041. DOI: 10.1109/ICPR.2000.906252.
13. Feng Y., Feng J., Chen X., et al. A Novel Fingerprint Matching Scheme Based on Local Structure Compatibility // Proceedings of the 18th International Conference on Pattern Recognition, ICPR'06. 2006. P. 374–377. DOI: 10.1109/ICPR.2006.137.

14. Cao J., Feng J. A Robust Fingerprint Matching Algorithm Based on Compatibility of Star Structures // Proceedings of the Sixth International Symposium on Multispectral Image Processing and Pattern Recognition, MIPPR 2009. 2009. Vol. 7498. Remote Sensing and GIS Data Processing and Other Applications, 74983X. DOI: 10.1117/12.832357.
15. Ratha N.K., Pandit V.D., Bolle R.M., et al. Robust Fngerprint Authentication Using Local Structure Similarity // Workshop on Applications of Computer Vision. 2000. P. 29–34. DOI: 10.1109/WACV.2000.895399.
16. Chikkerur S., Cartwright A., Govindaraju V. K-plet and CBFS: A graph Based Fingerprint Representation // Proceedings of the International Conference on Biometrics, ICB 2006: Advances in Biometrics. 2006. P. 309–315. DOI: 10.1007/11608288_42.
17. Chen X., Wang L., Li M. An Efficient Graph-Based Algorithm for Fingerprint Representation and Matching // Proceedings of the 3rd International Conference on Multimedia Technology, ICMT 2013. 2013. P. 1019–1029. DOI: 10.2991/icmt-13.2013.125.
18. Leslie S., Sumathi C.P. A Robust Hierarchical Approach to Fingerprint Matching Based on Global and Local Structures // International Journal of Applied Engineering Research. 2018. Vol. 13, no. 7. P. 4730–4739.
19. Capelli R., Ferrara M., Maltoni D. Fingerprint Indexing Based on Minutia Cylinder-Code // IEEE transactions on pattern analysis and machine intelligence. 2011. Vol. 33, no. 5. P. 1051–1057. DOI: 10.1109/TPAMI.2010.228.
20. Zheng F., Yang C. Latent Fingerprint Match using Minutia Spherical Coordinate Code // International Conference on Biometrics, ICB 2015 (Phuket, Thailand, May, 19-22, 2015). 2015. P. 357–362. DOI: 10.1109/ICB.2015.7139061.
21. Tabassi E., Wilson C., Watson C. Fingerprint Image Quality. NIST Internal Report 7151, National Institute for Standards and Technology, 2004. URL: https://www.nist.gov/publications/fingerprint-image-qualitiy (дата обращения: 23.07.2020).
22. Гудков В.Ю., Аркабаев Д.И. Способ сравнения отпечатков папиллярных узоров. РФ Пат. 2331108, МПК G 06 K 9/62, 2008. Бюл. 22.
23. Новиков Ф.А. Дискретная математика для программистов: Учебник для вузов, 3-е изд. СПб: Питер, 2009. 384 с.
24. Warren H.S. Hacker's Delight, 2nd ed. Addison-Wesley Professional, 2018. 512 p.
25. Dorizzi B., Cappelli R., Ferrara M., et al. Fingerprint and On-Line Signature Verification Competitions at ICB 2009 // Proceedings of the International Conference on Biometrics, ICB 2009. 2009. P. 725–732. DOI: 10.1007/978-3-642-01793-3_74.
26. Гудков В.Ю. Методы первой и второй обработки дактилоскопических изображений. Миасс: Геотур, 2009. 237 с.
27. FVC — ongoing: on-line evaluation of fingerprint recognition algorithms. URL: https://biolab.csr.unibo.it/FVCOnGoing/UI/Form/Home.aspx (дата обращения: 23.07.2020).

Гудков Владимир Юльевич, д.ф.-м.н., профессор, кафедра ЭВМ, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

DOI: 10.14529/cmse210103

# ACCELERATED FINGERPRINT IDENTIFICATION METHOD

© 2021 V.J. Gudkov
*South Ural State University (pr. Lenina 76, Chelyabinsk, 454080 Russia)*
*E-mail: diana@sonda.ru*
Received: 11.08.2020

The article outlines a method for accelerated identification of fingerprint images based on templates as image models. They are formed as a result of automatic processing of images. The method is based on the properties of the nearest neighborhoods of minutiae in the form of endings and bifurcations and consists of two stages. At the first stage, each minutia of the query template is compared with each minutia of the reference template from the database and the similarity of such pairs of minutiae are estimated. To speed up computational operations, classes are introduced that allow you quickly accumulate the similarity of minutiae from these two templates in a histogram. Histograms are built for all reference templates from the database and one query template. At the second stage, based on histogram estimates, the most similar templates are selected, the number of which is much less than the size of the database. These templates are compared additionally taking into account the consolidation of minutiae and the compactness of the location of the corresponding pairs of minutiae. Significant acceleration of the identification algorithm is achieved by discarding dissimilar pairs of minutiae at the first stage and pairs of patterns with poor histogram estimates at the second stage. The results of experiments are presented, which are published on the Internet.

*Keywords: fingerprint, identification, minutia, histogram.*

**FOR CITATION**

Gudkov V.J. Accelerated Fingerprint Identification Method. *Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering.* 2021. Vol. 10, no. 1. P. 32–48. (in Russian) DOI: 10.14529/cmse210103.



## References

1. Bolle R.M., Connel J.Y., Pankanti S., et al. Guide to Biometrics. New York, Springer-Verlag, 2004. 364 p. DOI: 10.1007/978-1-4757-4036-3.
2. Maltoni D., Maio D., Jain A.K., et al. Handbook of Fingerprint Recognition. London, Springer-Verlag, 2009. 494 p. DOI: 10.1007/978-1-84882-254-2.
3. ISO/IEC 19794-2:2011. Information technology – Biometric data interchange formats – Part 2: Finger minutiae data (accessed: 23.07.2020).
4. Bae G., Lee H., Hwang S.D., et al. Secure and Robust User Authentication Using Partial Fingerprint Matching. Proceedings of 2018 IEEE International Conference on Consumer Electronics, ICCE. 2018. P. 1–6. DOI: 10.1109/icce.2018.8326078.
5. Hidayat R., Souvanlit K., Bejo A. An Improvement of Minutiae-based Fingerprint Matching: Two Level of Scoring System. Proceedings of 2016 International Symposium on Electronics and Smart Devices, ISESD. 2016. P. 264–267. DOI: 10.1109/ISESD.2016.7886730.
6. Singh P., Kaur L. Fingerprint Feature Extraction Using Morphological Operations. Proceedings of 2015 International Conference on Advances in Computer Engineering and Applications. 2015. P. 764–767. DOI: 10.1109/ICACEA.2015.7164805.

7. Gudkov V.J. Methods for Mathematical Description and Identification of Fingerprints. Editor Arlazarov V.L., Emeljanov N.E. Image Processing and Data Analysis: Proceedings of ISA RSA. LIBROKOM. 2008. Vol. 38. P. 336–356. (in Russian)
8. Liao C.C., Chiu C.T. Fingerprint Recognition with Ridge Features and Minutiae on Distortion. Proceedings of 2016 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP. 2016. P. 2109–2113. DOI: 10.1109/ICASSP.2016.7472049.
9. Barman S., Chattopadhyay S., Samanta D., et al. An Efficient Fingerprint Matching Approach Based on Minutiae to Minutiae Distance Using Indexing With Effectively Lower Time Complexity. Proceedings of 2014 International Conference on Information Technology. 2014. P. 179–183. DOI: 10.1109/ICIT.2014.46.
10. Tran M.H., Duong T.N., Nguyen D.M., et al. A Local Feature Vector for an Adaptive Hybrid Fingerprint Matcher. 2017 International Conference on Information and Communications, ICIC. 2017. P. 249–253. DOI: 10.1109/INFOC.2017.8001668.
11. Gudkov V.J. Ridge Counting Model Based on the Topology of a Fingerprint Image. Bulletin of Chelyabinsk State University. 2011. Vol. 13. P. 99–108. (in Russian)
12. Jiang X., Yau W.Y. Fingerprint Minutiae Matching Based on the Local and Global Structures. Proceedings of the 15th International Conference on Pattern Recognition, ICPR-2000. 2000. Vol. 2. P. 1038–1041. DOI: 10.1109/ICPR.2000.906252.
13. Feng Y., Feng J., Chen X., et al. A Novel Fingerprint Matching Scheme Based on Local Structure Compatibility. Proceedings of the 18th International Conference on Pattern Recognition, ICPR'06. 2006. P. 374–377. DOI: 10.1109/ICPR.2006.137.
14. Cao J., Feng J. A Robust Fingerprint Matching Algorithm Based on Compatibility of Star Structures. Proceedings of the Sixth International Symposium on Multispectral Image Processing and Pattern Recognition, MIPPR 2009. 2009. Vol. 7498. Remote Sensing and GIS Data Processing and Other Applications, 74983X. DOI: 10.1117/12.832357.
15. Ratha N.K., Pandit V.D., Bolle R.M., et al. Robust Fingerprints Authentication Using Local Structure Similarity. Workshop on Applications of Computer Vision. 2000. P. 29–34. DOI: 10.1109/WACV.2000.895399.
16. Chikkerur S., Cartwright A., Govindaraju V. K-plet and CBFS: A graph Based Fingerprint Representation. Proceedings of the International Conference on Biometrics, ICB 2006: Advances in Biometrics. 2006. P. 309–315. DOI: 10.1007/11608288_42.
17. Chen X., Wang L., Li M. An Efficient Graph-Based Algorithm for Fingerprint Representation and Matching. Proceedings of the 3-rd International Conference on Multimedia Technology, ICMT 2013. 2013. P. 1019–1029. DOI: 10.2991/icmt-13.2013.125.
18. Leslie S., Sumathi C.P. A Robust Hierarchical Approach to Fingerprint Matching Based on Global and Local Structures. International Journal of Applied Engineering Research. 2018. Vol. 13, no. 7. P. 4730–4739.
19. Capelli R., Ferrara M., Maltoni D. Fingerprint Indexing Based on Minutia Cylinder-Code. IEEE transactions on pattern analysis and machine intelligence. 2011. Vol. 33, no. 5. P. 1051–1057. DOI: 10.1109/TPAMI.2010.228.
20. Zheng F., Yang C. Latent Fingerprint Match using Minutia Spherical Coordinate Code, International Conference on Biometrics, ICB 2015 (Phuket, Thailand, May, 19-22, 2015). 2015. P. 357–362. DOI: 10.1109/ICB.2015.7139061.
21. Tabassi E., Wilson C., Watson C. Fingerprint Image Quality. NIST Internal Report 7151, National Institute for Standards and Technology, 2004. URL: https://www.nist.gov/publications/fingerprint-image-qualitiy (accessed: 23.07.2020).
22. Gudkov V.J., Arkabaev D.I. Method for Comparing Fingerprints of Papillary Patterns. Patent RF, no. 2331108, G 06 K 9/62, 2008. Vol. 22. (in Russian)

23. Novikov F.A. Discrete Mathematics for Programmers: Textbook for Higher Schools. St. Petersburg, Piter, 2009. 384 p. (in Russian)
24. Warren H.S. Hacker's Delight, 2nd ed., Addison-Wesley Professional, 2018. 512 p.
25. Dorizzi B., Cappelli R., Ferrara M., et al. Fingerprint and On-Line Signature Verification Competitions at ICB 2009. Proceedings of the International Conference on Biometrics, ICB 2009. 2009. P. 725–732. DOI: 10.1007/978-3-642-01793-3_74.
26. Gudkov V.J. Methods of the First and Second Processing of Fingerprint Images. Miass, Publishing of the Geotour, 2009. 237 p. (in Russian)
27. FVC – ongoing: on-line evaluation of fingerprint recognition algorithms. URL: https://biolab.csr.unibo.it/FVCOnGoing/UI/Form/Home.aspx (accessed: 23.07.2020).

**УДК 004.272.2, 530.145** **DOI: 10.14529/cmse210104**

# СРАВНЕНИЕ ПРОИЗВОДИТЕЛЬНОСТИ ПАКЕТОВ СИМУЛЯЦИИ КВАНТОВЫХ ВЫЧИСЛЕНИЙ QUEST И INTEL-QS*

**© 2021 А.В. Линев[1], П.Е. Ведруков[1], Д.С. Куландин[1], И.Б. Мееров[1], С. Денисов[1,2]**

[1]*Нижегородский государственный университет им. Н.И. Лобачевского (603950 Нижний Новгород, пр. Гагарина, д. 23),*

[2]*Столичный университет Осло (NO-0130 Осло, Норвегия, ул. Olavs plass, P.O. Box 4)*

*E-mail: alin@unn.ru, pavelvedrukov@mail.ru, kulandin08@gmail.com, meerov@vmk.unn.ru, sergiyde@oslomet.no*

Поступила в редакцию: 14.09.2020

В ближайшем будущем появятся квантовые компьютеры, пригодные для практического использования. Разработка квантовых алгоритмов может проводиться с использованием классических компьютеров и специализированного программного обеспечения, позволяющего симулировать работу квантовой схемы. Результаты моделирования могут использоваться для анализа алгоритма, а также способствуют ко-дизайну при разработке квантовых архитектур. Однако при планировании и выполнении численных экспериментов необходимо понимать возможности симуляторов и ограничения на параметры квантовой схемы, накладываемые характеристиками доступных классических вычислительных ресурсов. В работе представлены результаты вычислительных экспериментов по симуляции работы квантовых схем на идеальном квантовом компьютере с использованием пакетов QuEST и Intel-QS, а также собственной «наивной» реализации. Показаны ограничения на размер моделируемой квантовой системы $N$ при использовании вычислительных систем различного класса — виртуальной машины, вычислительного сервера, вычислительного сервера с графическим ускорителем, суперкомпьютера (маскимальный достигнутый размер $N = 33$). Приведены характеристики производительности и масштабируемости рассматриваемых реализаций на общей и распределенной памяти (наблюдаемая эффективность масштабирования — 30 % и 70 % соответственно). Для пакета QuEST и собственной реализации представлена производительность при использовании графических сопроцессоров.

*Ключевые слова: вычислительная квантовая физика, квантовые алгоритмы, высокопроизводительные вычисления, GPGPU, QuEST, Intel-QS.*

**ОБРАЗЕЦ ЦИТИРОВАНИЯ**

Линев А.В., Ведруков П.Е., Куландин Д.С., Мееров И.Б., Денисов С. Сравнение производительности пакетов симуляции квантовых вычислений QuEST и Intel-QS // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2021. Т. 10, № 1. С. 49–61. DOI: 10.14529/cmse210104.

## Введение

Важной составляющей разработки и анализа квантовых алгоритмов [1] является симуляция их выполнения на классических компьютерах [2]. Разработаны десятки приложений [3], которые позволяют описывать квантовые схемы с использованием специализированных языков квантовых вычислений [4–7], выполнять симуляцию работы квантового алгоритма [8–11] и даже поддерживать полный цикл разработки, включающий описание алгоритма, его оптимизацию, отображение на конкретную архитектуру кван-

*Статья рекомендована к публикации программным комитетом Международной конференции «Суперкомпьютерные дни в России – 2020».

тового компьютера, верификацию, симуляцию работы полученной квантовой схемы и оценку ее производительности [12].

Симуляция возможна только для квантовых систем, состоящих из малого числа кубитов. Даже использование суперкомпьютеров позволяет симулировать работу квантовых систем, состоящих всего лишь из 38 [8], 42 [9], 45 [13], 49 [14], возможно, 53 [15], а в специальных случаях — до 64 кубитов [16], в то время как Google уже представляет результаты экспериментов на реальном квантовом компьютере общего назначения из 53 кубитов [17], анонсирует квантовый процессор из 72 кубитов [18], а D-Wave systems — специализированную систему из 2000 кубитов, реализующую алгоритм квантового отжига [19].

Несмотря на различную демонстрируемую производительность [20], разнообразие приложений позволяет выбрать наиболее подходящее для конкретной задачи и имеющихся вычислительных ресурсов. При выборе программных средств моделирования квантовых вычислений мы провели ряд экспериментов для определения производительности различных пакетов по выполнению симуляции работы квантовой схемы на идеальном квантовом компьютере [2] на системах различного класса — от настольного компьютера до вычислительного кластера. Мы рассматривали пакеты, которые, с одной стороны, позволяют оценить потребление ресурсов при выполнении отдельных шагов моделирования, с другой стороны — имеют параллельные версии для общей и распределенной памяти. Множество пакетов не удовлетворяют одному из данных требований (IBM Qiskit, LIQ$Ui|\rangle$, ProjectQ, DG-M и т.д.) или недоступны для свободного использования [21]. Мы выбрали для анализа два пакета, удовлетворяющие нашим требованиям и положительно характеризуемые в публикациях — QuEST и Intel-QS. Полученные результаты будут использованы для формирования набора инструментов, которые будут использоваться для разработки и анализа квантовых алгоритмов и обучения студентов и аспирантов ННГУ им. Н.И. Лобачевского.

Работа построена следующим образом. В разделе 1 приведено описание тестовой задачи. В разделе 2 — описание проведенных экспериментов. Раздел 3 содержит характеристики вычислительных систем. Раздел 4 — результаты экспериментов и комментарии к ним. В последнем разделе представлено обобщение результатов.

## 1. Тестовая задача

Квантовый компьютер хранит информацию в виде кубитов, которые можно считать квантовой версией битов. Базовые операции над кубитами включают квантовые гейты — аналоги классических битовых операций — и операцию измерения кубита.

Мы будем рассматривать симуляцию идеального квантового компьютера, в котором отсутствуют шумы и декогеренция, присущие реальным устройствам. В этом случае состояние квантового компьютера с $N$ кубитами описывается квантовым состоянием (также называемым «волновой функцией») системы кубитов, представленным в виде вектора комплексных чисел размером $2^N$ $(a_i)$, $i \in \{0,1\}^N$, $a_i \in \mathbb{C}$, удовлетворяющего условию нормировки $\sum_{i\in\{0,1\}^N}|a_i|^2 = 1$.

Действие квантового гейта в общем случае вызывает изменение всех $2^N$ элементов вектора, согласно природе физического процесса описывается унитарным преобразованием и может быть представлено унитарной матрицей. Квантовые компьютеры реализуют набор базовых гейтов, управляющих состоянием одного–трех кубитов и представляемых сильно разреженной матрицей преобразования системы в целом (комбинация кронекеровских произведений матрицы воздействия на кубиты и единичных матриц). Для повышения производительности симуляция квантовых гейтов обычно выполняется

не универсальным умножением разреженной матрицы на плотный вектор, а реализацией специализированного алгоритма, уникального для каждого типа квантового гейта.

В тестовом примере мы использовали однокубитовый гейт преобразования Адамара ($H$), двухкубитовый гейт контролируемого отрицания ($CNOT$) и вычисление вероятности измерения нулевого кубита в состояниях $|0>$ и $|1>$ ($P_0$ и $P_1$ соответственно), которые описываются следующими матрицами и формулами:

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}, C\text{--}NOT = \begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{pmatrix}, P_0 = \sum_{i_1,\ldots,i_{N-1}} a_{0,i_1,\ldots,i_{N-1}}, P_1 = \sum_{i_1,\ldots,i_{N-1}} a_{1,i_1,\ldots,i_{N-1}}.$$

Первый из них воздействует на один кубит и требует вычислений с использованием и обновлением всех элементов квантового состояния, второй — воздействует на два кубита и не предполагает выполнения вычислений. Для выполнения одной операции Адамара для кубитной системы требуется выполнение $2^N$ умножений и $2^N$ сложений комплексных чисел, т.е. $8 * 2^N$ FLOP. При этом требуется считать из памяти $16 * 2^N$ байт и записать в память $16 * 2^N$ байт (например, для $N = 30$ и 50 операций — 400 GFLOP и 32 ГБ). Для выполнения одной операции контролируемого отрицания требуется выполнить 0 FLOP, считать из памяти $8 * 2^N$ байт и записать $8 * 2^N$ байт (для $N = 30$ и 50 операций — 0 GFLOP и 16 ГБ). Фактическое число операций и объем передаваемых данных зависят от конкретной архитектуры вычислительной системы.

