Суперкомпьютерный консорциум университетов России
Российская академия наук

Суперкомпьютерные дни в России 2025

МГУ 270 1755 2025

Труды международной конференции

29–30 сентября 2025 г., Москва

Суперкомпьютерный консорциум университетов России
Российская академия наук

СУПЕРКОМПЬЮТЕРНЫЕ ДНИ В РОССИИ

Труды международной конференции

29–30 сентября 2025 г., Москва

МАКС ПРЕСС

МОСКВА – 2025

УДК 519.7
ББК 22.18
С89

https://elibrary.ru/nzvyjl

Под редакцией члена-корреспондента РАН *Вл. В. Воеводина*
Ответственный редактор *А.С. Антонов*

Суперкомпьютерные дни в России : Труды международной конференции. 29–30 сентября 2025 г., Москва / Под. ред. Вл. В. Воеводина. – Москва : МАКС Пресс, 2025. – 268 с.

С89

ISBN:978-5-317-07451-7 e-ISBN: 978-5-317-07452-4
https://doi.org/10.29003/m4750.978-5-317-07451-7

Данный сборник содержит полные статьи на русском языке, короткие статьи и аннотации стендовых докладов, включенных в программу Международной конференции «Суперкомпьютерные дни в России».

УДК 519.7
ББК 22.18

Russian Supercomputing Days : Proceedings of the International Conference. September 29–30, 2025, Moscow, Russia / Ed. by Vl. Voevodin. – Moscow : MAKS Press, 2025. – 268 p.

ISBN:978-5-317-07451-7 e-ISBN: 978-5-317-07452-4
https://doi.org/10.29003/m4750.978-5-317-07451-7

The book contains full papers in Russian, short papers and poster abstracts included in the agenda of the International Conference “Russian Supercomputing Days”.

Подробную информацию о конференции можно найти в сети Интернет по адресу http://RussianSCDays.org

ISBN: 978-5-317-07287-27
e-ISBN: 978-5-317-07283-4
https://doi.org/10.29003/m4296.978-5-317-07283-4

Полные и короткие статьи

AlEM: новый параллельный алгоритм линейного программирования для кластерных вычислительных систем

А.Э. Жулев, Л.Б. Соколинский

Южно-Уральский государственный университет
(национальный исследовательский университет)

Работа посвящена новому масштабируемому проекционному алгоритму AlEM (Along Edges Movement) для линейного программирования на кластерных вычислительных системах. Алгоритм начинает свою работу в произвольной вершине многогранника допустимых решений и строит оптимальный путь по ребрам до точки оптимума. Представлено формализованное описание алгоритма. Описана его параллельная реализация. Исследована масштабируемость параллельной реализации на кластерной вычислительной системе. Приведены результаты вычислительных экспериментов, подтверждающие высокую параллельную эффективность алгоритма AlEM.

Ключевые слова: линейное программирование, алгоритм AlEM, проекционный алгоритм, параллельная реализация, MPI, кластерная вычислительная система, исследование масштабируемости.

1. Введение

Линейное программирование (ЛП) является одной из наиболее востребованных оптимизационных математических моделей, используемых для решения практических масштабных задач в индустрии, экономике, науке и других областях [1–3]. Наиболее популярными подходами к решению задач ЛП являются симплекс-метод [4] и метод внутренних точек [5]. Однако параллельные алгоритмы, основанные на этих методах, в общем случае не поддаются эффективному распараллеливанию на больших вычислительных системах с распределенной памятью. В соответствии с этим остается актуальной задача разработки новых высоко-масштабируемых методов и алгоритмов для решения задач ЛП.

Одним из перспективных подходов является использование проекционных методов для решения задач ЛП. Впервые проекционные методы были применены для решения систем линейных алгебраических уравнений Стефаном Качмажом [6] и Джанфранко Чиммино [7] в 1930-х годах. Идея проекционного метода состоит в последовательном или совместном ортогональном проектировании текущего приближения на гиперплоскости, соответствующие линейным алгебраическим уравнениям, образующим систему. Результатом будет следующее приближение, находящееся ближе к точному решению, чем предыдущее. Доказано, что такой итерационный процесс сходится к точному решению, если система совместна. Шмуэль Агмон [8] обобщил

проекционный метод для систем линейных неравенств (задача линейной совместности). В последние десятилетия класс проекционных методов активно развивался [9].

И.И. Ерёмин одним из первых предложил использовать проекционные методы для решения задач ЛП [10]. Затем этот подход был развит в целом ряде работ (см., например, [11–15]). Главным недостатком проекционных методов является их низкая линейная скорость сходимости, зависящая от угла между гиперплоскостями [16]. С другой стороны, проекционные методы допускают эффективное распараллеливание на многопроцессорных системах с распределенной памятью, так как ортогональные проекции могут вычисляться независимо [17]. Однако, как показано в [15], граница эффективной масштабируемости параллельного проекционного метода не превышает $O(\sqrt{m})$ процессорных узлов с распределенной памятью, где m — число ограничений задачи ЛП.

В работе [15] был представлен апекс-метод — принципиально новый проекционный метод решения задач ЛП, строящий на поверхности допустимого многогранника (многогранника, ограничивающего область допустимых решений) оптимальный путь к точке максимума целевой функции. Для построения оптимального пути апекс-метод использует операцию псевдопроекции, $\boldsymbol{\rho}$, являющуюся обобщением понятия метрической проекции на выпуклое множество. Псевдопроекция является предельной точкой последовательности приближений, получаемых путем ортогонального проектирования на гиперплоскости, ограничивающие область допустимых решений. Основная идея апекс-метода представлена на рис. 1а.

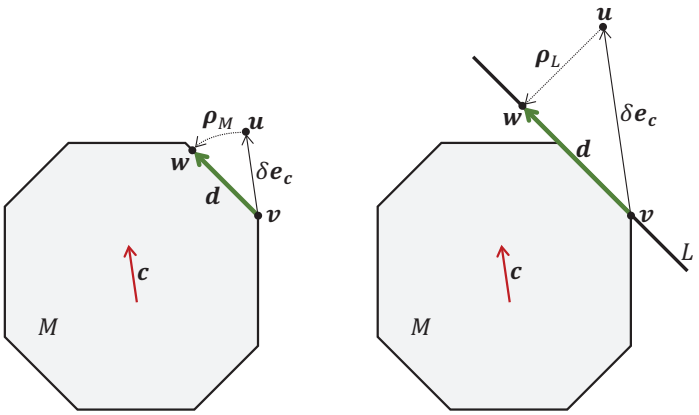


Рис. 1: Построение вектора движения $\boldsymbol{d}$: а) в апекс-методе; б) в алгоритма AlFaMove. M — многогранник, ограничивающий область допустимых решений; $\boldsymbol{c}$ — градиент линейной целевой функции; $\boldsymbol{e_c}$ — единичный вектор, сонаправленный с вектором $\boldsymbol{c}$; δ — положительная вещественная константа; $\boldsymbol{\rho}$ — операция построения псевдопроекции на выпуклое множество.

К точке текущего приближения $\boldsymbol{v}$ прибавляется вектор $\delta \boldsymbol{e_c}$, сонаправленный с градиентом $\boldsymbol{c}$ линейной целевой функции. Здесь $\boldsymbol{e_c}$ — единичный вектор, δ — положительная вещественная константа, являющаяся параметром алгоритма. Из получившейся точки $\boldsymbol{u}$ строится псевдопроекция $\boldsymbol{w}$ на допустимый многогранник M:

$$\boldsymbol{w} = \boldsymbol{\rho}_M(\boldsymbol{v}).$$

Вектор $\boldsymbol{d} = \boldsymbol{w} - \boldsymbol{v}$ указывает направление движения по поверхности многогранника M к следующему приближению, в качестве которого берется граничная точка,

максимально удаленная от $\boldsymbol{v}$ в направлении вектора $\boldsymbol{d}$. За конечное число шагов этот алгоритм приходит к решению задачи ЛП при условии, что $M \neq \emptyset$. Указанный метод имеет два недостатка. Во-первых, доказано, что метод сходится, если псевдопроекцию заменить на метрическую проекцию [15]. При использовании псевдопроекции апекс-метод может в отдельных случаях зацикливаться. Во-вторых, апекс-метод требует, чтобы точка $\boldsymbol{w}$ находилась на той же грани, что и точка $\boldsymbol{v}$. Следовательно, малый размер грани потребует малого значения вещественного параметра δ, что приводит к значимой потере точности при вычислении вектора движения $\boldsymbol{d}$.

В недавней статье [18] был представлен новый проекционный алгоритм AlFaMove (Along Faces Movement), устраняющий описанные недостатки. Принципиальное отличие алгоритма AlFaMove от апекс-метода заключается в том, что псевдопроекция из точки $\boldsymbol{u}$ строится не на допустимый многогранник M, а на линейное многообразие L, порождаемое пересечением ограничивающих гиперплоскостей, проходящих через точку $\boldsymbol{v}$ (см. рис. 1б). В этом случае величина положительной константы δ может быть сколь угодно большой, что обеспечивает достаточную точность вычисления вектора $\boldsymbol{d}$ даже для очень маленьких граней. Более того, в работе [19] было доказано, что в случае линейного многообразия псевдопроекция совпадает с ортогональной проекцией, что обеспечивает сходимость алгоритма. Основной недостаток алгоритма AlFaMove заключается в том, что он имеет экспоненциальную вычислительную сложность. Действительно, если через точку $\boldsymbol{v}$ проходит k ограничивающих гиперплоскостей, то при выборе оптимального направления движения нам необходимо рассмотреть $(2^k - 1)$ линейных многообразий.

В этой статье предлагается и исследуется новый проекционный алгоритм линейного программирования AlEM (Along Edges Movement), ориентированный на кластерные вычислительные системы. В отличие от алгоритма AlFaMove алгоритм AlEM использует для построения оптимального пути только ребра допустимого многогранника, соответствующие одномерным линейным многообразиям. Алгоритм AlEM позволяет существенно уменьшить вычислительную сложность по сравнению с алгоритмом AlFaMove. Если в n-мерном евклидовом пространстве через вершину допустимого многогранника проходят q ограничивающих гиперплоскостей, то для выбора оптимального пути необходимо рассмотреть C_q^{n-1} сочетаний, где

$$C_q^{n-1} = \frac{q!}{(q-n+1)!(n+1)!}.$$

Так, например, в случае гиперкуба AlFaMove в каждой вершине должен обработать $2^{(n-1)}$ комбинаций в то время, как AlEM должен будет обработать только n комбинаций.

Статья организована следующим образом. В разделе 2 представлен необходимый теоретический базис. Раздел 3 содержит формализованное описание алгоритма AlEM. Раздел 4 посвящен описанию параллельной версии алгоритма AlEM. В разделе 5 представлены информация о программной реализации параллельной версии алгоритма AlEM и результаты экспериментов на кластерной вычислительной системе по исследованию ее масштабируемости. В заключении суммируются полученные результаты и намечаются направления дальнейших исследований.

2. Теоретический базис

Данный раздел содержит необходимый теоретический базис, используемый для формализованного описания алгоритма AlEM.

В евклидовом пространстве $\mathbb{R}^n$ рассматривается задача ЛП в общем виде с системой ограничений

$$\begin{cases} \langle \boldsymbol{a}_1, \boldsymbol{x} \rangle \leqslant b_1 \\ \ldots\ldots \\ \langle \boldsymbol{a}_k, \boldsymbol{x} \rangle \leqslant b_k \\ \langle \boldsymbol{a}_{k+1}, \boldsymbol{x} \rangle = b_{k+1} \\ \ldots\ldots \\ \langle \boldsymbol{a}_{k+l}, \boldsymbol{x} \rangle = b_{k+l}, \end{cases} \tag{1}$$

включающей в себя k линейных неравенств и l линейных уравнений. Здесь и далее $\langle \cdot, \cdot \rangle$ обозначает скалярное произведение двух векторов. Предполагается, что система (1) включает в себя также неравенства вида

$$\begin{aligned} -x_1 &\leqslant 0; \\ &\cdots\cdots\cdots \\ -x_n &\leqslant 0. \end{aligned} \tag{2}$$

Общее число ограничений m определяется по формуле

$$m = k + l, \tag{3}$$

где $k > 0$ и $l \geqslant 0$. В матричной форме систему ограничений (1) можно представить следующим образом:

$$\begin{cases} \hat{A}\boldsymbol{x} \leqslant \hat{\boldsymbol{b}} \\ \bar{A}\boldsymbol{x} = \bar{\boldsymbol{b}}, \end{cases} \tag{4}$$

где $\hat{A} \in \mathbb{R}^{k \times n}$, $\bar{A} \in \mathbb{R}^{l \times n}$, $\hat{\boldsymbol{b}} \in \mathbb{R}^k$ и $\bar{\boldsymbol{b}} \in \mathbb{R}^l$. Матрицу $\hat{A}$ будем называть граничной, а матрицу $\bar{A}$ — опорной.

Решением задачи ЛП считается точка, удовлетворяющая системе ограничений (1), в которой достигается максимум линейной целевой функции

$$F(\boldsymbol{x}) = \langle \boldsymbol{c}, \boldsymbol{x} \rangle . \tag{5}$$

Неравенствам из системы ограничений (1) соответствует k гиперплоскостей[1], называемых граничными:

$$\begin{aligned} H_1 &= \{\boldsymbol{x} \in \mathbb{R}^n | \langle \boldsymbol{a}_1, \boldsymbol{x} \rangle = b_1\}; \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ H_k &= \{\boldsymbol{x} \in \mathbb{R}^n | \langle \boldsymbol{a}_k, \boldsymbol{x} \rangle = b_k\}. \end{aligned} \tag{6}$$

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

Без ограничения общности мы можем предполагать, что среди граничных гиперплоскостей нет совпадающих. Граничные гиперплоскости отделяют k закрытых полупространств:

$$\begin{gathered}\hat{H}_1 = \{\boldsymbol{x} \in \mathbb{R}^n | \langle \boldsymbol{a}_1, \boldsymbol{x} \rangle \leqslant b_1\}; \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \hat{H}_k = \{\boldsymbol{x} \in \mathbb{R}^n | \langle \boldsymbol{a}_k, \boldsymbol{x} \rangle \leqslant b_k\}.\end{gathered}$$

Пересечение этих полупространств образует замкнутый выпуклый многогранник $\hat{M}$:

$$\hat{M} = \bigcap_{i=1}^{k} \hat{H}_i.$$

Уравнениям из системы ограничений (1) соответствует l гиперплоскостей, называемых опорными:

$$\begin{gathered}H_{k+1} = \{\boldsymbol{x} \in \mathbb{R}^n | \langle \boldsymbol{a}_{k+1}, \boldsymbol{x} \rangle = b_{k+1}\}; \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ H_{k+l} = \{\boldsymbol{x} \in \mathbb{R}^n | \langle \boldsymbol{a}_{k+l}, \boldsymbol{x} \rangle = b_{k+l}\}.\end{gathered} \tag{7}$$

Если $l > 0$, то пересечение опорных гиперплоскостей образует опорное линейное многообразие $\bar{L}$:

$$\bar{L} = \bigcap_{i=1}^{l} H_{k+i}. \tag{8}$$

Если уравнения в системе ограничений отсутствуют ($l = 0$), полагаем $\bar{L} = \mathbb{R}^n$. Область допустимых решений задачи ЛП представляет собой замкнутый выпуклый многогранник M, называемый допустимым:

$$M = \bar{L} \cap \hat{M}.$$

Мы будем предполагать, что допустимый многогранник M является непустым ограниченным множеством. В этом случае задача ЛП имеет решение.

Следующие уравнения, соответствующие граничным гиперплоскостям (6), будем называть граничными:

$$\begin{gathered}\langle \boldsymbol{a}_1, \boldsymbol{x} \rangle = b_1; \\ \ldots\ldots\ldots\ldots \\ \langle \boldsymbol{a}_k, \boldsymbol{x} \rangle = b_k.\end{gathered} \tag{9}$$

Они образуют систему уравнений

$$\hat{A}\boldsymbol{x} = \hat{\boldsymbol{b}}.$$

Следующие уравнения, соответствующие опорным гиперплоскостям (7), будем называть опорными:

$$\begin{gathered}\langle \boldsymbol{a}_{k+1}, \boldsymbol{x} \rangle = b_{k+1}; \\ \ldots\ldots\ldots\ldots\ldots \\ \langle \boldsymbol{a}_{k+l}, \boldsymbol{x} \rangle = b_{k+l}.\end{gathered} \tag{10}$$

Они образуют систему уравнений

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

Везде далее мы будем предполагать, что

$$\operatorname{rank}\left(\bar{A}\right) = l. \tag{11}$$

В этом случае опорное линейное многообразие $\bar{L}$, определяемое формулой (8), имеет размерность

$$\dim\left(\bar{L}\right) = n - l.$$

При этом мы также можем считать, что

$$l < n, \tag{12}$$

так как в противном случае допустимый многогранник M вырождается в точку. Поскольку система ограничений (1) включает в себя n неравенства вида (2), имеем

$$k \geqslant n. \tag{13}$$

Введем следующие обозначения:

$$\hat{\mathcal{I}} = \{1, \ldots, k\}; \tag{14}$$
$$\bar{\mathcal{I}} = \{k+1, \ldots, k+l\}. \tag{15}$$

Для произвольного $\mathcal{J} \subseteq \hat{\mathcal{I}}$ обозначим через $A_{\mathcal{J}}$ матрицу системы, включающую в себя граничные уравнения с индексами из множества $\mathcal{J}$ и все опорные уравнения. Везде далее мы будем предполагать, что система ограничений (1) является общей, то есть удовлетворяет условию

$$\forall \mathcal{J} \subseteq \hat{\mathcal{I}}\left(|\mathcal{J}| = n - l \Rightarrow \operatorname{rank}\left(A_{\mathcal{J}}\right) = n\right). \tag{16}$$

Другими словами, если к опорным уравнениям (10) добавить $n - l$ произвольных граничных уравнений (9), то матрица получившейся системы будет иметь ранг n. В этих предположениях справедливы следующее три утверждения, являющиеся теоретической основой алгоритма AlEM.

Предложение 1. *Пусть выполняется условие* (16), $\mathcal{J} \subset \hat{\mathcal{I}}$ *и* $|\mathcal{J}| = n - l - 1$. *Положим*

$$L_{\mathcal{J}} = \bigcap_{i \in \mathcal{J} \cup \bar{\mathcal{I}}} H_i. \tag{17}$$

Тогда $L_{\mathcal{J}}$ *является линейным многообразием размерности 1:*

$$\dim\left(L_{\mathcal{J}}\right) = 1,$$

то есть $L_{\mathcal{J}}$ *— прямая в пространстве* $\mathbb{R}^n$.

Доказательство. Выберем любой $i' \in \hat{\mathcal{I}}$ такой, что $i' \notin \mathcal{J}$. Он существует в силу (13). Положим

$$\mathcal{J}' = \mathcal{J} \cup \{i'\}.$$

Обозначим через $A_{\mathcal{J}'}$ матрицу системы, включающей в себя граничные уравнения с индексами из множества $\mathcal{J}'$ и все опорные уравнения. По предположению (16) имеем

$$\operatorname{rank}\left(A_{\mathcal{J}'}\right) = n.$$

Матрица $A_{\mathcal{J}'}$ имеет n строк и n столбцов. Удалив любую строку из этой матрицы, мы получим матрицу ранга $n-1$. Удалив строку, соответствующую добавленному неравенству с индексом i', получим матрицу $A_{\mathcal{J}}$ ранга $n-1$:

$$\operatorname{rank}(A_{\mathcal{J}}) = n-1.$$

Отсюда следует, что $\dim(L_{\mathcal{J}}) = 1$.

Обозначим через $\Gamma(M)$ множество граничных точек допустимого многогранника M. Следующее условие является необходимым и достаточным для того, чтобы граничная точка $\boldsymbol{v} \in \Gamma(M)$ была вершиной допустимого многогранника M.

Предложение 2. *Пусть выполняется условие* (16) *и* $\boldsymbol{v} \in \Gamma(M)$. *Тогда следующее условие является необходимым и достаточным для того, чтобы точка* $\boldsymbol{v}$ *являлась вершиной допустимого многогранника* M*:*

$$\exists\, \mathcal{J} \subseteq \hat{\mathcal{I}} \left(|\mathcal{J}| = n-l \ \wedge\ \boldsymbol{v} \in \bigcap_{i \in \mathcal{J}} H_i \right). \tag{18}$$

Другими словами, граничная точка $\boldsymbol{v} \in \Gamma(M)$ *тогда и только тогда является вершиной допустимого многогранника* M*, когда через нее проходят* $n-l$ *ограничивающих гиперплоскостей.*

Доказательство. Сначала докажем «необходимость». Пусть граничная точка $\boldsymbol{v} \in \Gamma(M)$ является вершиной допустимого многогранника M. Это означает, что выполняется следующее условие:

$$\exists\, \mathcal{J}' \subseteq \hat{\mathcal{I}} \left(|\mathcal{J}'| \geqslant n-l \ \wedge\ \boldsymbol{v} = \bigcap_{i \in \mathcal{J}' \cup \bar{\mathcal{I}}} H_i \right). \tag{19}$$

Выберем произвольное $\mathcal{J} \subseteq \mathcal{J}'$ такое, что $|\mathcal{J}| = n-l$. Из (19) следует, что

$$\boldsymbol{v} \in \bigcap_{i \in \mathcal{J}} H_i,$$

то есть условие (18) выполняется.

Теперь докажем «достаточность». Пусть для граничной точки $\boldsymbol{v} \in \Gamma(M)$ выполняется условие (18). Поскольку $\boldsymbol{v} \in M$, имеем

$$\boldsymbol{v} \in \bigcap_{i \in \bar{\mathcal{I}}} H_i.$$

Сопоставляя это с (18), получаем

$$\boldsymbol{v} \in \bigcap_{i \in \mathcal{J} \cup \bar{\mathcal{I}}} H_i. \tag{20}$$

Рассмотрим матрицу $A_{\mathcal{J}}$ из коэффициентов уравнений, соответствующих гиперплоскостям в (20). Эта матрица имеет n строк и n столбцов. В соответствии с (16) имеем

$$\operatorname{rank}(A_{\mathcal{J}}) = n.$$

Это означает, что

$$\dim\left(\bigcap_{i\in\mathcal{J}\cup\bar{\mathcal{I}}} H_i\right) = 0.$$

Отсюда и из (20) следует

$$\boldsymbol{v} = \bigcap_{i\in\mathcal{J}\cup\bar{\mathcal{I}}} H_i,$$

то есть выполняется условие (19), означающее, что $\boldsymbol{v}$ является вершиной допустимого многогранника M.

Следующее утверждение будет использоваться для идентификации ребер допустимого многогранника M.

Предложение 3. *Пусть выполняется условие (16). Пусть D — ребро допустимого многогранника M, $\boldsymbol{v}'$ — его начальная вершина и $\boldsymbol{v}''$ — его конечная вершина:*

$$D = \left\{\boldsymbol{x}\in\mathbb{R}^n \,|\boldsymbol{x} = (1-\lambda)\boldsymbol{v}' + \lambda\boldsymbol{v}'',\ 0\leqslant\lambda\leqslant 1,\ \boldsymbol{v}'\neq\boldsymbol{v}''\right\}.$$

Определим прямую L, включающую в себя ребро D:

$$L = \left\{\boldsymbol{x}\in\mathbb{R}^n \,|\boldsymbol{x} = \boldsymbol{v}' + \delta(\boldsymbol{v}''-\boldsymbol{v}'),\ \delta\in\mathbb{R},\ \boldsymbol{v}'\neq\boldsymbol{v}''\right\}.$$

Тогда

$$\exists\,\mathcal{J}\subseteq\hat{\mathcal{I}}\left(|\mathcal{J}| = n-l-1\ \wedge\ \forall i\in\mathcal{J}\,(\boldsymbol{v}'\in H_i)\ \wedge\ L = \bigcap_{i\in\mathcal{J}\cup\bar{\mathcal{I}}} H_i\right). \tag{21}$$

Другими словами, если из всего множества граничных гиперплоскостей выбирать $n-l-1$ гиперплоскостей, проходящих через точку $\boldsymbol{v}'$, добавлять к ним все опорные гиперплоскости, и путем их пересечения получать прямые, то среди этих прямых обязательно найдется прямая L, содержащая ребро D.

Доказательство. Так как прямая L включает в себя ребро D допустимого многогранника M, то выполняется следующее условие:

$$\exists\,\mathcal{J}'\subseteq\hat{\mathcal{I}}\left(|\mathcal{J}'|\geqslant n-l-1\ \wedge\ \forall i\in\mathcal{J}'\,(\boldsymbol{v}'\in H_i)\ \wedge\ L = \bigcap_{i\in\mathcal{J}'\cup\bar{\mathcal{I}}} H_i\right). \tag{22}$$

Выберем произвольное $\mathcal{J}\subseteq\mathcal{J}'$ такое, что $|\mathcal{J}| = n-l-1$. Тогда из (22) следует

$$L\subseteq\bigcap_{i\in\mathcal{J}\cup\bar{\mathcal{I}}} H_i. \tag{23}$$

В соответствии с утверждением 1 имеем

$$\dim\left(\bigcap_{i\in\mathcal{J}\cup\bar{\mathcal{I}}} H_i\right) = 1. \tag{24}$$

Из (23) и (24) следует

$$L = \bigcap_{i\in\mathcal{J}\cup\bar{\mathcal{I}}} H_i, \tag{25}$$

то есть условие (21) выполняется.

3. Формализованное описание алгоритма AlEM

Данный раздел содержит формализованное описание нового алгоритма AlEM, строящего из произвольной вершины допустимого многогранника оптимальный путь по его ребрам к вершине, являющейся решением задачи ЛП. Оптимальность пути заключается в том, что всегда выбирается ребро, имеющее максимальное значение целевой функции в своей конечной точке.

Основной функцией алгоритма AlEM является функция MoveAlongEdge$(\cdot)$, осуществляющая переход к вершине с максимальным значением целевой функции. Функция MoveAlongEdge$(\cdot)$, в свою очередь, использует следующие две функции: $\boldsymbol{\rho}_{\mathcal{J}}(\cdot)$ — вычисление псевдопроекции на линейное многообразие и $Jump(\cdot)$ — перемещение по вектору.