При выполнении экспериментов производилась симуляция выполнения на системе из $N$ кубитов квантовой схемы, состоящей из 50 операций Адамара и 50 операций контролируемого отрицания. Рассматривались следующие этапы симуляции:

- инициализация программного окружения ПО симуляции;
- создание системы из N кубитов;
- последовательное применение оператора Адамара к кубитам 0, 1,..., $N-1$, 0, 1,... (далее по циклу); общее число операций — 50;
- последовательное применение оператора CNOT к парам кубитов (control=0, target=1), (1,2),..., $(N-2, N-1)$, $(N-1, 0)$, (0,1),... (далее по циклу); общее число операций — 50;
- вычисление вероятностей нахождения нулевого кубита в состояниях $|0>$ и $|1>$;
- удаление системы кубитов;
- завершение программного окружения ПО симуляции.

Оценивались время работы и потребление оперативной памяти на всех этапах.

## 2. Описание экспериментов

Для оценки характеристик программных средств симуляции работы квантовой схемы мы использовали 3 реализации:

- QuEST — симулятор квантовых компьютеров с поддержкой параллельного выполнения на общей и распределенной памяти [22], при работе на одном узле способен использовать один графический ускоритель;
- Intel Quantum Simulator (Intel-QS) — симулятор, оптимизированный для многоядерных архитектур и вычислительных кластеров [23];
- «наивная» реализация, выполненная нами (не использует векторизацию, оптимизацию использования кеша и т.д.); используется для оценки производительности качественных реализаций.

Все пакеты компилировались и запускались с параметрами по умолчанию.

Проводились следующие виды экспериментов.

1. Определение зависимости времени работы и объема используемой оперативной памяти от числа кубитов. Выполнялись тесты для числа кубитов от 2 до максимально возможного на вычислительной системе. На многоядерных системах использовались все ядра. Измерялись время работы каждого этапа и объем памяти, используемый процессом в конце этапа.

2. Масштабируемость на общей памяти. Использовалось 30 кубитов и число потоков OpenMP от 1 до максимально возможного на вычислительной системе. Измерялись время работы каждого этапа и максимальный объем памяти, используемый процессом.

3. Масштабируемость на распределенной памяти. Использовалось 30 кубитов и число MPI-процессов от 1 до максимально возможного на вычислительной системе. Измерялись время работы каждого этапа и объем памяти, используемый одним процессом в конце этапа.

4. Определение зависимости времени работы от числа кубитов при использовании графического ускорителя. Измерялось полное время симуляции.

На различных вычислительных системах выполнялся различный состав экспериментов. На системах 1–3 (см. раздел 3) проводились все возможные для них виды экспериментов. На системе 4 проводился только четвертый эксперимент.

## 3. Вычислительные системы

Для оценки возможности использования программных средств симуляции работы квантовой схемы мы использовали 4 вычислительные системы, существенно отличающиеся с точки зрения производительности.

**1. Виртуальная машина (VM), запущенная на рабочей станции.**

– Конфигурация: Intel Core i3-7100 (Kaby Lake), 3,90 ГГц, 1 ядро, 1 ГБ DDR4-2132 (1066 МГц).
– Производительность: 62,4 GFLOPS/ядро, 17 064 МБ/с * 2 канала.
– Интерконнект: предоставлямый средством виртуализации.
– Операционная система: CentOS 7.4.1708.
– Компилятор: gcc 4.8.5-39, gcc 7.3.1-5.
– MPI: mpich-3.0.

**2. Dell PowerEdge R815 (64-ядерный сервер).**

– Конфигурация: 4xAMD Opteron 6366 HECPU, 3,60 ГГц, 4x16 ядер, 256 ГБ DDR3-1600.
– Производительность: 921,6 GFLOPS (14,4/ядро), 12 800 МБ/с * 4 канала* 4 ЦП (До 1600 МТ/с, 140 ГБ/с согласно описанию платформы).
– Интерконнект: 4xBroadcom 5709C (Gigabit Ethernet).
– Операционная система: Ubuntu 18.04.1 LTS.
– Компилятор: gcc7.3.0-27.
– MPI: OpenRTE 3.0.0 (openmpi-based)

**3. Intel Endeavour (вычислительный кластер)**

– Конфигурация: 2xIntel Xeon Platinum 8260L (Cascade Lake), 2,40 ГГц, 2x24 физических ядер, 256 ГБ DDR4-2933.
– Производительность: 2 304 GFLOPS (48/ядро), 23 466 МБ/с * 6 каналов * 2 ЦП.
– Интерконнект: Mellanox Technologies MT28908 Family (HDR Infiniband).
– Операционная система: CentOS Linux 7 (Core).
– Компилятор: icc 19.0.5.281 20190815.
– MPI: Intel(R) MPI Library for Linux OS, Version 2019 Update 5 Build 20190806.

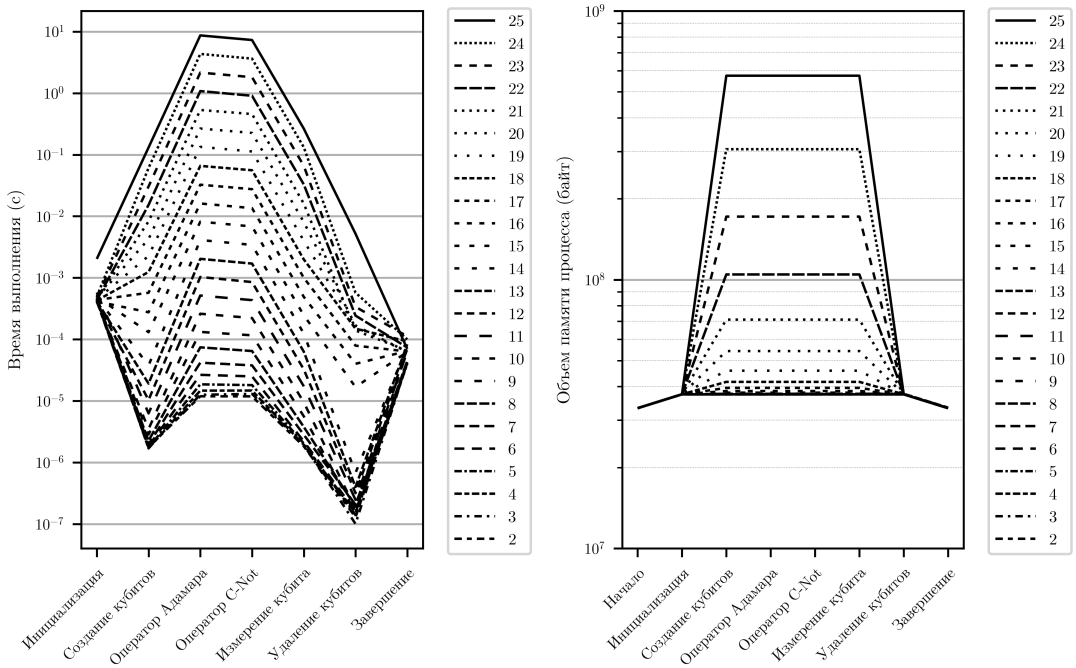
**4. DGX-2 (сервер с графическими ускорителями).**

- Конфигурация: 2xIntel Xeon Platinum 8168 (Skylake), 2,70 ГГц, 2x24 физических ядер, 1,5 ТБ DDR4-2666.
- Производительность: 2 918,4 GFLOPS (60,8/ядро), 21 333 МБ/с * 6 каналов * 2 ЦП.
- Графические сопроцессоры: 16xNVIDIA Tesla V100-SXM3-32GB, 16*32 ГБ HBM2.
- Производительность графических сопроцессоров: 124,8 TFLOPS DP (7,8/GPU), 900 ГБ/с * 16 GPU.
- Интерконнект: 8x Mellanox Technologies MT27800 Family (100 Гб/с Infiniband).
- Операционная система: Ubuntu 18.04.5 LTS (Bionic Beaver).
- Компилятор: gcc 7.5.0, Cuda compilation tools, release 10.1, V10.1.243.
- MPI: нет.

# 4. Результаты экспериментов

## 4.1. Виртуальная машина

В виртуальной машине проводился только первый эксперимент для квантовых систем размером от 2 до 25 кубитов. На рис. 1 приведены результаты для пакета QuEST. Здесь и далее все графики зависимости измеряемых значений от числа кубитов приведены в логарифмической системе координат.




а) Время выполнения этапов б) Объем памяти на этапах

**Рис. 1.** Зависимость времени работы этапов алгоритма и объема используемой оперативной памяти от числа кубитов; пакет QuEST; виртуальная машина

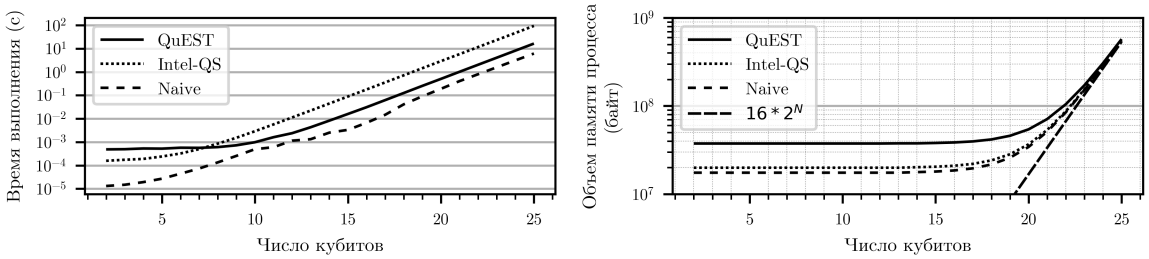
Наблюдается экспоненциальная зависимость времени выполнения содержательных этапов от числа кубитов (отличия по времени для малого числа кубитов могут быть связаны с особенностями архитектуры системы, например, с размещением всех данных в кеше). Объем потребляемой памяти приближенно определяется как сумма константы (код и данные за исключением квантового состояния) и $16 * 2^N$ байт (размер вектора квантового состояния).

Время выполнения операций инициализации и завершения не зависит от числа кубитов. Удаление кубита требует на 2–4 порядка меньше времени, чем остальные содержательные операции. Создание и измерение кубита требует в несколько десятков раз

меньше времени, чем выполнение операторов. Оператор *CNot* выполняется в 2–3 раза быстрее оператора Адамара.

Для других экспериментов и типов архитектур в большинстве случаев эти соотношения качественно сохраняются, поэтому далее мы их приводить не будем.

На рис. 2 приведены результаты экспериментов для различных пакетов. Используется суммарное время выполнения всех этапов и максимальное потребление оперативной памяти.




а) Общее время расчета б) Необходимый объем памяти

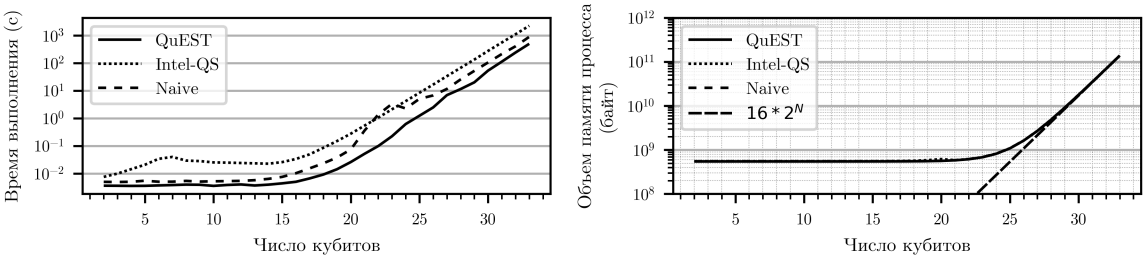
**Рис. 2.** Выполнение расчетов для различного числа кубитов; виртуальная машина

Экспоненциальная зависимость времени выполнения и объема потребляемой памяти от числа кубитов сохраняется, причем объем памяти фактически определяется размером вектора квантового состояния.

«Наивная» реализация показала более высокую производительность, чем другие пакеты. Таким образом, при моделировании простых квантовых схем собственные реализации могут показывать производительность, сравнимую с универсальными профессиональными пакетами.

## 4.2. Сервер Dell PowerEdge R815

На этом 64-ядерном сервере проводились эксперименты 1 и 2. На рис. 3 приведены результаты первого эксперимента для квантовых систем размером от 2 до 33 кубитов.



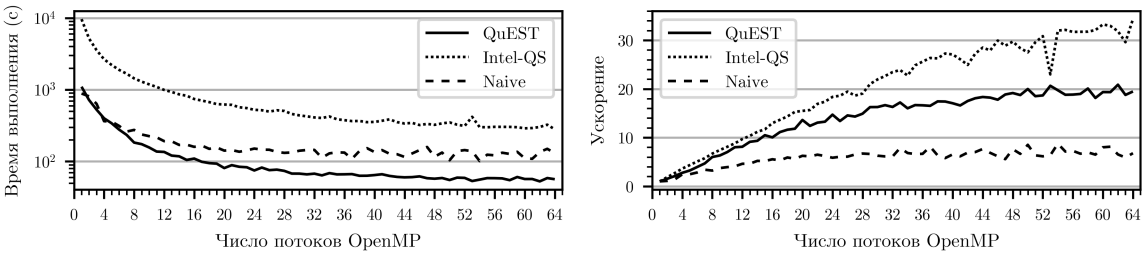

а) Общее время расчета б) Необходимый объем памяти

**Рис. 3.** Выполнение расчетов для различного числа кубитов; Dell PowerEdge R815

Скорость работы пакетов может отличаться в 4–5 раз. Низкая производительность Intel-QS по-видимому объясняется оптимизацией его кода для архитектур и компиляторов Intel.

Результаты эксперимента 2, оценивающего сильную масштабируемость при распараллеливании на общей памяти, приведены на рис. 4.

Эффективность распараллеливания пакета QuEST, показавшего лучшие результаты на данной системе, составляет около 30 %. Базовые квантовые гейты реализуются алгоритмами с низкой арифметической интенсивностью, поэтому их производительность ограничивается возможностями оперативной памяти, а не числом одновременно работающих потоков. Для более сложных схем продемонстрирована возможность достижения эффективности распараллеливания более 80 % и производительности в 47 % от теоретической [14].

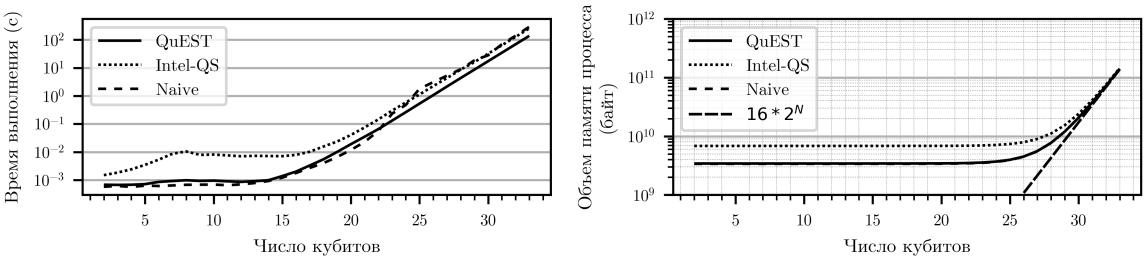



а) Общее время расчета б) Ускорение на общей памяти

**Рис. 4.** Производительность пакетов на системе с общей памятью; Dell PowerEdge R815

### 4.3. Вычислительный кластер Intel Endeavour

На вычислительном кластере Intel Endeavour проводились эксперименты 1–3. На рис. 5 приведены результаты первого эксперимента для квантовых систем размером от 2 до 33 кубитов.



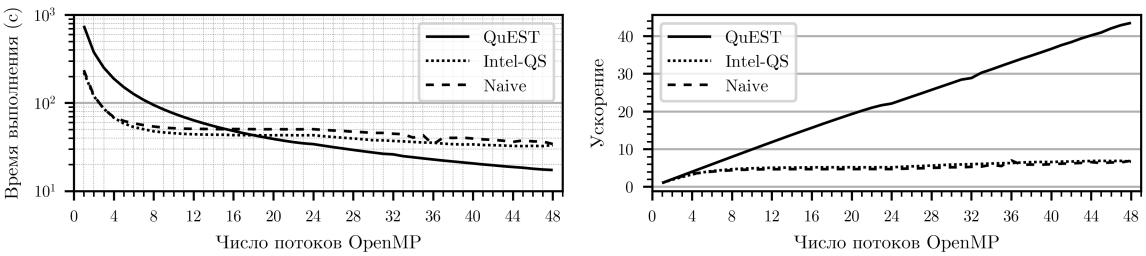

а) Общее время расчета б) Необходимый объем памяти

**Рис. 5.** Выполнение расчетов для различного числа кубитов; 1 узел Intel Endeavour

Для числа кубитов больше 20 скорость работы различных реализаций отличается не более чем в 2 раза.

На рис. 6 показаны результаты эксперимента 2, оценивающего сильную масштабируемость при распараллеливании на общей памяти.

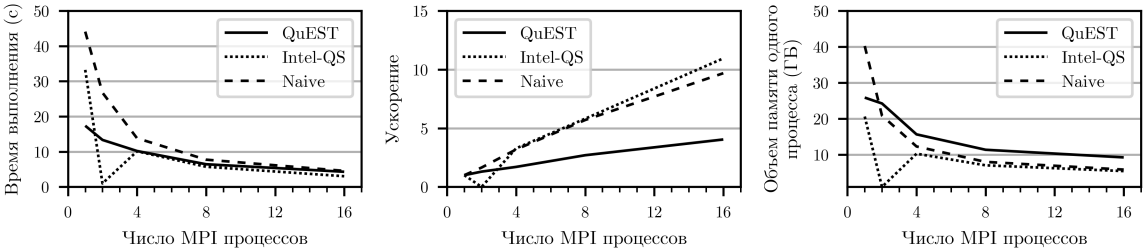
QuEST показывает отличную эффективность распараллеливания (≈90 %), остальные пакеты — очень низкую (≈15 %). Для «наивной» реализации это объясняется отсутствием оптимизаций. Для Intel-QS — высокой производительностью однопоточной реализации и однопоточной реализацией операции вычисления вероятности нахождения кубита в одном из состояний. Без учета этой операции эффективность масштабирования Intel-QS составляет ≈23 %, а время выполнения при максимальном числе потоков больше чем у QuEST всего на 9 %.




а) Общее время расчета б) Ускорение на общей памяти

**Рис. 6.** Производительность пакетов на системах с общей памятью; 1 узел Intel Endeavour

На рис. 7 показаны результаты эксперимента 3 по изучению сильной масштабируемости при распараллеливании на распределенной памяти. Intel-QS не отработал при запуске с одним и двумя MPI-процессами. В целях оценки масштабируемости, для одного процесса приведены данные версии для общей памяти.




а) Общее время расчета б) Ускорение в) Объем памяти

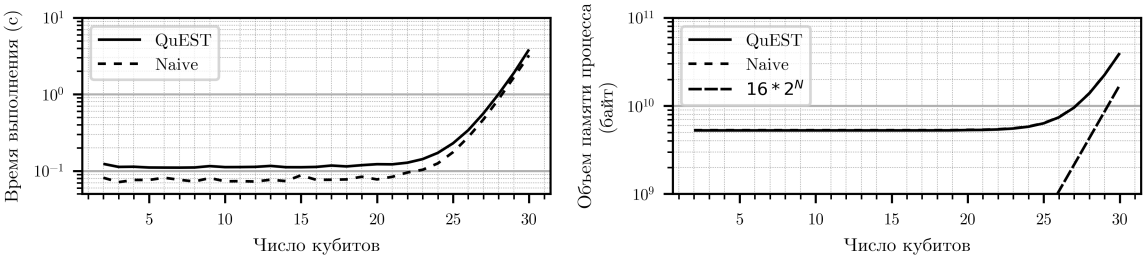
**Рис. 7.** Характеристики выполнения пакетов на системе с распределенной памятью; Intel Endeavour

Все реализации показывают сравнимую производительность, но QuEST показывает в 2,5 раза худшую масштабируемость. Время выполнения эксперимента слишком мало для точной оценки производительности, но при увеличении числа кубитов пакеты на доступных нам ресурсах перестают отрабатывать корректно для многих сочетаний параметров эксперимента. Эффективность QuEST существенно отличается от заявляемой авторами; возможно, необходимо подбирать параметры его запуска или использовать более сложные квантовые схемы.

Объем памяти, используемой одним MPI-процессом, уменьшается примерно пропорционально их числу. Удвоение числа процессов позволяет сократить объем памяти каждого процесса примерно в два раза или использовать тот же объем для моделирования системы размером на один кубит больше.

## 4.4. Сервер с графическими ускорителями DGX-2

На данном сервере проводился эксперимент 4. Из рассматриваемых пакетов только QuEST поддерживает использование графических ускорителей и способен использовать только один ускоритель, поэтому мы выполнили «наивную» реализацию для одного GPGPU и сравнили производительность. На рис. 8 приведены результаты четвертого эксперимента для квантовых систем размером от 2 до 30 кубитов (30 — максимальный размер системы кубитов, состояние которой можно разместить в памяти одного GPU).




а) Общее время расчета б) Необходимый объем памяти ЦП

**Рис. 8.** Выполнение расчетов для различного числа кубитов; 1 GPU DGX-2

Производительность QuEST и «наивной» реализации сопоставимы, объем используемой памяти ЦП отличается менее чем на 1 %.

Время работы QuEST для 27–30 кубитов примерно в 4 раза меньше, чем на одном узле Intel Endeavour при использовании максимального числа потоков. В нашем экспе-

рименте скорость вычислений ограничена производительностью работы с памятью, и отличие показателей ЦП и GPU примерно соответствует отношению теоретической пропускной способности памяти использованных систем, а полученные фактические значения говорят о возможности дополнительной оптимизации работы с памятью. Время работы наивной реализации для GPU меньше в 9–11 раз, но это объясняется низкой производительностью параллельной версии для ЦП. Можно предположить, что использование нескольких GPU будет показывать масштабируемость, сравнимую с MPI-версиями, а в случае использования нескольких GPU одного узла — даже более высокую, что позволит моделировать на сервере DGX-2 системы из 32–33 кубитов с производительностью, сопоставимой с производительностью более 50 двухпроцессорных узлов без GPU.

## Заключение

Результаты проведенных экспериментов позволяют сделать следующие выводы относительно возможностей симуляции работы квантовой схемы на идеальном квантовом компьютере с использованием различных вычислительных систем.

- Время моделирования экспоненциально зависит от числа кубитов.
- Объем необходимой памяти экспоненциально зависит от числа кубитов и будет являться основным ограничивающим фактором при его увеличении.
- Масштабируемость на общей памяти имеет среднюю эффективность (≈30 %), но известно, что на более сложных квантовых схемах можно достичь эффективности более 80 %.
- Эффективность масштабируемости на распределенной памяти достигает 70 %.
- Использование распределенной памяти позволяет увеличить размер моделируемых систем. При удвоении числа узлов можно симулировать систему размером на 1 кубит больше.
- Графические сопроцессоры могут обеспечить значительный рост производительности (8–9 раз по сравнению с ЦП), но критически ограничены объемом установленной на них оперативной памяти.
- Переход от использования виртуальной машины с 1 Гб оперативной памяти к суперкомпьютеру «Ломоносов-2» [24] позволит увеличить размер моделируемой системы с 25 до 40 кубитов, что не имеет принципиального значения при разработке, отладке и тестировании квантовых схем.
- «Наивные» реализации могут быть использованы для моделирования простых квантовых схем и в целях обучения. Исследование больших и сложных схем может потребовать профессиональных пакетов.
- В наших экспериментах Intel-QS и QuEST показали сравнимую производительность, лучший результат на распределенной памяти достигнут Intel-QS.

Полученные результаты будут использованы для формирования набора инструментов, которые будут использоваться для разработки и анализа квантовых алгоритмов и обучения студентов и аспирантов ННГУ им. Н.И. Лобачевского.

*Исследования выполнены при поддержке гранта РНФ № 19-72-20086 с использованием вычислительных ресурсов СК Intel Endeavor и Simula Research Laboratory.*

## Литература


1. Mermin N.D. Quantum computer science: an introduction. Cambridge University Press, 2007. 233 p.
2. Trieu D.B. Large-scale simulations of error prone quantum computation devices. Forschungszentrum Jülich, 2010. Vol. 2. 173 p.
3. Wiki Q. List of QC simulators. 2015. URL: https://quantiki.org/wiki/list-qc-simulators (дата обращения: 01.09.2020).
4. Green A.S., Lumsdaine P.L., Ross N.J., et al. Quipper: a scalable quantum programming language // Proceedings of the 34th ACM SIGPLAN conference on Programming language design and implementation. 2013. P. 333–342. DOI: 10.1145/2491956.2462177.
5. Cross A.W., Bishop L.S., Smolin J.A., et al. Open quantum assembly language // arXiv preprint. 2017. arXiv:1707.03429.
6. Svore K., Geller A., Troyer M., et al. Q# Enabling Scalable Quantum Computing and Development with a High-level DSL // Proceedings of the Real World Domain Specific Languages Workshop 2018. 2018. P. 1–10. DOI: 10.1145/3183895.3183901.
7. Abhari A.J., Faruque A., Dousti M.J., et al. Scaffold: Quantum programming language. Princeton Univ NJ Dept of Computer Science, 2012. 43 p.
8. Jones T., Brown A., Bush I., et al. Quest and high performance simulation of quantum computers // Scientific reports. 2019. Vol. 9, no. 1. P. 1–11. DOI: 10.1038/s41598-019-47174-9.
9. Guerreschi G.G., Hogaboam J., Baruffa F., et al. Intel Quantum Simulator: A cloud-ready high-performance simulator of quantum circuits // Quantum Science and Technology. 2020. Vol. 5, no. 3. P. 034007. DOI: 10.1088/2058-9565/ab8505.
10. Smelyanskiy M., Sawaya N.P.D., Aspuru-Guzik A. qHiPSTER: the quantum high performance software testing environment // arXiv preprint. 2016. arXiv:1601.07195.
11. Aleksandrowicz G., Alexander T., Barkoutsos P., et al. Qiskit: An open-source framework for quantum computing. 2019. DOI: 10.5281/zenodo.2562110.
12. Amy M., Gheorghiu V. staq–A full-stack quantum processing toolkit // Quantum Science and Technology. 2020. Vol. 5, no. 3. P. 034016. DOI: 10.1088/2058-9565/ab9359.
13. Häner T., Steiger D.S. 5 petabyte simulation of a 45-qubit quantum circuit // Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. 2017. P. 1–10. DOI: 10.1145/3126908.3126947.
14. Pednault E., Gunnels J.A., Nannicini G., et al. Breaking the 49-qubit barrier in the simulation of quantum circuits // arXiv preprint. 2017. arXiv:1710.05867.
15. Pednault E., Gunnels J., Maslov D., et al. On "Quantum Supremacy". 2019. URL: https://www.ibm.com/blogs/research/2019/10/on-quantum-supremacy (дата обращения: 01.09.2020).
16. Chen Z.Y., Zhou Q., Xue C., et al. 64-qubit quantum circuit simulation // Science Bulletin. 2018. Vol. 63, no. 15. P. 964–971. DOI: 10.1016/j.scib.2018.06.007.
17. Arute F., Arya K., Babbush R., et al. Quantum supremacy using a programmable superconducting processor // Nature. 2019. Vol. 574, no. 7779. P. 505–510. DOI: 10.1038/s41586-019-1666-5.
18. Google AI Blog: A Preview of Bristlecone, Google's New Quantum Processor. URL: https://ai.googleblog.com/2018/03/a-preview-of-bristlecone-googles-new.html (дата обращения: 01.09.2020).
19. The D-Wave 2000Q™ System. URL: https://www.dwavesys.com/d-wave-two-system (дата обращения: 01.09.2020).

20. de Avila A.B., Reiser R.H., Pilla M.L., et al. State-of-the-art quantum computing simulators: Features, optimizations, and improvements for D-GM // Neurocomputing. 2020. Vol. 393. P. 223–233. DOI: 10.1016/j.neucom.2019.01.118.
21. Häner T., Steiger D.S., Smelyanskiy M., et al. High performance emulation of quantum circuits // Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC'16. IEEE, 2016. P. 866–874. DOI: 10.1109/SC.2016.73.
22. GitHub — QuEST-Kit/QuEST: A multithreaded, distributed, GPU-accelerated simulator of quantum computers. URL: https://github.com/QuEST-Kit/QuEST (дата обращения: 01.09.2020).
23. GitHub — iqusoft/intel-qs: High-performance simulator of quantum circuits. URL: https://github.com/iqusoft/intel-qs (дата обращения: 01.09.2020).
24. Voevodin Vl.V., Antonov A.S., Nikitenko D.A., et al. Supercomputer Lomonosov-2: large scale, deep monitoring and fine analytics for the user community // Supercomputing Frontiers and Innovations. 2019. Vol. 6, no. 2. P. 4–11. DOI: 10.14529/jsfi190201.

Линев Алексей Владимирович, заведующий лабораторией, кафедра программной инженерии, Нижегородский государственный университет им. Н.И. Лобачевского (национальный исследовательский университет) (Нижний Новгород, Российская Федерация)

Ведруков Павел Евгеньевич, студент, Институт информационных технологий, математики и механики, Нижегородский государственный университет им. Н.И. Лобачевского (национальный исследовательский университет) (Нижний Новгород, Российская Федерация)

Куландин Денис Сергеевич, студент, Институт информационных технологий, математики и механики, Нижегородский государственный университет им. Н.И. Лобачевского (национальный исследовательский университет) (Нижний Новгород, Российская Федерация)

Мееров Иосиф Борисович, к.т.н., доцент, кафедра математического обеспечения и суперкомпьютерных технологий, Нижегородский государственный университет им. Н.И. Лобачевского (национальный исследовательский университет) (Нижний Новгород, Российская Федерация)

Денисов Сергей, к.ф.-м.н., профессор, кафедра прикладной математики, Нижегородский государственный университет им. Н.И. Лобачевского (национальный исследовательский университет) (Нижний Новгород, Российская Федерация), доцент, факультет компьютерных наук, Столичный университет Осло (Осло, Норвегия)

DOI: 10.14529/cmse210104

# QUEST AND INTEL-QS QUANTUM COMPUTATION SIMULATION PACKAGES PERFORMANCE COMPARISON

© 2021 A.V. Liniov[1], P.E. Vedrukov[1], D.S. Kulandin[1], I.B. Meyerov[1], S. Denisov[1,2]

[1]*Lobachevsky State University of Nizhny Novgorod (pr. Gagarina 23, Nizhnij Novgorod, 603950 Russia),*

[2]*Oslo Metropolitan University (P.O. Box 4, st. Olavs plass, NO-0130 Oslo, Norway)*

*E-mail: alin@unn.ru, pavelvedrukov@mail.ru, kulandin08@gmail.com, meerov@vmk.unn.ru, sergiyde@oslomet.no*

Received: 14.09.2020

In the nearest future quantum computers will be suitable for practical use. The development of quantum algorithms can be carried out using classical computers and specialized software that allows simulating of a quantum circuit functioning. Simulation results can be used to analyze the algorithm and also contribute to co-design when developing quantum architectures. However, when planning and performing numerical experiments, it is necessary to understand the capabilities of simulators and the limitations on the parameters of the quantum circuit imposed by the characteristics of the available classical computational resources (computers). This paper presents the results of computational experiments on simulating the operation of quantum circuits on an ideal quantum computer using the QuEST and Intel-QS packages, as well as our own "naïve" implementation. Restrictions on the size of a simulated quantum system $N$ are shown when using computing systems of various classes — a virtual machine, a computing server, a computing server with a graphics accelerator (GPU), a supercomputer (the maximum achieved size is $N = 33$). The performance and scalability characteristics of the considered implementations on shared and distributed memory are given (the observed scaling efficiency is 30 % and 70 %, respectively). For the QuEST package and our own implementation the performance is presented for systems with graphics accelerator (GPU).

*Keywords: computational quantum physics, quantum algorithms, high-performance computing, GPGPU, QuEST, Intel-QS.*

**FOR CITATION**

Liniov A.V., Vedrukov P.E., Kulandin D.S., Meyerov I.B., Denisov S. QuEST and Intel-QS Quantum Computation Simulation Packages Performance Comparison. *Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering.* 2021. Vol. 10, no. 1. P. 49–61. (in Russian) DOI: 10.14529/cmse210104.



## References

1. Mermin N.D. Quantum computer science: an introduction. Cambridge University Press, 2007. 233 p.
2. Trieu D.B. Large-scale simulations of error prone quantum computation devices. Forschungszentrum Jülich, 2010. Vol. 2. 173 p.
3. Wiki Q. List of QC simulators. 2015. URL: https://quantiki.org/wiki/list-qc-simulators (accessed: 01.09.2020).
4. Green A.S., Lumsdaine P.L., Ross N.J., et al. Quipper: a scalable quantum programming language. Proceedings of the 34th ACM SIGPLAN conference on Programming language design and implementation. 2013. P. 333–342. DOI: 10.1145/2491956.2462177.

5. Cross A.W., Bishop L.S., Smolin J.A., et al. Open quantum assembly language. arXiv preprint. 2017. arXiv:1707.03429.
6. Svore K., Geller A., Troyer M., et al. Q# Enabling Scalable Quantum Computing and Development with a High-level DSL. Proceedings of the Real World Domain Specific Languages Workshop 2018. 2018. P. 1–10. DOI: 10.1145/3183895.3183901.
7. Abhari A.J., Faruque A., Dousti M.J., et al. Scaffold: Quantum programming language. Princeton Univ NJ Dept of Computer Science, 2012. 43 p.
8. Jones T., Brown A., Bush I., et al. Quest and high performance simulation of quantum computers. Scientific reports. 2019. Vol. 9, no. 1. P. 1–11. DOI: 10.1038/s41598-019-47174-9.
9. Guerreschi G.G., Hogaboam J., Baruffa F., et al. Intel Quantum Simulator: A cloud-ready high-performance simulator of quantum circuits. Quantum Science and Technology. 2020. Vol. 5, no. 3. P. 034007. DOI: 10.1088/2058-9565/ab8505.
10. Smelyanskiy M., Sawaya N.P.D., Aspuru-Guzik A. qHiPSTER: the quantum high performance software testing environment. arXiv preprint. 2016. arXiv:1601.07195.
11. Aleksandrowicz G., Alexander T., Barkoutsos P., et al. Qiskit: An open-source framework for quantum computing. 2019. DOI: 10.5281/zenodo.2562110.
12. Amy M., Gheorghiu V. staq–A full-stack quantum processing toolkit. Quantum Science and Technology. 2020. Vol. 5, no. 3. P. 034016. DOI: 10.1088/2058-9565/ab9359.
13. Häner T., Steiger D.S. 5 petabyte simulation of a 45-qubit quantum circuit. Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. 2017. P. 1–10. DOI: 10.1145/3126908.3126947.
14. Pednault E., Gunnels J.A., Nannicini G., et al. Breaking the 49-qubit barrier in the simulation of quantum circuits. arXiv preprint arXiv:1710.05867. 2017.
15. Pednault E., Gunnels J., Maslov D., et al. On "Quantum Supremacy". 2019. URL: https://www.ibm.com/blogs/research/2019/10/on-quantum-supremacy/ (accessed: 01.09.2020).
16. Chen Z.Y., Zhou Q., Xue C., et al. 64-qubit quantum circuit simulation. Science Bulletin. 2018. Vol. 63, no. 15. P. 964–971. DOI: 10.1016/j.scib.2018.06.007.
17. Arute F., Arya K., Babbush R., et al. Quantum supremacy using a programmable superconducting processor. Nature. 2019. Vol. 574, no. 7779. P. 505–510. DOI: 10.1038/s41586-019-1666-5.
18. Google AI Blog: A Preview of Bristlecone, Google's New Quantum Processor. URL: https://ai.googleblog.com/2018/03/a-preview-of-bristlecone-googles-new.html (accessed: 01.09.2020).
19. The D-Wave 2000Q™ System. URL: https://www.dwavesys.com/d-wave-two-system (accessed: 01.09.2020).
20. de Avila A.B., Reiser R.H., Pilla M.L., et al. State-of-the-art quantum computing simulators: Features, optimizations, and improvements for D-GM. Neurocomputing. 2020. Vol. 393. P. 223–233. DOI: 10.1016/j.neucom.2019.01.118.
21. Häner T., Steiger D.S., Smelyanskiy M., et al. High performance emulation of quantum circuits. SC'16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE, 2016. P. 866–874. DOI: 10.1109/SC.2016.73.
22. GitHub – QuEST-Kit/QuEST: A multithreaded, distributed, GPU-accelerated simulator of quantum computers. URL: https://github.com/QuEST-Kit/QuEST (accessed: 01.09.2020).
23. GitHub – iqusoft/intel-qs: High-performance simulator of quantum circuits. URL: https://github.com/iqusoft/intel-qs (accessed: 01.09.2020).
24. Voevodin Vl.V., Antonov A.S., Nikitenko D.A., et al. Supercomputer Lomonosov-2: large scale, deep monitoring and fine analytics for the user community. Supercomputing Frontiers and Innovations. 2019. Vol. 6, no. 2. P. 4–11. DOI: 10.14529/jsfi190201.

УДК 519.6 DOI: 10.14529/cmse210105

# МОДЕЛИРОВАНИЕ ВЛИЯНИЯ СИСТЕМЫ МОНИТОРИНГА ПРОИЗВОДИТЕЛЬНОСТИ НА ВЫПОЛНЕНИЕ КОЛЛЕКТИВНЫХ MPI ОПЕРАЦИЙ*

© 2021 А.А. Худолеева, К.С. Стефанов
*Московский государственный университет имени М.В. Ломоносова*
*(119991 Москва, ул. Ленинские Горы, д. 1)*
*E-mail: khudoleeva.anna98@gmail.com, cstef@parallel.ru*
Поступила в редакцию: 02.10.2020

Изучение параллельных программ с помощью средств мониторинга производительности — распространенная практика. Агент системы мониторинга для сбора данных о работе приложения периодически активируется во время счета этого приложения, внося помехи и занимая ресурсы. Однако вопрос об уровне влияния этих помех является слабо изученным, разработчики систем мониторинга зачастую не проводят исследования в этом направлении. В данной статье рассматриваются подходы к изучению влияния системы мониторинга производительности суперкомпьютера на пользовательские приложения. В качестве инструмента для измерения влияния агента системы мониторинга предлагается использовать коллективные MPI операции. Так, кроме обнаружения шума системы мониторинга, можно исследовать влияние системы мониторинга на сильно синхронизированные приложения. Время выполнения коллективных MPI операций изучается в присутствии программного средства, моделирующего работу агента системы мониторинга производительности. Оценивается уровень шума, который каждая из рассматриваемых коллективных операций в выбранной конфигурации запуска способна зафиксировать. В работе приводятся данные запусков инструмента с коллективными MPI операциями All-to-All, All-Reduce, Barrier. Найдено, что хорошей стабильностью и чувствительностью обладают операции All-to-All и Barrier.

*Ключевые слова: суперкомпьютер, мониторинг производительности, шум системы мониторинга, замедление параллельных задач, моделирование влияния системы мониторинга.*

**ОБРАЗЕЦ ЦИТИРОВАНИЯ**

Худолеева А.А., Стефанов К.С. Моделирование влияния системы мониторинга производительности на выполнение коллективных MPI операций // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2021. Т. 10, № 1. С. 62–74. DOI: 10.14529/cmse210105.

## Введение

Современные суперкомпьютеры — это очень сложные, большие вычислительные комплексы, состоящие из множества компонентов. Рост возможностей системы, под которую пишется приложение, затрудняет его разработку и усложняет структуру этого приложения. Это ведет к повышению сложности оптимизации приложения под целевую платформу. Достижение максимальной производительности и наиболее полное использование предоставленных ресурсов — важная задача. Невозможность эффективно использовать средства суперкомпьютера ведет к замедлению работы программ и, следовательно, к задержке проведения научного исследования. Для решения данной проблемы оптимизации приложения можно использовать систему мониторинга производительности суперкомпьютера.

При проектировании средства мониторинга разработчики преследуют цель создания масштабируемого, эффективного инструмента, который будет предоставлять для анализа

*Статья рекомендована к публикации программным комитетом Международной конференции «Суперкомпьютерные дни в России – 2020».

необходимую информацию об используемых пользовательским процессом ресурсах. Данные могут характеризовать, например, взаимодействие приложения с различными частями системы: с процессором, сетью, памятью. Агент системы мониторинга запускается совместно с исследуемым приложением на тех же вычислительных узлах и с определенным периодом выполняет сбор данных и их отправку для дальнейшей обработки. Так как средство мониторинга разделяет ресурсы с приложением, оно вносит помехи в работу исследуемого приложения и негативно влияет на производительность. К данному моменту разработано достаточное количество различных систем мониторинга. Однако авторы этих систем часто утверждают, что влияние средств мониторинга незначительно, не приводя описания методов, позволяющих это влияние измерить.