Рассмотрим сначала функцию вычисления псевдопроекции $\boldsymbol{\rho}_{\mathcal{J}}(\boldsymbol{x})$ точки $\boldsymbol{x}$ на линейное многообразие

$$L_{\mathcal{J}} = \bigcap_{j \in \mathcal{J}} H_j,$$

где $\mathcal{J} \subseteq \{1, \ldots, m\}$. Основной элементарной операцией, используемой при вычислении псевдопроекции, является ортогональное проектирование $\boldsymbol{\pi}_i(\boldsymbol{x})$ точки $\boldsymbol{x}$ на гиперплоскость H_i, вычисляемое по известной формуле [20]:

$$\boldsymbol{\pi}_i(\boldsymbol{x}) = \boldsymbol{x} - \frac{\langle \boldsymbol{a}_i, \boldsymbol{x} \rangle - b_i}{\|\boldsymbol{a}_i\|^2} \boldsymbol{a}_i.$$

На основе ортогональной проекции строится проекционное отображение $\boldsymbol{\varphi}_{\mathcal{J}}(\cdot)$:

$$\boldsymbol{\varphi}_{\mathcal{J}}(\boldsymbol{x}) = \frac{1}{|\mathcal{J}|} \sum_{j \in \mathcal{J}} \boldsymbol{\pi}_j(\boldsymbol{x}). \tag{26}$$

Известно [11], что отображение $\boldsymbol{\varphi}_{\mathcal{J}}(\boldsymbol{x})$ является непрерывным $L_{\mathcal{J}}$-фейеровским отображением, и последовательность точек

$$\left\{ \boldsymbol{x}_t = \boldsymbol{\varphi}_{\mathcal{J}}^t \left(\boldsymbol{x}_0 \right) \right\}_{t=1}^{\infty}, \tag{27}$$

порождаемая этим отображением начиная с произвольной точки $\boldsymbol{x}_0 \in \mathbb{R}^n$, сходится к точке, принадлежащей $L_{\mathcal{J}}$:

$$\boldsymbol{x}_t \to \tilde{\boldsymbol{x}} \in L_{\mathcal{J}}.$$

В соответствии с этим мы можем определить псевдопроекцию $\boldsymbol{\rho}_{\mathcal{J}}(\boldsymbol{x})$ точки $\boldsymbol{x}$ на линейное многообразие $L_{\mathcal{J}}$ как предельную точку последовательности (27):

$$\lim_{k \to \infty} \left\| \boldsymbol{\rho}_{\mathcal{J}}(\boldsymbol{x}) - \boldsymbol{\varphi}^k(\boldsymbol{x}) \right\| = 0.$$

В работе [19] было доказано, что в случае линейного многообразия псевдопроекция совпадает с ортогональной проекцией:

$$\boldsymbol{\rho}_{\mathcal{J}}(\boldsymbol{x}) = \boldsymbol{\pi}_{\mathcal{J}}(\boldsymbol{x}). \tag{28}$$

Реализация приближенного вычисления псевдопроекции представлена в виде алгоритма 1. Прокомментируем шаги этого алгоритма. На шаге 2 счетчик итераций t устанавливается в значение ноль. Шаг 3 задает начальное приближение $\boldsymbol{x}_0$. Шаг 4

Алгоритм 1 Вычисление псевдопроекции $\boldsymbol{\rho}_{\mathcal{J}}(\boldsymbol{x})$

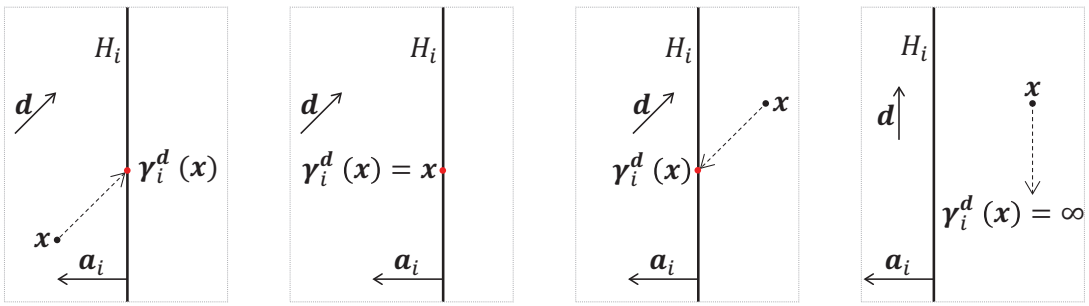
Require: $\mathcal{J} \subseteq \{1, \ldots, m\}$; $\mathcal{J} \neq \emptyset$
1: **function** $\boldsymbol{\rho}_{\mathcal{J}}(\boldsymbol{x})$
2: $t := 0$
3: $\boldsymbol{x}_0 := \boldsymbol{x}$
4: **repeat**
5: $\boldsymbol{\Sigma} := 0$
6: **for** $j \in \mathcal{J}$ **do**
7: $\boldsymbol{\Sigma} := \boldsymbol{\Sigma} + \boldsymbol{\pi}_j(\boldsymbol{x})$
8: **end for**
9: $\boldsymbol{\varphi} := \boldsymbol{\Sigma} / |\mathcal{J}|$
10: $\boldsymbol{x}_{(t+1)} := \boldsymbol{\varphi}$
11: $t := t + 1$
12: **until** $\|\boldsymbol{x}_t - \boldsymbol{x}_{(t-1)}\| \leqslant \varepsilon$
13: **return** $\boldsymbol{x}_t$
14: **end function**

открывает итерационный цикл вычисления псевдопроекции. Шаги 5–9 вычисляют по формуле (26) отображение $\boldsymbol{\varphi}_{\mathcal{J}}(\cdot)$ для текущего приближения $\boldsymbol{x}_k$. На шаге 10 вычисляется следующее приближение $\boldsymbol{x}_{k+1}$. Шаг 11 увеличивает счетчик итераций t на единицу. Шаг 12 завершает итерационный цикл, когда расстояние между соседними приближениями станет меньше малого параметра ε.

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

$$\boldsymbol{\gamma}_i^{\boldsymbol{d}}(\boldsymbol{x}) = \begin{cases} \boldsymbol{x} - \frac{\langle \boldsymbol{a}_i, \boldsymbol{x} \rangle - b_i}{\langle \boldsymbol{a}_i, \boldsymbol{d} \rangle} \boldsymbol{d}, & \text{если} \quad \langle \boldsymbol{a}_i, \boldsymbol{d} \rangle \neq 0; \\ \infty, & \text{если} \quad \langle \boldsymbol{a}_i, \boldsymbol{d} \rangle = 0 \end{cases} \tag{29}$$

(см. определение 1 и утверждение 2 в [19]). Различные случаи построения d-проекции $\boldsymbol{\gamma}_i^{\boldsymbol{d}}(\boldsymbol{x})$ показаны на рис. 2. Реализация функции $\mathrm{Jump}(\boldsymbol{x}, \boldsymbol{d})$ представлена в виде ал-



(а) (б) (в) (г)

Рис. 2: Различные случаи построения d-проекции $\boldsymbol{\gamma}_i^{\boldsymbol{d}}(\boldsymbol{x})$: а) $\langle \boldsymbol{a}_i, \boldsymbol{x} \rangle - b_i > 0$; б) $\langle \boldsymbol{a}_i, \boldsymbol{x} \rangle - b_i = 0$; в) $\langle \boldsymbol{a}_i, \boldsymbol{x} \rangle - b_i < 0$; г) $\langle \boldsymbol{a}_i, \boldsymbol{d} \rangle = 0$.

Алгоритм 2 Реализация функции Jump$(\boldsymbol{x}, \boldsymbol{d})$

Require: $\boldsymbol{x} \in M$; $\forall j \in \{1, \ldots, l\} : \boldsymbol{x} + \boldsymbol{d} \in H_{k+j}$

1: **function** Jump$(\boldsymbol{x}, \boldsymbol{d})$
2: $\quad \boldsymbol{x}_{min} := \boldsymbol{x}$
3: $\quad$ **for** $i = 1 \ldots k$ **do**
4: $\quad\quad$ **if** $\langle \boldsymbol{a}_i, \boldsymbol{d} \rangle > 0$ **then**
5: $\quad\quad\quad \boldsymbol{x}_\gamma := \boldsymbol{\gamma}_i^{\boldsymbol{d}}(\boldsymbol{x})$
6: $\quad\quad\quad$ **if** $\|\boldsymbol{x}_\gamma - \boldsymbol{x}\| < \|\boldsymbol{x}_{min} - \boldsymbol{x}\|$ **then**
7: $\quad\quad\quad\quad \boldsymbol{x}_{min} := \boldsymbol{x}_\gamma$
8: $\quad\quad\quad$ **end if**
9: $\quad\quad$ **end if**
10: $\quad$ **end for**
11: $\quad$ **return** $\boldsymbol{x}_{min}$
12: **end function**

горитма 2. Для работы алгоритма необходимо, чтобы точка $\boldsymbol{x}$ принадлежала допустимому многограннику M и выполнялось условие

$$\forall j \in \{1, \ldots l\} : \boldsymbol{x} + \boldsymbol{d} \in H_{k+j}, \tag{30}$$

которое означает, что точка, получающаяся путем прибавления вектора $\boldsymbol{d}$ к точке $\boldsymbol{x}$, должна принадлежать всем опорным гиперплоскостям (7). Другими словами, точка $(\boldsymbol{x} + \boldsymbol{d})$ обязана удовлетворять всем уравнениям из системы ограничений (1). Схема работы алгоритма 2 представлена на рис. 3.

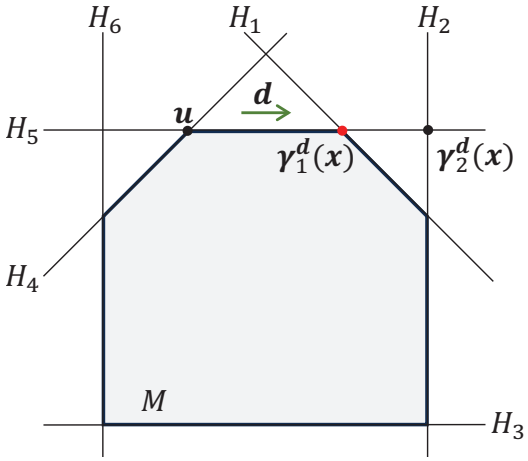


Рис. 3: Действие функции Jump: Jump$(\boldsymbol{x}, \boldsymbol{d}) = \boldsymbol{\gamma}_1^{\boldsymbol{d}}(\boldsymbol{u})$.

В примере, показанном на рисунке, система ограничений не включает в себя уравнения. Граничные гиперплоскости H_3–H_6 не удовлетворяют неравенству

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

проверяемому на шаге 4 алгоритма 2. Это означает, что точка $(\boldsymbol{x} + \lambda \boldsymbol{d})$ будет принадлежать полупространствам $\hat{H}_3$–$\hat{H}_6$ при любом положительном λ. Среди оставшихся граничных гиперплоскостей $\hat{H}_1$ и $\hat{H}_2$ на шагах 5–8 алгоритма 2 выбирается

d-проекция, в результате которой получается точка, находящаяся на минимальном расстоянии от точки $\boldsymbol{x}$. В примере на рис. 3 это — проекция на гиперплоскость H_1.

Реализация функции MoveAlongEdge($\boldsymbol{v}$), осуществляющей переход от вершины $\boldsymbol{v}$ к смежной оптимальной вершине $\boldsymbol{v}_{max}$, представлена в виде алгоритма 3. Опти-

Алгоритм 3 Проход по ребру

Require: $\boldsymbol{v} \in \Gamma(M);\ \exists\, \mathcal{J} \subseteq \{1, \ldots, k\}\Big(|\mathcal{J}| = n - l \wedge \boldsymbol{v} \in \bigcap\limits_{i \in \mathcal{J}} H_i\Big)$

1: **function** MoveAlongEdge($\boldsymbol{v}$)
2: $\quad\boldsymbol{u} := \boldsymbol{v} + \delta\boldsymbol{c}/\|\boldsymbol{c}\|$
3: $\quad\mathcal{V} := \emptyset$
4: $\quad$**for** $i = 1 \ldots k$ **do**
5: $\quad\quad$**if** $\langle \boldsymbol{a}_i, \boldsymbol{v} \rangle = b_i$ **then**
6: $\quad\quad\quad\mathcal{V} := \mathcal{V} \cup \{i\}$
7: $\quad\quad$**end if**
8: $\quad$**end for**
9: $\quad F_{max} := -\infty$
10: $\quad\boldsymbol{v}_{max} := \boldsymbol{v}$
11: $\quad$TWIDDLE_Init($\mathcal{V}, n - l - 1$)
12: $\quad$**repeat**
13: $\quad\quad\mathcal{J} :=$ TWIDDLE_NextCombination()
14: $\quad\quad\mathcal{J} := \mathcal{J} \cup \{k + 1, \ldots, k + l\}$
15: $\quad\quad\boldsymbol{w} := \boldsymbol{\rho}_{\mathcal{J}}(\boldsymbol{u})$
16: $\quad\quad\boldsymbol{d} := \boldsymbol{w} - \boldsymbol{v}$
17: $\quad\quad\boldsymbol{v}_{next} :=$ Jump($\boldsymbol{v}, \boldsymbol{d}$)
18: $\quad\quad$**if** $\langle \boldsymbol{c}, \boldsymbol{v}_{next} \rangle > F_{max}$ **then**
19: $\quad\quad\quad\boldsymbol{v}_{max} := \boldsymbol{v}_{next}$
20: $\quad\quad\quad F_{max} := \langle \boldsymbol{c}, \boldsymbol{v}_{max} \rangle$
21: $\quad\quad$**end if**
22: $\quad$**until** TWIDDLE_Done()
23: $\quad$**return** $\boldsymbol{v}_{max}$
24: **end function**

мальность означает, что алгоритм всегда выбирает то ребро, инцидентное вершине $\boldsymbol{v}$, конечная точка которого имеет максимальное значение целевой функции. В реализации алгоритма 3 используется генератор TWIDDLE [21], строящий все возможные сочетания из элементов заданного конечного множества. Дадим краткий комментарий по шагам алгоритма 3. Начальная точка $\boldsymbol{v}$ должна являться вершиной, то есть удовлетворять условиям утверждения 2. На шаге 2 строится точка $\boldsymbol{u}$ (см. рис. 1б). Здесь δ — «большой» положительный параметр: чем больше δ, тем точнее будет вычислен вектор перемещения $\boldsymbol{d}$, используемый функцией Jump на шаге 17. Шаги 4–8 строят множество $\mathcal{V}$, включающее в себя индексы всех граничных гиперплоскостей, проходящих через точку $\boldsymbol{v}$. Шаг 9 присваивает переменной F_{max}, обозначающей максимум целевой функции, значение $-\infty$. На шаге 10 вектору $\boldsymbol{v}_{max}$ в качестве начального значения присваивается вектор $\boldsymbol{v}$. Шаг 11 устанавливает генератор сочетаний TWIDDLE в начальное состояние. Шаг 12 открывает цикл **repeat**/**until**, вычисляющий оптимальное ребро и следующую вершину $\boldsymbol{v}_{max}$. На шаге 13 генератор TWIDDLE возвращает очередное сочетание $(n - l - 1)$ из k индексов граничных

гиперплоскостей, именуемое как множество $\mathcal{J}$. На шаге 14 в множество $\mathcal{J}$ добавляются индексы всех опорных гиперплоскостей. В результате получается множество, содержащее $n-1$ индексов различных гиперплоскостей, пересечение которых образует прямую, соответствующую некоторому ребру, инцидентному вершине $\boldsymbol{v}$. На шаге 15 из точки $\boldsymbol{u}$ на эту прямую строится псевдопроекция $\boldsymbol{w}$, которая в этом случае будет совпадать с ортогональной проекцией. Шаг 16 вычисляет вектор движения по ребру $\boldsymbol{d}$ в направлении увеличения значения целевой функции. Шаг 17 осуществляет переход в конечную точку ребра $\boldsymbol{v}_{next}$. Если значение целевой функции в этой точке больше F_{max}, то на шагах 18-21 обновляются значения $\boldsymbol{v}_{max}$ и F_{max}. Шаг 22 завершает выполнение цикла **repeat**/**until** после того, как генератор TWIDDLE выдал последнее сочетание. Шаг 23 возвращает вектор $\boldsymbol{v}_{max}$ в качестве результата функции MoveAlongEdge($\boldsymbol{v}$). Отметим, что корректность алгоритма 3 обеспечивается утверждением 3.

Реализация алгоритма AlEM, строящего оптимальный путь по ребрам допустимого многогранника к решению задачи ЛП (1), представлена в виде алгоритма 4. Начальная точка $\boldsymbol{v}_0$ должна являться вершиной допустимого многогранника, то есть

Алгоритм 4 AlEM (Along Edges Movement)

Require: $\boldsymbol{v}_0 \in \Gamma(M);\ \exists\, \mathcal{J} \subseteq \{1,\ldots,k\}\Big(|\mathcal{J}| = n-l \wedge \boldsymbol{v}_0 \in \bigcap\limits_{i\in\mathcal{J}} H_i\Big)$

1: $t := 0$
2: **repeat**
3: $\quad \boldsymbol{v}_{t+1} := \text{MoveAlongEdge}(\boldsymbol{v}_t)$
4: $\quad t := t+1$
5: **until** $\boldsymbol{v}_t = \boldsymbol{v}_{t-1}$
6: **output** $\boldsymbol{v}_t$
7: **stop**

удовлетворять условиям утверждения 2. На шаге 1 счетчик итераций t устанавливается в значение 0. Шаг 2 открывает цикл **repeat**/**until**, строящий с помощью функции MoveAlongEdge($\cdot$) оптимальный путь по ребрам допустимого многогранника от начальной точки $\boldsymbol{v}_0$ к решению задачи ЛП. Шаг 5 завершает выполнение цикла **repeat**/**until**, когда следующее приближение $\boldsymbol{v}_k$ совпадет с предыдущим $\boldsymbol{v}_{k-1}$. Шаг 6 выводит последнее приближение в качестве решения задачи ЛП. Шаг 7 завершает работу алгоритма AlEM.

Как уже было сказано выше, алгоритм AlEM требует в качестве начальной точки вершину допустимого многогранника M. Мы получаем такую вершину в два этапа. На первом этапе используется алгоритм BIP (Block-iterative projection) [17], реализованный нами в виде параллельной программы на языке C++, исходные тексты которой доступны в репозитории GitHub[1]. На вход алгоритму BIP подается произвольная точка $\boldsymbol{x}_0 \in \mathbb{R}^n$, не являющаяся внутренней по отношению к допустимому многограннику M. В случае задачи ЛП в качестве такой точки всегда может служить точка $\mathbf{0}$. Алгоритм BIP возвращает точку $\boldsymbol{z}_0$, лежащую на границе допустимого многогранника M. На втором этапе мы применяем разработанный нами параллельный алгоритм VeSP (Vertex Search by Projecting), исходные коды которого также доступ-

[1] `https://github.com/leonid-sokolinsky/BIP`

ны в репозитории GitHub[1]. В качестве начальной точки алгоритм VeSP использует произвольную граничную точку $\boldsymbol{z}_0$ допустимого многогранника M. Затем алгоритм VeSP идет по граням допустимого многогранника M в направлении уменьшения их размерности и за конечное число итераций доходит до вершины допустимого многогранника M, которая и служит начальной точкой $\boldsymbol{v}_0$ алгоритма AlEM. Отметим, что алгоритм VeSP использует подходы, примененные при разработке алгоритма AlFaMove[18].

4. Параллельная версия алгоритма AlEM

Наиболее ресурсоемкой операцией функции MoveAlongEdge$(\cdot)$ является операция вычисления псевдопроекции на линейное многообразие на шаге 15 (см. алгоритм 3). Выполнение вектор-функции $\boldsymbol{\rho}_{\mathcal{J}}(\cdot)$ заключается в последовательном применении проекционного отображения $\varphi(\cdot)$, задаваемого формулой (26), к исходной точке (см. алгоритм 1, реализующий вычисление псевдопроекции). Известно, что в случае больших задач ЛП проекционный метод может потребовать значительных временных затрат [22]. Кроме того, следует отметить, что алгоритм 3 в цикле на шаге 13 перебирает все сочетания из k по $(n-l-1)$ индексов граничных гиперплоскостей. Число таких сочетаний вычисляется по формуле

$$C_k^{n-l-1} = \frac{k!}{(k-n+l+1)!(n-l-1)!}$$

и может достигать больших значений при решении практических задач ЛП. Комбинаторный перебор может потребовать использования суперкомпьютерных мощностей. Поэтому мы разработали параллельную версию алгоритма AlEM. Работа параллельных узлов организуется по схеме SIMD (single instruction, multiple data): все процессорные узлы выполняют один и тот же код, но над различными данными. Сочетания, конструируемые генератором TWIDDLE, распределяются между процессорными узлами по принципу round-robin. Для этого мы преобразовали функцию MoveAlongEdge$(\cdot)$ (алгоритм 3) в функцию $RoundRobinMove(\cdot)$, реализация которой представлена в виде алгоритма 5. Кроме исходной вершины $\boldsymbol{v}$, в функцию $RoundRobinMove(\cdot)$ передаются номер процессорного узла *myRank* и количество процессорных узлов *numOfNodes*. Функция без параметров TWIDDLE_NextCombination() преобразована в функцию с одним параметром TWIDDLE_Combination(*twiddleNo*), возвращающую сочетание с порядковым номером *twiddleNo*. Если значение переменной *twiddleNo* превышает общее число сочетаний, то функция TWIDDLE_Combination(*twiddleNo*) возвращает последнее сочетание, а функция TWIDDLE_Done() возвращает значение «истина». Во всех остальных случаях функция TWIDDLE_Done() возвращает значение «ложь». В соответствии с принципом round-robin переменной *twiddleNo* на шаге 2 присваивается номер процессорного узла[2], а затем на каждой итерации цикла **while**, она увеличивается на количество процессорных узлов *numOfNodes* (шаг 23). Таким образом, множества сочетаний, обрабатываемых на различных процессорных узлах, не будут пересекаться, а их объединение будет равно множеству всех возможных сочетаний.

[1] `https://github.com/leonid-sokolinsky/VeSP`

[2] Мы предполагаем, что нумерация процессорных узлов начинается с нуля, равно как и нумерация сочетаний, выдаваемых генератором TWIDDLE, также начинается с нуля.

Алгоритм 5 Переход к следующей вершине по принципу round-robin

Require: $\boldsymbol{v} \in \Gamma(M);\ \exists\, \mathcal{J} \subseteq \{1, \ldots, k\}\Big(|\mathcal{J}| = n - l \wedge \boldsymbol{v} \in \bigcap\limits_{i \in \mathcal{J}} H_i\Big)$

function RoundRobinMove($\boldsymbol{v}$, *myRank*, *numOfNodes*)
  $twiddleNo := myRank$
  $\boldsymbol{u} := \boldsymbol{v} + \delta \boldsymbol{c}/\|\boldsymbol{c}\|$
  $\mathcal{V} := \emptyset$
  for $i = 1 \ldots k$ **do**
    if $\langle \boldsymbol{a}_i, \boldsymbol{v} \rangle = b_i$ **then**
      $\mathcal{V} := \mathcal{V} \cup \{i\}$
    end if
  end for
  $F_{max} := -\infty$
  $\boldsymbol{v}_{max} := \boldsymbol{v}$
  TWIDDLE_Init($\mathcal{V}, n - l - 1$)
  $\mathcal{J} :=$ TWIDDLE_Combination($twiddleNo$)
  while not TWIDDLE_Done() **do**
    $\mathcal{J} := \mathcal{J} \cup \{k + 1, \ldots, k + l\}$
    $\boldsymbol{w} := \boldsymbol{\rho}_{\mathcal{J}}(\boldsymbol{u})$
    $\boldsymbol{d} := \boldsymbol{w} - \boldsymbol{v}$
    $\boldsymbol{v}_{next} :=$ Jump($\boldsymbol{v}, \boldsymbol{d}$)
    if $\langle \boldsymbol{c}, \boldsymbol{v}_{next} \rangle > F_{max}$ **then**
      $\boldsymbol{v}_{max} := \boldsymbol{v}_{next}$
      $F_{max} := \langle \boldsymbol{c}, \boldsymbol{v}_{max} \rangle$
    end if
    $twiddleNo := twiddleNo + numOfNodes$
    $\mathcal{J} :=$ TWIDDLE_Combination($twiddleNo$)
  end while
  return $\boldsymbol{v}_{max}$
end function

Алгоритм 6 Параллельная версия алгоритма AlEM

Require: $\boldsymbol{v}_0 \in \Gamma(M);\ \exists\, \mathcal{J} \subseteq \{1, \ldots, k\}\Big(|\mathcal{J}| = n - l \wedge \boldsymbol{v}_0 \in \bigcap\limits_{i \in \mathcal{J}} H_i\Big)$

$t := 0$
$myRank :=$ SYSTEM_Rank()
$numOfNodes :=$ SYSTEM_NumberOfNodes()
repeat
  $\boldsymbol{v}_{t+1} :=$ RoundRobinMove($\boldsymbol{v}_t, myRank, numOfNodes$)
  $rank_{max} :=$ SYSTEM_AllReduceMax ($\langle \boldsymbol{c}, \boldsymbol{v}_{t+1} \rangle$)
  SYSTEM_Broadcast($rank_{max}, \boldsymbol{v}_{t+1}$)
  $t := t + 1$
until $\boldsymbol{v}_t = \boldsymbol{v}_{t-1}$
if $myRank = rank_{max}$ **then**
  output $\boldsymbol{v}_t$
end if
stop

Параллельная версия алгоритма AlEM представлена в виде алгоритма 6. Этот алгоритм выполняется на всех используемых процессорных узлах. Кратко прокомментируем шаги параллельного алгоритма 6. Для его работы по-прежнему требуется, чтобы начальная точка $\boldsymbol{v}_0$ была вершиной допустимого многогранника M. На шаге 1 счетчику циклов t присваивается значение 0. На шаге 2 вызывается системная функция SYSTEM_Rank(), возвращающая номер процессорного узла, который сохраняется в переменной *myRank*. На шаге 3 вызывается системная функция SYSTEM_NumberOfNodes(), возвращающая количество используемых процессорных узлов. Полученное значение сохраняется в переменной *numOfNodes*. Шаг 4 открывает цикл **repeat/until**, строящий оптимальный путь по ребрам допустимого многогранника от начальной точки $\boldsymbol{v}_0$ к решению задачи ЛП. На шаге 5 каждый узел по принципу round-robin перебирает свои комбинации и находит следующую вершину $\boldsymbol{v}_{t+1}$ с локальным максимумом целевой функции. На шаге 6 выполняется системная функция SYSTEM_AllReduceMax $(\langle c, \boldsymbol{v}_{t+1}\rangle)$, возвращающая номер узла, на котором целевая функция достигает наибольшего значения в точках локальных максимумов $\boldsymbol{v}_{t+1}$. Шаг 7 выполняет системную функцию SYSTEM_Broadcast$(rank_{max}, \boldsymbol{v}_{t+1})$, в результате которой процессорный узел с номером $rank_{max}$ передает остальным узлам вектор $\boldsymbol{v}_{t+1}$. Шаг 9 завершает выполнение цикла **repeat/until**, когда следующее приближение $\boldsymbol{v}_k$ совпадет с предыдущим $\boldsymbol{v}_{k-1}$. Это означает, что достигнут максимум целевой фцнкции. На шагах 10–12 процессорный узел с номером *myRank* выводит последнее приближение в качестве решения задачи ЛП. Шаг 13 завершает работу параллельного алгоритма AlEM.

5. Реализация и вычислительные эксперименты

Мы реализовали параллельную версию алгоритма AlEM на языке C++ с использованием библиотеки параллельных вычислений MPI 3.0. Исходные коды параллельной программы доступны в репозитории GitHub по адресу `https://github.com/zhulevae/AlEM`.

С помощью разработанной параллельной программы мы исследовали масштабируемость алгоритма AlEM. В экспериментах мы использовали параметризованную задачу ЛП «гиперкуб с отсеченной вершиной», для которой размерность пространства n является параметром. Ограничения этой задачи содержат $2n + 1$ неравенств следующего вида:

$$\begin{cases} x_1 & & & & & \leqslant & 200 \\ & x_2 & & & & \leqslant & 200 \\ & & & & & \vdots & \\ & & & x_n & & \leqslant & 200 \\ x_1 + & x_2 & \cdots & + & x_n & \leqslant & 200(n-1) + 100 \\ x_1 \geqslant 0, & x_2 \geqslant 0, & \cdots & , x_n \geqslant 0. & & & \end{cases}$$

Градиент целевой функции задается вектором

$$\boldsymbol{c} = (1, 2, \ldots, n)\,.$$

Задача предполагает нахождение максимума целевой функции и имеет единственное решение в точке $(100, 200, \ldots, 200)$ со значением целевой функции, равным

$100(n^2 + n - 1)$. Для произвольного n эта задача может быть получена в формате MTX [23] с помощью генератора FRaGenLP[1] [24], если в качестве количества случайных неравенств задать 0. Эти задачи ЛП для различных n доступны по адресу `https://github.com/leonid-sokolinsky/Set-of-LP-Problems/tree/main/Rnd-LP` под именами `lp_rnd<n>-0`, где в качестве `<n>` указана размерность пространства.

Масштабные вычислительные эксперименты проводились на вычислительном кластере «Торнадо ЮУрГУ» [25], характеристики которого представлены в табл. 1. Для сборки программы использовался компилятор g++, распространяемый в рамках пакета компиляторов GCC 10, и библиотека Intel MPI 5.0. Компиляция выполнялась с опцией оптимизации O3.

Таблица 1: Характеристики кластера «Торнадо ЮУрГУ»

Параметр	Значение
Количество доступных процессорных узлов	260
Процессоры	Intel Xeon X5680 (6 cores, 3.33 GHz)
Число процессоров на узел	2
Память на узел	24 GB DDR3
Соединительная сеть	InfiniBand QDR (40 Gbit/s)
Операционная система	Linux CentOS

В описанной среде нами была выполнена серия вычислительных экспериментов, в которой для задач ЛП различной размерности исследовались ускорение и параллельная эффективность в зависимости от количества используемых процессорных узлов кластера. На каждом процессорном узле выполнялся только один MPI-процесс. Это искусственно увеличивало нагрузку на соединительную сеть и одновременно повышало трудоемкость вычислений. Результаты этих экспериментов представлены на рис. 4.

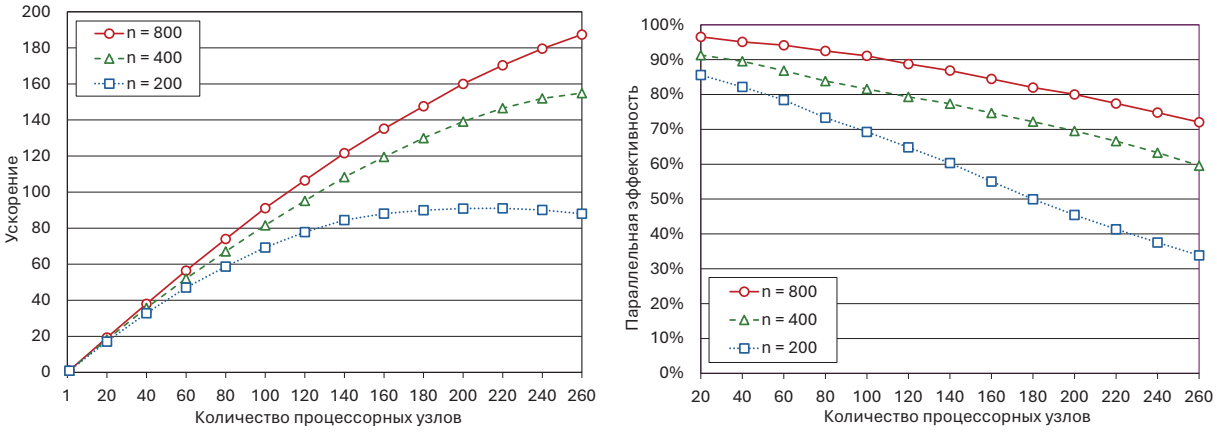


Рис. 4: Ускорение и эффективность параллельной реализации алгоритма AlEM.

Ускорение $\alpha(P)$ определялось как отношение времени $T(1)$ решения задачи на конфигурации с одним процессорным узлом ко времени $T(P)$ решения той же задачи

[1] `https://github.com/leonid-sokolinsky/FRaGenLP`

на конфигурации с P процессорными узлами:

$$\alpha(P) = \frac{T(1)}{T(P)}.$$

Параллельная эффективность вычислялась по формуле

$$\epsilon(P) = \frac{T(1)}{P \cdot T(P)}.$$

Вычисления проводились для следующих размерностей: 200, 400 и 800. Число ограничений соответственно составило 401, 801 и 1601. Эксперименты показали, что алгоритм AlEM хорошо масштабируется на задачах больших размерностей. Задача размерности $n = 800$ показала на 260 процессорных узлах ускорение, близкое к линейному, с эффективностью распараллеливания 72%. Однако, при уменьшении размерности задачи граница масштабируемости снижается и эффективность распараллеливания падает. Так для задачи размерности $n = 400$ граница масштабируемости наблюдалась в районе 260 процессорных узлов с эффективностью распараллеливания 60%. А для задачи размерности $n = 400$ граница масштабируемости упала до 200 процессорных узлов. Параллельная эффективность в этом случае составила только 50%. Время решения задачи «гиперкуб с отсеченной вершиной» для различных размерностей приведено в табл. 2. В случае, когда достигалась граница

Таблица 2: Время решения задачи «гиперкуб с отсеченной вершиной»

Размерность	Кол-во узлов	Время (сек)
200	200	1
400	260	2.5
800	260	18.5

масштабируемости (размерности 200 и 400), время решения оказалось сравнимым с симплекс-методом. Для достижения границы масштабируемости алгоритма AlEM при решении задачи размерности 800 необходим вычислительный кластер с большим числом процессорных узлов, чем «Торнадо ЮУрГУ».

Заключение

В статье предложен новый масштабируемый алгоритм линейного программирования, получивший название «AlEM». Ключевой особенностью этого метода является построение оптимального пути по ребрам допустимого многогранника от начальной точки к решению задачи линейного программирования. Под оптимальным путем понимается путь в направлении максимального увеличения значений целевой функции. Практическая значимость предложенного метода состоит в том, что он открывает возможность применения искусственных нейронных сетей прямого распространения для решения нестационарных многомерных задач линейного программирования в режиме реального времени.

Теоретической основой алгоритма AlEM является операция построения псевдопроекции на линейные многообразия, формирующие ребра допустимого многогранника. Псевдопроекция реализуется на основе фейеровского процесса и является обобщением понятия метрической проекции на выпуклое множество. В случае линейного

многообразия псевдопроекция сходится к точке ортогональной проекции. Приведен алгоритм проекционного типа для построения псевдопроекции на линейные многообразия, образуемые пересечением гиперплоскостей. Представлено формализованное описание алгоритма AlEM, строящего оптимальный путь по ребрам допустимого многогранника. В основе алгоритма AlEM лежит функция перехода по ребру допустимого многогранника из текущей вершины к следующей вершине с максимальным значением целевой функции. Дано формализованное описание этой функции.

Алгоритмы проекционного типа характеризуются низкой скоростью сходимости, зависящей от углов между гиперплоскостями, образующими линейное многообразие. Также отмечено, что при вычислении вектора движения возникает переборная задача комбинаторного типа, имеющая высокую временную сложность. Представлена параллельная версия алгоритма AlEM, ориентированная на кластерные вычислительные системы. Параллельная версия реализована на языке C++ с использованием библиотеки MPI. Проведены эксперименты по исследованию масштабируемости параллельной версии алгоритма AlEM на кластерной вычислительной системе. Вычислительные эксперименты показали, что задача линейного программирования, включающая 800 переменных и 1601 ограничение, демонстрирует ускорение близкое к линейному на 260 процессорных узлах кластера. В случае, когда достигается граница масштабируемости, время решения задачи ЛП методом AlEM сравнимо с симплекс-методом.

В качестве направлений дальнейших исследований выделим следующие. Для ускорения вычислений мы предполагаем использовать технологию OpenMP при вычислении псевдопроекции. Также мы планируем разработать новый, более эффективный метод построения пути по ребрам допустимого многогранника, приводящего к решению задачи линейного программирования. Основная идея состоит в построении многомерного образа задачи ЛП и использования искусственной нейронной сети прямого распространения для идентификации ребер допустимого многогранника. Это позволит отказаться от дорогостоящей операции псевдопроекции и превзойти симплекс-метод по скорости решения больших задач ЛП.

Список литературы

[1] Xu Y. Solving Large Scale Optimization Problems in the Transportation Industry and Beyond Through Column Generation // Optimization in Large Scale Problems. Springer Optimization and Its Applications, vol. 152. Springer, 2019. P. 269–292. https://doi.org/10.1007/978-3-030-28565-4_23.

[2] Chung W. Applying large-scale linear programming in business analytics // 2015 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM). IEEE, 2015. P. 1860–1864. https://doi.org/10.1109/IEEM.2015.7385970.

[3] Gondzio J., Gruca J.A., Hall J.A.J. et al. Solving large-scale optimization problems related to Bell's Theorem // Journal of Computational and Applied Mathematics. 2014. Vol. 263. P. 392–404. https://doi.org/10.1016/j.cam.2013.12.003.

[4] Dantzig G.B. Linear programming and extensions. Princeton, N.J.: Princeton university press, 1998. 656 p.

[5] Зоркальцев В.И., Мокрый И.В. Алгоритмы внутренних точек в линейной оптимизации // Сибирский журнал индустриальной математики. 2018. Т. 21, 1 (73). С. 11–20. https://doi.org/10.17377/sibjim.2018.21.102.

[6] Kaczmarz S. Angenherte Auflsung von Systemen linearer Gleichungen // Bulletin Interna- tional de l'Acadmie Polonaise des Sciences et des Lettres. Classe des Sciences Mathmatiques et Naturelles. Srie A, Sciences Mathmatiques. 1937. Vol. 35. P. 355–357.

[7] Cimmino G. Calcolo approssimato per le soluzioni dei sistemi di equazioni lineari // La Ricerca Scientifica, XVI, Series II, Anno IX, 1. 1938. P. 326–333.

[8] Agmon S. The relaxation method for linear inequalities // Canadian Journal of Mathematics. 1954. Vol. 6. P. 382–392. https://doi.org/10.4153/CJM-1954-037-2.

[9] Censor Y., Chen W., Combettes P.L. et al. On the effectiveness of projection methods for convex feasibility problems with linear inequality constraints // Computational Optimization and Applications. 2011. Vol. 51, no. 3. P. 1065–1088. https://doi.org/10.1007/S10589-011-9401-7.

[10] Ерёмин И.И. Применение метода фейеровских приближений к решению задач выпуклого программирования с негладкими ограничениями // Журнал вычислительной математики и математической физики. 1969. Т. 9, № 5. С. 1153–1160.

[11] Васин В.В., Ерёмин И.И. Операторы и итерационные процессы фейеровского типа. Теория и приложения. Екатеринбург: УрО РАН, 2005. 211 с.

[12] Ерёмин И.И., Попов Л.Д. Фейеровские процессы в теории и практике: обзор последних результатов // Известия вузов. Математика. 2009. № 1. С. 44–65. https://www.mathnet.ru/php/archive.phtml?wshow=paper&jrnid=ivm&paperid=1253.

[13] Nurminski E.A. Single-projection procedure for linear optimization // Journal of Global Optimization. 2016. Vol. 66, no. 1. P. 95–110. https://doi.org/10.1007/S10898-015-0337-9.

[14] Censor Y. Can linear superiorization be useful for linear optimization problems? // Inverse Problems. 2017. Vol. 33, no. 4. P. 044006. https://doi.org/10.1088/1361-6420/33/4/044006.

[15] Соколинский Л.Б., Соколинская И.М. О новой версии апекс-метода для решения задач линейного программирования // ВестникЮУрГУ. Серия: Вычислительная математика и информатика. 2023. Т. 12, № 2. С. 5–46. https://doi.org/10.14529/cmse230201.

[16] Deutsch F., Hundal H. The rate of convergence for the cyclic projections algorithm I: Angles between convex sets // Journal of Approximation Theory. 2006. Vol. 142, no. 1. P. 36–55. https://doi.org/10.1016/J.JAT.2006.02.005.

[17] Aharoni R., Censor Y. Block-iterative projection methods for parallel computation of solutions to convex feasibility problems // Linear Algebra and its Applications. 1989. Vol. 120. P. 165–175. https://doi.org/10.1016/0024-3795(89)90375-3.

[18] Соколинский Л.Б., Ольховский Н.А., Соколинская И.М. Численная реализация метода поверхностного движения для решения задач линейного программирования // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2024. Т. 13, № 3. С. 5–31. https://doi.org/10.14529/cmse240301.

[19] Ольховский Н.А., Соколинский Л.Б. О новом методе линейного программирования // Вычислительные методы и программирование. 2023. Т. 24, № 4. С. 408–429. https://doi.org/10.26089/NumMet.v24r428.

[20] Мальцев А.И. Основы линейной алгебры. Москва: Наука. Главная редакция физико-математической литературы, 1970. 402 с.

[21] Chase P.J. Algorithm 382: combinations of M out of N objects [G6] // Communications of the ACM. 1970. Vol. 13, no. 6. P. 368. https://doi.org/10.1145/362384.362502.

[22] Gould N.I. How good are projection methods for convex feasibility problems? // Computational Optimization and Applications. 2008. Vol. 40, no. 1. P. 1–12. https://doi.org/10.1007/S10589-007-9073-5.

[23] Boisvert R.F., Pozo R., Remington K.A. The Matrix Market Exchange Formats: Initial Design: tech. rep. / NISTIR 5935. National Institute of Standards; Technology. Gaithersburg, MD, 1996. P. 14. https://nvlpubs.nist.gov/nistpubs/Legacy/IR/nistir5935.pdf.

[24] Соколинский Л.Б., Соколинская И.М. О генерации случайных задач линейного программирования на кластерных вычислительных системах // Вестник Южно-Уральского государственного университета. Серия: Вычислительная математика и информатика. 2021. Т. 10, № 2. С. 38–52. https://doi.org/10.14529/cmse210103.

[25] Dolganina N., Ivanova E., Bilenko R., Rekachinsky A. HPC Resources of South Ural State University // Parallel Computational Technologies. PCT 2022. Communications in Computer and Information Science, vol. 1618 / ed. by L. Sokolinsky, M. Zymbler. Cham: Springer, 2022. P. 43–55. https://doi.org/10.1007/978-3-031-11623-0_4.

Evolutionary optimization in semiclassical quantum modeling of the optical properties of organic pigments

V.A. Kurkov, D.D. Chesalin, R.Y. Pishchalnikov

Prokhorov General Physics Institute of RAS,
Lomonosov Moscow State University, Faculty of Biology

The theoretical study of the physico-chemical properties of organic pigments and crystals in most cases involves modeling the optical response of electronic transitions. Considering that pigments contain tens and even hundreds of atoms, *ab initio* quantum mechanical calculations of electronic excited states for such systems will be quite complicated, especially if we address such cases when it is necessary to evaluate the influence of a solvent or proteins on the properties of electronic transitions of the pigments. The use of semiclassical quantum theories allows simplifying the calculations without losing the quality of the developed models. To make the semiclassical models as plausible as possible, it is necessary to conduct a detailed comparison of the calculated and experimental spectra. Evolutionary algorithms allow for highly accurate fitting of the measured spectra while simultaneously evaluating the statistical significance of the parameters of the quantum model. Differential evolution is one of the most efficient evolutionary optimization algorithms. Assuming the discrepancy between experimental and calculated data as the function to be optimized, the application of this algorithm allowed us to simulate the optical response of photosynthetic pigments and proteins. Using the example of designing semiclassical quantum models for photosynthetic pigments (carotenoids, chlorophylls, and bacteriochlorophylls), we demonstrate significant influence of the tuning of the internal parameters of differential evolution for its efficient performance.

Keywords: evolutionary optimization, differential evolution, multimode Brownian oscillator model, optical response, photosynthetic pigments

1. Introduction

Numerical modeling in fundamental sciences (physics, chemistry, and biology) is necessary when the problem cannot be solved by analytical methods [1–3]. Pigment-protein complexes of photosynthetic organisms consisting of many molecules are one of the illustrative examples of successful application of numerical simulations and multi-parameter optimization [4–6]. The efficiency of optimization algorithms can be demonstrated both on a simple example of calculating absorption spectra of monomeric pigments in organic solvents (carotenoids) and on systems of interacting pigments (photosynthetic pigment-protein complexes) [7, 8].

The aim of theoretical analysis of optical properties of any pigment or protein is to determine quantum microparameters such as the energy of electron excited states and their lifetime, as well as the magnitude and direction of dipole moments of transition from one state to another [9–12]. These microparameters determine the profile of the pigment absorption spectra and its potential ability to exchange energy with other pigments. The use of modern *ab initio* quantum calculations [13, 14] for modeling the electronic and vibrational excited states of organic pigments allows us to simulate the optical properties of monomeric molecules with high accuracy, but such calculations for systems of interacting pigments become extremely time-consuming.

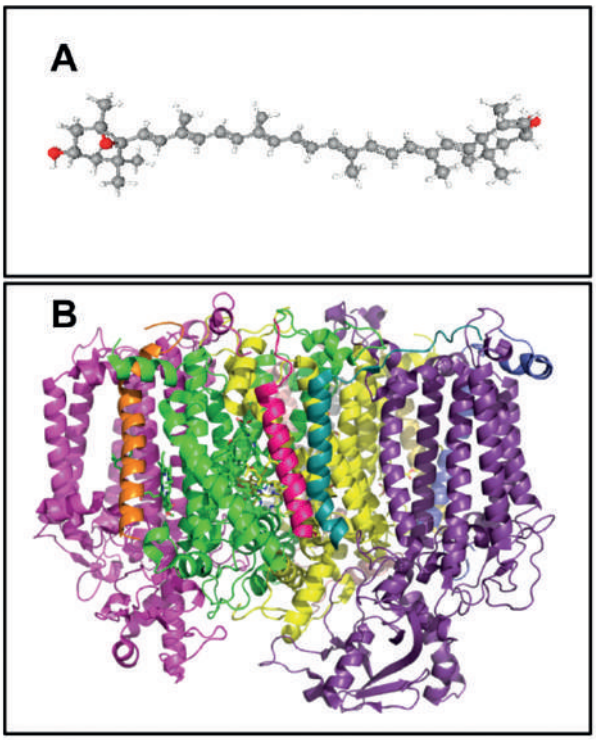


Figure 1: The chemical structure of lutein 5,6-epoxide (A) and the crystal structure of the reaction center of photosystem II (B)

To speed up the calculations, an alternative to *ab initio* calculations can be considered modeling the optical response using the semiclassical quantum theory of the interaction of electromagnetic radiation with matter [3–6]. This theory is based on the concept of the spectral density function, which carries information about the effective vibrational modes characterizing the studied electronic excitation of the molecule. Each vibrational mode is defined by three quantities: frequency, attenuation coefficient and intensity of interaction with the electronic excitation [4, 5]. Due to the phenomenological nature of this approach, a comparison of the calculated spectra and measured spectra is necessary to estimate the parameters of the system under study

Eventually, the problem of modeling the optical properties of photosynthetic pigments can be solved using evolutionary optimization algorithms, the heuristic nature of which enables us to tune the optimal mode of operation of the optimizer without going into the nuances and details of the specific problem being optimized [15–18]. The differential evolution algorithm, designed to find the global minimum of nonlinear and nondifferentiable

functions from many variables [19–22], was taken as a basis for the modeling. The algorithm is based on the method of selecting a trial mutant vector of optimized parameters; to obtain the trial vector, the algorithm adds the scaled difference of two random vectors to a third, randomly selected population vector.

Using the example of modeling the absorption spectra of β-carotene and the reaction center of photosystem II (Figure 1), we demonstrate that the simultaneous application of semiclassical quantum theory and differential evolution allows us to determine statistically significant parameters of the carotenoid spectral density and exciton model of the pigment-protein complex.

2. Materials and methods

2.1. Differential evolution

The classic optimization algorithm of Differential Evolution (DE) includes parts of initialization and the basic cycle, which in turn consists of three consecutive procedures: mutation, crossing and selection [19, 20]. DE initialization is performed only once, while the basic cycle procedures are repeated sequentially as many times as necessary to minimize the objective function and the appropriate simulated spectra.

During initialization, the matrix of model parameters $\mathbf{X}$ is filled with random values in a given interval, the size of this matrix is $D \times Np$ and it is based on a vector of real parameters $\mathbf{x} = \{x_0, x_1, \ldots, x_{D-1}\}$ of size D that need to be optimized, while Np is an adjustable parameter called the size of a population. In this case, the matrix can be defined as $\mathbf{X} = \{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{(Np-1)}\}$. The elements of the matrix are chosen taking into account the boundary conditions, which consider the physical limits of the parameters to find: $\mathbf{B} = \{B_j^{\text{lower}}, B_j^{\text{upper}}\}$. In this regard, the initial values of the ith vector $\mathbf{x}_i^0$ are selected using the following formula:

$$x_j i^0 = B_j^{\text{lower}} + \text{rand}_{ji}(0,1) \times (B_j^{\text{upper}} - B_j^{\text{lower}}). \tag{1}$$

Here, j is in the range from 0 to $D-1$, and $\text{rand}_{ji}(0,1)$ is a number that is randomly selected for each parameter x_{ji}^0 from the range $[0,1]$ by a homogeneous random number generator. After initialization, the objective functions for $\mathbf{x}_i^0$ are calculated and the minimum value is stored.

Then the main DE cycle begins its work. At the first stage, the cycle algorithm generates a new matrix $\mathbf{X}^g$ of mutant parameters. In the classical version of DE, the mutant vectors that make up the matrix are calculated according to one of the following five expressions:

$$\mathbf{v}_i^g = \mathbf{x}_{r0}^g + F(\mathbf{x}_{r1}^g - \mathbf{x}_{r2}^g), \tag{2}$$

$$\mathbf{v}_i^g = \mathbf{x}_{\text{best}}^g + F(\mathbf{x}_{r1}^g - \mathbf{x}_{r2}^g), \tag{3}$$

$$\mathbf{v}_i^g = \mathbf{x}_{r0}^g + F(\mathbf{x}_{\text{best}}^g - \mathbf{x}_{r0}^g) + F(\mathbf{x}_{r1}^g - \mathbf{x}_{r2}^g), \tag{4}$$

$$\mathbf{v}_i^g = \mathbf{x}_{\text{best}}^g + F(\mathbf{x}_{r1}^g - \mathbf{x}_{r2}^g) + F(\mathbf{x}_{r3}^g - \mathbf{x}_{r4}^g), \tag{5}$$

$$\mathbf{v}_i^g = \mathbf{x}_{r0}^g + F(\mathbf{x}_{r1}^g - \mathbf{x}_{r2}^g) + F(\mathbf{x}_{r3}^g - \mathbf{x}_{r4}^g). \tag{6}$$

Here, $g = 0, 1, \ldots, g_{\max}$ is the index for numbering generations, thus x_{best}^g is a vector corresponding to the best solution, that is, the minimum of the objective function.

F is the weighting factor and F$\in [0, 1]$; however, in general it can be any positive real number. $\mathbf{x}^g_{r0}$, $\mathbf{x}^g_{r1}$, and $\mathbf{x}^g_{r2}$ are the randomly chosen vectors of the current population, and each time they are selected, the following rule is followed for their indexes $(i \neq r0 \neq r1 \neq r2) \in [0, Np]$.

The trial vector population's diversity can be increased by using a crossover approach. To do this, the DE algorithm creates a new trial vector $\mathbf{u}^g_j$ by exchanging the parameter values of each target vector of the current population with those of a mutant vector. The crossover rate, Cr $\in [0, 1]$, determines the number of exchanged values in the trial vector.

There are two types of crossover: binomial and exponential. In the case of binomial crossover, the jth element of a trial vector is substituted by that of a mutant vector only if the random number generated by the uniform random number generator is less than or equal to Cr, or else the jth element is simply copied from a target vector:

$$u^g_{ji} = \begin{cases} v_j i^g, (\text{rand}_{ji}(0,1) \leq \text{Cr}) \,\text{OR}(j = j_{\text{rand}}), \\ x^g_{ji}. \end{cases} \tag{7}$$

At the same time, the exponential crossover method utilizes two integer values, that are chosen from the interval of the vector indexes. The first integer $n \in [1, D]$ corresponds to the starting point of a target vector; substitution begins with the nth element. And the second integer $L \in [1, D]$ equals the number of the elements to be copied. This value is calculated according to the following algorithm: starting from $L = 0$, a do-while cycle with $L = L + 1$ as a body works until the condition $(\text{rand}(0, 1) \leq \text{Cr}) \,\text{AND}(L \leq D)$ becomes false. Ultimately, the trial vector is

$$u^g_{ji} = \begin{cases} v^g_{ji}, \ \forall j \in \{\langle n \rangle_D, \ \langle n+1 \rangle_D, \ \ldots, \ \langle n+L-1 \rangle_D\}, \\ x^g_{ji}, \end{cases} \tag{8}$$

where $\langle \ldots \rangle_D$ is a modulo function with modulus D. And finally, the target vector of a new generation $g + 1$ is created by comparing values of the objective functions $f(\mathbf{u}^g_i)$ and $f(\mathbf{x}^g_i)$.

Combinations of two types of crossover and five equations defining a mutant vector can be considered 10 strategies to create a new generation of parameters $\mathbf{X}^{g+1}$. The names of the strategies are constructed as follows: $DE/x/y/z$, where x is the choice of a base vector (rand, best, rand-to-best), y is the number of vector differences (1 or 2), and z is a crossover type (exponential or binomial). Thus, the convergence of DE can be managed through the selection of a well-suited strategy and through adjustments to the weighting factor and crossover rate.

To fit absorption spectra for biological pigments, the objective function is defined using the following equation:

$$f(\mathbf{x}^g_i) = \frac{1}{N} \sum_{n=1}^{N} \big(I(\omega_n) - \sigma_{\text{abs}}(\omega_n, \mathbf{x}^g_i)\big)^2. \tag{9}$$

Here, $I(\omega_n)$ is a measured spectrum of a pigment molecule at frequencies ω_n; $\sigma_{\text{abs}}(\omega_n, \mathbf{x}^g_i)$ is a simulated spectrum in terms of the MBOM; and N is the number of points in the spectra. After the objective functions are evaluated, $\mathbf{x}^{g+1}_i$ vector allocation is performed according to the following requirements:

$$\mathbf{x}^{g+1}_i = \begin{cases} \mathbf{u}^g_i, & f(\mathbf{u}^g_i) \leq f(\mathbf{x}^g_i), \\ \mathbf{x}^g_i, & f(\mathbf{u}^g_i) > f(\mathbf{x}^g_i), \end{cases} \tag{10}$$

After a new population is fully processed, the subsequent DE cycle continues either until the objective function reaches its minimum value or the predetermined maximum number of generations is achieved.

2.2. Theory of the pigment optical response

A general expression for the absorption spectrum of a monomeric pigment can be written in an integral form:

$$\sigma_{\text{abs}}(\omega) = \int_{-\infty}^{\infty} dt\, S^{(1)}(t_1) e^{i\omega t}, \tag{11}$$

here $S^{(1)}(t_1)$ is a linear optical response function, which is calculated in terms of the correlation function of an electronic transition moment of a pigment [2, 5, 6]:

$$S^{(1)}(t_1) = \frac{i}{\hbar}\theta(t_1) e^{-i\omega_{eg}t_1 - g(t_1)} + c.c. \tag{12}$$

$$g(t) = \int_0^t d\tau_2 \int_0^{\tau_2} d\tau_1\, C(\tau_1), \tag{13}$$

where $g(t)$ is the lineshape function; $C(\tau_1)$ is the two-time correlation function of transition moment. $C(\tau_1)$ is a complex function; and according to the fluctuation-dissipation theorem the general expression can be written as following:

$$C(t) = \int_{-\infty}^{\infty} d\omega\, \cos(\omega t) \coth(\beta\hbar\omega/2) C''(\omega) + i \int_{-\infty}^{\infty} d\omega\, \sin(\omega t) C''(\omega), \tag{14}$$

where $C''(\omega)$ is the imaginary part of $C(\omega)$ and can be treated classically. This feature of $C''(\omega)$ makes it quite suitable for the modeling of the optical response. Thus, the equation for $g(t)$ in terms of $C''(\omega)$ is written as:

$$g(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} d\omega \frac{1-\cos\omega t}{\omega^2} \coth(\beta\hbar\omega/2) C''(\omega) - \frac{i}{2\pi} \int_{-\infty}^{\infty} d\omega \frac{\sin(\omega t) - \omega t}{\omega^2} C''(\omega). \tag{15}$$

Taking into account Equations (2)–(6), the final expression for numerical simulation of the absorption lineshape is given by:

$$\sigma_{\text{abs}}(\omega) = \frac{1}{\pi} \operatorname{Re} \int_0^{\infty} dt\, e^{i(\omega-\omega_{eg})t} e^{-g(t)}. \tag{16}$$

Several approaches can be applied to evaluate the imaginary part of C(t) called the spectral density function:

$$C''(\omega) = \sum_j \frac{2 S_j \omega_j^3 \omega \gamma_j}{(\omega_j^2 - \omega^2)^2 + \omega^2 \gamma_j^2}. \tag{17}$$

This expression already contains a set of effective parameters $\{\omega_j, S_j, \gamma_j\}$ representing the effect of vibrational modes on the electronic transition. ω_j is the frequency of the

jth vibrational mode; S_j is the Huang–Rhys factor, which is proportional to the electron-phonon coupling; γ_j is the damping factor of the jth vibrational mode.

Finally, to calculate the absorption spectrum of a monomeric pigment (16) one has to evaluate the spectral density (17), the lineshape function (15).

2.3. Exciton theory

To model the optical response of a system of interacting pigments, we can use the theory of molecular excitons considering electronic transition and interaction energies between pigment molecules and allows calculating the contributions of each pigment molecule to the resulting spectra and population kinetics. If the system consists of N two-level molecules and each molecule can be either in a ground $|0\rangle$ or in an excited $|n\rangle$ state, n runs from 1 to N. Denoting $B_n^+ = |n\rangle\langle 0|$ as the exciton creation operator and $B_n = |0\rangle\langle n|$ as that of annihilation. The exciton Hamiltonian is expressed as:

$$H_{\text{ext}} = \sum_n \Omega_n B_n^+ B_n + \frac{1}{2}\sum_{n\neq m} J_{mn}(B_m^+ B_n + B_n^+ B_m) \tag{18}$$

where $B_n^+ = |n\rangle\langle 0|$ and $B_n = |0\rangle\langle n|$ are the exciton creation and annihilation operator, which obey the commutation rules: $[B_n, B_n^+] = 1$. J_{mn} is a matrix of coupling energies calculated employing the extended dipole approximation. This method of calculating the interaction energies using the values of partial charges is much more accurate than the classical dipole–dipole approximation.

Diagonalizing the Hamiltonian, we obtain the eigenstates c_n^α and eigenvalues ϵ_α that allow for the transformation of the system parameters from the site representation to the exciton representation. Thereby, the lineshape function (Equation (15)) in the exciton representation is $g_{\mu\eta\alpha\beta}(t) = \sum_{mnkl} c_m^\mu c_n^\nu c_k^\alpha c_l^\beta g_{mnkl}(t)$, where $\alpha, \beta, \ldots = 1 \ldots N$ are indices of the exciton states. Ultimately, we will express the exciton absorption and circular dichroism spectra by summing over the exciton states.

$$\sigma_{\text{abs}}^{\text{ext}}(\omega) \approx \frac{\omega}{\pi}\sum_\alpha^N \mathbf{d}_\alpha^2 \,\text{Re}\int_0^\infty dt\, e^{i(\omega-\epsilon_\alpha)t} e^{-g_{\alpha\alpha\alpha\alpha}(t)} e^{-0.5K_{\alpha\alpha}t}, \tag{19}$$

$$\sigma_{\text{CD}}^{\text{ext}}(\omega) \approx \frac{\omega}{\pi}\sum_\alpha^N \mathbf{R}_\alpha \,\text{Re}\int_0^\infty dt\, e^{i(\omega-\epsilon_\alpha)t} e^{-g_{\alpha\alpha\alpha\alpha}(t)} e^{-0.5K_{\alpha\alpha}t}, \tag{20}$$

$K_{\alpha\alpha} = \sum_\beta K_{\alpha\beta}$ are the exciton relaxation rates; $\mathbf{d}_\alpha = \sum_n c_n^\alpha \mathbf{d}_n$ is the Q_y transition moments transformed to the exciton representation; and $\mathbf{R}_\alpha = \sum_{nm} c_n^\alpha c_m^\alpha r_{nm}(\mathbf{d}_n \times \mathbf{d}_m)$ is a matrix of the rotational strength necessary for CD spectra simulation.