Вопросу обнаружения влияния системы мониторинга производительности суперкомпьютера с помощью использования коллективных MPI операций и установления уровня этого влияния указанным методом посвящена эта работа.

Статья организована следующим образом. Раздел 1 посвящен описанию исследований влияния некоторых существующих систем мониторинга производительности и шума ОС на пользовательские задачи. В разделе 2 описываются выбранный инструмент для обнаружения шума, эксперименты, проведенные на СК, и результаты этих экспериментов. В заключении приводится краткая сводка результатов, полученных в работе, и указаны направления дальнейших исследований.

## 1. Обзор работ

За последние 20 лет было уделено много внимания созданию различных средств мониторинга производительности: Supermon [9], NWPerf [6], HPCToolkit [1], Performance Co-Pilot [13], LIKWID [9, 11], LDMS [2]. Авторы статей об инструментах HPCToolkit, Performance Co-Pilot, LIKWID говорят, что эффект, оказываемый этими средствами мониторинга на пользовательские приложения, является незначительным, однако не приводят результатов исследований, связанных с этим вопросом. Только для двух систем из списка [2, 6] проводилось исследование их влияния на пользовательские задачи.

Агентом системы мониторинга NWPerf [6] является модуль ядра Linux. Стандартный интервал сбора данных для этого средства — одна минута. Для этого инструмента было проведено исследование его влияния на коллективные операции All-to-All и All-Reduce. На 128 узлах, на 256 вычислительных ядрах запускался цикл из 10 000 коллективных операций с и без средства мониторинга. При частоте сбора данных (1 раз в 6 секунд), в десять раз превышающей стандартный режим, время выполнения операции All-to-All увеличилось на 27%, а операции All-Reduce — на 9,46%. Но при стандартной частоте прочие помехи в системе перекрывали шум от NWPerf, что свидетельствует о приемлемости частоты работы агента раз в минуту.

Достаточно подробно было исследовано влияние средства мониторинга LDMS [2] на работу различных компонент высокопроизводительной системы. Средой проведения экспериментов выступали две большие вычислительные системы. Тестирование с использованием бенчмарков показало, что инструмент LDMS при частоте работы 1 раз в секунду имеет незначительное влияние на производительность MPI коммуникаций.

Разница в полученных результатах в вопросе влияния системы мониторинга на MPI коммуникации значительная. Она может быть вызвана аппаратными и сетевыми отличиями суперкомпьютеров, на которых запускались системы мониторинга. Исследование системы LDMS [2] является более актуальным, так как проводилось на 10 лет позже исследования системы NWPerf [6]. Тем не менее, полученные в работах [2, 6] результаты

сложно обобщить, и вопрос влияния помех, вносимых системами мониторинга на производительность параллельных программ, остается открытым.

Хорошо исследованной является область обнаружения уровня влияния шума операционной системы на работу параллельных программ. Методы, применяемые в этой области, можно распространить на изучение шума системы мониторинга. В работе [3] при использовании большого количества процессов удалось установить, что несинхронизированный шум от операционной системы, возникающий периодически, является причиной значительного замедления коммуникаций Barrier и All-Reduce. В исследовании [7], посвященном оптимизации приложения SAGE на суперкомпьютере ASCI Q, было показано, что источником замедления работы приложения была низкая производительность операции All-Reduce. Авторы исследования [7] сделали вывод о том, что на некоторые хорошо синхронизированные приложения небольшой, но частый синхронный шум ОС оказывает сильное влияние.

Можно предположить, что коллективные операции являются чувствительными даже к малому шуму, создаваемому агентом системы мониторинга производительности, и их можно использовать для обнаружения помех, вносимых системой мониторинга в работу пользовательской задачи.

## 2. Методология исследования

Одним из самых популярных средств при программировании приложений для вычислительных кластеров является MPI — интерфейс, поддерживающий работу параллельных процессов с помощью передачи сообщений между ними. Из функционала, который доступен при использовании MPI, широко используются коллективные операции. В исследовании [5] проводился анализ 110 параллельных приложений с открытым исходным кодом. Минимальный функционал, который используют все рассмотренные в работе приложения — вызов коллективных операций.

Приложение может выполняться несимметрично, то есть в зависимости от ранга процессам назначаются роли, реализованные разными фрагментами кода. На большом вычислительном кластере нельзя гарантировать бесперебойность работы каждого узла, симметрию выполнения приложения на каждом ядре. Поэтому даже в приложениях, где процессы выполняют один и тот же фрагмент кода, стартовав синхронно, время выполнения последовательной части программы будет разным для разных процессов. Так как при запуске коллективной операции все процессы должны одновременно выполнить пересылку сообщений или синхронизацию, часть процессов будет простаивать до начала коммуникации, ожидая готовности к выполнению операции «опаздывающих» процессов. Поэтому время работы параллельного приложение измеряется по самому медленному процессу.

В силу тех фактов, что использование коллективных MPI операций является распространенной практикой и они могут стать весомым источником замедления в работе хорошо синхронизированной программы, важным является вопрос рассмотрения влияния агента системы мониторинга на выполнение коллективных операций. Задачей данной работы является исследование влияния системы мониторинга производительности на коллективные операции при использовании небольшого числа узлов суперкомпьютера. Возможность использования коллективных MPI операций, как средства для обнаружения шума, создаваемого системой мониторинга, и его уровня, исследуется ниже.

## 2.1. Программное средство, детектирующее шум

На рис. 1 представлена схема реализации программного средства — детектора шума — с помощью которого предлагается исследовать влияние шума системы мониторинга производительности. Измеряемой величиной при работе детектора шума является время выполнения цикла из *count* числа коллективных MPI операций. Рассматриваемыми коллективными MPI операциями в рамках данной работы являются: Barrier, All-Reduce, All-to-All.

```
start timer
do over iteration count
     MPI collective operation
stop timer
```

**Рис. 1.** Схема реализации инструмента для обнаружения шума

Для разработанного программного средства существует несколько свободных параметров, выбрав которые, можно определить конфигурацию запуска детектора шума. Конфигурацию инструмента будем считать подходящей, если при запуске в стандартных условиях, в отсутствии искусственно внедренного шума, эксперимент является воспроизводимым, то есть от запуска к запуску время выполнения коллективных операций изменяется в пределах допустимой погрешности. Также важной является способность инструмента фиксировать присутствие в системе дополнительного шума, чем выше чувствительность, тем лучше.

Свободные параметры включают в себя:

– выбор коллективной операции;
– количество узлов, на которых запущено программное средство, количество процессов;
– количество коллективных операций;
– для операций All-Reduce и All-toAll длина передаваемого сообщения.

## 2.2. Программное средство, моделирующее шум агента системы мониторинга

Было решено начать исследование возможности детектора обнаружить шум системы мониторинга с изучения времени работы детектора совместно с искусственным шумом. На рис. 2 представлен фрагмент кода источника шума — программы, которая моделирует влияние агента системы мониторинга производительности на работу пользовательского приложения.

В основе работы этого средства лежит цикл, в котором вычисляется сумма (значение суммы хранится в переменной *sum*) массива **a* объемом 250 000 слов (значение константы *SIZE*) типа long long. В *WORK_TIME* хранится доля секунды, которую работает нагрузка — суммирование массива. В параметр *tw* процедуры *nanosleep()* записана доля секунды, в течение которой сумма массива не вычисляется. Это значение дополняет *WORK_TIME* до 1 секунды. *BREAK_TIME* — общее время работы источника шума. Уровень шума в процентах — значение 100x*WORK_TIME*. Так, если речь идет об искусственном шуме 1%, значит каждую секунду цикл суммирования массива работает в течение 0,01 с и оставшуюся часть секунды бездействует. Стоит заметить, что величина искусственного шума 1%, создаваемого источником шума, оценивается на порядок выше, чем величина шума от агента реальной системы мониторинга.

```
start_time = MPI_Wtime();
do
{
     t_1 = MPI_Wtime();
     do
     {
        for (int i = 0; i < SIZE; ++i)
          sum += a[i];
        t_2 = MPI_Wtime();
     }
     while ((t_2 - t_1) < WORK_TIME);
     nanosleep(&tw, &tc);
     t_2 = MPI_Wtime();
}
while ((t_2 - start_time) < BREAK_TIME);
```

**Рис. 2.** Фрагмент реализации программного средства, моделирующего влияние агента системы мониторинга

## 2.3. Описание проведенных экспериментов

Исследование разработанных программных средств было выполнено на суперкомпьютере Ломоносов-2 [12]. Запуски проводились в тестовом разделе суперкомпьютера. Так удалось избежать ожидания в загруженной очереди основного раздела. На каждом узле установлен процессор Intel Haswell-EP E5-2697v3, имеющий 14 ядер. Лимит времени выполнения заданий на тестовом разделе составляет 15 минут. Этим фактом обусловлен выбор числа итераций в цикле для детектора шума. При запуске на суперкомпьютере использовалась библиотека OpenMPI [14].

Исследование детектора шума проходило в два этапа. Сначала необходимо было оценить стабильность работы разных вариантов детектора шума. В случае сильного разброса времени выполнения коллективных операций одного из вариантов детектора на системе Ломоносов-2, нужно отказаться от такой конфигурации, так как она не подходит для многократного использования в качестве инструмента, предназначенного для обнаружения шума. Причиной этого является отсутствие возможности получить воспроизводимое значение времени работы детектора без шума и, следовательно, отличить его от времени работы детектора в присутствии сторонней нагрузки. Далее, выбранные варианты детектора шума запускались совместно с источником шума. Полученная разница во времени выполнения коллективных операций в этих двух случаях сравнивалась с помощью статистических критериев, описанных в работе [4], и определялись чувствительные к дополнительной нагрузке конфигурации детектора шума.

В силу большого разнообразия возможностей проведения запусков была выбрана следующая конфигурация: программные средства запускались на 4 узлах суперкомпьютера Ломоносов-2, по 2 процесса на ядро (один процесс на виртуальное ядро, Hyper Threading включен). Для программы с операцией All-to-All также проводились тесты на 8 узлах. Модель агента системы мониторинга запускалась совместно с детектором на тех же узлах, но на одном ядре одного узла.

## 2.4. Результаты измерений

На рисунках ниже и в табл. 1, 2 представлены результаты измерений для каждой конфигурации детектора.

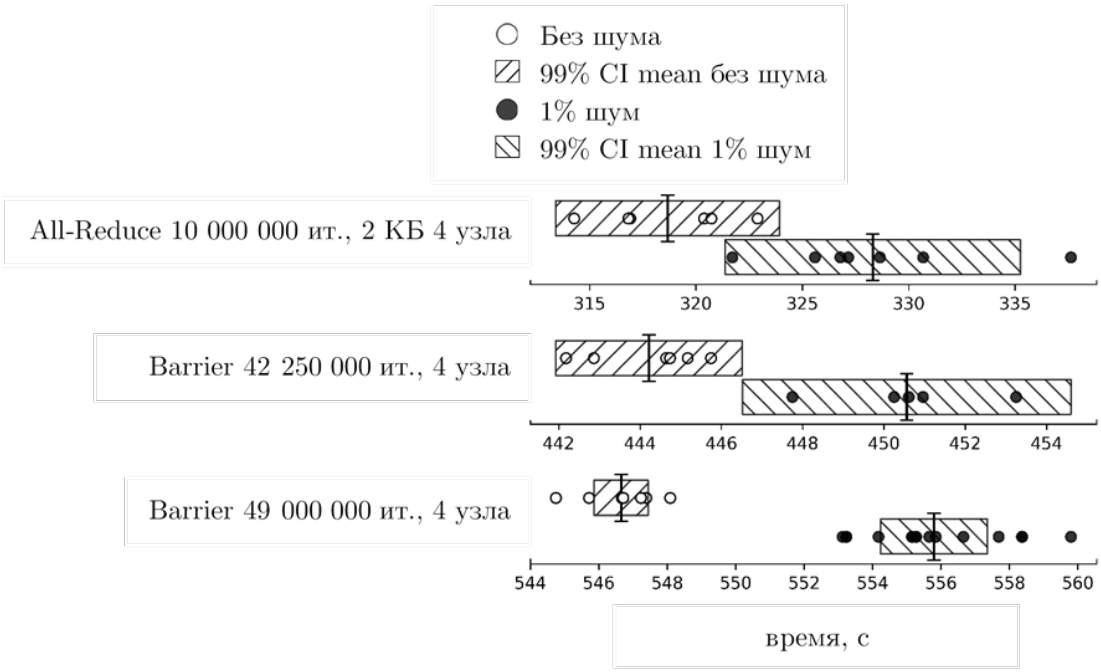



**Рис. 3**. Распределение времени для операций All-Reduce, Barrier

**Таблица 1**

Результаты запусков для I: All-Reduce, 10 000 000 ит., 2 КБ, 4 узла;
II: Barrier, 42 250 000 ит., 4 узла; III: Barrier, 49 000 000 итераций, 4 узла

| | | I | II | III |
|---|---|---|---|---|
| Запуски без шума | Среднее значение, с | 318,7 | 444,2 | 546,66 |
| | Стандартное отклонение, с | 3,19 | 1,39 | 1,03 |
| | Количество запусков | 6 | 6 | 8 |
| Запуски с шумом 1% | Среднее значение, с | 328,3 | 450,6 | 555,90 |
| | Стандартное отклонение, с | 4,95 | 1,96 | 2,09 |
| | Количество запусков | 7 | 5 | 11 |

All-Reduce, 1 000 000 операций (рис. 3, табл. 1) демонстрирует разброс во времени выполнения детектора без шума. Также конфигурация не обнаруживает шум 1%–99% CI mean времени работы детектора без шума и с шумом 1% пересекаются. Данная конфигурация дальше не используется. Для детектора с 42 250 000 итераций (рис. 3, табл. 1) операции Barrier доверительные интервалы касаются. Было решено проверить эту конфигурацию детектора при большем числе операций Barrier. Так, для операции Barrier, 49 000 000 итераций, 4 узла на рис. 3 видно, что значения времени работы детектора с шумом 1% превышают доверительный интервал времени работы детектора без шума. Эта конфигурация детектора обнаруживает искусственный шум 1% и время работы этого детектора воспроизводимо.

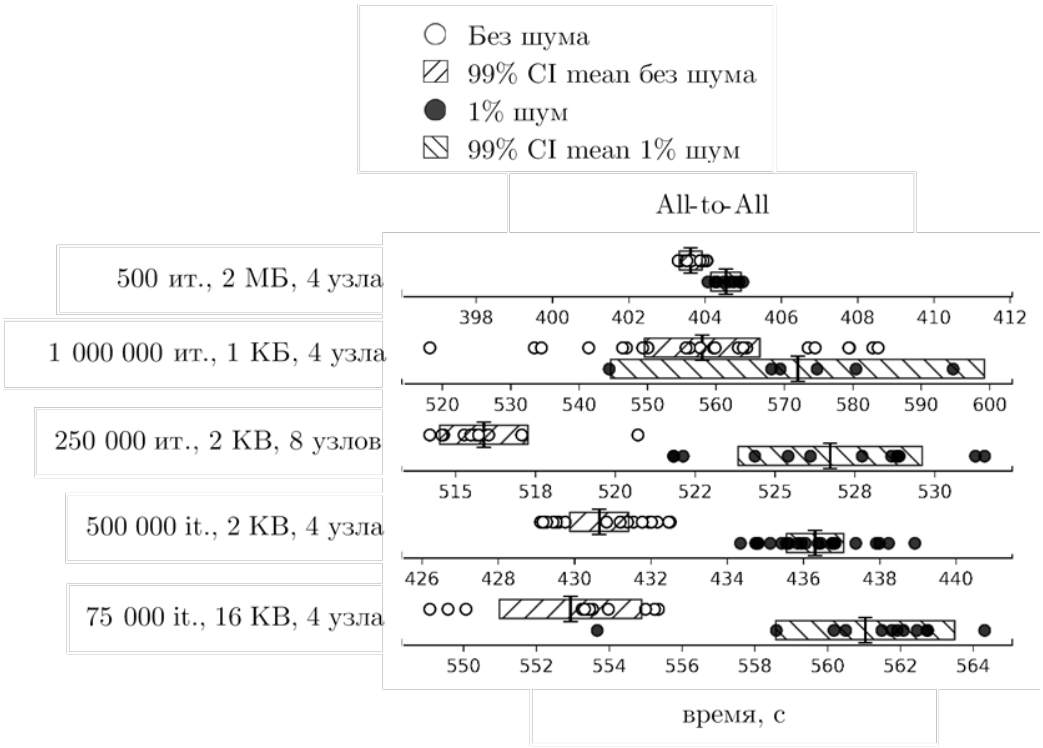



**Рис. 4.** Распределение времени для операции All-to-All

**Таблица 2**

Результаты запусков для I: All-to-All 500 ит., 2 МБ, 4 узла;
II: All-to-All 1 000 000 ит., 1 КБ, 4 узла; III: All-to-All 250 000 ит., 2 КБ, 8 узлов;
IV: All-to-All 500 000 ит., 2 КБ, 4 узла; V: All-to-All 75 000 ит., 16 КБ, 4 узла

| | | I | II | III | IV | IV |
|---|---|---|---|---|---|---|
| Чистые запуски | Среднее значение, с | 403,6 | 558,0 | 515,9 | 430,7 | 552,9 |
| | Стандартное отклонение, с | 0,3 | 14,3 | 1,6 | 1,3 | 2,2 |
| | Количество запусков | 10 | 23 | 13 | 22 | 12 |
| Запуски с шумом 1% | Среднее значение, с | 404,5 | 572,0 | 526,7 | 436,3 | 561,0 |
| | Стандартное отклонение, с | 0,3 | 16,6 | 3,4 | 1,2 | 2,7 |
| | Количество запусков | 8 | 6 | 13 | 22 | 12 |

Для All-to-All, 500 итераций, 2 МБ, 4 узла на рис. 4 видно, что время работы детектора с шумом 1% отличается меньше чем на 1 секунду от времени работы детектора без шума. Эта конфигурация детектора не чувствительна к шуму 1%. В случае All-to-All, 1 000 000 итераций, 1 КВ, 4 узла время «чистых» запусков имеет сильный разброс. При таких параметрах детектора уловить разницу между «чистыми» запусками и запусками с шумом 1% нельзя. Причины плохой воспроизводимости времени работы детектора в этой конфигурации неизвестны. All-to-All, 75 000 операций, 16 КБ, 4 узла (рис. 4, табл. 2) можно считать пригодным инструментом для обнаружения шума. Детекторы All-to-All, 250 000 итераций, 8 узлов и All-to-All, 500 000 итераций, 4 узла (рис. 4, табл. 2) с размером сообщения 2 КБ показывают наиболее стабильный результат, с хорошей возможностью обнаружения шума 1%.

Во время запусков на суперкомпьютере выбросы появлялись для всех выбранных коллективных операций. Возможно, на появление разницы времени запусков влияют конкретные узлы, которые выделяются на суперкомпьютере, задержки во взаимодействии между ними. Причины появления выбросов подробно не исследовались.

## 2.5. Тестирование чувствительности

Выше были приведены результаты, демонстрирующие способность разработанных программных средств улавливать искусственный шум 1%, создаваемый моделью агента системы мониторинга производительности. Ниже на графиках приведены результаты экспериментов, в которых уровень шума постепенно снижался до порога чувствительности (до верхней границы доверительного интервала среднего значения времени работы детектора без шума). Для тестирования были выбраны детекторы, с помощью которых можно обнаружить шум 1%:

– All-to-All 500 000 итераций, 2 КБ, 4 узла (табл. 3);
– All-to-All 250 000 итераций, 2 КБ, 8 узла (табл. 4);
– All-to-All 75 000 итераций, 16 КБ, 4 узла (табл. 5);
– Barrier 49 000 000 итераций, 4 узла (табл. 6).