3. Computational details

The computational complexity of the classical DE algorithm is calculated based on several factors. At the initialization stage, there are two loops, one nested within the other. The outer loop has a complexity of $O(Np) = O(10n) \to O(n)$, where n is the number of variables being optimized, and the inner loop has a complexity of $O(n)$. Therefore, the total complexity of this stage is $O(n^2)$. After the initialization, a while loop iterates over

the generations of the DE ($O(g_{\max})$), where $g_{\max}$ is the maximum number of generations. This loop contains another loop with a complexity of $O(Np)$, resulting in an overall complexity of $O(ng_{\max})$. Therefore, the overall computational complexity of classical DE can be expressed as $O(n^2 + ng_{\max})$.

A PC with the following configuration was used for the simulations: Intel Core i9-14900K processor (Raptor Lake), with 8 performance cores and 16 efficient cores, for a total of 32 threads. The motherboard is the Rog Strix Z790-A, with a PCI-Express 5.0 bus (16 G/s) and 64 GB DDR5 memory. The operating system is Linux Workstation 6.11.0-24-generic #24∼24.04.1-Ubuntu.To simulate the spectra of carotenoids and pigment-protein complexes, we wrote a C++ program using the OpenMP API for parallelization, implemented in the GNU Offloading and Multiprocessing Runtime Library. This library makes it easier to parallelize simulations of the optical response and set different configurations at runtime. Unlike with MPI, we didn't need to completely rewrite the original program or do time-consuming tests for possible parallelization locations.

4. Results and discussion

To perform the fitting of experimental data the spectra of lutein 5,6-epoxide in tetrahydrofuran and the reaction center of photosystem II measured at room temperature were used.

According to our previous modeling of the linear optical response of monomeric photosynthetic pigments, *DE/rand-to-best/1/exp* strategy has demonstrated the best convergence rates. Preliminary trial runs of the optimization algorithm for lutein 5,6-epoxide and the reaction center of photosystem II showed that, in general, the results of convergence are similar to those we obtained for other monomeric pigments. In the case of reaction center, it was decided to run a strategy test where control parameters varied within the range of values that proved to be the most optimal for the monomer, namely from 0.55 to 0.85 for F and from 0.8 to 1.0 for Cr with a discrete step of 0.05.

Although the theories used to model the monomers and systems differ, we had the same number of free parameters when modeling the 5,6-epoxide and reaction center of photosystem II. For the monomer 12 parameters were used: the electronic transition energy Ω_{eg}, FWHM of inhomogeneous broadening, three values of the lowest vibrational mode $\{\omega_{\text{low}}, S_{\text{low}}, \gamma_{\text{low}}\}$, and seven Huang–Rhys factors S_j. For the pigment-protein complex: eight electronic transition energies Ω_n, the effective dielectric constant, which is used to calculate the coupling energies between pigments, and three values of $\{\omega_{\text{low}}, S_{\text{low}}, \gamma_{\text{low}}\}$ for the lowest vibrational mode in the spectral density, which is considered to be the same for all molecules. The size of the population was calculated as $Np = 10 * n$, where n is the number of free parameters.

The same number of free parameters does not mean at all that the calculation time of the absorption spectrum will be the same for the monomer and the system. Since the calculation of the relaxation rates between electronic excitations is included in the total calculation of the optical response of the system, it requires more computational time. The maximum number of generations was 300 in both cases. The results of the fitting are shown in Figure 2.

Despite the initial discrepancies, in the case of lutein in tetrahydrofuran the objective function turned out to be less than 10^{-4}, which corresponds to a very good agreement between the experimental and simulated spectrum. In the case of artificial absorption

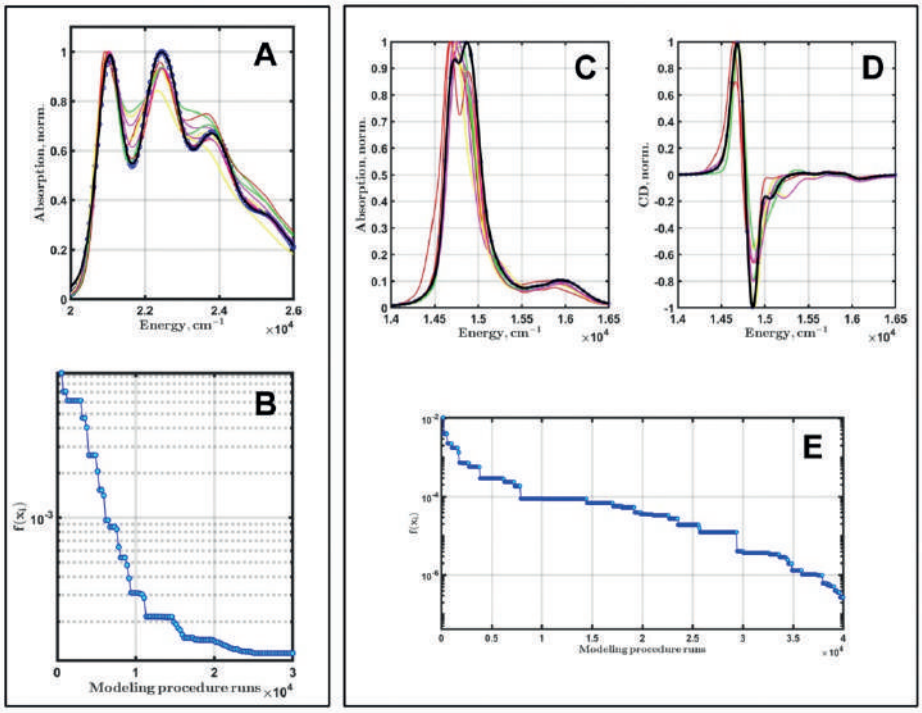


Figure 2: The final simulated spectrum (blue line with circles) and the intermediate results (colored thin lines) for lutein 5,6-epoxide in tetrahydrofuran (A). Dependence of the objective function on the number of modeling procedure runs (B). Simulated absorption (C) and circular dichroism spectra (D) of the reaction center photosystem II. Dependence of the objective function on the number of modeling procedure runs (E).

spectra and circular dichroism, the objective function was below 10^{-7} after 40,000 runs of the simulation. Moreover, in the case of a longer program running time, the residual function successfully reaches machine epsilon, while in the case of lutein 5,6-epoxide, the objective function asymptotically tends to a limit depending on the quality of measured spectra.

5. Conclusions

We have shown that the use of a heuristic evolutionary algorithm such as DE in modeling the optical properties of PPCs allows us to obtain high-quality calculated spectra and, at the same time, to assess the uniqueness of the obtained parameters of the exciton model of the energy transfer in PPCs. In this study, differential evolution was used for simulation of the linear optical response of lutein 5,6-epoxide in tetrahydrofuran and the system of interacting chlorophyll pigments (reaction center of photosystem II). Applying the semiclassical quantum theory and the theory of molecular excitons, we have demonstrated that the linear optical response of the reaction center (absorption, circular dichroism) can be simulated by the algorithm with high accuracy. To explore the effectiveness of differential evolution, we used the simulated experimental data as the target functions instead of those of actually measured. The best tuning parameters were determined to run the full optimization of the linear optical response modeling of the pigment-protein complex. Finally, using the DE/rand-to-best/1/exp strategy, we found the exact solution for the quantum model of lutein 5,6-epoxide and the reaction center of photosystem II.

6. Acknowledgement

R.Y. Pishchalnikov and D.D. Chesalin were supported by the Russian Science Foundation (RSF # 25-21-00375, https://rscf.ru/en/project/25-21-00375/)

References

[1] Caycedo-Soler F., Mattioni A., Lim J., Renger T., Huelga S.F., and Plenio M.B. Exact simulation of pigment-protein complexes unveils vibronic renormalization of electronic parameters in ultrafast spectroscopy // Nat. Commun., 13, 2022. https://doi.org/10.1038/s41467-022-30565-4.

[2] Jang S.J. and Mennucci B. Delocalized excitons in natural light-harvesting complexes // Rev. Mod. Phys., 90, 035003, 2018. https://doi.org/10.1103/RevModPhys.90.035003.

[3] Mirkovic T., Ostroumov E.E., Anna J.M., van Grondelle R., Govindjee, and Scholes G.D. Light Absorption and Energy Transfer in the Antenna Complexes of Photosynthetic Organisms // Chem. Rev., 117, 249-293, 2017. https://doi.org/10.1021/acs.chemrev.6b00002.

[4] Renger T. Semiclassical Modified Redfield and Generalized Forster Theories of Exciton Relaxation/Transfer in Light-Harvesting Complexes: The Quest for the Principle of Detailed Balance // J. Phys. Chem. B, 125, 6406-6416, 2021. https://doi.org/10.1021/acs.jpcb.1c01479.

[5] Zhang W.M., Meier T., Chernyak V., and Mukamel S. Exciton-migration and three-pulse femtosecond optical spectroscopies of photosynthetic antenna complexes // J. Chem. Phys., 108, 7763-7774, 1998. https://doi.org/10.1063/1.476212.

[6] Meier T., Chernyak V., and Mukamel S. Femtosecond photon echoes in molecular aggregates // J. Chem. Phys., 107, 8759-8780, 1997. https://doi.org/10.1063/1.475169.

[7] Chesalin D.D., Kulikov E.A., Yaroshevich I.A., Maksimov E.G., Selishcheva A.A., and Pishchalnikov R.Y. Differential evolution reveals the effect of polar and non-polar solvents on carotenoids: A case study of astaxanthin optical response modeling // Swarm Evol. Comput., 75, 101210, 2022. https://doi.org/10.1016/j.swevo.2022.101210.

[8] Chesalin D.D. and Pishchalnikov R.Y. Searching for a Unique Exciton Model of Photosynthetic Pigment–Protein Complexes: Photosystem II Reaction Center Study by Differential Evolution // Mathematics, 10, 2022. https://doi.org/10.3390/math10060959.

[9] Bruggemann B., Sznee K., Novoderezhkin V., van Grondelle R., and May V. From structure to dynamics: Modeling exciton dynamics in the photosynthetic antenna PS1 // J. Phys. Chem. B, 108, 13536-13546, 2004. https://doi.org/10.1021/jp0401473.

[10] Raszewski G., Saenger W., and Renger T. Theory of optical spectra of photosystem II reaction centers: Location of the triplet state and the identity of the primary electron donor // Biophys. J., 88, 986-998, 2005. https://doi.org/10.1529/biophysj.104.050294.

[11] Vaitekonis S., Trinkunas G., and Valkunas L. Red chlorophylls in the exciton model of photosystem I // Photosynth. Res., 86, 185-201, 2005. https://doi.org/10.1007/s11120-005-2747-x.

[12] Adolphs J. and Renger T. How proteins trigger excitation energy transfer in the FMO complex of green sulfur bacteria // Biophys. J., 91, 2778-2797, 2006. https://doi.org/10.1529/biophysj.105.079483.

[13] Cremer D. and Pople J.A. General definition of ring puckering coordinates // J. Am. Chem. Soc, 97, 1354-1358, 1975. https://doi.org/10.1021/ja00839a011.

[14] Ditchfield R., Hehre W.J., Pople J.A. Self-consistent molecular-orbital methods. 9. Extended gaussi-an-type basis for molecular-orbital studies of organic molecules // J. Chem. Phys. 54, 724-728, 1971. https://doi.org/10.1063/1.1674902.

[15] Vesterstrom J. and Thomsen R. A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems // in Proceedings of the 2004 Congress on Evolutionary Computation, 2004, 1980-1987. https://doi.org/10.1109/CEC.2004.1331139.

[16] Darwish A., Hassanien A.E., and Das S. A survey of swarm and evolutionary computing approaches for deep learning // Artif. Intell. Rev., 53, 1767-1812, 2020. https://doi.org/10.1007/s10462-019-09719-2.

[17] Del Ser J., Osaba E., Molina D., Yang X.S., Salcedo-Sanz S., Camacho D. et al. Bio-inspired computation: Where we stand and what's next // Swarm Evol. Comput., 48, 220-250, 2019. https://doi.org/10.1016/j.swevo.2019.04.008.

[18] Ali M.M. and Torn A. Population set-based global optimization algorithms: Some modifications and numerical studies // Comp. Oper. Res., 31, 1703-1725, 2004. https://doi.org/10.1016/S0305-0548(03)00116-3.

[19] Storn R. System design by constraint adaptation and differential evolution // IEEE Trans. Evol. Comput., 3, 22-34, 1999. https://doi.org/10.1109/4235.752918.

[20] Storn R. and Price K. Differential evolution — A simple and efficient heuristic for global optimization over continuous spaces // J. Glob. Optim., 11, 341-359, 1997. https://doi.org/10.1023/A:1008202821328.

[21] Opara K.R. and Arabas J. Differential Evolution: A survey of theoretical analyses // Swarm Evol. Comput., 44, 546-558, 2019. https://doi.org/10.1016/j.swevo.2018.06.010.

[22] Trivedi A., Srinivasan D., Biswas S., and Reindl T. A genetic algorithm — Differential evolution based hybrid framework: Case study on unit commitment scheduling problem // Inf. Sci., 354, 275-300, 2016. https://doi.org/10.1016/j.ins.2016.03.023.

III D моделирование образования и эволюции сверхплотных ядер, возникающих в филаментах при сверхзвуковом соударении молекулярных облаков

Б.П. Рыбакин

НИЦ “Курчатовский институт”, Россия,
МГУ имени М.В. Ломоносова, Россия

В работе представлены результаты моделирования сверхзвукового соударения молекулярных облаков в трехмерной постановке на многопроцессорных компьютерах. В этой модели плотность ядра увеличивается за счет того, что она аккрецирует газ из окружающей среды, увеличивая свою массу и уменьшая свою длину Джинса. Недавние наблюдения показывают, что столкновения плотных ядер, которые ранее считались второстепенными при звездообразовании, на самом деле играют значительную роль в формировании структур вокруг протозвездных оболочек и формировании двойных звезд. Используя данные, полученные при наблюдении звезды Barnard 68 и близлежащих областей звездообразования, можно определить частоту столкновений ядер. Полученные результаты показывают, что ядра, вероятно, будет подвергаться множественным взаимодействиям с другими ядрами на протяжении всего периода своего существования. Для дальнейшего исследования процесса столкновения ядра мы используем гидродинамическое моделирование с помощью адаптивного измельчения сетки. Соударения молекулярных облаков, их столкновения с ударными волнами приводят к формированию сверхзвуковой турбулентности и образованию сильно сжатых областей, плотность в которых на несколько порядков превышает плотность газа в облаке. Эти процессы вызывают появление пленочных и нитевидных структур — филамент. Расчеты проводятся на гетерогенных компьютерах, на сетках с разрешением до 1024^3 узлов. В расчетах используется алгоритм адаптивного уточнения сетки (AMR), с возможностью уточнения размеров сетки до 5 уровней.

Ключевые слова: образования звезд, параллельные вычисления, численные методы, астрофизическая гидродинамика

1. Введение

Исследования, проведенные в последние десятилетия, показывают, что динамическая эволюция и морфологическое строение газа зависят от сверхзвуковой турбулентности и что новые звезды формируются в результате гравотурбулентной фрагмента-

ции [1]. Эти наблюдения, которые проводились в последние десять — пятнадцать лет с помощью наземных телескопов, таких как JCMT [2] и космических телескопов Spitzer и Herschel [3, 4] значительно улучшили наше понимание начальных процессов формирования звезд и звездных систем. Но, с другой стороны, стало понятно, что эти условия значительно отличается от тех моделей, которые использовались до сих пор. Выяснилась необходимость исследовать процесс образования и эволюции протозвездных ядер, включая изучение процессов их соударения между собой.

В большом масштабе несколько теоретических и наблюдательных исследований показали, что плотные ядра могут накапливать дополнительную массу из окружающей среды. В работах [5, 6] было проведено численное моделирование формирования звездных кластеров, что подтвердило идею о том, что ядра могут приобретать значительное количество газа из окружающей среды. Эти исследования, а также результаты, приведенные в работах [7, 8] показали, что образовавшиеся филаменьные структуры являются местами образования новых звездных систем. Формирование новых звезд начинается при достижении плотности газа величин порядка 10^{-20}–10^{-19} г/см^3, что на 5–6 порядков выше плотности межзвездного газа. При этом типичные наблюдаемые размеры образовавшихся сжатых областей составляет порядка нескольких парсек (1 парсек равен 3.26 световых лет), и они имеют массу порядка от 10 до 10^3 масс Солнца.

Наши расчеты проводятся на декартовых сетках с разрешением порядка 1024^3, на гетерогенных вычислительных системах. В расчетах используется алгоритм адаптивного уточнения сетки (AMR), с возможностью уточнения размеров сетки до 5 уровней. Такое уточнение сетки применяется для гарантирования выполнения ограничения на длину Джинса. Длина Jeans $L_J = c_s(\pi/(G\rho))^{1/2}$ связывает размеры гравитационно связанного ядра с его массой $M_J = 4/3\pi\rho r^3$. Сила гравитации в ядре уравновешивается тепловым давлением, но, если радиус ядра становится меньше длины Джинса и при этом имеет массу M_J и больше, силы гравитации начинают преобладать над тепловым давлением. В этом случае ядро начинает коллапсировать и этот процесс завершается за время $t_{ff} = (3\pi/32G\rho)^{1/2}$ [8]. При соударении молекулярных облаков (МО) на больших скоростях (более 5 км/сек) указанного уточнения сетки не хватает для выполнения условия Джинса. Гравитационный коллапс протозвездных ядер приводит к необходимости дальнейшего повышения разрешения расчетной сетки и соответственно, увеличения расчетного времени.

Целью данного исследования является создание параллельного алгоритма и проведение численного моделирования сверхзвукового соударения молекулярных облаков, взаимных столкновений образующихся сверхплотных ядер между собой и с ударными волнами. В таких процессах преобладает сверхзвуковая турбулентность [8]. Соударения сопровождаются периодическими возмущениями в распределении плотности вещества в новообразованных сгустках. В таких сгустках, которые содержат большое количество конденсированного вещества, плотность газа может достигать значений, находящихся на начальном уровне дозвездных образований. При этом плотность в них повышается на шесть и более порядков, что является предзвездной плотностью. Моделирование астрофизических процессов требует использования все более современных инструментов для поддержки трудоемких расчетов при максимально возможном распараллеливании операций. В работе используется собственный адаптивный магнитогидродинамический код, основанный на методах Годунова

высокого порядка точности. Возникающее движение описывается системой трехмерных уравнений Эйлера с учетом гравитации.

Формирование новых звезд и звездных систем происходит в плотных скоплениях газа — молекулярных облаках. Это связано с тем, что плотность газа в молекулярных облаках существенно превышает плотность в межзвездной среде, поэтому образование предзвездных сгустков здесь более вероятно. Процессы образования звезд и звездных скоплений еще до конца не очень хорошо изучены. Новые наблюдения дают данные для понимания этого процесса [9, 10]. Эти наблюдения показывают, что звезды и звездные скопления формируются в турбулентных, плотных сгустках, которые находятся в намагниченных молекулярных облаках. Такие сгустки содержат массы порядка нескольких масс Солнца и в дальнейшем они фрагментируются в скопление ядер, часть из которых обладает достаточной массой, и для них выполняется условие Jeans, что может привести к их коллапсу.

Кроме того, проведенные наблюдения показывают, что эти сверхплотные ядра испытывают постоянные соударения друг с другом [11]. В качестве примера можно привести звезду Barnard 68 [12], которая служит хорошим наблюдательным примером роста ядер в результате их столкновений. До недавнего времени существовало мнение о том, что столкновения плотных ядер происходят крайне редко и не играют значительную роль в формировании структур вокруг протозвездных оболочек. Наблюдения показывают, что ядра в Barnard 68 подвергаются множественным взаимодействиям с другими ядрами на протяжении всего времени своего существования. В работе [13] приведена оценка частоты столкновений в области звездообразования, с учетом того, что все ядра имеют сферическую форму со средним радиусом $\langle Rc \rangle$ и средней скоростью $\langle Vc \rangle$ и равномерно распределены в трехмерном пространстве. Обозначим через $N_c = \rho_c/m$ числовую плотность ядер, которая имеет размерность длина в минус третьей степени $[L^{-3}]$. Здесь ρ_c обычная плотность, m — масса протона. Принимается, что масса протона равна 2.3 и имеет размерность массы. Таким образом, числовая плотность $N_c = 10^3$ равна обычной плотности $\rho_c = 4.008 \times 10^{-21}$ г/см^3. В этом случае время столкновения оценивается как: $\tau_{\text{col}} = 1/(4\pi R_c^2 N_c \langle V_c \rangle)$. Время жизни предзвездного ядра до его коллапса определяется как: $t_{ff} = (3\pi/32G\rho_c)^{1/2}$ [8]. Для протоядер массой в несколько масс Солнца, это время равно ~ 0.5–1.0 млн. лет. Статистические исследования времени жизни предзвезд показывают, что их типичное время жизни колеблется от 1 до 10 t_{ff}. Таким образом можно оценить количество соударений, которые происходят в областях звездообразования. Оно сильно зависит от начальной массы молекулярных облаков, скорости соударения, влияния магнитных полей и целого ряда других факторов. Таким образом, при числовой плотности ядра от 10^4 до 10^5 см$^{-3}$ время жизни ядра составляет около 1 млн лет. Частота столкновений определяется как отношение времени жизни ядра к времени столкновения. Проведенное моделирование показало, что частота соударений ядер за время их жизни, колеблется (в зависимости от начальных условий) в диапазоне от 2.1 до 5.7 раз. Это достаточно хорошо коррелирует с данными, приведенными в [13]. Например, для области Serpens South количество соударений равно 2.6, а для AGAL014.492-00.139 частота столкновений оценивается как 5.7. Существуют области, для которых частота соударений существенно отличается. Например, для ρOph частота равна 15.3 [13]. Наблюдения молекулярных линий этих областях звездообразования позволяет предположить, что большинство ядер в ней гравитационно не связаны [14, 15].

2. Постановка задачи

Моделирование осуществляется в предположении, что ядра погружены в среду с постоянной плотностью и их плотность увеличивается как в результате соударения между собой, так и с помощью аккреции газа из этой среды. На рис. 1 приведены результаты численного моделирования нецентрального соударения двух сферических ядер со скоростью 4.5 км/сек. Слева приведен результат моделирования на начальные моменты времени, порядка $t = 100$ тысяч лет, справа — на момент времени $t = 750$ тысяч лет. В результате соударения происходит рост плотности ядер.

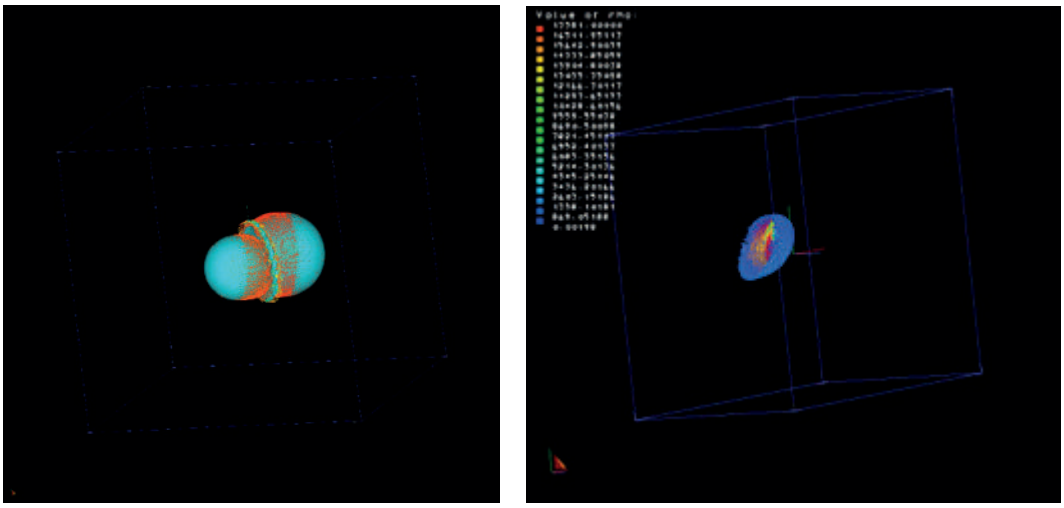


Рис. 1: Моделирование процесса нецентрального соударения двух сферических ядер. На левом графике представлена начальная стадия соударения, справа — на момент времени $t = 750000$ лет

К моменту времени $t = 750$ тысяч лет максимальная плотность отдельных ядер повысилась на 3 порядка. При этом, соответственно, уменьшилась Джинсовская длина. Часть таких сверхплотных структур становится гравитационно связанными, и в дальнейшем из них могут сформироваться новые звезды и звездные системы. Данные, полученные с космических телескопов Spitzer и Herschel [3, 4], показали, что взаимодействие таких протозвездных структур между собой играют значительную роль в формировании молодых звездных объектов (Young Star Objects — YSO) и двойных звездных систем [13, 15]. Кроме того, наблюдения показывают, что фаза аккреции часто сопровождается мощным выбросом биполярных струй вещества — джетов (рис. 2). На левом рисунке показан начальный процесс формирования джета, приведена плотность вещества, на которую наложено поле скорости. Справа показан момент проникания джета в эллиптическое ядро. Такие осесимметричные струи, возникающие вблизи формирующихся протозвезд, оказывают большое влияние на взаимодействия протозвездных ядер. Джеты и соударяющиеся ядра (сгустки) часто связаны с одними и теми же молодыми звездными объектами [16, 17]. Такой выброс большого количества вещества (уносится порядка 8–10% вещества) из формирующейся звезды в виде струй пока еще недостаточно хорошо изученный процесс. Часто первым наблюдаемым признаком рождения новой звезды является образование биполярной струи и истечение молекулярного газа. Кроме того, учитывая высокие потоки массы и достаточно большие импульсы, их повсеместность и длительность, струи и взаимные соударения играют основную роль в формировании звезд.

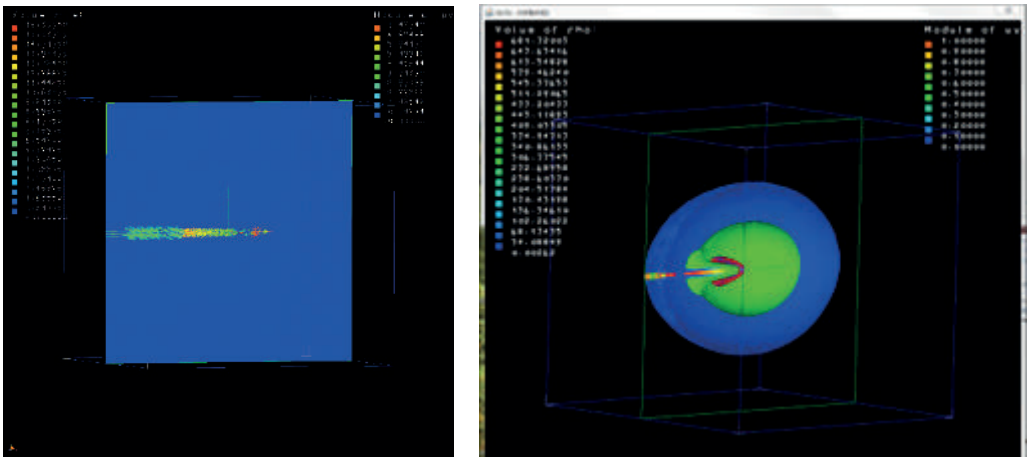


Рис. 2: Образование джета около протозвезды (слева). Взаимодействие джета, с эллиптическим ядром (справа).

Процессы формирования плотных ядер, их столкновения между собой и образование джетов моделируются с помощью схем высокого разрешения с использованием метода адаптивного измельчения сетки для трехмерных уравнений Эйлера, записанных с учетом сил гравитации и МГД [18]. Столкновения сопровождается периодическими возмущениями распределения плотности вещества в новообразованных сгустках. В таких сгустках, содержащих большое количество конденсированного вещества, плотность газа может достигать значений начального уровня дозвездных образований, порядка 10^{-19}–10^{-20} г/см^3. Проведение таких исследований необходимо для улучшения понимания физики процессов и совершенствования математических моделей образования новых звездных систем.

Возникающее движение описывается системой уравнений идеальной магнитной гидродинамики, описывающей движение сжимаемой проводящей жидкости под действием гравитационных сил и магнитных полей. Диссипативные процессы в системе уравнений в данных исследованиях не рассматриваются, предполагается, что жидкость не обладает вязкостью. Кроме того, предполагается, что магнитное поле «вморожено» в вещество, то есть не учитывается амбиполярная диффузия. Систему законов сохранения массы, количества движения и энергии в трехмерной декартовой системе координат, с учетом гравитации и идеальной магнитной гидродинамики (МГД) можно записать следующим образом:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \rho \mathbf{u} = 0,$$
$$\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u}\rho\mathbf{u} - bb + P_{\text{tot}}) = -\rho \nabla \bar{\Phi},$$
$$\frac{\partial \rho E}{\partial t} + \nabla \cdot [(\rho E + P_{\text{tot}})\mathbf{u} - b(\mathbf{u} \cdot b)] = -\rho \mathbf{u} \cdot \nabla \bar{\Phi},$$
$$\frac{\partial b}{\partial t} - \nabla \times (\mathbf{u} \times b) = 0$$

Здесь ρ — плотность, $\mathbf{u} = (u, v, w)$ — вектор скорости, $E = e + |\mathbf{u}|^2/2 + b^2/2$ — полная энергия, Давление $P = p + b^2/2$ есть сумма давления газа и магнитного давления. Уравнение состояния имеет вид: $p = \rho e(\gamma - 1)$. Правая часть уравнений сохранения импульса и энергии определяет влияние гравитационных сил. Гравитационный

потенциал определяется из уравнения Пуассона $\nabla^2\bar{\Phi} = 4\pi G\rho$. Уравнения МГД сохраняют дивергенцию магнитного поля, так что начальное условие $\nabla \cdot b = 0$ остается справедливым и согласуется с физическим наблюдением, что магнитные монополи никогда не наблюдались.