На рис. 5 и в табл. 3–6 при каждом указанном уровне нагрузки представлено среднее арифметическое времени работы детектора шума по всем соответствующим запускам. По графикам на рис. 5 видно, что при нагрузке в 0,5% шума для детекторов, использующих операции Barrier и All-to-All, время с шумом и без можно различить с хорошей надежностью. При уровне шума 0,3% для детектора, основанного на операции All-to-All, 2 КБ наблюдается достижение границы чувствительности. При дальнейшем понижении уровня шума доверительные интервалы среднего значения времени работы детектора с шумом и без начинают пересекаться. Детекторы All-to-All, 16 КБ и Barrier, 49 000 000 итераций также обнаруживает шум до 0,3%. Причем на время работы операции Barrier шум 0,01% и шум 0,3% влияет одинаково. Чувствительность операции Barrier к столь малому шуму 0,01% требует более тщательной проверки — проведения повторных экспериментов — так как ни для какого другого варианта детектора шума такая чувствительность не была замечена. Видно, что все рассмотренные конфигурации детектора шума обнаруживают малый шум до 0,3%.

**Таблица 3**

Результаты тестирования чувствительности All-to-All 500 000 итераций, 2 КБ, 4 узла

| Уровень шума | 0% | 0,01% | 0,05% | 0,1% | 0,3% | 0,5% | 0,7% | 1% |
|---|---|---|---|---|---|---|---|---|
| Среднее значение, с | 431 | 432 | 432 | 433 | 434 | 435 | 436 | 436 |
| Стандартное отклонение, с | 1,3 | 1,1 | 2,1 | 1,5 | 1,7 | 1,5 | 2,2 | 1,2 |
| Количество запусков | 22 | 10 | 4 | 9 | 6 | 7 | 9 | 22 |

**Таблица 4**

Результаты тестирования чувствительности All-to-All 250 000 итераций, 2 КБ, 8 узлов

| Уровень шума | 0% | 0,01% | 0,05% | 0,1% | 0,3% | 0,5% | 0,7% | 1% |
|---|---|---|---|---|---|---|---|---|
| Среднее значение, с | 516 | 517 | 523 | 519 | 519 | 526 | 529 | 527 |
| Стандартное отклонение, с | 1,6 | 3,4 | 0,7 | 4,3 | 1,1 | 0,9 | 2,7 | 3,4 |
| Количество запусков | 13 | 5 | 5 | 5 | 5 | 5 | 4 | 13 |

**Таблица 5**

Результаты тестирования чувствительности Barrier, 49 000 000 итераций, 4 узла

| Уровень шума | 0% | 0,01% | 0,05% | 0,1% | 0,3% | 0,5% | 0,7% | 1% |
|---|---|---|---|---|---|---|---|---|
| Среднее значение, с | 547 | 552 | 549 | 551 | 552 | 553 | 555 | 556 |
| Стандартное отклонение, с | 1,0 | 1,5 | 1,3 | 1,0 | 1,7 | 0,7 | 0,9 | 2,1 |
| Количество запусков | 8 | 5 | 5 | 5 | 5 | 5 | 5 | 11 |

**Таблица 6**

Результаты тестирования чувствительности All-to-All 75 000 ит., 16 КБ, 4 узла

| Уровень шума | 0% | 0,01% | 0,05% | 0,1% | 0,3% | 0,5% | 0,7% | 1% |
|---|---|---|---|---|---|---|---|---|
| Среднее значение, с | 553 | 555 | 556 | 557 | 560 | 560 | 561 | 561 |
| Стандартное отклонение, с | 2,2 | 0,2 | 0,3 | 1,2 | 1,9 | 0,3 | 0,7 | 2,7 |
| Количество запусков | 12 | 5 | 5 | 5 | 5 | 5 | 5 | 12 |

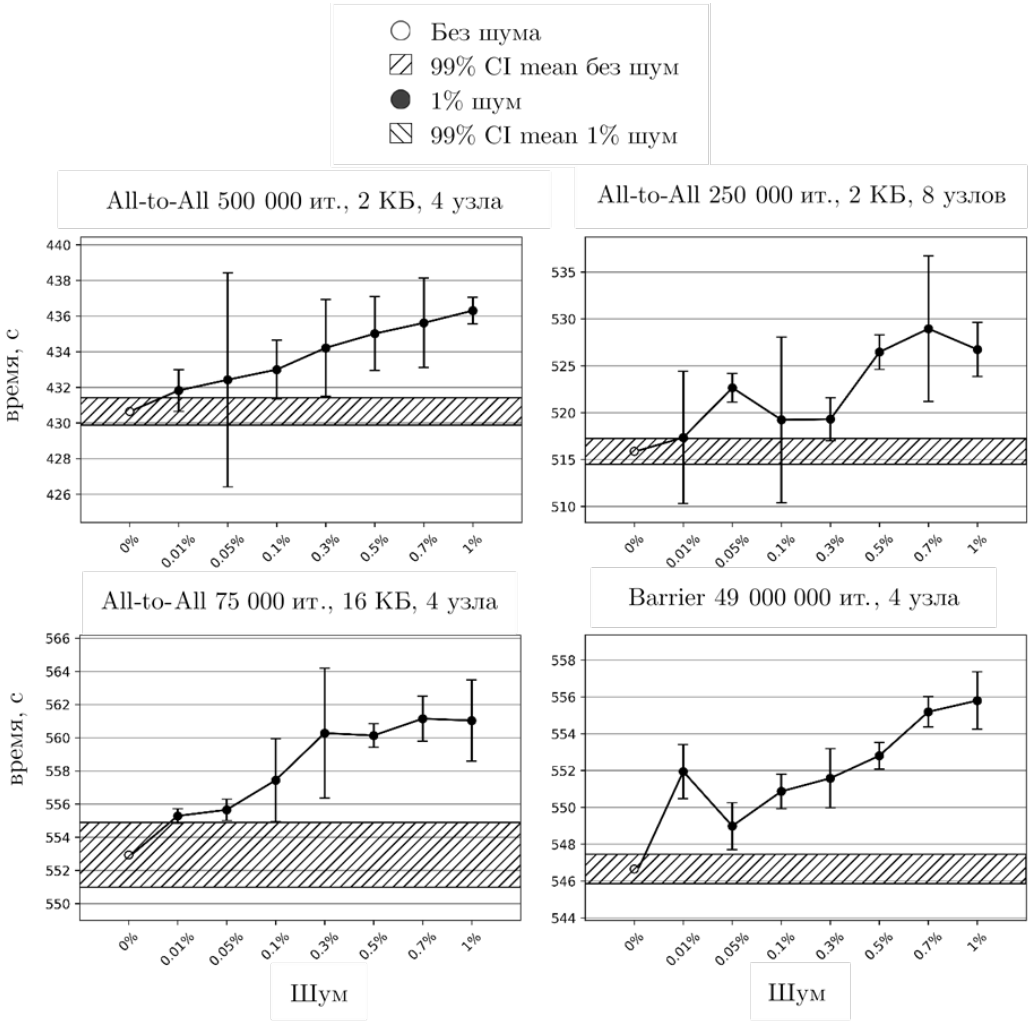



**Рис. 5.** Тестирование чувствительности

## Заключение

Были разработаны два программных средства — источник шума, предназначенный для моделирования влияния работы агента системы мониторинга производительности, и детектор шума. В качестве детектора шума были опробованы программы с различными коллективными MPI операциями. В работе рассматривалось замедление времени выполнения детектора с шумом относительно времени выполнения детектора без дополнительной нагрузки. Так, если замедление работы коллективных MPI операций статистически значимо, то коллективная операция пригодна для обнаружения дополнительной малой нагрузки в системе.

Для операции All-to-All проводились запуски с различной длиной сообщения. Оказалось, что наиболее стабильным из рассмотренных является детектор с длиной пересылаемого сообщения 2 КБ, протестированный на 4-х и 8-ми узлах суперкомпьютера.

Детектор, использующий операцию Barrier, при большом числе итераций (время выполнения от 7 минут) также является чувствительным к искусственному шуму 1%. При использовании All-Reduce стабильная конфигурация для суперкомпьютера не была найдена.

Для операции All-to-All была обнаружена граница чувствительности к нагрузке, которую создает источник шума. Это значение составляет 0,3% для операции All-to-All. Операция Barrier также чувствительна к шуму 0,3%. Шум 0,3% является малым, поэтому можно сказать, что операции All-to-All и Barrier обладают хорошей чувствительностью. Эти коллективные операции являются подходящими для обнаружения влияния малой дополнительной нагрузки, и их можно использовать для изучения шума от системы мониторинга производительности суперкомпьютера.

В дальнейшей работе планируется исследовать чувствительность операций All-to-All и Barrier к шуму реальной системы мониторинга производительности суперкомпьютера Ломоносов-2 [10]. Будет исследовано время работы различных конфигураций детектора шума и отобраны те из них, которые являются чувствительными к шуму агента системы мониторинга, работающего в стандартном режиме. С помощью отобранных конфигураций детектора планируется сравнить уровень влияния разных режимов работы агента системы мониторинга производительности на синхронизированные приложения.

*Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 19-07-00940.*

*Работа выполнена с использованием оборудования Центра коллективного пользования сверхвысокопроизводительными вычислительными ресурсами МГУ имени М.В. Ломоносова.*

## Литература

1. Adhianto L., Banerjee S., Fagan M., et al. HPCTOOLKIT: tools for performance analysis of optimized parallel programs // Concurr. Comput. Pract. Exp. 2009. Vol. 22, no. 6. P. 685–701. DOI: 10.1002/cpe.1553.
2. Agelastos A., Allan B., Brandt J., et al. The Lightweight Distributed Metric Service: A Scalable Infrastructure for Continuous Monitoring of Large Scale Computing Systems and Applications // International Conference for High Performance Computing, Networking, Storage and Analysis, SC14 (New Orleans, LA, USA, Nov. 2014). IEEE, 2014. P. 154–165. DOI: 10.1109/SC.2014.18.
3. Beckman P., Iskra K., Yoshii K., et al. Benchmarking the effects of operating system interference on extreme-scale parallel machines // Cluster Comput. 2008. Vol. 11, no. 1. P. 3–16. DOI: 10.1007/s10586-007-0047-2.
4. Hoefler T., Belli R. Scientific benchmarking of parallel computing systems // Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis on, SC '15 (New York, New York, USA, Nov. 2015). ACM Press, 2015. P. 1–12. DOI: 10.1145/2807591.2807644.
5. Laguna I., Marshall R., Mohror K., et al. A large-scale study of MPI usage in open-source HPC applications // Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (New York, NY, USA, Nov. 2019). ACM, 2019. P. 1–14. DOI: 10.1145/3295500.3356176.
6. Mooney R., Schmidt K.P., Studham R.S. NWPerf: a system wide performance monitoring tool for large Linux clusters // 2004 IEEE International Conference on Cluster Computing. (San Diego, CA, USA, Sept. 2004). IEEE, 2004. P. 379–389. DOI: 10.1109/CLUSTR.2004.1392637.

7. Petrini F., Kerbyson D.J., Pakin S. The Case of the Missing Supercomputer Performance // Proceedings of the 2003 ACM/IEEE conference on Supercomputing, SC '03 (New York, New York, USA, Nov. 2003). ACM Press, 2003. P. 55. DOI: 10.1145/1048935.1050204.
8. Rohl T., Eitzinger J., Hager G., et al. LIKWID Monitoring Stack: A Flexible Framework Enabling Job Specific Performance monitoring for the masses // 2017 IEEE International Conference on Cluster Computing, CLUSTER (Honolulu, HI, USA, Sept. 2017). IEEE, 2017. P. 781–784. DOI: 10.1109/CLUSTER.2017.115.
9. Sottile M.J., Minnich R.G. Supermon: a high-speed cluster monitoring system // Proceedings. IEEE International Conference on Cluster Computing (Chicago, IL, USA, USA, Sept. 2002). IEEE Comput. Soc, 2002. P. 39–46. DOI: 10.1109/CLUSTR.2002.1137727.
10. Stefanov K., Voevodin V., Zhumatiy S., et al. Dynamically Reconfigurable Distributed Modular Monitoring System for Supercomputers (DiMMon) // Procedia Computer Science. Elsevier B.V., 2015. P. 625–634. DOI: 10.1016/j.procs.2015.11.071.
11. Treibig J., Hager G., Wellein G. LIKWID: A Lightweight Performance-Oriented Tool Suite for x86 Multicore Environments // 2010 39th International Conference on Parallel Processing Workshops (San Diego, CA, USA, Sept. 2010). IEEE, 2010. P. 207–216. DOI: 10.1109/ICPPW.2010.38.
12. Voevodin Vl.V., Antonov A.A., Nikitenko D.A., et al. Supercomputer Lomonosov-2: Large Scale, Deep Monitoring and Fine Analytics for the User Community // Supercomput. Front. Innov. 2019. Vol. 6, no. 2. P. 4–11. DOI: 10.14529/jsfi190201.
13. Performance Co-Pilot. URL: https://pcp.io/ (дата обращения: 27.09.2020)
14. Open MPI: Open Source High Performance Computing. URL: https://www.open-mpi.org/ (дата обращения: 27.09.2020)

Худолеева Анна Александровна, студентка, кафедра суперкомпьютеров и квантовой информатики, Московский государственный университет имени М.В. Ломоносова (Москва, Российская Федерация)

Стефанов Константин Сергеевич, к.ф.-м.н., старший научный сотрудник, научно-исследовательский вычислительный центр Московского государственного университета имени М.В. Ломоносова (Москва, Российская Федерация)

DOI: 10.14529/cmse210105

# MODELING INFLUENCE OF MONITORING SYSTEM ON PERFORMANCE OF MPI COLLECTIVE OPERATIONS

© 2021 A.A. Khudoleeva, K.S. Stefanov
*Lomonosov Moscow State University (GSP-1, Leninskie Gory 1, Moscow, 119991 Russia)*
*E-mail: khudoleeva.anna98@gmail.com, cstef@parallel.ru*
Received: 02.10.2020

Studying parallel program with the means of monitoring systems is a common practice. To collect data about application, monitoring system agent activates periodically during the run of application, occupying resources and causing perturbation. Monitoring system developers often ignore studying the problem of monitoring tools interference into application performance, this problem remains poorly examined. This article discusses ways to study influence of supercomputer monitoring system on users' applications. We suggest to use MPI collective operations as a tool to measure this influence. This method also allows to estimate influence of monitoring system noise on a synchronized application. MPI collective operations are measured in presence of injected noise generated by the program that imitates interference of monitoring tool. We estimate the noise level that each of the used collective operations is capable to sense in chosen configuration. All-to-All, All-Reduce and Barrier are used in the noise detection tool. We find parameters for All-to-All and Barrier operations to perform stably and detect low noise level.

*Keywords: supercomputer, performance monitoring, monitoring system noise, parallel job slowdown, modeling influence of monitoring system.*

**FOR CITATION**

Khudoleeva A.A., Stefanov K.S. Modeling Influence of Monitoring System on Performance of MPI Collective Operations. *Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering.* 2021. Vol. 10, no. 1. P. 62–74. (in Russian) DOI: 10.14529/cmse210105.



## References

1. Adhianto L., Banerjee S., Fagan M., et al. HPCTOOLKIT: tools for performance analysis of optimized parallel programs. Concurr. Comput. Pract. Exp. 2009. Vol. 22, no. 6. P. 685–701. DOI: 10.1002/cpe.1553.
2. Agelastos A., Allan B., Brandt J., et al. The Lightweight Distributed Metric Service: A Scalable Infrastructure for Continuous Monitoring of Large Scale Computing Systems and Applications. International Conference for High Performance Computing, Networking, Storage and Analysis, SC14 (New Orleans, LA, USA, Nov. 2014). IEEE, 2014. P. 154–165. DOI: 10.1109/SC.2014.18.
3. Beckman P., Iskra K., Yoshii K., et al. Benchmarking the effects of operating system interference on extreme-scale parallel machines. Cluster Comput. 2008. Vol. 11, no. 1. P. 3–16. DOI: 10.1007/s10586-007-0047-2.
4. Hoefler T., Belli R. Scientific benchmarking of parallel computing systems. Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis on, SC '15 (New York, New York, USA, Nov. 2015). ACM Press, 2015. P. 1–12. DOI: 10.1145/2807591.2807644.
5. Laguna I., Marshall R., Mohror K., et al. A large-scale study of MPI usage in open-source HPC applications. Proceedings of the International Conference for High Performance

Computing, Networking, Storage and Analysis (New York, NY, USA, Nov. 2019). ACM, 2019. P. 1–14. DOI: 10.1145/3295500.3356176.

6. Mooney R., Schmidt K.P., Studham R.S. NWPerf: a system wide performance monitoring tool for large Linux clusters. 2004 IEEE International Conference on Cluster Computing (San Diego, CA, USA, Sept. 2004). IEEE, 2004. P. 379–389. DOI: 10.1109/CLUSTR.2004.1392637.
7. Petrini F., Kerbyson D.J., Pakin S. The Case of the Missing Supercomputer Performance. Proceedings of the 2003 ACM/IEEE conference on Supercomputing, SC '03 (New York, New York, USA, Nov. 2003). ACM Press, 2003. P. 55. DOI: 10.1145/1048935.1050204.
8. Rohl T., Eitzinger J., Hager G., et al. LIKWID Monitoring Stack: A Flexible Framework Enabling Job Specific Performance monitoring for the masses. 2017 IEEE International Conference on Cluster Computing, CLUSTER (Honolulu, HI, USA, Sept. 2017). IEEE, 2017. P. 781–784. DOI: 10.1109/CLUSTER.2017.115.
9. Sottile M.J., Minnich R.G. Supermon: a high-speed cluster monitoring system. Proceedings. IEEE International Conference on Cluster Computing (Chicago, IL, USA, USA, Sept. 2002). IEEE Comput. Soc, 2002. P. 39–46 DOI: 10.1109/CLUSTR.2002.1137727.
10. Stefanov K., Voevodin V., Zhumatiy S., et al. Dynamically Reconfigurable Distributed Modular Monitoring System for Supercomputers (DiMMon). Procedia Computer Science. Elsevier B.V., 2015. P. 625–634 DOI: 10.1016/j.procs.2015.11.071.
11. Treibig J., Hager G., Wellein G. LIKWID: A Lightweight Performance-Oriented Tool Suite for x86 Multicore Environments. 2010 39th International Conference on Parallel Processing Workshops (San Diego, CA, USA, Sept. 2010). IEEE, 2010. P. 207–216. DOI: 10.1109/ICPPW.2010.38.
12. Voevodin Vl.V., Antonov A.A., Nikitenko D.A., et al. Supercomputer Lomonosov-2: Large Scale, Deep Monitoring and Fine Analytics for the User Community. Supercomput. Front. Innov. 2019. Vol. 6, no. 2. P. 4–11. DOI: 10.14529/jsfi190201.
13. Performance Co-Pilot. URL: https://pcp.io/ (accessed: 27.09.2020).
14. Open MPI: Open Source High Performance Computing. URL: https://www.open-mpi.org/ (accessed: 27.09.2020).

**УДК 51.73** **DOI: 10.14529/cmse210106**

# ПРОГРАММНЫЙ КОМПЛЕКС РАДУГА-Т ДЛЯ МОДЕЛИРОВАНИЯ ПОЛЕЙ НЕЙТРОНОВ В ЯДЕРНО-ЭНЕРГЕТИЧЕСКИХ УСТАНОВКАХ

**© 2021 О.В. Николаева, С.А. Гайфулин, Л.П. Басс**
*Институт прикладной математики им. М.В. Келдыша РАН*
*(123047 Москва, Миусская пл., д. 4)*
*E-mail: nika@kiam.ru, sagarp@yandex.ru, bass@kiam.ru*
Поступила в редакцию: 14.07.2020

При проектировании и сопровождении эксплуатации ядерно-энергетических установок (ЯЭУ) необходимо выполнять моделирование в этих установках потоков нейтронов. При задании геометрии ЯЭУ необходимо учитывать границы разномасштабных конструктивных элементов, состоящих из материалов с существенно различными свойствами. Из-за больших размеров ЯЭУ для расчетов желательно использовать параллельные компьютеры. Для выполнения такого моделирования развиваются алгоритмы и программы численного решения краевой задачи для интегро-дифференциального уравнения переноса нейтронов на неструктурированных сетках. В статье приводится описание реализованных в программном комплексе РАДУГА-Т алгоритмов решения такой задачи. Представлены сетки, сеточные схемы, итерационные методы решения систем сеточных уравнений. Рассмотрены методы распараллеливания вычислений на гибридных компьютерах (используются технологии MPI и OpenMP). Представлены методы работы с пространственными сетками (построение, улучшение качества, декомпозиция, визуализация). Описаны особенности программной реализации. Проведено сравнение используемых в программном комплексе РАДУГА-Т алгоритмов с алгоритмами в других аналогичных программных комплексах. Приведены результаты исследования эффективности распараллеливания вычислений в задаче расчета коэффициента размножения нейтронов в модели легководного реактора. Исследования выполнены на многопроцессорном компьютере МВС-10П (МСЦ РАН). Приведены значения ускорения вычислений каждого из используемых в расчете алгоритмов и суммарного ускорения всего расчета.