Уравнения Эйлера решаются на эйлеровой сетке с помощью адаптивного решателя Roe с использованием схем типа (TVD). Высокоточные разностные схемы гарантируют сохранение монотонности законов. В работе использована схема TVD второго порядка точности в областях гладкого решения и обладающую свойством не возрастания осцилляций на фронтах ударных волн, скачков плотности. Такая схема обеспечивает корректное описание ударных волн и предотвращает нефизические колебания. Суммарное изменение дискретного решения, определяемого как мера общей суммарной осцилляции в скорости u,

$$TV(u^t) = \sum_{i=1}^{N} |u_{i+1}^t - u_i^t|,$$

Условие не возрастания осцилляций гарантирует ограничение общего количества осцилляций. Для ограничения таких осцилляций используются различные ограничители: minmod, superbee, vanleer. Проведенные исследования показали, что для данного класса задач, в наших расчетах наилучшие результаты показал ограничитель Van Leer. Для верификации программного кода было проведено большое количество различных двух и трехмерных тестов. Проведенное тестирование показало хорошее совпадение результатов расчетов с аналитическими решениями и экспериментальными данными. Фронты ударных волн размазывались на величину 3–4 ячейки, контактные разрывы на 3–5 ячеек.

Для верификации программного кода было проведено большое количество различных двух и трехмерных тестов. Проведенное тестирование показало хорошее совпадение результатов расчетов с аналитическими решениями и экспериментальными данными [18]. Численные расчеты проводились в основном на сетках $512 \times 512 \times 512$ и $1024 \times 1024 \times 1024$. Размеры вычислительных сеток выбираются таким образом, чтобы избежать численной Джинсовской неустойчивости. Кроме того, для получения достаточного разрешения внутри молекулярных облаков, их радиус был равен, как минимум, 164 ячейкам. Количество узлов в этом случае превышает уровень пространственного разрешения, который необходим для корректного описания флуктуаций плотности, скорости и образования турбулентности в высокоэнергетических слоях газа.

3. Параллельные алгоритмы

Для предложенной программы были построены параллельные алгоритмы для CPU и GPU [17]. Распараллеливание осуществлялось с помощью технологий OpenMP, OpenACC и Coarray [16]. С помощью этих алгоритмов и программы было проведено численное моделирование астрофизических процессов в трехмерной постановке. Для решения уравнений Эйлера использовался метод адаптивного уточнения сетки (AMR). Решение уравнения Пуассона проводится на графической карте nVidia RTX 4070 Ti Super 16GB с помощью технологии OpenACC [18]. Для проверки кода было проведено тестирование, в котором было промоделировано гравитационное сжатие

и проведено сравнение с аналитическими решениями [18]. Расчеты проводились на процессоре AMD Ryzen 9 7950X3D, с 16 ядрами и 32 потоками и дополнительной кэш-памятью третьего уровня в отдельном кристалле, присоединенном к вычислительному блоку. Размер кэш-памяти уровня L3 равен 2×96 Mbytes (рис. 3). Был использован оптимизирующий драйвер 3D V-Cache, который позволяет ускорить работу при активации интенсивного вычислительного режима. В процессоре была реализована технология установки кэш-памяти в вертикальный стек. Это позволило значительно увеличить общий объем L3-кэша, который стал доступен для ядер чиплета. Модель Ryzen 9 7950X3D имеет 16 вычислительных ядер, кроме того, удалось ограничить энергопотребление. Он потребляет всего 120 Вт, что существенно меньше, чем у предыдущих моделей. Кроме того, этот процессор имеет 28 линий PCIe 5.0. Ryzen 9 7950X3D может работать только с память стандарта DDR5. Несмотря на то, что Ryzen 9 7950X3D официально работает с памятью DDR5 с частотой 5200 MHz, удается использовать память с более высокой тактовой частотой, до 6200 MHz. В наших расчетах использовалась память DDR5, размером 192 GB Kingston Fury и частотой 6000 MHz.

Рис. 3: Скриншоты программы CPU-Z с параметрами процессора Ryzen 9 7950X3D и памятью DDR5, 192 GB Kingston Fury 6000MHz, на котором были выполнены данные расчеты

На рис. 3 приведены скриншоты основных параметров процессора Ryzen 9 7950X3D. С учётом таких возможностей современных процессоров CPU и графических процессоров GPU, в программу были внесены изменения для повышения производительности и уменьшения времени реализации команд. Эта модификация позволила сгладить проблему влияния размера кэша памяти в вычислительных потоках.

4. Основные результаты и дальнейшая работа

При формировании протозвезд, внутри гравитационно нестабильных ядер, аккрецирующий газ образует вращающийся околозвездный диск и начинают формироваться джеты. При выполнении условий Джинса эти протозвездные объекты начинают коллапсировать, причем образующиеся джеты уносят до 8–10% массы протозвезды. Это согласуется с недавними открытиями крупномасштабных стримеров вокруг протозвезд [9]. В результате столкновений протоядер, которые содержат большое

количество конденсированного вещества, плотность газа существенно повышается и достигает значений, находящихся на уровне дозвездных образований [19].

Исходя из полученных результатов, сделано предположение, что в областях формирования сверхплотных ядер, сверхзвуковая турбулентность поддерживается за счет импульсов, возникающих при соударениях ядер, столкновений с ударными волнами и с образующимися джетами. Поэтому можно сделать вывод о том, что звездообразование формируется в результате сложного взаимодействием между собственной гравитацией и противодействующими факторами, такими как сверхзвуковая турбулентность, магнитные поля и тепловое давление газа. На рис. 2 приведен результат взаимодействия коллимированного джета с холодным ($T = 90$ K) молекулярным облаком, которое имеет эллиптическую форму (справа). Отметим, что формируется сильная ударная волна, которая оказывает сильное влияние на формирующиеся сверхплотные ядра. Плотность в молекулярном облаке распределена по радиусу по определенному закону. На этом рисунке слева приведена сформировавшаяся коллимированная струя, интенсивность которой меняется со временем.

За сильными ударными волнами в газовом следе возникает неустойчивость Рихтмайера–Мешкова (РМ), на внешних слоях начинает развиваться неустойчивость Кельвина–Гельмгольца (КГ) (рис. 4). Неустойчивость КГ становится основным инициатором и триггером резкого увеличения завихрённости и соответствующей вихревой турбулентности вещества молекулярных облаков. Процессы соударения сопро-

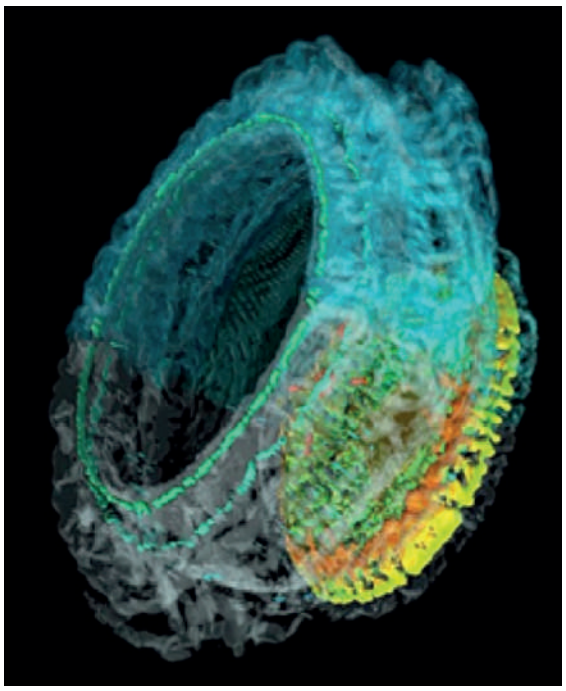


Рис. 4: Сверхзвуковая неустойчивость РМ и КГ, филаменты и сверхплотные плотные ядра, сформировавшиеся при соударении молекулярных облаков

вождаются осциллирующими неустойчивыми возмущениями в распределении плотности вещества в новообразованных плотных сгустках и филаментах. На рис. 4 приведены возникающие филаменты и плотные сгустки, которые содержат большое количество конденсированного вещества. Часть из этих сгустков со временем увеличивает свою плотность, а часть распадается в результате соударений, взаимодействия с ударными волнами и из-за воздействия джетов. Такая генерация когерент-

ных структур в неуравновешенных после столкновения новых сверхплотных образованиях определяется сверхзвуковой турбулентностью и гидродинамической неустойчивостью, которые вызывают осцилляции плотности в ударно сжатых ядрах и на границах зон соударения.

Для моделирования соударения протоядер используется собственный код [17], в котором используется метод адаптивного уточнения сетки. В этом варианте сетка верхнего уровня имеет 512^3 ячеек с пятью дополнительными уровнями уточнения. Образование сеток более высокого уровня происходит при выполнении условия Джинса. Соударения происходят по оси X, поэтому ударно сжатые слои газа развиваются в плоскости Y-Z. На рисунке ось X обозначена красным цветом. Плотность возникающих ядер на много порядков выше плотности окружающего газа. Из-за большого контраста плотности шаг по времени становится очень маленьким. Это приводит к значительному замедлению времени расчета. Введение дополнительных уровней уточнения AMR позволяет немного сгладить эту проблему.

Моделирование нецентрального соударения двух молекулярных облаков дает результаты похожие на данные, полученные при центральном соударении. Различие в том, что сформировавшийся слой имеет изогнутую форму (рис. 5), а максимальные плотности, при одинаковых скоростях соударения меньше. Дальнейшие расчеты, и

Рис. 5: Нецентральное соударение двух молекулярных облаков с разным смещениями центов. Приведено сечение плоскостью Z–Y на момент времени $t = 0.725$ (слева) и $t = 1.125$ млн. лет (справа). Рисунок справа “вырезан” для более детального изучения

учет взаимодействия плотных ядер (пиковых пятен) приводит к тому, что плотности одних ядер возрастает, а другие ядра разрушаются. Размеры этих пиковых пятен уменьшается, а их плотность значительно увеличивается. Когда требуется более высокий уровень детализации, создается частица-сток. На максимальном уровне размер ячейки равен примерно 37 астрономических единиц [13]. В работах [13, 19] предложен алгоритм поиска и создания таких сгустков плотности, что позволяет однозначно определить дискретный набор сверхплотных ядер, которые являются потенциальными местами образования частиц стоков (sink particle). Область, сжатая ударной волной между ядрами, развивается в рукава, которые соединяются с диском.

К моменту времени $t = 2.495$ миллиона лет, количество таких ядер с большой плотностью становится меньше (рис. 6), а плотность становится предзвездной, создается частица-сток, и масса частицы-стока продолжает расти за счет аккреции

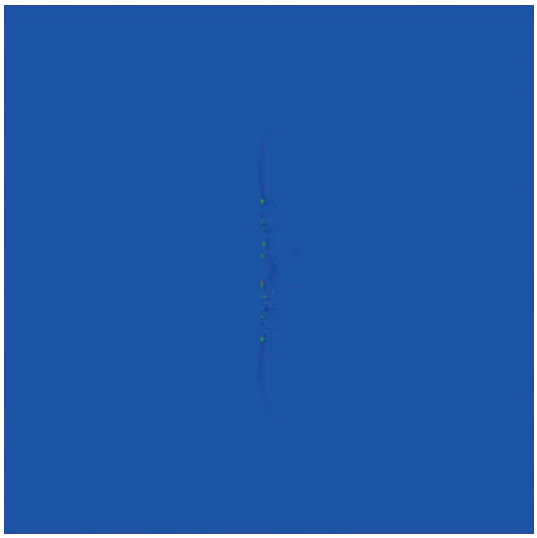


Рис. 6: Сверхплотные ядра, сформировавшиеся при соударениях на момент времени $t = 2.495$ млн лет. Яркие точки — сверхплотные ядра, которые заменяются частицами-стоками. Рисунок "вырезан" из расчетной области для более детального изучения ядер

окружающего газа [20]. Для более детального изучения изображение "вырезано" из графика. Максимальное значение плотности для сверхплотных ядер достигает значения порядка 6.75×10^{-19} г/см^3.

Моделирование астрофизических процессов, связанных с гравитационным коллапсом и формированием новых звезд связано с очень большим изменением как плотности, так и масштабов. Это, в частности, связано с тем, что относительно крупные фрагменты становятся гравитационно нестабильными и либо коллапсируют, либо распадаются на более мелкие ядра и поэтому сильно уменьшаются в размерах [21, 22]. Формирование ядер, по-видимому, происходит в три этапа: во-первых, рост гравитационной неустойчивости приводит к быстрой фрагментации ударно сжатого слоя; во-вторых, относительно небольшие фрагменты набирают массу за счёт аккреции газа и/или слияния; в-третьих, достаточно массивные фрагменты становятся восприимчивыми к гравитационной неустойчивости и, либо коллапсируют, либо фрагментируются, образуя более мелкие ядра.

Начальные размеры молекулярных облаков измеряются парсеками, а коллапсирующие объекты имеют размеры на несколько порядков меньше, и измеряются в астрономических единицах (а.е. — расстояние от Земли до Солнца). Использование методов адаптивного уточнения сетки лишь немного помогают в решении этой проблемы. Поэтому долгосрочная эволюция таких систем, которые содержат участки гравитационного коллапса, очень затруднена, и в некоторых случаях невозможна.

Как показано в работе [14], учет влияния магнитного поля не оказывает значительного влияния на процессы формирования протозвездных ядер. Численное моделирование, проведенное в этой работе, дает основание полагать, что существует некоторая зависимость структур, образующихся в процессе столкновения, от параметров магнитного поля. Часть плотных ядер, которые образуются внутри турбулентного молекулярного облака, в конечном итоге коллапсируют, что приводит к образованию новых звезд. Таким образом, определение физических критериев для ядер, которые становятся нестабильными, и анализ критических свойств ядер, представляет собой необходимый шаг в теории звездообразования.

В дальнейшем планируется построить усовершенствовать параллельный алгоритм с учетом sink particles, используя для этого Coarray Fortran 2018. Планируется включение моделирования магнитных полей в сочетании с частицами-стоками.

5. Благодарности

Работа выполнена за счет субсидии, выделенной НИЦ «Курчатовский институт» — НИИСИ на выполнение государственного задания по теме No. FNEF-2024-0002 «Математическое моделирование многомасштабных динамических процессов и системы виртуального окружения».

Список литературы

[1] Nakamura F., Zhi-Yun Li. Present-day star formation: Protostellar outflows and clustered star formation // AIP Conf. Proc. 1480, pp. 30–37 (2012). https://doi.org/10.1063/1.4754324

[2] Ward-Thompson D., Di Francesco J., Hatchell J., Hogerheijde M.R. et al. The James Clerk Maxwell Telescope Legacy Survey of Nearby Star-forming Regions in the Gould Belt. (2007), 60 p. arXiv:0707.0169 [astro-ph]. https://doi.org/10.48550/arXiv.0707.0169

[3] Ward-Thompson D., Kirk J.M., André P., Saraceno P., Didelon P., Könyves V. et al. A Herschel study of the properties of starless cores in the Polaris Flare dark cloud region using PACS and SPIRE // Astronomy and Astrophysics, 518, L92, (2010). https://doi.org/10.1051/0004-6361/201014618

[4] Schneider N., Röllig M., Simon R., Wiesemeyer H., Gusdorf A., Stutzki J., Güsten R., Bontemps S., Comerón F., Csengeri T., Adams J.D., Richter H. Anatomy of the massive star-forming region S106 // Astronomy and Astrophysics, 617, A45 (2018). https://doi.org/10.1051/0004-6361/201732508

[5] Pelkonen V.M., Padoan P., Haugbølle T., Nordlund A. Monthly Notices of the Royal Astronomical Society, Volume 504, Issue 1, pp. 1219–1236, (2021). https://doi.org/10.1093/mnras/stab844

[6] Bonnell I.A., Bate M.R. Star formation through gravitational collapse and competitive accretion // Monthly Notices of the Royal Astronomical Society, Vol. 370, Iss. 1, (2006), pp. 488–494. https://doi.org/10.1111/j.1365-2966.2006.10495.x

[7] Kuznetsova A., Bae J., Hartmann L. Mordecai-Mark Mac Low. Anisotropic Infall and Substructure Formation in Embedded Disks // The Astrophysical Journal, V. 928, N. 1 (2022). https://doi.org/10.3847/1538-4357/ac54a8

[8] Klessen R.S., Ballesteros-Paredes J. Gravoturbulent Fragmentation. (2004). https://doi.org/10.48550/arXiv.astro-ph/0402038

[9] Abe D., Inoue T., Enokiya R., Fukui Y. The Effect of Shock-wave Duration on Star Formation and the Initial Condition of Massive Cluster Formation // The Astrophysical Journal, 940:106, (2022). https://doi.org/10.3847/1538-4357/ac9e55

[10] Maity A.K., Inoue T., Fukui Y., Dewangan L.K., Sano H., Yamada R.I., Tachihara K., Bhadari N.K., Jadhav O.R. Cloud-Cloud Collision: Formation of Hub-Filament Systems and Associated Gas Kinematics Mass-collecting cone: A new signature of Cloud-Cloud Collision. (2024), arXiv:2408.06826v1 [astro-ph.GA]

[11] Moon S., Ostriker E. Prestellar Cores in Turbulent Clouds II. Properties of Critical Cores. (2024) arXiv:2411.07350v1 [astro-ph.GA]

[12] Burkert A., Alves J. The inevitable future of the starless core Barnard 68 // The Astrophysical Journal, (2009), v. 695, 1308.

[13] Yano Y., Nakamura F., Kinoshita Sh.W. Dense Core Collisions in Molecular Clouds: Formation of Streamers and Binary Stars // The Astrophysical Journal, 964:119 (12pp), (2024) https://doi.org/10.3847/1538-4357/ad2a54

[14] Kinoshita S.W., Nakamura F. MHD Simulations of Dense Core Collision // The Astrophysical Journal, 937:69 (15pp), (2022). https://doi.org/10.3847/1538-4357/ac8c95

[15] Takemura H., Nakamura F., Arce H.G., Schneider N., Ossenkopf-Okada V. CARMA-NRO Orion Survey: Unbiased Survey of Dense Cores and Core Mass Functions in Orion A // Astrophysical Journal Supplement Series, 264:35 (44pp), (2023)

[16] Rybakin B. Formation of prestellar regions in collisions of molecular clouds simulations on heterogeneous computers // Acta Astronautica, v. 204, pp 926-932, (2023)

[17] Rybakin B. Formation and propagation of jets arising from protostellar cores collapse // Lobachevskii Journal of Mathematics, Springer Nature, (2025), (In print).

[18] Rybakin B. Dynamic evolution and morphological analysis of supersonic turbulence arising during the collision of prolate and spherical clouds // Acta Astronautica, (2024), 215, 325-332.

[19] Bleuler A., Teyssier R. Towards a more realistic sink particle algorithm for the RAMSES code // Mon. Not. R. Astron. Soc. (2014), arXiv:1409.6528v1 [astro-ph.SR]

[20] Naranjo-Romero R., Vázquez-Semadeni E., Loughnane R.M. Filamentary collapse flow in molecular clouds. MNRAS, Preprint (2020). https://doi.org/10.48550/arXiv.2012.12819

[21] Anathpindika S. Formation of prestellar cores via non-isothermal fragmentation // Publications of the Astronomical Society of Australia. (2015); 32:e002. https://doi.org/10.1017/pasa.2015.1

[22] Naranjo-Romero R., Vazquez-Semadeni E., Loughnane R.M. Hierarchical gravitational fragmentation. I. Collapsing cores within collapsing clouds // The Astrophysical Journal (2015) https://doi.org/10.1088/0004-637X/814/1/48

Paralleling the Monte Carlo simulation of the THz conductivity

N.N. Reutskii[1], E.A. Nikulchin[2], V.V. Bulgakova[2], P.A. Chizhov[2], A.A. Ushakov[2], R.Y. Pishchalnikov[2]

[1]Lomonosov Moscow State University, Faculty of Physics,
[2]Prokhorov General Physics Institute of RAS

The Monte Carlo simulation is a robust and effective method for estimating the dynamics of charge carriers in nanomaterials exposed to THz radiation and for calculating the conductivity of a given sample. The main feature of this method is the need to average the behavior of a large number of particles over long time periods. This type of simulation can be significantly accelerated by modifying the executable code to allow for multithreaded calculations. We have developed a software product that enables us to calculate the complex THz conductivity, taking into account the unique molecular structure of each sample. By using the GNU Offloading and Multiprocessing Runtime Library implementation of the OpenMP parallelization API, we were able to speed up the necessary calculations while modeling the dynamics of non-interacting charge carriers within an elementary volume, and obtain desired system parameters with reasonable accuracy.

Keywords: Monte Carlo simulation, THz complex conductivity, OpenMP, parallel programming

1. Introduction

Charge transfer is a fundamental process in many scientific applications, including photovoltaic and optoelectronic devices, but also in more conventional components such as simple wires and other circuits [1, 2]. The development of nanotechnology has led to the creation of very complex materials consisting of a large number of different nanoscale elements, in which many charge transfer processes occur at various time and spatial scales [3–5]. The ability of materials to transfer charges over macroscopic distances is typically characterized by their conductivity. On a microscopic level, this conductivity decreases as the charge carriers encounter obstacles as they move from one electrode to the other. In bulk materials with extended electronic states, such as metals or doped semiconductors, charge motion can be approximately described using the Drude model [5, 6]. In this model, the ballistic movement of charges is interrupted by collisions with impurities or phonons, resulting in a finite average charge velocity under a static electric field. This means that in these simple materials, the conductivity value characterizes both the microscopic properties of the material, such as the concentration of charge carriers and their mobility, as well as the characteristics of devices built from these materials, such as their ability to transfer charge [3, 7–12].

Monte Carlo simulation of charged particle motion [13, 14] in a weakly confining environment has been used as a reliable classical model. In these calculations, charge carriers are allowed to move freely in the bulk of the material (isotropic scattering), and at the surface of the material (possibly anisotropic scattering). The scattering at the boundaries is determined by the probability of forward scattering (charge transfer to a neighboring crystal) and backscattering (charge carrier remains within the original crystal). An effective method is to calculate the velocity correlation function of the thermal movement of carriers in the absence of an applied field, and then use the Kubo formula to determine the conductivity spectrum [15–17]. The advantage of this approach is fast convergence and obtaining results in the limit of truly vanishing electric fields (strictly linear mode) even in an arbitrary potential landscape. Optionally, it is possible to simulate the full motion of the carriers in an alternating electric field, as described in other papers [1]. This approach mainly allows us to study high-field effects.; On the other hand, it is less efficient for linear response research: large probing electric fields (and/or powerful computing resources) are required to achieve an acceptable signal-to-noise ratio in the resulting conductivity. Both types of calculations allow taking into account the potential profile, as well as using Maxwell-Boltzmann or Fermi-Dirac statistics.

2. Materials and methods

2.1. Computational facilities

PC with the following processor configuration was used for simulations: Intel Core i9-14900K (Raptor Lake); the number of cores is 8P(performance)+16E(efficient); 32 threads. Motherboard is Rog Strix Z790-A; Bus is PCI-Express 5.0 (16.0 G/s); DDR5 Memory of 64 GBytes size. The operational system is Linux workstation 6.11.0-24-generic #24~24.04.1-Ubuntu. To simulate the conductivity at different frequencies, a C++ program was written using the OpenMP parallelization API, implemented in GNU Offloading and Multi-processing Runtime Library. This library made it easier to parallelize the simulation of the charge carriers dynamics in the applied field and to set different configurations at runtime. Unlike MPI, we did not need to completely rewrite the original version of the program or to do the time-consuming tests of the possible parallelization locations.

2.2. Theory of the pigment optical response

Using the Monte Carlo method, we simulated the dynamics of charged, classical, non-interacting particles that form the so-called electron gas. To simplify the calculations, we used a two-dimensional model in which each particle was described by four variables: $\dot{\mathbf{r}} = \{v_x, v_y\}$, coordinates in space and velocities $\mathbf{r} = \{x, y\}$ in the x and y directions.

At the beginning of the simulation, we initialized the particles using a Maxwell-Boltzmann distribution with an average velocity of 0 and a variance v_T for each direction. Within the framework of our model, $v_T = \sqrt{\frac{kT}{m^*}}$. For each particle, the velocity components were calculated independently. The parameters of a simple bounded plane are defined by the side length, L, and the reflective boundaries. At the start of the simulation, particles are randomly placed within the 2D frame. When a particle reaches a boundary, it can either reflect off of it or pass through. If it passes through, it will appear

on the other side of the plane, simulating the process of tunneling or transitioning to a new particle or free movement within the volume.

In accordance with Brownian motion, charged particles scatter during movement, corresponding to collisions with phonons and lattice defects. To simulate this scattering, a velocity of $1/\tau$ is introduced, where τ is the scattering time of the carrier. In our code, the probability that a particle will scatter during a certain time step is given by t/τ, where t is the size of the time step in the simulation. If an electron scatters at a certain time step, its velocity values are then re-generated using the Maxwell-Boltzmann distribution based on the x and y positions it occupied at the time of scattering. The x and y velocity components are selected separately, just as they were at the beginning of the simulation. This ensures that the particle has an equal probability of scattering in all directions within the plane.

To determine the conductivity of the sample as a function of the field frequency:

$$\ddot{\mathbf{r}}(\mathbf{r}, t) = \frac{q_e}{m^*}\mathbf{E}(\mathbf{r}, t), \tag{1}$$

where $\mathbf{E}(\mathbf{r}, t)$ is the electric field, q_e is the electron charge, m^* is the effective mass used in simulations.

In order to simplify the calculation, the effect of the external field was only considered for v_y. Therefore, the velocity of each particle was recalculated at each time step, with y being the velocity component at time step n.

$$v_y^{n+1} = v_y^n + \frac{q_e}{m^*}\mathbf{E}(\mathbf{r}, t)\Delta t, \tag{2}$$

here n is the time step, and Δt. We used $\mathbf{E}(\mathbf{r}, t) = \mathbf{E}(t) = E_0 \cos(\omega \Delta t n)$, where $\omega = 2\pi f_i$, f_i are in the range from 0.05 to 16 THz.

The conductivity according to the Kubo theory is calculated in terms of correlation functions of velocity of charge carriers as

$$\sigma(\omega) \sim \frac{1}{\mathbf{E}(\omega)} \int_0^\infty \langle v_y(0), v_y(t) \rangle e^{i\omega t}\, dt \tag{3}$$

Eventually, the operating formula looks like this

$$\sigma(\omega) = \sigma_1(\omega) + i\sigma_2(\omega) = \frac{\sum_{k=1}^{M} \langle v_y(0), v_y(t_k) \rangle [\cos(\omega t_k) + i \sin(\omega t_k)]}{\sum_{l=1}^{M} E_0 \cos(\omega t_l)[\cos(\omega t_l) + i \sin(\omega t_l)]} \tag{4}$$

3. Results and discussion

Figure 1 shows the results of modeling the dynamics of elementary charge carriers using the example of the tracks of a single particle. Plots A and C illustrate the particle's motion in the absence of scattering centers, where the particle moves freely. Plots B and D show a more realistic representation of the particle's trajectory, characterized by a scattering time $\tau = 30$ fs. The modeling was performed over 10,000 time steps for each frequency. The time step is set to 10–16 seconds. Thus, each simulation, where $f \cdot 0.05$ THz, contains at least one period of the driving frequency. Each simulation uses 80,000 particles.