*Ключевые слова: уравнение переноса, неструктурированные сетки, сеточные схемы, итерационные методы, параллельные вычисления.*

**ОБРАЗЕЦ ЦИТИРОВАНИЯ**

Николаева О.В., Гайфулин С.А., Басс Л.П. Программный комплекс РАДУГА-Т для моделирования полей нейтронов в ядерно-энергетических установках // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2021. Т. 10, № 1. С. 75–89. DOI: 10.14529/cmse210106.

## Введение

При проектировании и сопровождении эксплуатации ядерно-энергетических установок (ЯЭУ) необходимо выполнять моделирование в этих установках потоков нейтронов. Нейтроны в работающих ЯЭУ возникают в результате ядерных реакций; в дальнейшем они вызывают такие же реакции и тем самым порождают энерговыделение в активной зоне (АЗ) ЯЭУ. Кроме того, нейтроны повреждают конструктивные элементы как в АЗ, так и в окружающих ее слоях радиационной защиты. Нейтроны, покинувшие ЯЭУ, влияют на радиационный фон в ее окрестности.

Для определения потоков нейтронов в АЗ часто используется уравнение диффузии. Такое приближение применимо в предположении, что потоки слабо зависят от направления скорости нейтрона. Однако такое предположение часто не выполнено даже в АЗ из-за наличия щелей и зазоров между конструктивными элементами [10]. Тем более оно не выполнено в слоях защиты и в окрестности ЯЭУ. В этом случае для определения потоков нейтронов используется интегро-дифференциальное уравнение переноса.

Для задания распределения материалов в 3D области расчета с декартовой геометрией используются либо структурированные сетки с ячейками-параллелепипедами, либо неструктурированные сетки с ячейками-тетраэдрами, гексаэдрами, призмами. Структурированные сетки легче в построении. Кроме того, сеточные схемы на ячейках-параллелепипедах обладают меньшей погрешностью аппроксимации, чем на ячейках-тетраэдрах и призмах [9]. С другой стороны, грани ячеек-параллелепипедов не могут точно передать границы между материалами. Только с помощью неструктурированных сеток могут быть корректно описаны локальные неоднородности в области расчета [7]. Однако при переходе от структурированных сеток к неструктурированным возникают дополнительные трудности, связанные с построением сеточных схем, итерационных алгоритмов, распараллеливанием вычислений и обработкой результатов расчетов.

В настоящей работе описаны возможности и алгоритмы программого комплекса РАДУГА-Т, предназначенного для решения интегро-дифференциального уравнения переноса в 3D областях на неструктурированных сетках на гибридных многопроцессорных системах. Статья имеет следующую структуру. В разделе 1 представлена постановка краевой задачи для уравнения переноса нейтронов. В разделе 2 обсуждаются методы ее решения; описаны методы, используемые в комплексе РАДУГА-Т и методы, применяемые в других комплексах. В разделе 3 перечислены возможности и алгоритмы программного комплекса РАДУГА-Т. В разделе 4 приведены результаты исследования эффективности распараллеливания вычислений в программном комлексе РАДУГА-Т. В заключении приводится краткая сводка результатов, полученных в работе, и указаны направления дальнейших исследований.

## 1. Краевая задача для уравнения переноса нейтронов

Уравнение переноса нейтронов имеет следующий вид

$$\partial\Psi/\partial\mathbf{\Omega} + \sigma(\mathbf{r},E)\Psi(\mathbf{r},\mathbf{\Omega},E) = \iint \Psi(\mathbf{r},\mathbf{\Omega}',E')\,\sigma_s(\mathbf{r},\mathbf{\Omega}\cdot\mathbf{\Omega}',E,E')d\mathbf{\Omega}'\,dE' + \tag{1}$$

$$+\frac{1}{4\pi\,k_{eff}}\,\chi(E)\iint \Psi(\mathbf{r},\mathbf{\Omega}',E')\,\sigma_f(\mathbf{r},E')d\mathbf{\Omega}'\,dE' + F(\mathbf{r},\mathbf{\Omega},E),\ E_{\min} < E < E_{max},\ \mathbf{r}\in G,\ \mathbf{\Omega}\in\Omega.$$

Здесь решение $\Psi(\mathbf{r},\mathbf{\Omega},E)$ — плотность потока нейтронов кинетической энергии $E$ из интервала $(E_{\min},E_{\max})$ в пространственной точке $\mathbf{r}$ выпуклой области $G$, движущихся в направлении единичного вектора $\mathbf{\Omega}$, $\Omega$ — единичная сфера.

Величина $\sigma(\mathbf{r},E)$ — полное сечение взаимодействия со средой в точке $\mathbf{r}$ нейтрона энергии $E$; левая часть уравнения (1) отвечает уменьшению числа нейтронов в единичном объеме фазового пространства $(\mathbf{r},\mathbf{\Omega},E)$ за счет пролета нейтронов без столкновения с атомами среды и за счет их поглощения и рассеяния атомами.

Первый интеграл в правой части уравнения (1) отвечает за увеличение числа нейтронов в единичном объеме фазового пространства при изменении энергии $E'$ и направления движения $\mathbf{\Omega}'$ нейтрона за счет рассеяния на атомах среды. Коэффициент $\sigma_s(\mathbf{r},\mathbf{\Omega}\cdot\mathbf{\Omega}',E,E')$ есть сечение рассеяния. Отметим, что сечение рассеяния зависит от скалярного произведения $\mathbf{\Omega}\cdot\mathbf{\Omega}'$ векторов $\mathbf{\Omega}$ и $\mathbf{\Omega}'$ , то есть от косинуса угла рассеяния. Второй интеграл в правой части уравнения (1) отвечает увеличению числа нейтронов за счет деления атомов. Коэффициент $\sigma_f(\mathbf{r},E')$ есть сечение деления, функция $\chi(E)$ задает спектр образующихся при делении нейтронов, $k_{eff}$ — коэффициент размножения. Функция $F(\mathbf{r},\mathbf{\Omega},E)$ есть плотность источника нейтронов.

Краевые условия для уравения (1) задаются на границе $\partial G$ области $G$

$$\Psi(\mathbf{r}^*,\mathbf{\Omega},E) = f(\mathbf{r}^*,\Psi(\mathbf{r}^*,\mathbf{\Omega},E)) \ \text{ при } \ \mathbf{r}^*\in\partial G,\ \ \mathbf{n}(\mathbf{r}^*)\cdot\mathbf{\Omega} < 0. \tag{2}$$

Здесь $\mathbf{n}(\mathbf{r}^*)$ — внешняя нормаль к границе $\partial G$ в точке $\mathbf{r}^*$, $f$ — кусочно-линейная функция Таким образом, краевые условия в каждой точке $\mathbf{r}^*$ границы $\partial G$ задаются для тех направлений скорости $\mathbf{\Omega}$, которые имеют тупой угол с внешней нормалью, то есть отвечают входящим в область $G$ нейтронам.

Отметим, что существует два варианта задачи (1), (2).

– В предположении $k_{eff} = 1$ имеем задачу с источником $F(\mathbf{r}, \mathbf{\Omega}, E)$.
– В предположении $F(\mathbf{r}, \mathbf{\Omega}, E) \equiv 0$ имеем задачу определения коэффициента $k_{eff}$.

Для решения задачи (1), (2) вводится сетка (как правило, неравномерная) по энергии $E$, состоящая из $Q$ интервалов (групп)

$$E_{\max} = E_1 > E_2 > \cdots > E_Q > E_{Q+1} = E_{\min}. \tag{3}$$

Задача (1), (2) принимает следующий вид

$$\hat{L}_q[\Psi_q] = \sum_{p=1}^{Q} \hat{S}_{p,q}[\Psi_p] + \hat{S}_f[\Psi_p]/k_{eff} + F_q(\mathbf{r}, \mathbf{\Omega}), \tag{4}$$

$$\hat{L}_q[\Psi_q] = \partial\Psi_q/\partial\mathbf{\Omega} + \sigma_q(\mathbf{r})\Psi_q(\mathbf{r}, \mathbf{\Omega}), \qquad \hat{S}_{p,q}[\Psi_p] = \int \Psi_p(\mathbf{r}, \mathbf{\Omega}')\, \sigma_{s,p,q}(\mathbf{r}, \mathbf{\Omega}\cdot\mathbf{\Omega}')d\mathbf{\Omega}', \tag{5}$$

$$\hat{S}_f[\Psi_p] = \frac{1}{4\pi}\, \chi_q \sum_{p=1}^{Q} \sigma_{f,p}(\mathbf{r}) \int \Psi_p(\mathbf{r}, \mathbf{\Omega}')\, d\mathbf{\Omega}', \tag{6}$$

$$\Psi_q(\mathbf{r}^*, \mathbf{\Omega}) = f(\mathbf{r}^*, \Psi_q(\mathbf{r}^*, \mathbf{\Omega})) \ \text{ при } \ \mathbf{r}^* \in \partial G, \ \ \mathbf{n}(\mathbf{r}^*)\cdot\mathbf{\Omega} < 0, \quad q = 1, \ldots, Q. \tag{7}$$

Здесь $\Psi_q(\mathbf{r}, \mathbf{\Omega})$ — плотность группового потока нейтронов с энергией из интервала $E_q < E < E_{q-1}$, функция $F_q(\mathbf{r}, \mathbf{\Omega})$ — плотность источника нейтронов с энергией из $q$-го интервала. Величины $\sigma_q(\mathbf{r})$, $\sigma_{s,p,q}(\mathbf{r}, \mathbf{\Omega}\cdot\mathbf{\Omega}')$, $\sigma_{f,p}(\mathbf{r})$ суть групповые сечения взаимодействия, рассеяния и деления, $\chi_q$ — групповой спектр образующихся при делении нейтронов.

Решение задачи (4)–(7) может быть найдено методом Зейделя

$$\hat{L}_q[\Psi_q^{i+1}] = \sum_{p=1}^{Q} \hat{S}_{p,q}[\Psi_p^{i+1}] + \hat{S}_f[\Psi_p^{i}]/k_{eff}^{i} + F_q(\mathbf{r}, \mathbf{\Omega}), \quad k_{eff}^{i+1} = k_{eff}^{i}\ \hat{S}_f[\Psi_p^{i+1}]/\hat{S}_f[\Psi_p^{i}], \tag{8}$$

$$\Psi_q^{i+1}(\mathbf{r}^*, \mathbf{\Omega}) = f(\mathbf{r}^*, \Psi_q^{i+1}(\mathbf{r}^*, \mathbf{\Omega})) \ \text{ при } \ \mathbf{r}^* \in \partial G, \ \ \mathbf{n}(\mathbf{r}^*)\cdot\mathbf{\Omega} < 0, \quad q = 1, \ldots, Q, \tag{9}$$

где $i$ — индекс итерации. Эти итерации первого уровня. Они также называются внешними итерациями. Для решения задачи (8), (9) тоже используется метод Зейделя

$$\hat{L}[\Psi_q^{k+1}] = \sum_{p=1}^{q} \hat{S}_{p,q}[\Psi_p^{k+1}] + \sum_{p=q+1}^{Q} \hat{S}_{p,q}[\Psi_p^{k}] + \hat{S}_f[\Psi_p]/k_{eff} + F_q(\mathbf{r}, \mathbf{\Omega}), \tag{10}$$

$$\Psi_q^{k+1}(\mathbf{r}^*, \mathbf{\Omega}) = f(\mathbf{r}^*, \Psi_q^{k+1}(\mathbf{r}^*, \mathbf{\Omega})) \ \text{ при } \ \mathbf{r}^* \in \partial G, \quad \mathbf{n}(\mathbf{r}^*)\cdot\mathbf{\Omega} < 0, \quad q = 1, \ldots, Q, \tag{11}$$

где $k$ — индекс итерации. Это итерации второго уровня.

Поскольку только медленные нейтроны при взаимодействии со средой могут увеличить свою кинетическую энергию, а группы нумерованы по убыванию энергии, см. (3), то при $p > q$ большинство сечений рассеяния $\sigma_{s,p,q} = 0$. Если при $p > q$ все сечения $\sigma_{s,p,q} = 0$, то в методе (10), (11) выпоняется одна итерация. В любом случае в методе (10), (11) на каждой итерции последовательно для всех энергетических групп решаются задачи

$$\hat{L}_q[\Psi_q] = \hat{S}_{q,q}[\Psi_q] + \tilde{F}_q(\mathbf{r}, \mathbf{\Omega}), \tag{12}$$

$$\Psi_q(\mathbf{r}^*, \mathbf{\Omega}) = f(\mathbf{r}^*, \Psi_q(\mathbf{r}^*, \mathbf{\Omega})) \ \text{ при } \ \mathbf{r}^* \in \partial G, \ \ \mathbf{n}(\mathbf{r}^*)\cdot\mathbf{\Omega} < 0, \quad q = 1, \ldots, Q. \tag{13}$$

Здесь правая часть $\tilde{F}_q(\mathbf{r}, \mathbf{\Omega})$ содержит как плотность источника $F_q(\mathbf{r}, \mathbf{\Omega})$, так и интегралы от плотностей потока в других группах. Базовой при моделировании потоков нейтронов является краевая задача (12), (13).

# 2. Методы решения базовой задачи

## 2.1. Сетки и сеточные алгоритмы

При решении сеточным методом задачи (12), (13) на единичной сфере $\Omega$ вводится набор узлов $\{\mathbf{\Omega}_m\}$, $m$=1,...,$M$, каждая функция $\Psi_q(\mathbf{r},\mathbf{\Omega})$ заменяется функциями $\Psi_{q,m}(\mathbf{r}) = \Psi_q(\mathbf{r},\mathbf{\Omega}_m)$, интеграл в правой части (12) заменяется *квадратурной формулой* [1].

Далее в пространственной области $G$ вводится *неструктурированная сетка* из ячеек-выпуклых многогранников. Используются ячейки-тетраэдры [2, 4, 12, 14, 20], произвольные гексаэдры, треугольные и четырехугольные призмы [4]. Тетраэдры универсальны, они позволяют аппроксимировать любые границы между материалами. Треугольные призмы хорошо приспособлены для задания тепловыделяющих элементов (ТВЭЛ) в АЗ и удобны для моделирования движения стержней в ТВЭЛах. Использование произвольных гексаэдров позволяет уменьшить число ячеек в однородных подобластях.

В каждой ячейке функция $\Psi_{q,m}(\mathbf{r})$ представляется своими значениями в каких-либо выбранных точках, например, в вершинах. Аналогично эта функция представляется на каждой грани каждой ячейки. Причем лучше использовать так называемые «разрывные» сеточные аппроксимции, когда значения решения в вершинах грани, общей для двух ячеек, предполагаются различными в каждой из этих ячеек.

Сеточная схема для уравнения переноса (12) строится отдельно в каждой ячейке. При этом используется та особенность уравнения переноса (12), что для каждого вектора направления скорости $\mathbf{\Omega}_m$ грани каждой ячейки разделяются на «входные», для которых $\mathbf{\Omega}_m \cdot \mathbf{n} < 0$, и «выходные», для которых $\mathbf{\Omega}_m \cdot \mathbf{n} > 0$, где $\mathbf{n}$ — внешняя нормаль к грани. Для уравнения (12) в ячейке краевые условия вида (13) задаются на «входных» гранях.

При построении сеточных схем для уравнения переноса применяется либо метод конечных элементов, либо метод характеристик, либо slice-balance метод. В *методе конечных элементов* (МКЭ) [2, 5] в ячейках и на гранях искомое решение и правая часть представляются разложением по базисным функциям. При этом коэффициентами разложения служат значения решения и правой части в вершинах ячеек и граней. Сеточные уравнения находятся после подстановки разложений в уравнение (12) и интегрирования получившегося соотношения с различными весовыми функциями. В *методе характеристик* (МХ) [15, 20] уравнение (12) решается отдельно в каждой ячейке в предположении, что на «входных» гранях значения решения известны. Далее находятся значения построенного решения в вершинах ячейки и «выходных» граней. В *slice-balance методе* [19] ячейка разбивается на подъячейки плоскостями, параллельными вектору $\mathbf{\Omega}_m$. Тогда в каждой подъячейке имеется только одна «входная» и одна «выходная» грани. Сеточная схема строится отдельно в каждой подъячейке методом конечных элементов и выполняется интерполяция на всю ячейку решений в подъячейках. Slice-balance метод является гибридом методов конечных элементов и характеристик.

В программном комплексе РАДУГА-Т расчет может быть выполнен на сетке из ячеек-выпуклых многогранников, среди которых могут быть тетраэдры и треугольные призмы. В комплексе реализованы схемы метода характеристик для ячеек-тетраэдров, метода конечных элементов для ячеек-тетраэдров и треугольных призм [7, 9]. Для ячеек иных типов в настоящее время используется шаговая схема первого порядка точности [1].

## 2.2. Итерационные методы решения базовой задачи

Грани ячеек, аппроксимирующие границу $\partial G$, будем называть граничными. Остальные грани будем называть внутренними. Для каждого вектора направления скорости $\mathbf{\Omega}_m$ все граничные грани делятся на «входные», где $\mathbf{\Omega}_m \cdot \mathbf{n}(\mathbf{r}^*) < 0$, и «выходные», где $\mathbf{\Omega}_m \cdot$

$\mathbf{n}(\mathbf{r}^*) > 0$. Определим следующие вектора.

- Вектор $\mathbf{\Psi}$ содержит значения решения для всех векторов $\mathbf{\Omega}_m$ в ячейках, на внутренних гранях и на «выходных» граничных гранях.
- Вектор $^{-}\mathbf{\Psi}$ содержит значения решения на граничных «входных» гранях.
- Вектор $^{+}\mathbf{\Psi}$ содержит значения решения на граничных «выходных» гранях.
- Вектор $\mathbf{f}$ содержит значения граничного источника на «входных» гранях.
- Вектор $\mathbf{F}$ содержит значения правой части в ячейках.

Тогда сеточные уравнения, аппроксимирующие задачу (12), (13), можно записать в форме

$$\hat{\mathbf{L}}\,\mathbf{\Psi} = \hat{\mathbf{S}}\,\mathbf{\Psi} + \mathbf{F}, \qquad {}^{-}\mathbf{\Psi} = \mathbf{f}({}^{+}\mathbf{\Psi}). \tag{14}$$

Матрицы $\hat{\mathbf{L}}$ и $\hat{\mathbf{S}}$ аппроксимируют операторы $\hat{L}$ и $\hat{\mathrm{S}}$ в уравнении (12) соответственно.

Для решения системы уравнений (14) можно использовать метод простой итерации

$$\hat{\mathbf{L}}\,\mathbf{\Psi}^{n+1} = \hat{\mathbf{S}}\,\mathbf{\Psi}^{n} + \mathbf{F}, \qquad {}^{-}\mathbf{\Psi}^{n+1} = \mathbf{f}({}^{+}\mathbf{\Psi}^{n}). \tag{15}$$

Здесь $n$ — номер итерации. Это итерации третьего уровня или внутренние итерции.

Одна итерация в методе (15) требует обращения оператора $\hat{\mathbf{L}}$. Для каждого узла $\mathbf{\Omega}_m$ элементы вектора $\mathbf{\Psi}^{n+1}$ находятся последовательным расчетом во всех пространственных ячейках. Первой рассчитывается прилегающая к границе $\partial G$ ячейка, для которой все граничные грани являются «входными». Найденные при этом значения решения на «выходных» гранях являются значениями решения на «входных» гранях для соседних ячеек. Подчеркнем, что порядок расчета ячеек зависит от вектора $\mathbf{\Omega}_m$.