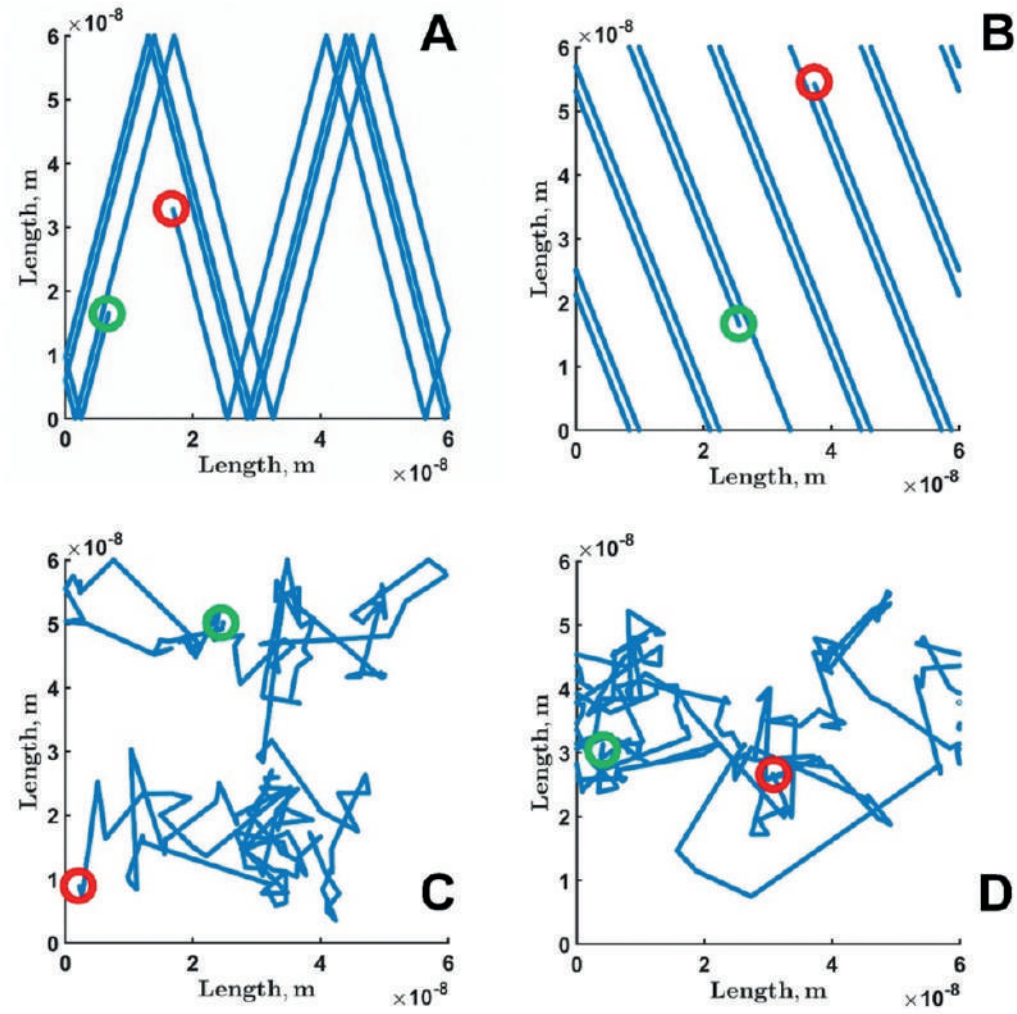


Figure 1: Simulation of the charge carries dynamics without the external THz pulse. A and B plots show the tracks of a carrier when the collisions with phonons and the scattering effects do not take into account. C and D plots show the tracks if the collisions are introduced by the effective scattering time τ. The simulations were performed for two cases: the full internal reflection (A and C) and no reflection. Red and green circles are the starting and the ending points of the modeling.

Figure 2 shows the dynamics of changing the direction and magnitude of the velocity of a moving particle. In the absence of an external exciting field, the velocity changes have a step-like character, obviously it is connected with the fact that the velocity vector changes only when the particle hits the wall. In the case when an electric field changing according to the harmonic law is applied, we see either an increase or decrease of the velocity amplitude

Figure 3 shows the results of calculations of the real and imaginary parts of the conductivity of the investigated sample. It is well seen how strongly the quality of modeling depends on the number of charge carriers and the time over which the integration was carried out. It was found that the quality of the final result depends much more strongly on the length of time integration than on the number of particles. Parallelization was implemented in the procedures for calculating particle trajectories during a free run and under the influence of an external field. The separation of simulations of individual par-

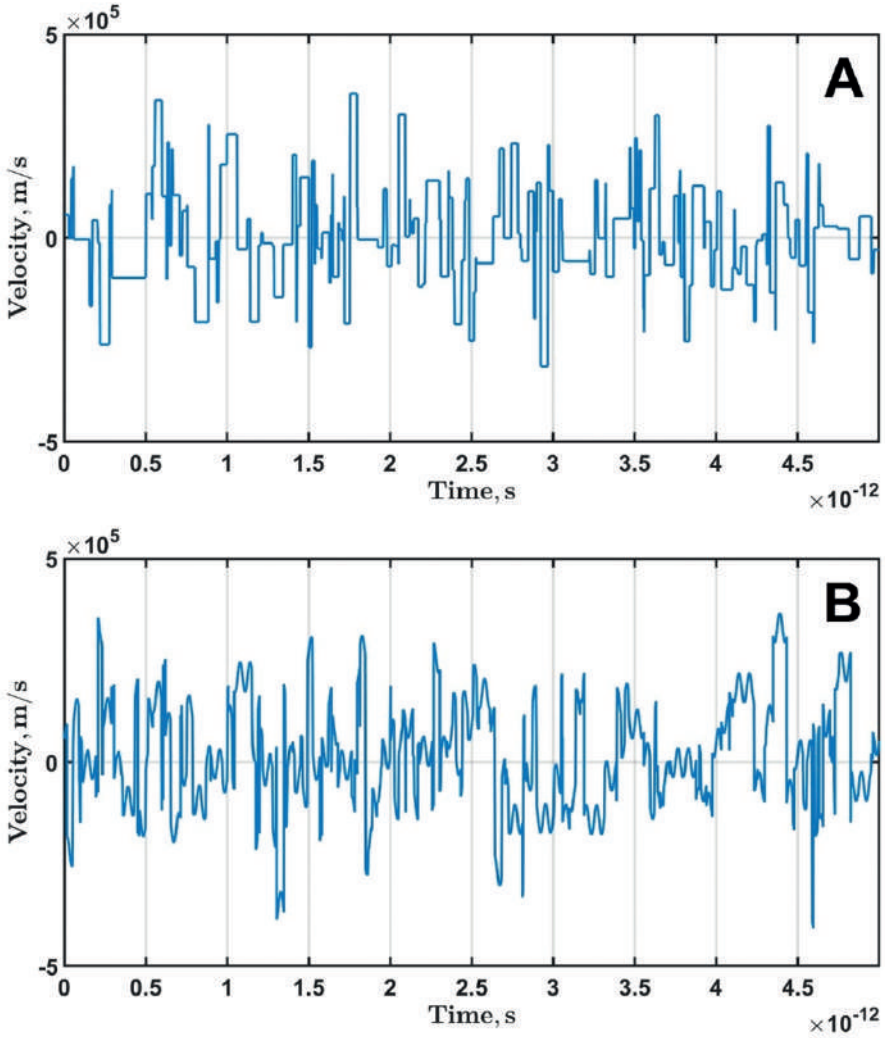


Figure 2: $v_y(t)$ evolution of a free carrier (A) and when the THz pulse with the frequency of 16 THz is applied.

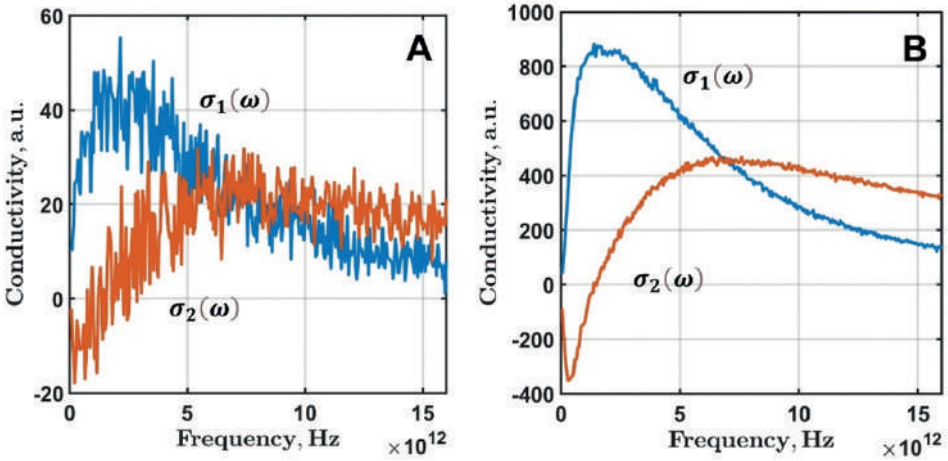


Figure 3: Monte Carlo simulation of the complex conductivity $\sigma(\omega) = \sigma_1(\omega) + i\sigma_2(\omega)$. (A) 5 charge carries, 50000 time intervals; (B) 200 charge carriers, 200000 time intervals.

ticles proved to be the most effective, as each one could be calculated independently from the others. OpenMP allows you to declare a block to be parallelized in this way:

#pragma omp parallel{}.

While subsequent loops can be declared parallel in this way:

#pragma omp for

And code blocks necessary for calculation on one core can be declared as follows:

#pragma omp single

Thus, without resorting to a complete change of the program structure and the need to manually adjust the inter-thread interaction, we have achieved the following results on speeding up the computational (simulation) part of the program and determined the optimal number of threads for our system, controlling their number at runtime with the environment variable: OMP_NUM_THREADS=4. The reported CPU load being lower than the number of threads multiplied by 100% indicates underutilization of cores and threading setup overhead, which become a significant issue as the number of cores increases, to a point where it hinders the performance so severely that any gains from using this many threads are negated (Figure 4). Considering parameters of the processor configuration (Intel Core i9-14900K: 8P(performance)+16E(efficient) cores), it can be argued that the number of performance cores is crucial to speed up the program. To verify this, it is enough to compare the running time of the program for 8 and 16 cores. The technique used does not fully leverage the "Efficient" cores of the CPU, indicating that other ways for utilizing them in parallelization should be explored. Regarding the modeling of charge carrier dynamics in THz spectroscopy, the first order electric field (Equation (2)), which drives the charge particles, allows us to simulate their dynamics independently. In this case, parallel Monte Carlo simulations can be efficiently processed on a computer cluster [18], allowing for the simulation of a large number of particles.

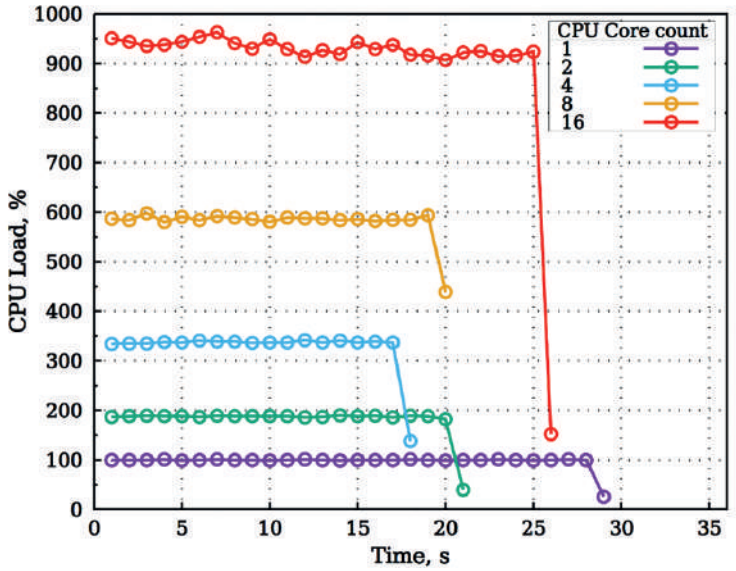


Figure 4: The result of parallelization of the simulation algorithm. CPU Load represents the percentage of total CPU time utilized by the simulation program's process over a 1 second time frame. 100% of CPU load corresponds to the unparallel version of the program. 100% of CPU Load corresponds to 100% of the CPU time of a given CPU core. Optimal performance is achieved when the program is executed on 4 threads.

4. Conclusions

As a result of the Monte Carlo simulation of charged particle motion we have shown that parallelization of the algorithm for calculating terahertz conductivity using the GNU Offloading and Multi Processing Runtime Library leads to a significant acceleration of calculations. OpenMP standard allowed for quicker transition from original sequential code to a new parallel version. Performance gains achieved are significant compared to execution time of initial approach, but several improvements are yet to be made. Considering that the problem at hand is well-suited for parallel independent calculations, it is reasonable to aim for even faster execution. As we are limited by overhead caused by the simple nature of our approach and the lack of control over the produced executable, it is worth exploring other, more sophisticated solutions, such as MPI, OpenCL compute shaders or SYCL.

5. Acknowledgement

R.Y. Pishchalnikov was supported by the Russian Science Foundation (RSF # 25-21-00375, https://rscf.ru/en/project/25-21-00375/)

References

[1] Kužel P., Němec H. Terahertz spectroscopy of nanomaterials: A close look at charge-carrier transport. Adv. Opt. Mater. 2020, 8.

[2] Lloyd-Hughes J., Jeon T.I. A review of the terahertz conductivity of bulk and nanomaterials // J. Infrared. Millim. Terahertz Waves 2012, 33, 871-925.

[3] Pushkarev V., Němec H., Paingad V.C., Maňák J., Jurka V., Novák V., Ostatnický T., Kužel P. Charge transport in single-crystalline gaas nanobars: Impact of band bending revealed by terahertz spectroscopy. Adv. Funct. Mater. 2022, 32.

[4] Quick M.T., Owschimikow N., Achtstein A.W. Terahertz charge carrier mobility in 1d and 2d semiconductor nanoparticles // J. Phys. Chem. Lett. 2021, 12, 7688-7695.

[5] Joyce H.J., Boland J.L., Davies C.L., Baig S.A., Johnston M.B. A review of the electrical properties of semiconductor nanowires: Insights gained from terahertz conductivity spectroscopy // Semicond Sci Technol 2016, 31.

[6] Smith N.V. Classical generalization of the drude formula for the optical conductivity // Phys. Rev. B Condens. Matter Mater. Phys. 2001, 64.

[7] Nsengiyumva W., Zhong S., Zheng L., Liang W., Wang B., Huang Y., Chen X., Shen Y. Sensing and nondestructive testing applications of terahertz spectroscopy and imaging systems: State-of-the-art and state-of-the-practice // IEEE Trans. Instrum. Meas. 2023, 72.

[8] Mitra S., Avazpour L., Knezevic I. Terahertz conductivity of two-dimensional materials: A review // J. Phys. Condens. Matter. 2025, 37.

[9] Wach Q., Quick M.T., Ayari S., Achtstein A.W. Field-dependent thz transport nonlinearities in semiconductor nano structures // Phys. Chem. Chem. Phys. 2024, 26, 13995-14005.

[10] Quick M.T., Ayari S., Owschimikow N., Jaziri S., Achtstein A.W. Quantum nature of thz conductivity: Excitons, charges, and trions in 2d semiconductor nanoplatelets and implications for thz imaging and solar hydrogen generation // ACS Appl. Nano Mat. 2022, 5, 8306-8313.

[11] Xu X., Fu Q., Gu H., Guo Y., Zhou H., Zhang J., Pan D., Wu S., Dong M., Guo Z. Polyaniline crystalline nanostructures dependent negative permittivity metamaterials // Polymer 2020, 188.

[12] Ostatnický T., Pushkarev V., Němec H., KuŽel P. Quantum theory of terahertz conductivity of semiconductor nanostructures // Phys. Rev. B 2018, 97.

[13] Cocker T.L., Baillie D., Buruma M., Titova L.V., Sydora R.D., Marsiglio F., Hegmann F.A. Microscopic origin of the drude-smith model // Phys. Rev. B 2017, 96.

[14] Johnston M.B., Whittaker D.M., Corchia A., Davies A.G., Linfield E.H. Simulation of terahertz generation at semiconductor surfaces // Phys. Rev. B Condens. Matter Mater. Phys. 2002, 65, 1-8.

[15] Quick M.T., Achtstein A.W. Challenging kubo-greenwood theory: Nonunitarian dynamics reshapes thz conductivity and transport in nanosystems // J. Phys. Chem. C 2023, 127, 24591-24597.

[16] Kužel P., Kadlec F., Němec H. Propagation of terahertz pulses in photoexcited media: Analytical theory for layered systems // J. Chem. Phys. 2007, 127.

[17] Kubo R. Generalized cumulant expansion method // Journal of the Physical Society of Japan 1962, 17, 1100-1120.

[18] Jungemann C., Meng F., Thomson M.D., Roskos H.G. Massively parallel FDTD full-band Monte Carlo simulations of electromagnetic THz pulses in p-doped silicon at cryogenic temperatures, Solid-State Electronics, 2022, 197, 108439.

Агентный подход в параллельном алгоритме для получения негильотинного размещения на основе точной модели оптимизации и декомпозиции набора деталей[1]

А.А. Андрианова, Т.М. Мухтарова[1]

[1]Казанский (Приволжский) федеральный университет

В работе предлагается параллельный алгоритм на основе синхронизации работы нескольких независимых агентов, умеющих решать отдельные вспомогательные задачи для получения негильотинного размещения набора деталей на полуполосе с помощью точной оптимизационной модели. Для постановки задач для агентов используется прием декомпозиции набора деталей.

Ключевые слова: негильотинное размещение набора прямоугольников, задача частично булевого линейного программирования, параллельный алгоритм с декомпозицией.

1. Введение

Задачи размещения набора деталей на некотором материале являются очень разнообразными по постановкам и методам решения и имеют важное экономическое значение. В зависимости от условий, которые накладываются на исходные данные задачи, в этом классе можно выделить задачи упаковки на лист заданного размера, размещения на полуполосе (ограничение ставится только по одному размеру), задачи с фиксированной ориентацией прямоугольников или с возможностью их поворота, с гильотинным размещением набора прямоугольников, требующих проводить разрезы материала от края до края, и с негильотинным размещением, определяющим более сложную конфигурацию разрезов. Достаточно подробный обзор современных постановок задач размещения представлен в [1].

Большинство задач размещения относятся к NP-трудным задачам. Поэтому, несмотря на то что исследования методов решения задач размещения проводятся уже давно, основное внимание уделено эвристическим алгоритмам их решения [2–7]. Среди них выделяются метаэвристические алгоритмы, базирующиеся не столько на свойствах самой решаемой задачи, сколько на общих характеристиках задач оптимизации. Наиболее популярны метаэвристические подходы в построении алгоритмов, которые основаны на применении методов локального поиска [4–6] и генетических алгоритмов [7]. Точным моделям уделено сильно меньшее внимание в связи с их сложностью и плохой применимостью для решения практических задач.

[1]Работа выполнена за счет средств Программы стратегического академического лидерства Казанского (Приволжского) федерального университета (“ПРИОРИТЕТ-2030”)

Несколько точных моделей постановок задач размещения можно найти в [8]. Такие модели характеризуются сложной системой ограничений, включая разреженные матрицы ограничений, большое количество переменных, включающих иногда трехуровневую индексацию, использование альтернативных ограничений, нелинейность и недифференцируемость целевой функции. В данной работе внимание уделено одной из точных моделей, в которой задача негильотинного размещения набора прямоугольных деталей на полуполосе представляется задачей частично булевого линейного программирования большой размерности [9]. Существуют точные методы решения такой задачи, однако с увеличением количества деталей вычислительная сложность ее решения растет очень быстро, так как они базируются на идее метода ветвей и границ, обеспечивающей последовательную фиксацию переменных модели и решение, таким образом, более простых задач того же вида. В [10] рассматривались эвристические подходы к уменьшению вычислительной сложности решения данной задачи, использующие генетические алгоритмы и алгоритмы локального поиска в рамках процедуры сокращения количества переменных задачи за счет одновременной фиксации значений серии переменных. В [11] был рассмотрен алгоритм, базирующийся на применении подходов к распараллеливанию алгоритма метода ветвей и границ, основанный на возможности обработки отдельных ветвей виртуального дерева решений по отдельности.

В данной работе предлагается использовать декомпозицию набора деталей как основной инструмент получения задач размещения меньшей сложности, решения которых могут комбинироваться различным образом, в том числе на основе постановки точной оптимизационной модели для композитных фрагментов, состоящих из уже уложенных исходных деталей.

Такой подход обеспечивает другие возможности для распараллеливания вычислительного процесса. Прием декомпозиции обеспечивает создание большого количества независимых задач, которые могут решаться достаточно быстро независимыми агентами — процессами, которые могут быть реализованы как отдельные потоки в моделях параллельной обработки с общей памятью, так и как независимые процессы, работающие в том числе на вычислительном кластере. Для синхронизации работы требуется некоторый ведущий процесс, своего рода посредник, который обеспечивает декомпозицию и постановку отдельных вспомогательных задач для решения агентами, сохранение результатов решения этих задач и компоновку из этих решений итогового результата. Такая схема работы имеет большую гибкость, позволяет комбинировать разные алгоритмы решения вспомогательных задач — так, отдельные агенты могут выполнять специализированные операции, другие могут использовать несколько различных алгоритмов для решения своих задач. Также можно экспериментировать с алгоритмами работы ведущего процесса.

2. Точная модель компактного размещения набора прямоугольников на полуполосе

В [9] была представлена следующая точная модель размещения набора прямоугольников на полуполосе заданной ширины.

Пусть даны набор n прямоугольников с размерами $a_i \times b_i$ $(a_i \geq b_i)$, $i = 1 \dots n$, и полуполоса с шириной B $(B \geq \max_{i=1\dots n} b_i)$. Требуется определить минимальную

длину листа, необходимого и достаточного для размещения данного набора прямоугольников, и план данного размещения, т.е. координаты всех прямоугольников при данном размещении.

2.1. Модель частично булевого линейного программирования

Пусть полуполоса размещается на координатной плоскости таким образом, что левый нижний угол полуполосы имеет координаты $(0, 0)$. Будем рассматривать варианты размещения прямоугольника с ориентацией либо вдоль листа (сторона прямоугольника длиной a_i расположена параллельно оси OX), либо с ориентацией поперек листа (сторона прямоугольника длиной a_i расположена параллельно оси OY).

Обозначим через (x_i, y_i) координаты левого нижнего угла прямоугольника с номером i, а через z_i — ее ориентацию на полуполосе — $z_i = 0$, если i-ый прямоугольник ориентирован вдоль листа, $z_i = 1$ — в противном случае.

Для формирования условий попарного непересечения набора прямоугольников вводятся дополнительные булевые переменные для каждой пары прямоугольников:

1) s_{ij} — переменная, определяющая предшествование i-ого прямоугольника j-ому ($s_{ij} = 0$) или предшествование j-ого прямоугольника i-ому ($s_{ij} = 1$);

2) t_{ij} — переменная, определяющая по горизонтали ($t_{ij} = 0$) или по вертикали ($t_{ij} = 1$) производится расхождение местоположения прямоугольников.

В силу прямой связи значений переменных при смене порядка индексов ($s_{ij} = 1 - s_{ji}$, $t_{ij} = t_{ji}$) при рассмотрении пар прямоугольников рассматриваются только индексы, удовлетворяющие условиям $i = 1 \dots n-1$, $j = i+1 \dots n$.

Задача линейного частично булевого программирования для размещения набора прямоугольников на полуполосе будет иметь следующий вид:

$$A \to \min$$

$$x_i + (b_i - a_i)z_i \le A - a_i, \quad i = 1 \dots n,$$

$$y_i + (a_i - b_i)z_i \le B - b_i, \quad i = 1 \dots n,$$

$$-x_i + x_j - (b_i - a_i)z_i + \bar{A}t_{ij} + \bar{A}s_{ij} \ge a_i, \quad i = 1 \dots n-1,\ j = i+1 \dots n,$$

$$x_i - x_j - (b_j - a_j)z_j + \bar{A}t_{ij} - \bar{A}s_{ij} \ge a_j - \bar{A}, \quad i = 1 \dots n-1,\ j = i+1 \dots n,$$

$$-y_i + y_j - (a_i - b_i)z_i - \bar{B}t_{ij} + \bar{B}s_{ij} \ge b_i - \bar{B}, \quad i = 1 \dots n-1,\ j = i+1 \dots n,$$

$$y_i - y_j - (a_j - b_j)z_j - \bar{B}t_{ij} - \bar{B}s_{ij} \ge b_i - 2\bar{B}, \quad i = 1 \dots n-1,\ j = i+1 \dots n,$$

$$x_i \ge 0, \quad i = 1 \dots n,$$

$$y_i \ge 0, \quad i = 1 \dots n,$$

$$z_i \in \{0, 1\}, \quad i = 1 \dots n,$$

$$s_{ij} \in \{0, 1\}, \quad i = 1 \dots n-1,\ j = i+1 \dots n,$$

$$t_{ij} \in \{0, 1\}, \quad i = 1 \dots n-1,\ j = i+1 \dots n.$$

В модели константы выбраны так, чтобы $\bar{B} \ge B$, $\bar{A}$ — величина, не меньшая оптимальной длины полуполосы, достаточной для размещения заданного набора прямоугольников, например, можно использовать для расчетов $\bar{B} = B$, $\bar{A} = \sum_{i=1}^{n} a_i$.

2.2. Особенности вычислений для задачи частично булевого линейного программирования

Оценим вычислительные характеристики представленной задачи частично булевого линейного программирования. Количество переменных в модели равно $n^2 + 2n + 1$, где n^2 переменных являются булевыми. Количество ограничений при этом также зависит от количества деталей n и вычисляется как $3n^2 + 2n$. Так, при $n = 5$ число переменных будет равно 36, 25 из которых будут являться булевыми. При этом количество ограничений задачи будет равно 86, из которых 25 ограничений будут условиями булевости переменных, 20 ограничений будут определять условия допустимости размещения деталей на результирующем листе, 40 ограничений будут определять условия попарного непересечения деталей друг с другом. При $n = 10$ число переменных уже будет равно 121, а ограничений — 320. Таким образом, видно, как сильно растет размер задачи частично булевого линейного программирования при увеличении размера набора деталей.

Универсальным методом решения задачи частично булевого линейного программирования является метод ветвей и границ Лэнд и Дойг. Алгоритм строит виртуальное дерево решений, где в каждом узле решается ослабленная задача линейного программирования, в которой ограничения булевости заменены либо на ограничение фиксации отдельных переменных в одно из допустимых значений — 0 или 1, либо на ослабленное условие включения переменной в интервал $[0, 1]$. Каждое ветвление узла виртуального дерева представляет фиксацию выбранной переменной в два значения, что разбивает все множество допустимых решений на два непересекающихся подмножества, которые могут быть обработаны отдельно друг от друга.

При использовании метода ветвей и границ в задаче размещения набора деталей на полуполосе размер виртуального дерева будет расти очень быстро вслед за увеличением количества деталей n. Так, при $n = 5$ глубина виртуального дерева решений составит 25, а общее количество узлов дерева составило бы 16 777 215, что потребовало бы решить такое же количество ослабленных задач линейного программирования. Как и любой метод ветвей и границ, метод Лэнд и Дойг имеет механизм отсечения неперспективных для получения оптимума ветвей путем сравнения нижней оценки возможного значения целевой функции в поддереве с полученным на момент сравнения лучшим решением. Поэтому реальные вычислительные характеристики процесса решения задач с $n = 5$ варьируется — количество решенных ослабленных задач линейного программирования по задачам эксперимента изменялось от 856 до 7974, время решения — от 1,5 до 80 секунд. Данное время можно считать приемлемым.

Уже при количестве деталей $n = 6$ количество решенных ослабленных задач возрастает до более 700000 по верхней границе, а время решения увеличивается примерно в 5 раз. При $n = 10$ эксперимент содержал в том числе задачи, для которых не находилось решение и за 20 часов. В этом случае приходилось останавливать процесс и принимать за приближенный оптимум наилучшее полученное решение.

Таким образом, приемлемым для решения с помощью точной оптимизационной модели в виде задачи частично булевого линейного программирования можно считать только задачи с размером набора прямоугольников $n \leq 5$.

3. Алгоритм построения негильотинного размещения набора деталей на полуполосе на основе декомпозиции набора деталей

3.1. Применение декомпозиции набора деталей

Применение декомпозиции набора деталей обеспечивает многоуровневую рекурсивную процедуру, в результате которой исходный набор деталей разбивается на поднаборы с размером, приемлемом для решения задачи частично булевого линейного программирования. Полученные размещением поднаборов фрагменты материала становятся новыми композитными деталями, размещение которых возможно также получить путем решения оптимизационной задачи того же типа. Если композитных деталей окажется много, возможно провести следующий этап декомпозиции. Таким образом, ожидается серьезное уменьшение вычислительной сложности, так как несмотря на то, что ослабленных задач линейного программирования ожидается много, они будут иметь небольшой размер, тем самым, можно будет их решить за незначительное время.

Для определения, требуется ли декомпозиция набора деталей, определяется значение m — максимальное количество деталей в поднаборе для решения задачи частично булевого линейного программирования. Оно определяется исходя из имеющихся возможностей вычислительной системы, на которой производится решение задачи.

Для всех задач с размером набора деталей $n \leq m$ формируется база знаний о размещениях различных поднаборов. База знаний $F(key)$ формируется в виде отображения бинарного кода key номера поднабора деталей k (бинарный код которого key кодирует включение деталей в поднабор (бит с номером i, равный 1, означает, что деталь с номером i включена в поднабор)), в кортеж с информацией об оптимальном размещении данного поднабора $\{B, A, Loss, Count, X\}$, где $Loss$ — площадь отходов материала при применении размещения, A и B — размеры фрагмента полуполосы, достаточной для размещения поднабора, ширина выбирается на основании минимизации отходов материала, $Count$ — количество деталей в поднаборе, X — массив троек (x, y, z) размещения деталей текущего поднабора.