Поскольку простые итерации могут сходиться очень медленно, вместо (15) может использоваться двухшаговый итерационный KP1 метод [3]

$$\hat{\mathbf{L}}\,\mathbf{\Psi}^{n+1/2} = \hat{\mathbf{S}}\,\mathbf{\Psi}^{n} + \mathbf{F}, \qquad {}^{-}\mathbf{\Psi}^{n+1/2} = \mathbf{f}({}^{+}\mathbf{\Psi}^{n}), \tag{16}$$

$$\hat{\mathbf{D}}\,\hat{\mathbf{L}}\,\boldsymbol{\psi}^{n+1/2} = \hat{\mathbf{D}}\,\hat{\mathbf{S}}\,\boldsymbol{\psi}^{n+1/2} + \hat{\mathbf{D}}\,\hat{\mathbf{S}}\left(\mathbf{\Psi}^{n} - \mathbf{\Psi}^{n+1/2}\right), \qquad \mathbf{\Psi}^{n+1} = \mathbf{\Psi}^{n+1/2} + \hat{\mathbf{P}}\,\boldsymbol{\psi}^{n+1/2}. \tag{17}$$

Здесь на втором P1-шаге (17) к решению, полученному методом простой итерации на первом K-шаге (16), добавляется ускоряющая поправка $\boldsymbol{\psi}^{n+1/2}$. Поправка $\boldsymbol{\psi}^{n+1/2}$ ищется как линейная функция компонент вектора направления скорости $\mathbf{\Omega}_m$ [3]. Уравнение для поправки находится суммированием по индексу $m$ уравнения для ошибки $\mathbf{\Psi} - \mathbf{\Psi}^{n+1/2}$ (это суммирование реализует оператор $\hat{\mathbf{D}}$). В пространство искомого решения $\mathbf{\Psi}$ поправка $\boldsymbol{\psi}^{n+1/2}$ переводится оператором $\hat{\mathbf{P}}$.

Также используется DSA (Diffusive Synthetic Acceleration) метод, где поправка $\boldsymbol{\psi}^{n+1/2}$ ищется как функция, не зависящая от вектора $\mathbf{\Omega}_m$.

Достоинство KP1 и DSA методов заключается в том, что ускоренный процесс (16), (17) сходится к тому же решению, что и неускоренный процесс (15). Более популярен метод грубосеточного ребаланса [2, 4, 15], не обладающий таким качеством. Однако в нем усреднение выполняется еще и по группам пространственных ячеек, что еще больше снижает размерность поправки. В программном комплексе РАДУГА-Т из двухшаговых итерационных методов используется только KP1 метод.

### 2.3. Распараллеливание вычислений на распределенной памяти

Разобьем пространственную сетку на $J$ непересекающихся подобластей, каждую подобласть присоединим к своему процессору. Метод простой итерации (15) принимает вид

$$\hat{\mathbf{L}}_j\,\mathbf{\Psi}_j^{n+1} = \hat{\mathbf{S}}_j\,\mathbf{\Psi}_j^{n} + \mathbf{F}_j, \qquad {}^{-}\mathbf{\Psi}_j^{n+1} = \mathbf{f}({}^{+}\mathbf{\Psi}_{j'}^{n},\ j' = 1,\ldots,J,\ j' \neq j), \quad j{=}1,\ldots,J, \tag{18}$$

или

$$\hat{\mathbf{L}}_j\,\boldsymbol{\Psi}_j^{n+1} = \hat{\mathbf{S}}_j\,\boldsymbol{\Psi}_j^{n} + \mathbf{F}_j, \qquad {}^{-}\boldsymbol{\Psi}_j^{n+1} = \mathbf{f}({}^{+}\boldsymbol{\Psi}_{j'}^{n+1},\ j' = 1,\dots,J,\ j' \neq j) \quad j{=}1,\dots,J. \tag{19}$$

Здесь $j$ и $j'$ — индексы подобластей. Уравнения (18) отвечают BJ (Block-Jacobi) методу, а уравнения (19) — BS (Block-Seidel) методу.

В *BJ методе* итерации выполняются как по правой части, так и по значениям решения на границе подобласти. В *BS методе* итерации выполняются только по правой части. Это достигается за счет того, что каждая итерация совершается за несколько проходов, между проходами выполняется обмен данными между соседними подобластями. Фактически в BS методе выполняется сквозной расчет по всей области, как при расчете на одной подобласти. При этом значения плотности потока нейтронов, выходящих из одной подобласти и входящих в другую, пересылаются между соответствующими процессорами.

В BJ методе число итераций растет с ростом числа подобластей. В BS методе число итераций не зависит от числа подобластей, но каждая итерация требует большего времени, чем итерация BJ метода. Задержки возникают вследствие того, что на разных проходах каждый процессор рассчитывает разное число ячеек. Кроме того, число актов обмена очень велико. Таким образом, часть времени процессор проводит в ожидании получения необходимых ему для расчета данных от других процессоров.

После завершения простой итерации (18) или (19) может быть вычислена ускоряющая поправка в соответствии с уравнением (17). Подчеркнем, что уравнение (17) решается во всей области. При применении на первом шаге BJ метода уравнение для поправки меняется при увеличении числа подобластей и поправка $\boldsymbol{\psi}^{n+1/2}$ ускоряет итерации как по правой части, так и по значениям решения на границах подобластей. При использовани на первом шаге BS метода уравнение для поправки не зависит от числа подобластей и поправка ускоряет только итерации по правой части.

Основная трудность при применении BS метода состоит в нахождении такой последовательности расчета ячеек, при котором было бы минимально время ожидания процессором данных от соседних процессоров. В то же время использование ускоряющей поправки позволяет снизить число простых итераций. Поэтому в некоторых комплексах применяется BJ метод [12, 15, 20]. Тем не менее развиваются и методы построения оптимальных последовательностей расчета ячеек для BS метода. Особенно популярен здесь KBA метод, названный по именам его авторов (Koch, Baker, Alcouffe) [13, 16, 17].

В программном комплексе РАДУГА-Т реализованы и BJ метод, и BS метод, причем каждый из них может быть выполнен как при с использованием, так и без использования ускоряющей процедуры (17). Таким образом, возможно применение одного из четырех итерационных процессов: BJ, BJ+P1, BS, BS+P1.

Решение уравнения для поправки в программном комплексе РАДУГА-Т также выполняется параллельно, тогда как в других комплексах ее решение поручается одному процессору [15]. В комплексе РАДУГА-Т для решения системы (17) используется параллельный метод минимальных невязок. Данный метод опирается на построение системы базисных векторов; причем основные совершаемые операции — это вычисление скалярных произведений и умножение матрицы на вектор. В параллельном счете каждый процессор выполняет операции со своими элементами векторов и матриц; далее после межпроцессорных обменов формируются единые для всей области базисные вектора.

## 2.4. Дополнительное распараллеливание на общей памяти

Поскольку процессоры обычно являются многоядерными, наряду с распараллеливанием на верхнем уровне на распределенной памяти возможно дополнительное распараллеливание на нижнем уровне на общей памяти.

При обращении оператора $\hat{\mathbf{L}}_j$ распараллеливание на нижнем уровне возможно и по пространственным ячейкам, и по узлам $\boldsymbol{\Omega}_m$, и по энергетическим группам. Распараллеливание

по энергетическим группам приводит к увеличению числа итераций в методе Зейделя (8), (9). Оно применяется в [15], где при построении ускоряющей поправки выполняется и усреднение по энергии, и, тем самым, ускорение итераций (8), (9). Распараллеливание по пространственым ячейкам на нижнем уровне привлекается, если используется BS метод сквозного счета [13, 16, 17]. Оптимально распараллеливание по узлам $\mathbf{\Omega}_m$, так как расчет для всех узлов выполняется независимо. В программном комплексе РАДУГА-Т, также как в [20], распараллеливание на нижнем уровне выполняется только по узлам $\mathbf{\Omega}_m$.

Распараллеливание нижнего уровня в методе минимальных невязок при решении системы линейных алгебраических уравнений для поправки (17) в программном комплексе РАДУГА-Т выполняется по элементам матриц и векторов. На обоих шагах (К и Р1) необходимо выполнять большое число актов обмена. Для выполнения обменов асинхронно выделяется одно процессорное ядро.

## 3. Программный комплекс РАДУГА-Т

Программный комплекс РАДУГА-Т предназначена для решения краевых задач для уравнения переноса нейтронов (1)–(2) в групповой аппроксимации (3)–(7) сеточным методом на параллельных компьютерах. Перечислим кратко его основные характеристики.

### 3.1. Основные возможности программного комплекса

В программном комплексе реализована возможность расчета на неструктурированной *сетке*, состоящей из ячеек-выпуклых многогранников. Единственное требование к сетке — отсутствие «висячих» вершин, то есть вершина каждой грани должна быть вершиной хотя бы одной другой грани. Допустимо присутствие в сетке ячеек разных типов.

В программном комплексе реализованы *сеточные схемы* метода конечных элементов (для ячеек-тетраэдров и треугольних призм), и сеточные схемы метода характеристик (для ячеек тетраэдров). Если для ячейки какого-либо типа не реализована сеточная схема, то расчет в ней проводится по шаговой схеме первого порядка точности [1].

Для решения системы сеточных уравнений может быть использован один из четырех *итерационных методов*:

– BJ (Block-Jacobi) — двойные итерации по правой части и граничным значениям решения без ускорения сходимости;
– BJ+P1 — BJ с ускорением сходимости внутренних итераций;
– BS (Block-Seidel) — итерации по правой части без ускорения сходимости;
– BS+P1 — BS с ускорением сходимости внутренних итераций.

Внешние итерации выполняются без ускорения.

При вычислениях используется двухуровневый *алгоритм распараллеливания*, где на верхнем уровне распараллеливание выполняется по пространственным подобластям, на нижнем уровне — по направлениям скорости $\mathbf{\Omega}_m$.

*Разбиение пространственной сетки* на подобласти выполняется в самом программном комплексе одним из трех методов: жадным, бисекции или координатным. Время расчета существенно зависит от использованного метода разбиения сетки [6].

Возможен *счет с продолжением*, счет с ненулевым начальным приближением.

Отличие программного комплекса РАДУГА-Т от других комплексов решения уравнения переноса на неструктурированных сетках [2, 5, 11, 12, 15, 20] состоит в том, что программный комплекс РАДУГА-Т включает большой набор алгоритмов и методов: сеточные схемы двух типов, 4 итерационных алгоритма, 3 встроенных алгоритма декомпозиции сетки, возможность расчета на сетках с ячейками разных типов. Сравнение характеристик программного комплекса РАДУГА Т и других комплексов представлено в таблице. Прочерк означает отсутствие данных.

**Таблица**

Сравнение характеристик программных комплексов для решения уравнения переноса нейтронов

| **Программный комплекс** | **Типы ячеек** | **Сеточные схемы** | **Итерационный метод** | **Средства и переменные распараллеливания** |
|---|---|---|---|---|
| ATTILA [11] | тетраэдр | МКЭ | BJ, BJ+DSA | OpenMP (—) |
| THOR [20] | тетраэдр | МХ | BJ | MPI (**r**, **Ω**) |
| ODETTA [2] | тетраэдр | МКЭ | BJ, BJ+Ребаланс | OpenMP (**Ω**) |
| ARES [12] | тетраэдр | МКЭ | BJ | — |
| APOLLO [15] | тетраэдр | МХ | BJ | MPI+OpenMP (**r**, **Ω**, $E$) |
| PMSNSYS [5] | тетраэдр | МКЭ | BS, BS+Ребаланс | MPI (**Ω**) + OpenMP (**r**) |
| **РАДУГА-Т** | **тетраэдр треугольная призма** | **МКЭ МХ** | **BJ, BJ+P1 BS, BS+P1** | **MPI (r) + OpenMP (Ω)** |

### 3.2. Особенности программной реализации

Программный комплекс написан на языке FORTRAN, для параллельных вычислений используются команды MPI (верхний уровень) и OpenMP (нижний уровень).

Иные пакеты не используются. Программный комплекс совместим с операционными системами Windows и Unix.

Для экономии оперативной памяти в каждый момент времени расчета в ней находятся только те значения решения, которые относятся к рассчитываемой энергетической группе. После завершения расчета каждой группы происходят обмены с жестким диском.

Все итерационные методы в программном комплексе реализованы в виде единой системы вложенных циклов. Включение пространственных ячеек нового типа требует только подключения модуля обращения оператора $\hat{\mathbf{L}}_j$ в ячейках такого типа.

### 3.3. Подготовка пространственной сетки

Геометрия области расчета в задачах о переносе нейтронов в АЗ и в радиационных защитах реальных ЯЭУ является очень сложной, содержит множество разномасштабных элементов. Эта геометрия может быть задана самыми разными способами:

– с помощью одного из CAD-форматов;
– посредством специального языка в одной из программ моделирования переноса нейтронов методом Монте-Карло (отечественной MCU или зарубежной MCNP);
– с помощью бумажных чертежей.

Для построения сетки используется свободно распространяемый пакет SALOME, в котором геометрия может быть задана в отдельном входном файле на языке Python при помощи библиотеки API. Использование этого подхода позволяет сократить трудозатраты на обработку больших объемов геометрических данных, а также адаптироваться к появлению их новых форматов.

Создана специальная программа CONVERTGEOM, переводящая описание геометрии с языка MCU на язык Python. При задании геометрии в виде чертежей разработан «служебный» формат RAD описания геометрических данных. Файл в формате RAD преобразуется той же программой CONVERTGEOM в файл на языке Python. На основе этого файла с помощью пакета SALOME строится неструктурированная сетка.

Такая сетка не всегда обладает высоким качеством. В частности, сетка может быть излишне сгущена в одной подобласти и слишком разрежена в другой. Для улучшения качества сетки используется пакет ANIMBA [21].

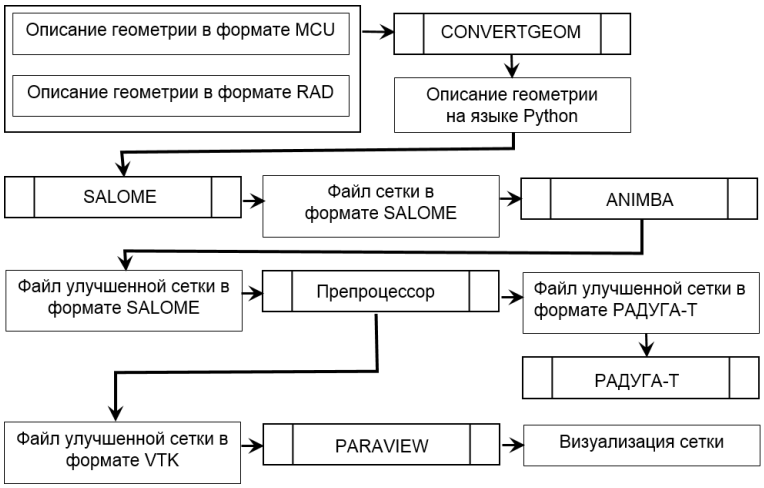



**Рис. 1.** Схема подготовки пространственной сетки

Следующий этап в подготовке сетки состоит в переводе ее в формат программного комплекса РАДУГА-Т. В этом формате содержится информация о материале каждой ячейки и о соседних с ней ячейках. Такой перевод выполняется Препроцессором программного комплекса РАДУГА-Т. Им же создаются файлы в формате VTK (Visualization ToolKit) для визуализации сетки с помощью пакета PARAVIEW. Схема подготовки пространственной сетки приведена на рис. 1.

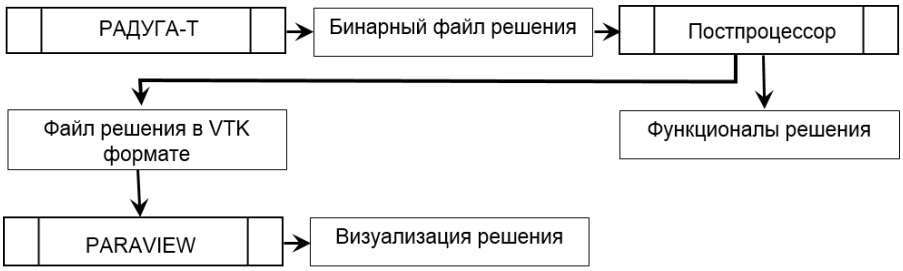



**Рис. 2.** Схема обработки решения Постпроцессором

После выполнения расчета уже Постпроцессором может быть выполнен расчет различных функционалов решения, в частности, средних значений решения в заданных телах (параллелепипедах, цилиндрах и др.). Также может быть выполнена интерполяция решения с неструктурированной сетки в точки с заданными координатами. Пакет PARAVIEW также может быть использован для визуализации решения, см. рис. 2.

## 4. Вычислительные эксперименты

В этом разделе приводятся результаты исследования эффективности алгоритма распараллеливания программного комплекса РАДУГА-Т. Рассматривается только метод BS+P1. Находится коэффициент $k_{eff}$ в двухгрупповой ($Q$=2) модели легководного реактора [18], см. рис. 3, с опущенным стержнем. Используется пространственная сетка из 108 тыс. ячеек, 271 тыс. граней. Сетка по направлениям скорости содержит $M = 80$ узлов.

Используется схема метода характеристик. Полное число неизвестных в уравнении переноса в одной энергетической группе — 74 млн., а в системе уравнений для поправки — 3,7 млн. Для ее решения используется 39 тыс. ортонормированных векторов.

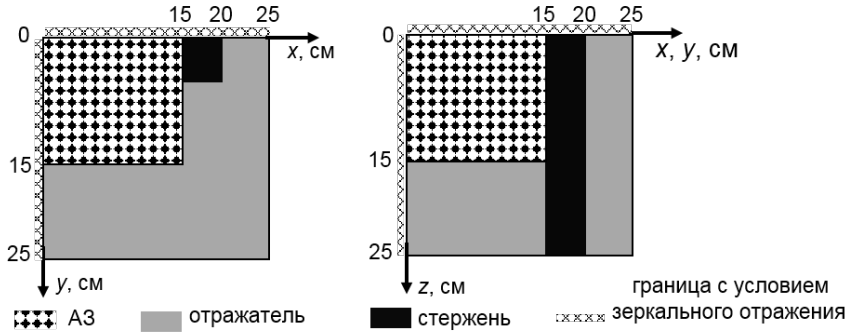



**Рис. 3.** Геометрия модели легководного реактора

Расчеты выполняются на многопроцессорном компьютере МВС-10П (МСЦ РАН). В расчете каждой подобласти используется 2 ядра. Одно из них выполняет вычисления, другое — межпроцессорные обмены. На одной внешней итерации (8), (9) выполняется 28 внутренних итераций (19).

Каждый расчет состоит из двух шагов: К и Р1. Каждый шаг разделяется на два этапа: вычисления и обмены, всего 4 этапа. Доли временных затрат (%) на каждый из 4-х этапов для различного числа подобластей $J$ приведены на рис. 4.

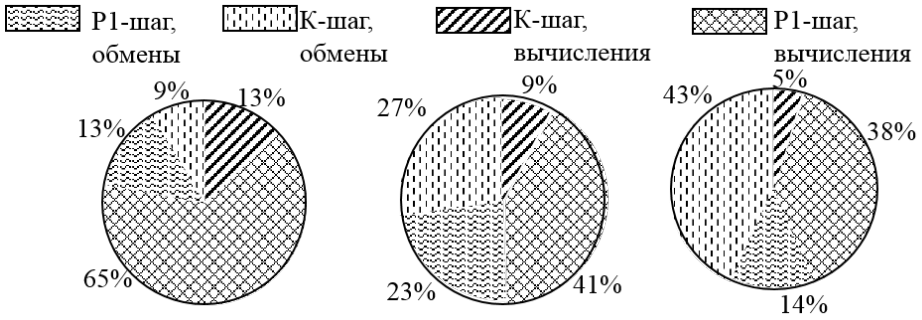



**Рис. 4.** Распределение времени по этапам расчета

Можно видеть, сколь большую часть времени занимают обмены, причем время обменов увеличивается с ростом числа подобластей $J$, особенно на К-шаге. Это связано ростом числа актов обмена, увеличивающегося из-за роста числа проходов, см. рис. 5. На Р1-шаге с ростом $J$ число базисных векторов не меняется. Здесь доля времени обменов сначала растет, затем снижается.