Следующий шаг алгоритма применения декомпозиции заключается в выборе разбиения набора D на поднаборы. Назовем покрывающим разбиением такое разделение на части $\{S_i\}$, чтобы $S_i \cap S_j = \emptyset \ \forall i, j,\ i \neq j,\ \bigcup S_i = D$. Разбиение осуществляется на основе анализа бинарных кодов key, по которым информация по уже полученным размещениям хранится в базе знаний. При наличии в разбиении нескольких подмножеств $S = \{S_i\}$, код key для следующего подмножества должен быть выбран таким образом, чтобы $\exists F(key),\ D_{key} \cap S_i = \emptyset \ \forall S_i \in S$, где D_{key} — множество индексов бит, равных 1. Отсутствие таких key, означает, что разбиение является покрывающим.

Каждое выбранное покрывающее разбиение обеспечивает представление нового набора деталей, состоящего из укрупненных деталей, включивших в себя уже некоторый набор исходных деталей. Данный набор деталей можно разместить также с помощью решения задачи частично булевого линейного программирования, если количество укрупненных деталей в нем будет приемлемым для решения ($\leq m$). Иначе можно провести рекурсивно декомпозицию уже набора укрупненных деталей.

При получении размещения укрупненных деталей (обозначим этот набор как набор троек (dx_i, dy_i, dz_i)) возможно получение координат размещения для всех исходных деталей, попавших в это укрупнение. Координаты пересчитываются по следующим правилам.

Пусть имеется покрывающее разбиение $S = \{S_i\}$. Для каждого $S_j \in S$ формируем код key_j.

Если для укрупненной детали, соответствующей S_j, $F(key_j).A > F(key_j).B$ (укрупненная деталь не поворачивалась) и $dz_j = 0$, для $i \in S_j$ характеристики размещения задаются кортежем $(F(key_j).X[i].x + dx_j,\ F(key_j).X[i].y + dy_j,\ F(key_j).X[i].z)$.

Если для укрупненной детали, соответствующей S_j, $F(key_j).A > F(key_j).B$ (укрупненная деталь не поворачивалась) и $dz_j = 1$, для $i \in S_j$ характеристики размещения задаются кортежем $(F(key_j).X[i].y + dx_j,\ F(key_j).X[i].x + dy_j,\ 1 - F(key_j).X[i].z)$.

Если для укрупненной детали, соответствующей S_j, $F(key_j).A \leq F(key_j).B$ (укрупненная деталь поворачивалась) и $dz_j = 0$, для $i \in S_j$ характеристики размещения задаются кортежем $(F(key_j).X[i].y + dx_j,\ F(key_j).X[i].x + dy_j,\ 1 - F(key_j).X[i].z)$.

Если для укрупненной детали, соответствующей S_j, $F(key_j).A \leq F(key_j).B$ (укрупненная деталь поворачивалась) и $dz_j = 1$, для $i \in S_j$ характеристики размещения задаются кортежем $(F(key_j).X[i].x + dx_j,\ F(key_j).X[i].y + dy_j,\ F(key_j).X[i].z)$.

Таким образом, зная размещение укрупненных деталей, однозначно определяется размещение исходных деталей набора.

Общая концепция применения декомпозиции имеет элементы эвристического алгоритма, так как зависит от выбора покрывающих разделений набора деталей. Так, выбор кода поднабора для включения в покрывающее разбиение можно осуществить, выбирая поднаборы с максимальным количеством деталей и минимальным размером отходов, однако реализация такого выбора может также быть трудоемким процессом. Можно ограничиться выполнением нескольких проб для генерации ключа и выбором того поднабора, которому соответствуют наилучшие значение двух выбранных для отбора характеристик.

Вычислительная процедура декомпозиции набора деталей может рассматриваться как принципиальная схема, в рамках которой могут быть использованы различные варианты выбора рассматриваемых покрывающих разбиений. Простым вариантом является выбор набора разбиений методом случайного поиска — создается определенное количество случайных покрывающих разбиений и выбирается из них лучшее по полученной итоговой длине полуполосы. Другим вариантом может быть применение алгоритма, базирующегося на использовании алгоритмов случайного поиска или генетических алгоритмов для выбора других вариантов разбиений. Таким образом, в целом, процедура является комбинированной, подразумевающей использование на различных шагах алгоритмов различной природы, включая параллельные алгоритмы.

3.2. Параллельный алгоритм на основе применения декомпозиции набора деталей

Алгоритм на основе применения декомпозиции набора деталей разбивает весь вычислительный процесс на решение нескольких вспомогательных задач, которые могут быть решены независимо друг от друга с помощью различных алгоритмов.

Для простоты определим в качестве алгоритма поведения независимого агента алгоритм метода ветвей и границ Лэнд и Дойг решения задачи частично булевого линейного программирования. Независимый агент получает в качестве исходных данных набор из n деталей с заданными размерами и пробную ширину полуполосы и возвращает кортеж с информацией об оптимальном размещении данного поднабора $\{B, A, Loss, Count, X\}$, где $Count = n$, $Loss = AB - \sum a_i b_i$, $X = \{(x_i, y_i, z_i)\}$ — значения из оптимального решения задачи частично булевого линейного программирования.

Для синхронизации работы агентов и распределения задач должен быть выделен один управляющий центр. В управляющем центре ведется декомпозиция наборов деталей и определение постановок задач частично булевого линейного программирования, ведется очередь свободных независимых агентов, формируется задача для агента и интерпретируется решение, полученное от него. Сформулируем данный алгоритм.

Алгоритм работы управляющего центра. На вход алгоритму подается набор деталей $D = \{1, 2 \dots n\}$ и их размеры $\{a_i \times b_i\}$ $i \in D$. Пусть Q — очередь свободных агентов, на начальном этапе в ней находятся параметры агентов (потоков, процессов и т.д.).

1. Если $n > m$, требуется декомпозиция набора. Пусть выбрано значение m — максимальное количество деталей в поднаборе для решения задачи частично булевого линейного программирования.

1.1. Формирование базы знаний о размещениях различных поднаборов.

1.1.1. Генерация примитивных поднаборов, т.е. поднаборов, содержащих одну деталь. Для каждого $i = 1 \dots n$ выполняется процедура:

1.1.1.1. Генерация кода $key = (0, 0, \dots, 1, \dots, 0)$, где $key_i = 1$.

1.1.1.2. Если $a_i < B$ (деталь может быть размещена поперек листа), добавить в базу знаний кортеж $F(key) = \{a_i, b_i, 0, 1, (0, 0, 1)\}$, в противном случае — $F(key) = \{b_i, a_i, 0, 1, (0, 0, 0)\}$.

1.1.2. Генерация композитных поднаборов, т.е. поднаборов, содержащих более одной детали. Для каждого номера поднабора i (изменяется от 1 до 2^n):

1.1.2.1. Вычисляем key как бинарный код номера i. Вычисляем $Count$ — количество бит в коде key, равных 1.

1.1.2.2. Если $Count = 1$ или $Count > m$, поднабор не требует включения в базу знаний, так как либо является примитивным, либо содержит количество деталей, более допустимого. Переход к п. 1.2.1 при $i = i + 1$.

1.1.2.3. Формируем поднабор деталей $D_{key} = \{i_{key}\}$ — множество номеров деталей, соответствующих значениям 1 в битах кода key.

1.1.2.4. Для любого значения ширины полуполосы $b \in [\max\{b_i,\ i \in D_{key}\}, B]$ ставится задача частично булевого линейного программирования и передается свободному агенту из очереди Q. Исходные данные для задачи определяются параметрами $(b,\ Count,\ a_i \times b_i,\ i \in D_{key})$.

1.1.2.5. При получении ответа от агента в виде кортежа с информацией о размещении $\{key, B, A, Loss, Count, X\}$ в базе знаний происходит поиск кортежа по ключу key, соответствующему другому размеру ширины полуполосы. Если такая запись существует и размеры отходов в ней указаны большие, чем получены как результат работы агента, то делаем замену записи кортежа по ключу key.

1.2. Получаем набор из k покрывающих разбиений $S = \{S_i\}$ на поднаборы набора D. Для каждого покрывающего разбиения:

1.2.1. Выбрать произвольно код key такой, что $\exists F(key)$, $D_{key} \cap S_i = \emptyset$ $\forall S_i \in S$, где D_{key} — множество индексов бит, равных 1. Если выбрать такой код невозможно, переход к п. 1.3.

1.2.2. Выбранный поднабор включается в покрывающее разбиение $S = S \cup D_{key}$. Переход к п. 1.2.1.

Для каждого полученного покрывающего разбиения S выполняем следующие шаги, выбирая то размещение, которое соответствует наилучшему количеству величины отхода материала.

1.3. Для каждого подмножества $S_i \in S$ определим код key_i и укрупненную деталь по размерам листа, достаточного для размещения поднабора $S_i - a_i = \max\{F(key_i).A, F(key_i).B\}$, $b_i = \min\{F(key_i).A, F(key_i).B\}$.

1.4. Создаем задачу для свободного агента на основании набора укрупненных деталей, построенных на шаге 1.3. При получении ее решения независимым агентом как результат получим набор кортежей (dx_i, dy_i, dz_i) соответственно для координат левого нижнего угла и ориентации укрупненных деталей, соответствующих разбиению S.

1.5. Определение размещения исходных деталей $i \in D$. Для каждого $S_j \in S$ формируем код key_j. Обозначим $k = |S_j|$ — мощность множества S_j (количество деталей в поднаборе).

1.5.1. Если для укрупненной детали, соответствующей S_j, $F(key_j).A > F(key_j).B$ (укрупненная деталь не поворачивалась) и $dz_j = 0$, для $i \in S_j$ характеристики размещения задаются кортежем $(F(key_j).X[i].x + dx_j,\ F(key_j).X[i].y + dy_j,\ F(key_j).X[i].z)$.

1.5.2. Если для укрупненной детали, соответствующей S_j, $F(key_j).A > F(key_j).B$ (укрупненная деталь не поворачивалась) и $dz_j = 1$, для $i \in S_j$ характеристики размещения задаются кортежем $(F(key_j).X[i].y + dx_j,\ F(key_j).X[i].x + dy_j,\ 1 - F(key_j).X[i].z)$.

1.5.3. Если для укрупненной детали, соответствующей S_j, $F(key_j).A \leq F(key_j).B$ (укрупненная деталь поворачивалась) и $dz_j = 0$, для $i \in S_j$ характеристики размещения задаются кортежем $(F(key_j).X[i].y + dx_j,\ F(key_j).X[i].x + dy_j,\ 1 - F(key_j).X[i].z)$.

1.5.4. Если для укрупненной детали, соответствующей S_j, $F(key_j).A \leq F(key_j).B$ (укрупненная деталь поворачивалась) и $dz_j = 1$, для $i \in S_j$ характеристики размещения задаются кортежем $(F(key_j).X[i].x + dx_j,\ F(key_j).X[i].y + dy_j,\ F(key_j).X[i].z)$.

2. Если $n \leq m$, декомпозиция не требуется:

2.1. Формируем для текущего множества деталей задачу частично булевого линейного программирования и передаем ее для решения свободному агенту.

2.2. Возвращаем характеристики решения — для всех деталей i формируем кортеж (x_i, y_i, z_i) и массив кортежей возвращается как результат работы алгоритма.

В алгоритме работы управляющего центра выделяется несколько крупных этапов. Этап под номером 1 определяет действия в ситуации, когда требуется декомпозиция набора деталей. Этап под номером 2 представляет собой простой случай, который можно решить без применения декомпозиции, своего рода условие выхода из рекурсии. На этапе 1 вычислительный процесс представляется последовательностью этапов формирования базы знаний о поднаборах размера $n \leq m$ (п. 1.1), включая примитивные поднаборы, состоящие из одной детали (п. 1.1.1) и композитные поднаборы (п. 1.1.2), получения покрывающих разбиений на поднаборы (п. 1.2), постановку и решение задачи частично булевого линейного программирования для получения размещения укрупненных деталей (пп. 1.3–1.4), определение координат деталей поднабора, из которых формируется укрупненная деталь, в итоговом размещении (п. 1.5).

Данный алгоритм работы управляющего центра подразумевает только один тип задач, которые может решать независимый агент — задачу частично булевого линейного программирования. Тем не менее, существуют в схеме алгоритма еще несколько этапов, которые возможно выполнять параллельно для различных данных. Так, можно предоставить независимым агентам выполнять задачу генерации покрывающего разбиения набора деталей в п. 1.2, обеспечивая доступ к базе знаний как к общей памяти для чтения. Аналогично, в п. 1.5 производится получение координат исходных деталей, находящихся внутри укрупненной детали. Для каждой укрупненной детали этим вычислением может заняться отдельный независимый агент.

Таким образом, предлагаемый алгоритм имеет достаточно большие перспективы применения механизмов параллельных вычислений на различных этапах алгоритма, а также возможность комбинировать разные подходы к выполнению этих этапов, что, в целом, может показать высокую эффективность применяемой эвристики.

4. Анализ экспериментов параллельного алгоритма на основе применения декомпозиции набора деталей

Эксперименты по решению задачи негильотинного размещения набора прямоугольных деталей на полуполосе проводились на наборе сгенерированных модельных задач (общее количество более 100). Основными критериями сравнения различных вариантов алгоритмов были общая вычислительная сложность решения в форме количества ослабленных задач линейного программирования, решенных в узлах виртуального дерева при решении задач частично булевого линейного программирования, а также время, затраченное на получение итогового размещения.

Для эксперимента была написана программа на языке программирования Python на основе многопоточной архитектуры. Для выполнения были использованы простые вычислительные средства — облачная среда выполнения программ Google Colaboratory, а также ноутбук с процессором Intel(R) Core(TM) i5-10210U CPU 1.60 GHz (количество логических ядер — 8), 8 ГБ оперативной памяти.

Проводилось несколько серий экспериментов, которые позволили сделать выводы о применимости алгоритма декомпозиции, а также об эффективности применения приемов параллельной обработки на этапе решения задачи частично булевого линейного программирования.

Эксперимент 1. В данном эксперименте оценивалось время решения задачи в зависимости от размера m допустимых поднаборов для построения базы знаний. Для серии задач с $n = 10$ и $m = 3$ число решенных ослабленных задач линейного программирования в среднем составило 42769 с временными затратами в среднем в 36 секунд. С увеличением числа деталей в допустимом подмножестве до $m = 4$ число решенных ослабленных задач линейного программирования достигла 520928 задач и потребовало примерно 10 минут. При $m = 5$ в среднем было решено около 10000000 задач за 2,5 часа.

По результатам эксперимента можно сделать вывод, что средние временные затраты на решение одной задачи линейного программирования практически не варьируются для разных значений m, определяющих количество деталей в допустимом

для расчетов поднаборе. В целом, на решение одной задачи линейного программирования уходило максимум 1,1 мс. Поэтому существенное увеличение количества решаемых задач такого размера приводит к преимущественно линейному увеличению времени работы программы.

Таким образом, построение базы знаний в п. 1.1 по сравнению с решением исходной задачи методом Лэнд и Дойг затрачивает значительно меньшее время. Поскольку задачи для разных поднаборов данных являются независимыми, применение параллельной версии алгоритма сократило время работы еще существеннее — уменьшение времени работы по серии задач эксперимента по сравнению с методом Лэнд и Дойг составило более 80%, что дало возможность получить приближенное решение хорошего качества за примерное время 30–40 минут.

Также оценивалась эффективность полученного таким образом размещения. При $n = 10$ по всем решенным задачам коэффициент полезного использования площади листа (доля площади, покрытая размещенными деталями) для всех тестовых задач превысил 70%, что можно считать приемлемым по эффективности размещением. В то же время наблюдалось тенденция, что при $m = 3$ программа работала дольше из-за большего размера построенной базы знаний, но при этом были получены лучшие показатели полезного использования площади (минимизация отходов материала). Причиной такого факта является наличие плотных фрагментов незначительного размера, которые оказалось легче совместить на более высоких уровнях размещения при декомпозиции.

Эксперимент 2. В этом эксперименте исследовалась зависимость вычислительных характеристик от количества деталей в исходном наборе. Увеличение количества деталей n при фиксированном допустимом количестве в поднаборе $m = 3$ привело к увеличению количества решенных задач линейного программирования при построении базы знаний для $n = 10$, $n = 15$, $n = 20$ на 42769, 194085, 547068 соответственно, а временные характеристик — 36 секунд, 3.5 минуты, 7.7 минут в параллельном варианте. Здесь наблюдается линейное увеличение времени, что, в целом, позволит решать задачи большой размерности за приемлемое время. Обратим внимание, что непосредственное решение задачи методов Лэнд и Дойг приводит к 20 часам работы при $n = 10$.

В целом, замечено, что порядка 95% времени решения в последовательном варианте относится к этапу формирования базы знаний. Поэтому применение параллельной генерации записей базы знаний и позволило так значительно улучшить временные показатели решения задачи.

Пока не очень высокие показатели полезного использования площади (более чем 70%) можно компенсировать тем, что достаточно быстро обеспечивается получение допустимого размещения, которое можно использовать как исходное размещение для алгоритмов улучшения известных решений. Причем наличие в этом допустимом решении плотных фрагментов за счет использования точной модели позволяет выполнить алгоритм улучшения более эффективно по времени.

5. Заключение

Таким образом, для решения задачи получения негильотинного размещения набора прямоугольных деталей на полуполосе предлагается алгоритм, параллельная реализация которого подразумевает гибкую и расширяемую схему построения узлов

вычислительной системы, которая при наличии управляющего узла позволяет всю сложную вычислительную работу перекладывать на узлы-агенты, что позволит использовать данный алгоритм как эффективную эвристику, получающую допустимое размещение набора деталей с плотными фрагментами за приемлемое время.

Принципиально схема алгоритма подразумевает несколько различных моделей распараллеливания процесса, позволяя выполнять алгоритм как в виде многопоточной программы с общей памятью, так и на вычислительном кластере. В последнем случае выделенный узел управляющего центра распоряжается основным хранилищем данных, а приложение-агент может оперировать небольшими объемами данных.

Список литературы

[1] Guo Baosu, Zhang Yu, Hu Jingwen, Li Jinrui, Wu Fenghe, Peng Qingjin, Zhang Quan. Two-dimensional irregular packing problems: A review // Frontiers in Mechanical Engineering. 2022. Vol. 8. https://doi.org/10.3389/fmech.2022.966691.

[2] Wu L., Tian X., Zhang J., Liu Q., Xiao W., and Yang Y. An improved heuristic algorithm for 2d rectangle packing area minimization problems with central rectangles // Eng. Appl. Artif. Intell. 2017. Vol. 66. P. 1–16. https://doi.org/10.1016/j.engappai.2017.08.012

[3] Arnaout J. P., ElKhoury C., Karayaz G. Solving the multiple level warehouse layout problem using ant colony optimization// Oper. Res. Int. J. 2020. Vol.20 (1). P. 473–490. DOI:10.1007/s12351-017-0334-5

[4] Zhang H., Liu Q., Wei L.J., Zeng J., Leng J., Yan D. An iteratively doubling local search for the two-dimensional irregular bin packing problem with limited rotations // Comput. Operations Res. 2022. Vol. 137. 105550. https://doi.org/10.1016/j.cor.2021.105550

[5] Hifi M. A local search-based method for sphere packing problems // European Journal of Operational Research. 2019. Vol. 274, No 2. P. 482–500.

[6] Zhang H., Liu Q., Wei L.J., Zeng J., Leng J., Yan D. An iteratively doubling local search for the two-dimensional irregular bin packing problem with limited rotations // Comput. Operations Res. 2022. Vol. 137. 105550. https://doi.org/10.1016/j.cor.2021.105550

[7] Чеканин В.А. Мультиметодный генетический алгоритм для решения задач раскроя и упаковки прямоугольных объектов // Вестник МГТУ «Станкин». 2019. № 4 (51). С. 14–18.

[8] Manuel Iori, Vinícius L. de Lima, Silvano Martello, Flávio K. Miyazawa, Michele Monaci, Exact solution techniques for two-dimensional cutting and packing // European Journal of Operational Research. 2021. Vol. 289, No. 2, P. 399-415, https://doi.org/10.1016/j.ejor.2020.06.050.

[9] Андрианова А.А., Мухтарова Т.М., Фазылов В.Р. Модели негильотинного размещения набора прямоугольных деталей на листе и полуполосе // Учен. зап. Казан. ун-та. Сер. Физ. матем. науки. 2013. Т. 155, кн. 2. С. 5–18.

[10] Андрианова А.А. Эвристические подходы к использованию точной модели для получения размещения набора прямоугольников на полуполосе // Информационные технологии. 2024. Т. 30, № 11. С. 555–564.

[11] Андрианова А.А. Параллельные алгоритмы решения задачи негильотинного размещения на основе точной модели // Труды международной конференции «Суперкомпьютерные дни в России» (23–24 сентября 2024 г. Москва). Москва : МАКС Пресс, 2024. С. 135–145.

Высокопроизводительная реализация функций exp и expm1 для процессоров архитектуры RISC-V

Е.А. Панова, В.Д. Волокитин, Е.А. Козинов, И.Б. Мееров

Нижегородский государственный университет им. Н.И. Лобачевского

Актуальным вопросом для развития области HPC в рамках архитектуры RISC-V является разработка высокопроизводительной векторной библиотеки стандартных математических функций. В данной работе приводится детальная реализация функций exp(x) и expm1(x), уделяется особое внимание аспектам точности вычислений и производительности. Выполняется подробное сравнение с другими существующими на данный момент библиотеками векторных математических функций под архитектуру RISC-V.

Ключевые слова: математические функции, экспонента, exp, expm1, векторные вычисления, высокопроизводительные вычисления, архитектура RISC-V

1. Введение

Реализации базовых математических функций востребованы при численном моделировании во многих предметных областях, включая вычислительную физику, химию, биологию, финансы. В зависимости от особенностей задачи, к таким реализациям могут предъявляться различные требования, как по точности, так и по производительности. Действительно, в некоторых задачах финансовой математики и компьютерной графики точность не является критическим фактором, влияющим на возможность решения задачи, тогда как время вычислений выходит на первый план. При решении задач вычислительной физики, напротив, к точности подсчета значений математических функций предъявляются повышенные требования. Понимая сложности, связанные с наличием двух противоречивых критериев (точность, время вычислений), разработчики компиляторов реализуют аппроксимации математических функций, ориентируясь на требования стандартов, и давая прикладным программистам возможность влиять на детали работы алгоритма путем установки соответствующих опций компилятора. Так, в компиляторах C и C++ поддерживаются одинарная и двойная (в некоторых реализациях еще и половинная) точность, удовлетворяются требования GNU C по числу правильно вычисленных бит мантиссы, установленные для каждой функции, а также поддерживается набор флагов, определяющий степень толерантности к арифметическим преобразованиям, необходимость учета специальных значений и другие особенности реализации.

Кажущаяся простота реализации математических функций, так или иначе сводящаяся к вычислению некоторого полинома в точке, не должна вводить в заблуждение. Достижение требуемой точности за как можно меньшее время вычислений, выполнение всех предъявляемых требований не является элементарной задачей [1, 2].

Исторически одним из первых классов алгоритмов для вычисления математических функций являлись итерационные методы, в частности, алгоритм Волдера [3]. Другие методы используют полиномиальную аппроксимацию с предварительной редукцией аргумента [4–7]. Подход с использованием табличных значений может быть применен в тех областях, где не требуется высокая точность, в частности, для 8-битной арифметики. Современные высокопроизводительные реализации эффективно комбинируют полиномиальную аппроксимацию и табличные значения [8–10]. Тем не менее, задача разработки реализации, удовлетворяющей требованиям по точности и производительности, все еще остается актуальной [11].

Наряду с реализациями, встроенные в компиляторы языков программирования высокого уровня, распространены следующие программные библиотеки. Так, в библиотеке MPFR [12] представлены реализации математических функций в произвольной точности. Библиотека CRLibm [13] предоставляет производительные реализации математических функций с корректным округлением в любой точке диапазона. В библиотеке SLEEF [14] представлены векторные реализации математических функций для различных платформ. Тем не менее, при появлении новых процессорных архитектур проблема разработки библиотеки математических функций, оптимизированной для этой архитектуры, сочетающей точность вычислений с высокой производительностью, позволяющей настраивать режимы точности, и, что очень важно, позволяющей компилятору векторизовывать вычислительные циклы, вновь выходит на передний план. Именно в данном направлении и выполняется проект, некоторые из текущих результатов которого представлены в данной статье.

Проект посвящен созданию высокопроизводительной библиотеки, реализующий стандартные функции из модуля LibM компиляторов C и C++ в виде, допускающем работу в векторизованных циклах (в терминологии Intel C++ Compiler такой модуль называется SVML, Short Vector Math Library). Данная библиотека ориентирована на перспективные процессоры открытой и свободной от патентных отчислений архитектуры RISC-V, которая активно развивается последние 10 лет большим международным сообществом, в котором Российские компании играют заметную роль. В данной статье представлены результаты реализации функций `exp(x)` и `expm1(x)=exp(x)-1`. В работе приведено детальное описание алгоритма, использованных оптимизаций и улучшений для повышения точности и производительности. Выполнено подробное сравнение с другими аналогичными библиотеками под архитектуру RISC-V, а именно SLEEF и VecLibm [15, 16]. Изучен вопрос о компромиссе между скоростью вычислений и точностью, показано, какие алгоритмические приемы влияют на достижение этого компромисса. Авторы надеются, что результаты могут быть полезны другим исследователям при реализации математических библиотек. Коды будут открыто доступны по мере их окончательной готовности.

2. Постановка задачи

Стандарт IEEE 754 [17] предоставляет требования к реализации ряда математических функций. Так функция `exp(x)` должна обрабатывать следующие диапазоны аргументов и особые случаи:

- $x \in (-\infty, x_{underflow}]$, $\exp(x) = +0$ или *subnormal* – ожидается исключение `FE_UNDERFLOW`;
- $x \in [x_{overflow}, +\infty)$, $\exp(x) = +\infty$ – ожидается исключение `FE_OVERFLOW`;

- $x \in (x_{underflow}, x_{overflow})$ – основной диапазон интереса (*normal*);
- $\exp(-\infty) = +0$, $\exp(+\infty) = +\infty$;
- $\exp(\pm 0) = +0$;
- $\exp(\texttt{qNaN}) = \texttt{qNaN}$;
- $\exp(\texttt{sNaN}) = \texttt{qNaN}$, ожидается исключение `FE_INVALID`.

В точках, где выполняется округление, ожидается исключение `FE_INEXACT`, которое вызывается автоматически. Значения $x_{underflow}$ и $x_{overflow}$ зависят от типа числа с плавающей точкой. В целях краткости обработка особых случаев остается за рамками данной статьи, рассматривается только диапазон *normal* значений.

Важным вопросом при проектировании математической библиотеки является точность реализации математических функций. Согласно стандарту, математическим функциям следует возвращать корректно округленные значения во всех точках допустимого диапазона. Для описания ошибок округления часто используют понятие *unit in the last place* (*ulp*), означающее расстояние между двумя соседними представимыми числами с плавающей точкой. Расстояние между точным математическим значением и ближайшим представимым числом, не превышающее 0.5 ulp, гарантирует, что будет выполнено корректное округление. Здесь и далее подразумевается режим округления к ближайшему представимому числу (*round-to-nearest mode*).

Ограничение в 0.5 ulp достаточно трудно обеспечить, что подробно рассматривается в разделе 3.5. Большинство математических библиотек удовлетворяют более слабым критериям. Так модуль LibM компилятора GCC [18] гарантирует ошибку не более 1 ulp для функций `exp` и `expm1` в одинарной и двойной точности. В своей реализации мы предъявляем к этим функциям следующие требования:

- допускается ошибка 1 ulp, но не более;
- точность реализации должна стремиться к ошибке 0.501 ulp, что эквивалентно одному неверному округлению из 1000 случайных чисел с плавающей точкой.

3. Алгоритм вычисления функций exp и expm1

3.1. Общий метод

Один из классических алгоритмов вычисления экспоненты включает в себя редукцию аргумента в отрезок в окрестности нуля и последующую аппроксимацию функции в этом отрезке полиномом [8, 10]. Редукция может быть выполнена за счет использования машинного представления числа с плавающей точкой: $e^x = 2^E e^y$, $y = x - E\ln 2$. При выборе $E = \lfloor x/\ln 2 \rceil$, где $\lfloor\cdot\rceil$ – округление к ближайшему целому в соответствии со стандартом IEEE, аргумент отображается в отрезок $y \in [-\ln 2/2, \ln 2/2]$, в котором функция e^y аппроксимируется достаточно точно полиномом небольшой степени. С целью уменьшения числа операций при вычислении полинома аргумент может быть отображен в меньший диапазон за счет использования предпосчитанных табличных значений:

$$e^x = 2^E 2^{2^{-k} f} e^y \approx 2^E T(f) P(y), \tag{1}$$

$$h = \left\lfloor \frac{x}{2^{-k}\ln 2} \right\rceil = 2^k E + f, \tag{2}$$

$$y = x - h \cdot 2^{-k} \ln 2. \tag{3}$$

Здесь $f = 0..2^k - 1$ – младшие k бит целого числа h, $2^k E$ – старшие биты h, $T(f) = 2^{2^{-k} f}$ – значение из таблицы размером 2^k, $P(y)$ – полиномиальная аппроксимация функции e^y, $y \in [-\ln 2/2^{k+1}, \ln 2/2^{k+1}]$. Значение параметра k выбирается с учетом архитектурных особенностей. Увеличение k приводит к уменьшению степени аппроксимирующего полинома и, следовательно, количеству арифметических операций, однако требует использования таблицы большего размера.