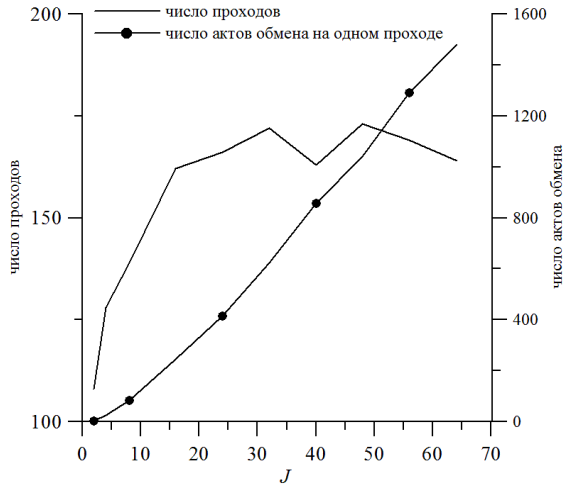



**Рис. 5.** Число проходов и актов обмена на К-шаге в зависимости от числа подобластей

На рис. 6 представлены величины $a$ ускорения вычислений всего расчета и отдельно каждого шага, найденные по формуле

$$a = T(2)/T(J). \tag{20}$$

Здесь $T(J)$ — время расчета на $J$ подобластях. Идеальное ускорение $a^{\text{Ideal}} = J/2$ на рис. 6 представлено штрихованной линией. Можно видеть, что эффективное ускорение вычислений на P1-шаге сочетается с малым ускорением на K-шаге. Более того, при увеличении числа подобластей ускорение вычислений на K-шаге прекращается. Это происходит за счет увеличения числа актов обмена данными между подобластями, см. рис. 5. Суммарное ускорение при небольшом числе подобластей определяется ускорением P1-шага. С ростом числа подобластей доля времени выполнения K-шага возрастает, см. рис. 4, и суммарное ускорение вычислений снижается.

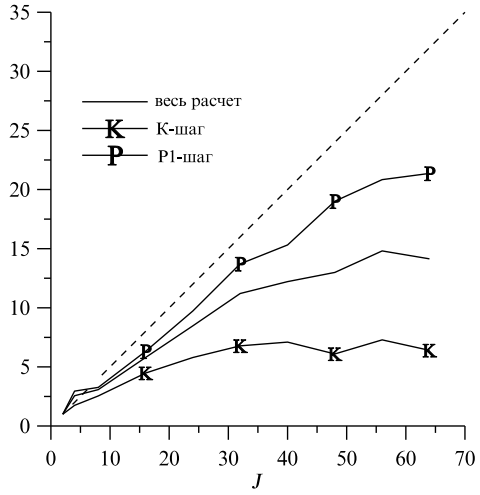



**Рис. 6.** Ускорения вычислений в зависимости о числа подобластей

Отметим, что выбранная для тестирования стратегия — использование в каждой подобласти двух ядер, одно из которых выполняет вычисления, а другое межпроцессорные обмены — представляется оптимальной для метода BS. В BS методе на каждом проходе для каждого вектора $\mathbf{\Omega}_m$ рассчитывается разное число пространственных ячеек, причем ячейки рассчитываются в строго заданном порядке. Использование для вычисления двух и более ядер в одной подобласти приводит к задержкам на синхронизацию. Для другого итерационного метода оптимальным будет другая стратегия.

## Заключение

Представлен программный комплекс РАДУГА-Т, предназначенный для решения краевой задачи для уравнения переноса нейтронов в 3D областях на неструктурированных сетках. Описаны способы задания геометрии области расчета в сеткопостроителе, используемые сеточные схемы и итерационные методы решения систем сеточных уравенений, методы распараллеливания вычислений.

Приведены результаты исследования эффективности распараллеливания одного из четырех итерационных методов на модельной задаче о легководном реакторе. Выбранный метод обладает тем преимуществом, что решаемые системы линейных алгебраических уравнений не зависят от числа пространственных подобластей. Таким образом, от числа подобластей не зависит и число итераций. Такой эффект, однако, достигается за счет роста числа актов межпроцессорных обменов при увеличении числа подобластей. В рас-

сматриваемой задаче время счета с увеличением числа подобластей от 2 до 64 уменьшеается всего в 14 раз. Для других итерационных методов результаты исследования эффективности распараллеливания будут иными.

Развитие комплекса РАДУГА-Т планируется по следующим направлениям.

- Разработка сеточной схемы для расчета в ячейке-четырехугольной пирамиде; это даст возможность выполнять расчет на комбинированных сетках, содержащих ячейки-тетраэдры и ячейки-призмы.
- Разработка методов ускорения сходимости внешних итераций (8), (9) и (10), (11).
- Повышение эффективности распараллеливания вычислений на K-шаге.

## Литература


1. Басс Л.П., Волощенко А.М., Гермогенова Т.А. Методы дискретных ординат в задачах о переносе излучения. Москва: ИПМ, 1986. 231 с.
2. Белоусов В.И., Грушин Н.А., Сычугова Е.П., и др. Некоторые результаты верификации кода ODETTA для неоднородных задач // Вопросы атомной науки и техники. Серия: Физика ядерных реакторов. 2018. № 3. С. 46–53.
3. Коконков Н.И., Николаева О.В. KP1 итерационный метод решения уравнения переноса в трехмерных областях на неструктурированных сетках // Журнал вычислительной математики и математической физики. 2015. Т. 55, № 10. С. 95–108. DOI: 10.7868/S0044466915100154.
4. Николаев А.А., Усенков В.В., Афанасьев П.Б., и др. Современное состояние развития программного обеспечения расчета переноса ионизирующего излучения в активных зонах и радиационной защите реакторных установок со свинцово-висмутовым теплоносителем // Вопросы атомной науки и техники. Серия: Ядерно-реакторные константы. 2017. № 1. С. 129–143.
5. Николаев А.А. Совершенствование геометрических опций SN кода PMSNSYS-II // Вопросы атомной науки и техники. Серия: Физика ядерных реакторов. 2017. № 1. С. 143–147.
6. Николаева О.В., Гайфулин С.А., Басс Л.П. О декомпозиции неструктурированной сетки при решении уравнения переноса нейтронов на параллельных компьютерах // Параллельные вычислительные технологии (ПаВТ'2019): Труды международной научной конференции (Калининград, 2–4 апреля 2019 г.). Челябинск: Издательский центр ЮУрГУ, 2019. С. 362–372.
7. Николаева О.В., Гайфулин С.А., Басс Л.П. Детальное моделирование эксперимента IRON 88 на установке ASPIS в (r,z)- и (x,y,z) геометриях // Известия вузов. Ядерная энергетика. 2019. № 3. С. 135–147. DOI: 10.26583/npe.2019.3.12.
8. Николаева О.В., Казанцева А.С. Сравнение свойств сеточных схем для решения уравнения переноса на неструктурированных тетраэдрических сетках // Вопросы атомной науки и техники. Серия: Математическое моделирование физических процессов. 2019. № 1. С. 3–18.
9. Николаева О.В., Казанцева А.С. Точность схем метода конечных элементов для решения уравнения переноса на неструктурированных тетраэдрических и призматических сетках // Вопросы атомной науки и техники. Серия: Математическое моделирование физических процессов. 2020. № 1. С. 3–18.
10. Селезнев Е.Ф., Березнев В.П. Использование диффузионного приближения при расчете реактора с полостями // Известия вузов. Ядерная энергетика. 2018. № 2. С. 67–77. DOI: 10.26583/npe.2018.2.07.

11. Vassiliev O.D., Wareing T.A., Davis I.M., et al. Feasibility of a Multigroup Deterministic Solution Method for 3D Radiotherapy Dose Calculations // International Journal of Radiative Oncology, Biology, Physics. 2008. Vol. 72 P. 220–227. DOI: 10.1016/j.ijrobp.2008.04.0572017.
12. Chen Y., Zhang B., Zhang L., et al. ARES: A Parallel Discrete Ordinates Transport Code for Radiation Shielding Applications and Reactor Physics Analysis // Hindawi Science and Technology of Nuclear Installations. 2017. Article ID 2596727. DOI: 10.1155/2017/2596727.
13. Colomer G., Borrell R., Trias F.X., et al. Parallel Algorithms for Sn Transport Sweeps on Unstructured Meshes // Journal of Computational Physics. 2013. Vol. 232. P. 118–135. DOI: 10.1016/j.jcp.2012.07.009.
14. Kim J.W., Lee Y.O. A Deep Penetration Problem Calculation Using AETIUS: An Easy Modeling Discrete Ordinates Transport Code UsIng Unstructured Tetrahedral Mesh, Shared Memory Parallel // EPJ Web of Conferences. 2017. Vol. 153. P. 06025. DOI: 10.1051/epjconf/20171530.
15. Lenain R., Masiello E., Damian F., et al. Domain Decomposition Method for 2D and 3D Transport Calculations Using Hybrid MPI/OPENMP Parallelelizm // Mathematics and Computation (M&C), Supercomputing in Nuclear Applications (SNA) and the Monte Carlo (MC) Method: Proc. of the Joint International Conference (Nashville, Tennessee, April, 19–23, 2015). LaGrange Park, IL, American Nuclear Society, 2015. URL: https://hal-cea.archives-ouvertes.fr/cea-02506817/document (дата обращения: 14.09.2020).
16. Pautz Sh.D. An Algorithm for Parallel Sn Sweeps on Unstructured Meshes // Nuclear Science and Engineering. 2002. Vol. 140, no 2. P. 111–136. DOI: 10.13182/NSE02-1.
17. Plimpton S.J., Hendrickson B., Burns Sh.P., et al. III, Rauchwerger L. Parallel Sn Sweeps on Unstructured Grids: Algorithms for Prioritization, Grid Partitioning, and Cycle Detection // Nuclear Science and Engineering. 2005. Vol. 150. P. 267–283. DOI: 10.13182/NSE150-267.
18. Takeda T., Ikeda H. 3-D Neutron Transport Benchmarks // Journal of Nuclear Science and Technology. 1991. Vol. 28, no 7. P. 656–669. DOI: 10.3327/jnst.28.656.
19. Vega R.M., Adams M.L. Transport Sweeps Using an Improved Slice Balance Approach with LDFE and GPU Acceleration // Mathematics & Computational Methods Applied to Nuclear Science & Engineering: Proc. of the International Conference (Jeju, Korea, April, 16–20, 2017). URL: https://www.kns.org/files/int_paper/paper/MC2017_2017_1/P056S01-11VegaR.pdf (дата обращения: 14.09.2020).
20. Yessayan R., Azmy Y., Schunert S. Development Of A Parallel Performance Model For The THOR Neutral Particle Transport Code // Mathematics & Computational Methods Applied to Nuclear Science & Engineering: Proc. of the International Conference (Jeju, Korea, April, 16–20, 2017). URL: https://www.osti.gov/servlets/purl/1369430 (дата обращения: 14.09.2020).
21. Adaptive Numerical Instruments 3D. URL: https://sf.net/p/ani3d/ (дата обращения: 01.09.2020).

Николаева Ольга Васильевна, к.ф.-м.н., с.н.с., Институт прикладной математики им. М.В. Келдыша РАН (Москва, Российская Федерация)

Гайфулин Сергей Андреевич, н.с., Институт прикладной математики им. М.В. Келдыша РАН (Москва, Российская Федерация)

Басс Леонид Петрович, к.ф.-м.н., с.н.с., Институт прикладной математики им. М.В. Келдыша РАН (Москва, Российская Федерация)

DOI: 10.14529/cmse210106

# CODE RADUGA T FOR SIMULATING NEUTRONS FLUXES IN NUCLEAR POWER STATIONS

© 2021 O.V. Nikolaeva, S.A. Gaifulin, L.P. Bass
*Keldysh Institute of Applied Mathamatics (Miusskaya Sq. 4, Moscow, 125047 Russia)*
*E-mail: nika@kiam.ru, sagarp@kiam.ru, bass@kiam.ru*
Received: 14.07.2020

Design and operation of nuclear power stations (NPS) are followed by simulation of neutron propagation in these stations. It is necessary to consider borders of multi-scale structural elements consisting of different materials. It is desirable to use parallel computer because of big size of NPSs. To keep such conditions, algorithms and codes for solving the integro-differential transport equation on unstructured grids are being developed. In the paper such algorithms included into the code RADUGA T are outlined. Grids, grid schemes, iterative methods to solve grid equations are presented. Calculation parallelization methods for hybrid computers are considered, MPI and OpenMP techniques are used. Methods of building, improvement, decomposition and visuzlisation of spatial grids are considered. Software implementation is described. The algorithms of the code RADUGA T are compared with methods of other codes. Computation parallelization efficiency study results are presented. The problem of neutron multiplication factor calculation in a light-water reactor model is solved. The multi-processor computer MVS-10P of the Joint SuperComputernal Center is used. Acceleration of each algorithm being used and summary acceleration are given.

*Keywords: transport equation, unstructured frids, grid schemes, iteration methods, parallel calculation.*

**FOR CITATION**

Nikolaeva O.V., Gaifulin S.A., Bass L.P. Code RADUGA T for Simulating Neutrons Fluxes in Nuclear Power Stations. *Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering.* 2021. Vol. 10, no. 1. P. 75–89. (in Russian) DOI: 10.14529/cmse210106.



## References

1. Bass L.P., Voloschenko A.M., Germogenova T.A. Discrete Ordinate Method in Radiation Transport Problem. Moscow, KIAM Press, 1986. 231 p. (in Russian)
2. Belousov V.I., Grushin N.A., Sychugova E.P., et al. Some Results of Verification of Code ODETTA for Shielding Calculations. Nuclear Science and Technology: Nuclear Reactor Physics. 2018. No. 3. P. 46–53. (in Russian)
3. Kokonkov N.I., Nikolaeva O.V. An iterative KP1 Method for Solving the Transport Equation in 3D Domains on Unstructured Grids. Computational Mathematics and Mathematical Physics. 2015. Vol. 55, no. 10. P. 1698–1712. DOI: 10.7868/S0044466915100154.
4. Nikolaev A.A., Usenkov V.V., Afanasiev P.B., et al. Actual State of Development of the Software for Radiation Transport Calculatons in Core and Shield of Reactor Facities with Lead-Bismuth Coolant. Nuclear Science and Technology: Nuclear Reactor Physics. 2017. No. 1. P. 143–147. (in Russian)
5. Nikolaev A.A. Improvement of Geometric Options of SN-code PMSNSYS-II. Nuclear Science and Technology: Nuclear Reactor Physics. 2017. No. 1. P. 129–143. (in Russian)
6. Nikolaeva O.V., Gaifulin S.A., Bass L.P. On decomposition of a unstructured grid to solving the neutron transport equation on parallel computers. Parallel Computational Technologies (PCT'2019): Proceedings of the International Scientific Conference (Kaliningrad, Russia, April, 2–4, 2019). Chelyabinsk, Publishing of the South Ural State University, 2019. P. 362–372. (in Russian)

7. Nikolaeva O.V., Gaifulin S.A., Bass L.P. Detailed Simulation of Winfrith IRON 88 Benchmark (ASPIS) in (r, z)- and (x, y, z)-Geometries. Izvestiya vuzov. Yadernaya Energetika. 2019. No. 3. P. 135–147. DOI: 10.26583/npe.2019.3.12. (in Russian)
8. Nikolaeva O.V., Kazantseva A.S. Comparison Between the Properties of Grid Schemes for Solving the Transport Equation on Unstructured Thetrahedral Grids. Nuclear Science and Technology. Series: Mathematical Modeling of Physical Processes. 2019. No. 1. P. 3–18. (in Russian)
9. Nikolaeva O.V., Kazantseva A.S. Accuracy of FEM Schemes for Solving the Transport Equation on Unstructured Thetrahedral and Prismatic Grids. Nuclear Science and Technology. Series: Mathematical Modeling of Physical Processes. 2020. No. 1. P. 3–19. (in Russian)
10. Seleznev E.F., Bereznev V.P. Using the Diffusive Approximation for Reactor with Cavities Calculation. Izvestiya Vuzov. Yadernaya Energetika. 2018. No. 2. P. 66–67. DOI: 10.26583/npe.2018.2.07. (in Russian)
11. Vassiliev O.D., Wareing T.A., Davis I.M., et al. Feasibility of a Multigroup Deterministic Solution Method for 3D Radiotherapy Dose Calculations. International Journal of Radiative Oncology, Biology, Physics. 2008. Vol. 72. P. 220–227. DOI: 10.1016/j.ijrobp.2008.04.0572017.
12. Chen Y., Zhang B., Zhang L., et al. ARES: A Parallel Discrete Ordinates Transport Code for Radiation Shielding Applications and Reactor Physics Analysis. Hindawi Science and Technology of Nuclear Installations. 2017. Article ID 2596727. DOI: 10.1155/2017/2596727.
13. Colomer G., Borrell R., Trias F.X., et al. Parallel Algorithms for Sn Transport Sweeps on Unstructured Meshes. Journal of Computational Physics. 2013. Vol. 232. P. 118–135. DOI: 10.1016/j.jcp.2012.07.009.
14. Kim J.W., Lee Y.O. A Deep Penetration Problem Calculation Using AETIUS: An Easy Modeling Discrete Ordinates Transport Code UsIng Unstructured Tetrahedral Mesh, Shared Memory Parallel. EPJ Web of Conferences. 2017. Vol. 153. P. 06025. DOI: 10.1051/epjconf/20171530.
15. Lenain R., Masiello E., Damian F., Sanchez R. Domain Decomposition Method for 2D and 3D Transport Calculations Using Hybrid MPI/OPENMP Parallelelizm. Mathematics and Computation (M&C), Supercomputing in Nuclear Applications (SNA) and the Monte Carlo (MC) Method: Proceedings of the Joint International Conference (Nashville, Tennessee, April, 19–23, 2015). LaGrange Park, IL, American Nuclear Society, 2015. URL: https://hal-cea.archives-ouvertes.fr/cea-02506817/document (accessed: 14.09.2020).
16. Pautz Sh.D. An Algorithm for Parallel Sn Sweeps on Unstructured Meshes. Nuclear Science and Engineering. 2002. Vol. 140, no. 2. P. 111–136. DOI: 10.13182/NSE02-1.
17. Plimpton S.J., Hendrickson B., Burns Sh.P., et al. Parallel Sn Sweeps on Unstructured Grids: Algorithms for Prioritization, Grid Partitioning, and Cycle Detection. Nuclear Science and Engineering. 2005. Vol. 150. P. 267–283. DOI: 10.13182/NSE150-267.
18. Takeda T., Ikeda H. 3-D Neutron Transport Benchmarks. Journal of Nuclear Science and Technology. 1991. Vol. 28, no. 7. P. 656–669. DOI: 10.3327/jnst.28.656.
19. Vega R.M., Adams M.L. Transport Sweeps Using an Improved Slice Balance Approach with LDFE and GPU Acceleration. Mathematics & Computational Methods Applied to Nuclear Science & Engineering: Procedeengs of the International Conference (Jeju, Korea, April, 16–20, 2017). URL: https://www.kns.org/files/int_paper/paper/MC2017_2017_1/P056S01-11VegaR.pdf (accessed: 14.09.2020).
20. Yessayan R., Azmy Y., Schunert S. Development Of A Parallel Performance Model For The THOR Neutral Particle Transport Code. Mathematics & Computational Methods Applied to Nuclear Science & Engineering: Procedeengs of the International Conference (Jeju, Korea, April, 16–20, 2017). URL: https://www.osti.gov/servlets/purl/1369430 (accessed: 14.09.2020).
21. Adaptive Numerical Instruments 3D. URL: https://sf.net/p/ani3d/ (accessed: 14.09.2020).

## 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: 05.13.11 – Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей (физико-математические науки), 05.13.17 – Теоретические основы информатики (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, издателя: 454080, г. Челябинск, проспект Ленина, 76, Издательский центр ЮУрГУ, каб. 32.*

## ПРАВИЛА ДЛЯ АВТОРОВ

1. Правила подготовки рукописей и пример оформления статей можно загрузить с сайта серии `http://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, кафедра СП,

   ответственному секретарю Цымблеру М.Л.
3. Адрес электронной почты редакции: `vestnikvmi@susu.ru`
4. **Плата с авторов за публикацию рукописей не взимается, и гонорары авторам не выплачиваются.**

ВЕСТНИК
ЮЖНО-УРАЛЬСКОГО
ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА
Серия
«ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА И ИНФОРМАТИКА»
Том 10, № 1
2021

16+

Техн. редактор *А.В. Миних*

Издательский центр Южно-Уральского государственного университета

Подписано в печать 11.02.2021. Дата выхода в свет 18.02.2021. Формат 60×84 1/8. Печать цифровая.
Усл. печ. л. 10,69. Тираж 500 экз. Заказ 18/56. Цена свободная.

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