Функция e^y на редуцированном отрезке обычно аппроксимируется минимаксным полиномом, для построения которого применяется алгоритм Ремеза [4]. Ошибка аппроксимации контролируется степенью полинома. Для определения степени и коэффициентов полинома при заданной точности аппроксимации мы использовали программное обеспечение Sollya [19].

Кроме ошибок аппроксимации, на точность результата существенно влияют ошибки, возникающие при выполнении операций над числами с плавающей точкой. Поддержка fused multiply–add (FMA) инструкций позволяет улучшить одновременно производительность и точность вычислений, последнее за счет выполнения одного округления вместо двух при эквивалентном умножении и сложении. Также на некоторых этапах вычисления функции для корректного округления требуется учитывать отбрасываемые младшие биты операндов.

Особое внимание с точки зрения точности следует уделить вычислению формулы (3). В случае, когда порядок y много меньше порядка x, при вычитании двух чисел возникает катастрофическая потеря точности (*cancellation*), приводящая к неверному результату. В методе Коди и Уэйта [6, 10] константа $c = 2^{-k} \ln 2$ представляется в виде суммы двух чисел с плавающей точкой c_h и c_l, где c_h хранит $m - d$ старших битов мантиссы c, а c_l следующие m битов (m – длина мантиссы, d ограничивает диапазон аргументов функции, $|x| < 2^d$). Доказано, что операция $y' = x - h \cdot c_h$ выполняется абсолютно точно, а ошибка, вносимая операцией $y = y' - h \cdot c_l$, уже не является катастрофической.

Вычисление полинома реализовано с использованием схемы Горнера и операций FMA. На некоторых архитектурах может присутствовать несколько устройств FMA, в этом случае выгодно выполнять вычисления параллельно, например, используя метод Эстрина [20]. Наша реализация адаптирована для одного или двух устройств FMA за счет выполнения вычислений независимо для четных и нечетных степеней полинома. Ошибки округления при вычислении полинома слабо влияют на результат, поскольку для функции e^y полином представляется в виде $P(y) = 1 + Q(y)$, где $|Q(y)| \ll 1$, следовательно, накопленная в младших битах $Q(y)$ ошибка не учитывается при сложении.

В следующих разделах представлены некоторые полезные приемы, использующиеся при вычислении экспоненты, а также несколько алгоритмов вычисления функции e^x, отличающихся соотношением точности и скорости работы.

3.2. Некоторые полезные техники

В данном разделе кратко описываются известные техники, полезные при реализации функции `exp`. Эти и другие приемы описаны более подробно, например, в [21, 22].

Для улучшения точности вычислений иногда необходимо сохранять и использовать младшие биты мантиссы, теряющиеся при выполнении стандартных арифметических операций. Далее операции над числами, представленными двумя числами

с плавающей точкой, будем называть double-FP арифметикой. Основные операции такой арифметики с доказательством корректности работы подробно описаны, например, в [21, 23]. Будем предполагать возможность использования операции FMA без выполнения промежуточного округления, что позволяет существенно сократить вычислительные затраты. В листингах 1-4 приведены некоторые функции double-FP арифметики, используемые в разделе 3.4, а именно:

- быстрое сложение двух чисел a и b (алгоритм `Fast2Sum`, [22]), где порядок a не меньше порядка b, с сохранением младших битов результата (`FP` + `FP` = `double-FP`, листинг 1);
- FMA с сохранением младших битов результата (`FP` · `FP` + `FP` = `double-FP`, листинг 2);
- умножение double-FP чисел, результат double-FP (`double-FP` · `double-FP` = `double-FP`, листинг 3);
- умножение double-FP чисел, результат FP (`double-FP` · `double-FP` = `FP`, листинг 4).

Листинг 1: Алгоритм сложения с результатом double-FP

```
function fastSum12(a, b):  # exponent a >= exponent b
    rh = a + b
    rl = b - (rh - a)
    return rh, rl  # a + b = rh + rl exactly
```

Листинг 2: Алгоритм fused-multiply add с результатом double-FP

```
function fma12(a, b, c):
    rh = FMA(a, b, c)
    rl = FMA(a, b, c - rh)
    return rh, rl
```

Листинг 3: Алгоритм умножения двух чисел double-FP с результатом double-FP

```
function mul22(ah, al, bh, bl):
    rh = ah * bh
    rl = FMA(ah, bh, -rh)
    rl += FMA(ah, bl, al * bh)
    return rh, rl
```

Листинг 4: Алгоритм умножения двух чисел double-FP с результатом FP

```
function mul21(ah, al, bh, bl):
    rh, rl = mul22(ah, al, bh, bl)
    return rh + rl
```

Операция округления к ближайшему целому в формуле (2) для ограниченного диапазона аргументов может быть выполнена быстро за счет выполнения сдвига мантиссы и использования машинного округления. Это достигается прибавлением к исходному числу значения $1.5 \cdot 2^{m-1}$, после чего искомое целое число представляется младшими $w-p-2$ битами результата. Здесь m – длина мантиссы, включая старший бит, p – количество битов в порядке числа, $w = m+p$ – общее количество битов числа с плавающей точкой. Пример использования данного приема приводится в функции `calculate_h` листинга 5.

3.3. Базовый алгоритм вычисления экспоненты

Алгоритм вычисления экспоненты (листинг 5) состоит из следующих этапов:

1) *Редукция.* Округление к ближайшему целому по формуле (2) выполняется в функции `calculate_h`, редукция аргумента по формуле (3) в функции `range_reduction`.

2) *Извлечение табличного значения.* Младшие k бит числа h, полученного на предыдущем шаге, составляют индекс в таблице $T(f)$ размером 2^k.

3) *Вычисление полинома.* Как было отмечено в разделе 3.1, значение полинома вычисляется параллельно за счет разделения четных и нечетных степеней. Общий вид полинома и схема вычислений для полинома степени 6 с использованием метода Горнера и операций FMA продемонстрированы в формуле (4) ($a_2 \geq a_3 \geq \ldots \geq a_6$). Единица учитывается на этапе реконструкции для обеспечения необходимой точности. Степень полинома зависит от выбора k. Функции `evaluate_polynomial_r` и `evaluate_polynomial` вычисляют значения $R(y)$ и $Q(y)$ соответственно, которые определяются формулой (4):

$$\begin{aligned} Q(y) = P(y) - 1 = y + a_2 y^2 + a_3 y^3 + a_4 y^4 + a_5 y^5 + a_6 y^6 = \\ = y + y^2[(a_2 + y^2(a_4 + a_6 y^2))) + y(a_3 + a_5 y^2))] = \\ = y + y^2[p_{even}(y^2) + y \cdot p_{odd}(y^2)] = y + y^2 R(y). \end{aligned} \tag{4}$$

4) *Реконструкция.* На данном этапе выполняется реконструкция результата по формуле (1) с использованием ранее полученных значений таблицы и полинома. Для увеличения точности умножение $T(f)$ и $P(y)$ производится с помощью операции FMA: $T(f)P(y) = T(f) + T(f)Q(y)$.

Листинг 5: Базовый алгоритм вычисления функции exp(x)

```
function calculate_h(x):
    h = FMA(x, INV_LOG2_K, FP2INT_CONST)
    hi = AS_INT(h & MASK_HI_BIT)  # hi is the nearest integer to x*
        INV_LOG2_K
    h -= FP2INT_CONST  # h is the nearest FP to x*INV_LOG2_K
    return h, hi

function range_reduction(x, h):
    yh_ = FMA(h, -LOG2_K_H, x)
    yh = FMA(h, -LOG2_K_L, yh_)
    return yh

function get_table_value(hi):
    fi = hi & MASK_FI_BIT  # table index
    th = TABLE_H[fi]
    return th

function evaluate_polynomial_r(yh, sqry):
    p_odd = ...   # evaluation of odd polynomial terms
    p_even = ...  # independent evaluation of even polynomial terms
    res = FMA(yh, p_odd, p_even)
    return res

```

```
function evaluate_polynomial(yh):
    sqry = yh * yh
    r = evaluate_polynomial_r(yh, sqry)
    ph = FMA(sqry, r, yh)
    return ph

function update_exponent(hi, res):   # res != 0
    Ei = hi >> k
    AS_INT(res) += Ei << (m-1)
    return res

function reconstruction(th, ph, hi):
    res = FMA(th, ph, th)
    res = update_exponent(hi, res)
    return res

function exp(x):
    # checks for overflow, underflow, NaN (omitted)
    h, hi = calculate_h(x)
    yh = range_reduction(x, h)
    th = get_table_value(hi)
    ph = evaluate_polynomial(yh)
    res = reconstruction(th, ph, hi)
    return res
```

В данном алгоритме используются следующие предвычисленные константы и параметры:

- `INV_LOG2_K` – округленное к ближайшему представимому числу значение $\frac{1}{2^{-k}\ln 2}$;
- `LOG2_K_H`, `LOG2_K_L` – старшие $m - d$ бит и младшие m бит мантиссы числа $2^{-k}\ln 2$, подробнее см. раздел 3.1;
- `FP2INT_CONST` – число с плавающей точкой, используемое для выполнения округления к ближайшему целому и равное $1.5 \cdot 2^{m-1}$;
- `MASK_HI_BIT`, `MASK_FI_BIT` – маски для выделения младших $p + k + 1$ и k бит соответственно;
- `TABLE_H` – таблица округленных к ближайшему представимому значений функции $T(f) = 2^{2^{-k}f}$, $f = 0..2^k - 1$;
- k – параметр редукции;
- m, p – длина мантиссы и порядка действующего типа числа с плавающей точкой.

В листинге 5 неявно задаются в качестве параметров степень и коэффициенты полинома $P(y)$. Степень полинома зависит от параметра k и желаемой точности аппроксимации. Для стандартных типов числа с плавающей точкой одинарной и двойной точности в базовой версии алгоритма были выбраны следующие параметры:

- `binary32` (`float`): $k = 4$, степень полинома 3;
- `binary64` (`double`): $k = 6$, степень полинома 5.

Экспериментально определено, что данные параметры для базовой версии алгоритма и рассматриваемого в статье программно-аппаратного окружения (раздел 5) дают оптимальный на наш взгляд результат с точки зрения баланса точности и производительности. В разделе 3.4.2 обосновывается необходимость увеличения степени полинома на 1 после ряда улучшений, однако для базовой версии алгоритма это бессмысленно.

Базовый алгоритм достаточно эффективен по производительности, однако показывает не самые лучшие результаты с точки зрения точности вычислений (раздел 5.1). Часто возникают ошибки округления, максимальная ошибка ограничена 2 ulp. В следующем разделе предлагаются различные приемы, позволяющие достигнуть требуемой точности 0.501 ulp.

3.4. Вариации алгоритма

В процессе разработки библиотеки математических функций необходимо решить вопрос, касающийся баланса точности вычислений и скорости работы. Реализация правильного округления во всех точках и теоретическое обоснование корректности функции является сложной задачей, противоречащей задаче обеспечения высокой производительности реализации. В частности, уточнение значения функции в достаточно редких «плохих» точках требует существенных дополнительных затрат, на порядок и более превосходящих ожидаемое время работы. Подробнее данный вопрос рассматривается в разделе 3.5.

В следующих разделах приводятся пошаговые техники, позволяющие увеличить точность реализации функции `exp`. Большинство этих техник подразумевает использование double-FP арифметики. Наилучшим с точки зрения точности вариантом является выполнение всех вычислений с учетом старших и младших битов числа [22], однако это неприемлемо в рамках высокопроизводительной библиотеки математических функций. Одной из задач при реализации высокопроизводительной библиотеки математических функций является поиск баланса между точностью и производительностью. В данной работе предлагается несколько версий алгоритма вычисления экспоненты, дающих разный результат по точности и производительности. Версии упорядочены по возрастанию точности. Каждая следующая версия заменяет операцию, приводящую в предыдущей версии к наибольшему числу ошибок округления, на ее более точный аналог. Выполнение более точных операций ухудшает производительность, что иллюстрируется в разделе 5.

3.4.1. Вариация 1. Хранение старших и младших битов таблицы

Более 95% ошибок округления в базовой версии алгоритма возникает из-за неточных операндов в последней операции FMA при реконструкции результата. В частности, большая потеря точности происходит из-за округления табличного значения к ближайшему представимому числу. Чтобы этого избежать, необходимо хранить младшие биты мантиссы табличного значения и учитывать их при реконструкции. Данный прием позволяет существенно увеличить точность результата без критических потерь в производительности за счет параллельного выполнения арифметических операций и операций загрузки из памяти.

В листинге 6 младшие биты таблицы хранятся в отдельном массиве `TABLE_L`. Принимается во внимание, что сложение операндов должно производиться в порядке от меньшего по модулю к большему во избежание потери точности.

Листинг 6: Вариация 1 алгоритма вычисления функции exp(x)

```
function get_table_value(hi):
    fi = hi & MASK_FI_BIT
    th = TABLE_H[fi]
    tl = TABLE_L[fi]
    return th, tl
```

```
6
7  function reconstruction(th, tl, ph, hi):
8      res = th + FMA(th, ph, tl)
9      res = update_exponent(hi, res)
10     return res
```

3.4.2. Вариация 2. Увеличение степени полинома

Данный этап предназначен для демонстрации эффекта от увеличения степени полинома на 1. Далее мы используем следующие параметры метода:

- `binary32` (`float`): $k = 4$, степень полинома 4;
- `binary64` (`double`): $k = 6$, степень полинома 6.

Стоит отметить, что хотя текущая вариация кода требует всего одной дополнительной операции FMA относительно предыдущей, это достаточно заметно влияет на время работы программы, т.к. зависимость по данным препятствует эффективной конвейерной обработке.

3.4.3. Вариация 3. Использование double-FP арифметики в редукции аргумента

Дальнейшее увеличение точности предполагает использование double-FP арифметики во избежание ошибок округления на этапах редукции аргумента и вычисления полинома. Известным приемом для уточнения значения полинома является точное выполнение последней операции в схеме Горнера (функция `evaluate_polynomial` листинга 7). На этапе реконструкции производится умножение двух чисел формата double-FP.

Листинг 7: Вариация 3 алгоритма вычисления функции exp(x)

```
1  function evaluate_polynomial(yh):
2      sqry = yh * yh
3      r = evaluate_polynomial_r(yh, sqry)
4      ph, pl = fma12(sqry, r, yh)
5      return ph, pl
6
7  function reconstruction(th, tl, ph, pl, hi):
8      sh, sl = fastSum12(1, ph)
9      sl += pl
10     res = mul21(th, tl, sh, sl)
11     res = update_exponent(hi, res)
12     return res
```

3.4.4. Вариация 4. Точное выполнение редукции методом Коди-Уэйта

В функции `range_reduction` листинга 5 выполняется редукция аргумента классическим методом Коди-Уэйта [6]. Доказано, что первая операция FMA выполняется точно, однако вторая вносит ошибку округления. При использовании double-FP арифметики потери точности можно избежать, сохранив младшие биты y (листинг 8).

Листинг 8: Вариация 4 алгоритма вычисления функции exp(x)

```
function range_reduction(x, h):
    yh_ = FMA(h, -LOG2_K_H, x)
    yh, yl = fma12(h, -LOG2_K_L, yh_)
    return yh, yl

function evaluate_polynomial(yh, yl):
    sqry = yh * yh
    r = evaluate_polynomial_r(yh, sqry)
    ph, pl = fma12(sqry, r, yh)
    pl += yl
    return ph, pl
```

3.4.5. Вариация 5. Идеальная редукция

Несмотря на то, что операции FMA редукции аргумента в функции `range_reduction` листинга 8 выполняются точно, редукция все еще вносит существенную ошибку при вычислениях в одинарной точности. Источником ошибки является недостаточное количество битов мантиссы при хранении константы $2^{-k}\ln 2$. В листинге 9 эта константа представляется в виде трех чисел с плавающей точкой, дополнительные младшие биты позволяют уточнить редуцированный аргумент.

Листинг 9: Вариация 5 алгоритма вычисления функции exp(x)

```
function range_reduction(x, h):
    yh_ = FMA(h, -LOG2_K_H, x)
    yh, yl = fma12(h, -LOG2_K_L, yh_)
    yl = FMA(h, -LOG2_K_LL, yl)
    yh, yl = fastSum2(yh, yl)
    return yh, yl
```

3.4.6. Вариация 6. Уточнение значения полинома с использованием double-FP арифметики

Точность вычисления значения полинома играет достаточно важную роль, что особенно заметно в случае функции `expm1` в окрестности нуля. Накапливаемая ошибка вычислений при выполнении последовательных операций FMA может быть уменьшена за счет выполнения операций в double-FP арифметике. В вариации 6 (листинг 10) в схеме Горнера заменяются две последние операции FMA на операции `fma12` (листинг 2). Порядок вычислений описывается формулой (5):

$$\begin{aligned} Q(y) = P(y) - 1 = y + a_2y^2 + a_3y^3 + a_4y^4 + a_5y^5 + a_6y^6 = \\ = y + y^2[a_2 + y(a_3 + \ldots)] = y + y\left[y \cdot (a_2 + yS(y))\right] \end{aligned} \tag{5}$$

Листинг 10: Вариация 6 алгоритма вычисления функции exp(x)

```
function evaluate_polynomial(yh, yl):
    sqry = yh * yh
    s = evaluate_polynomial_s(yh, sqry)
    sh, sl = fma12(yh, s, a2)
    sh, sl = mul22(yh, yl, sh, sl)
    ph, pl = fma12(yh, sh, yh)
    pl += FMA(yh, sl, yl)
    return ph, pl
```

3.5. Дальнейшее повышение точности. Обсуждение

Полученный в прошлом разделе результат можно уточнять за счет повсеместного использования double-FP арифметики. В CRLibm [22] («быстрая» фаза алгоритма) удалось достигнуть точности 68 бит результата для типа `double`. Тем не менее, даже такая точность не гарантирует корректного округления. В литературе данная проблема обозначается как *Table Maker's Dilemma (TMD)* [11, 24]. Трансцендентный результат z, найденный с некоторой точностью ε, может быть округлен как вверх, так и вниз, если середина отрезка между двумя представимыми точками попадает в интервал $(z-\varepsilon, z+\varepsilon)$.

Классическим решением проблемы TMD является итерационный алгоритм Зива [25], предполагающий повторение вычислений в расширенной точности. Лефевр [24] показал, что для экспоненты в двойной точности достаточно 157 битов мантиссы для гарантированного выполнения верного округления. В CRLibm корректная во всех точках реализация функции `exp` включает в себя «быструю» и «точную» фазы, последняя в расширенной арифметике. Необходимость выполнения второй фазы определяется некоторым условием после выполнения первой фазы. «Точная» фаза на порядок медленнее «быстрой», однако случаи, когда она требуется, достаточно редки: один на около 2 млн точек.

Мы не ставили себе целью получить реализацию, приближающуюся к CRLibm по точности. В то же время мы ставим себе задачу поиска компромисса между точностью и производительностью, как это принято в компиляторах языков программирования C и C++. В разделе 5.1 мы демонстрируем соотношение точности и скорости работы на примере представленных выше вариаций вычисления функции `exp`.

3.6. Функция expm1. Реконструкция аргумента

Функция `expm1` имеет во многом схожую реализацию с функцией `exp`, за исключением финального вычитания единицы из полученного результата. Затруднение вызывает тот факт, что единица вносит различный вклад для больших и маленьких по модулю аргументов. В [26] эти случаи обрабатываются в отдельных ветках, что неприемлемо с точки зрения производительности векторного кода. Мы используем вместо ветвлений операцию `fastSum12` с предварительным определением числа с наибольшим порядком. Последнее может быть выполнено достаточно быстро за счет маскированных операций. Алгоритм реконструкции аргумента для функции `expm1` представлен в листинге 11.

Листинг 11: Реконструкция аргумента для функции expm1(x)

```
function reconstruction(th, tl, ph, pl, hi):
    rh, rl = fastSum12(1, ph)
    rl += pl
    sh, sl = mul22(th, tl, rh, rl)
    sh = update_exponent(hi, sh)
    sl = update_exponent(hi, sl)
    smax, smin = sortNumbersByExponent(sh, -1)
    uh, ul = fastSum12(smax, smin)
    res = uh + (ul + sl)
    return res
```

4. Векторная реализация

Реализация под архитектуру RISC-V выполнена с использованием векторных функций-интринсиков, компилирующихся в векторные инструкции RVV 1.0. Реализация алгоритмов выполнена для одинарной и двойной точности, половинная точность в процессе разработки. Поддерживается режим `-ffast-math`, исключающий проверки на особые случаи и вычисления с *subnormal* числами. В будущем планируется поддержка нескольких реализаций с различной точностью, в том числе низкой точностью с максимальной ошибкой 3.5 ulp.

Особенностью архитектуры является возможность объединять логические векторные регистры за счет изменения параметра `LMUL`. Это может дать заметный прирост в производительности [27, 28]. Библиотека поддерживает 4 режима `LMUL` (*m1*, *m2*, *m4*, *m8*). В режиме с `LMUL`=8 выполняется двойной запуск реализации с `LMUL`=4 для ускорения времени работы, т.к. при `LMUL`=8 количества логических регистров недостаточно для создания эффективного машинного кода.

5. Эксперименты

5.1. Точность

В рамках проекта была разработана система тестирования на основе библиотеки MPFR с точностью 200 бит. Система генерирует равномерную тестовую выборку в заданных диапазонах и вычисляет в этих точках ошибку в ulp относительно эквивалентной функции MPFR. В таблице 1 приводятся результаты тестирования разработанных в предыдущих разделах вариаций функций `exp` и `expm1` в диапазоне $(x_{underflow}, x_{overflow})$. Вариации 4-6, а также 3 для двойной точности соответствуют требованию 0.501 ulp.

Таблица 1: Доля точек с неверно выполненным округлением для различных вариаций функций `exp(x)` и `expm1(x)` для чисел в одинарной («f») и двойной («d») точности в диапазоне $(x_{underflow}, x_{overflow})$. Для базовой версии максимальная ошибка ограничена сверху 2 ulp, для остальных 1 ulp. Точности 0.501 ulp соответствует значение в таблице $\leq 10^{-3}$.

	Базовая	**Вар. 1**	**Вар. 2**	**Вар. 3**	**Вар. 4**	**Вар. 5**	**Вар. 6**
expd	$2 \cdot 10^{-1}$	$5 \cdot 10^{-3}$	$2 \cdot 10^{-3}$	$7 \cdot 10^{-4}$	$3 \cdot 10^{-6}$	$3 \cdot 10^{-6}$	$2 \cdot 10^{-6}$
expf	$2 \cdot 10^{-1}$	$1 \cdot 10^{-2}$	$7 \cdot 10^{-3}$	$3 \cdot 10^{-3}$	$1 \cdot 10^{-3}$	$1 \cdot 10^{-4}$	$6 \cdot 10^{-5}$
expm1d	-	-	-	$7 \cdot 10^{-4}$	$3 \cdot 10^{-6}$	$3 \cdot 10^{-6}$	$1 \cdot 10^{-6}$
expm1f	-	-	-	$3 \cdot 10^{-3}$	$1 \cdot 10^{-3}$	$8 \cdot 10^{-5}$	$6 \cdot 10^{-5}$

Был проведен сравнительный анализ с другими библиотеками. Рассмотрены реализации из GNU C Library 2.39, модуля SVML пакета Intel oneAPI 2023, библиотек SLEEF, VecLibm. Библиотека VecLibm поддерживает только двойную точность, для которой предоставляет несколько версий функции `exp` с различной степенью точности. Мы рассмотрели версию по умолчанию и версию с табличной редукцией (`rvvlm_expD_tbl64`). Для тестирования точности использовалась опция компиляции `-fno-fast-math`. На рисунке 1 представлены результаты тестирования в различных

диапазонах области определения. Каждая функция в каждом диапазоне была протестирована на 10^6 случайных точках. Мы приводим здесь следующие характерные интервалы:

- $I_1 = (x_{underflow}, x_{overflow})$ – диапазон normal значений;
- $I_2 = (x_{underflow}, x_{underflow} + 4)$ – граница underflow;
- $I_3 = (x_{overflow} - 4, x_{overflow})$ – граница overflow;
- $I_4 = (-4, 4)$ – диапазон вблизи нуля;
- $I_5 = (-\ln 2/2^{k+1}, \ln 2/2^{k+1})$ – диапазон полиномиальной аппроксимации.

	expd							expf				
	LibM	SLEEF	SVML	VecLibm		Bap. 2	Bap. 6	LibM	SLEEF	SVML	Bap. 2	Bap. 6
				default	tbl64							
I1	0	60.38	5.26	17.15	5.29	1.67	0.002	0	94.55	40.72	6.80	0.056
I2	0	62.60	5.63	17.49	5.31	1.69	0.001	0	91.88	40.07	6.75	0.250
I3	0	63.39	4.04	17.15	5.22	1.68	0.000	0	93.59	27.88	7.23	0.253
I4	0	62.86	5.05	17.08	5.34	1.70	0.004	0	94.90	41.41	6.60	0.024
I5	0	0.59	5.11	0.003	5.40	1.46	0.000	0	6.10	6.08	6.11	0.000
	expm1d							expm1f				
	LibM	SLEEF	SVML	Veclibm default		Bap. 3	Bap. 6	LibM	SLEEF	SVML	Bap. 3	Bap. 6
I1	92.36	1.26	0.14	88.59		0.74	0.001	85.95	7.75	6.82	2.98	0.064
I2	0	172.71	0	0		0	0	0	172.71	0	0	0
I3	100.14	0.15	0.93	95.48		0.75	0.000	98.64	1.04	8.03	3.89	0.253
I4	96.62	0.33	0.30	65.84		1.44	0.002	96.02	1.21	0.28	3.90	0.028
I5	0.51	0.00	35.59	0.01		1.05	0.001	2.40	0.02	8.35	2.70	0.020

Рис. 1: Результаты тестирования функций `exp(x)` и `expm1(x)` для различных библиотек для чисел в двойной («d») и, если применимо, одинарной («f») точности. Количество результатов с неверным округлением на 1000 точек выборки. Ограничению 0.501 ulp соответствует значение в таблице, меньшее или равное единице.

Для функции `exp` модуля LibM тестовая система не обнаружила точек с ошибкой округления. Предположительно, в GNU реализована точная непроизводительная версия функции, по аналогии с CRLibm, или же точки с ошибкой округления крайне редки. Версия функции `exp` библиотеки VecLibm с табличной редукцией оказалась более точной, чем версия по умолчанию, однако более медленной, что продемонстрировано в разделе 5.2. Библиотека SLEEF показывает для типа `double` точность хуже, чем VecLibm. Наша реализация в вариациях 5 и 6 соответствует требованию 0.501 ulp во всех рассматриваемых диапазонах.

Точность функции `expm1` зависит от диапазона. LibM показывает хорошую точность в окрестности 0, однако в других диапазонах возникает достаточно много ошибок округления. SLEEF и SVML демонстрируют хорошую точность во всех диапазонах, за исключением небольшой окрестности underflow для SLEEF. Реализация VecLibm функции `expm1` дает в некоторых точках ошибку > 1 ulp. Наша реализация в вариации 6 удовлетворяет условию 0.501 ulp во всех перечисленных выше интервалах, при этом важную роль для достижения необходимой точности результата в окрестности нуля играет качество вычисленного значения полинома.

5.2. Производительность

Для замеров производительности была разработана система бенчмаркинга, измеряющая время работы математической функции в трех режимах. Первый тест вычисляет по массиву аргументов массив значений функции («array»), учитываются операции загрузки/выгрузки из памяти. Второй тест вычисляет функцию в цикле без операций загрузки/выгрузки, при этом аргумент следующего вызова зависит от результата предыдущего. Получаемое значение времени работы в тактах близко к показателю латентности функции («latency»). Третий тест направлен на измерение пропускной способности, в цикле вычисляются 4 независимых значения математической функции («throughput»).

Эксперименты производились на процессоре Banana Pi RISC-V SpacemiT K1 (RVV 1.0, длина векторного регистра 256 бит), использовался кросс-компилятор GCC 14.2. Особенностью процессора является in-order выполнение инструкций, в связи с чем тесты «latency» и «throughput» демонстрируют почти одинаковое время работы. Замеры производительности выполнялись в режиме `-ffast-math`. Для библиотеки VecLibm были вручную отключены проверки на особые аргументы. Время работы измерялось с помощью инструкции `rdtime`. Было выявлено экспериментально, что единица `rdtime` для приведенной конфигурации соответствует 75 тактам процессора. Для удобства результаты замеров далее приведены в тактах.

На рисунке 2 представлены замеры производительности для вариаций функций из раздела 3, а также для библиотек SLEEF и VecLibm. Параметр `LMUL` выбран оптимальным для каждой функции. Для нашей реализации и SLEEF оптимальный `LMUL` равен 2. VecLibm предлагает значения по умолчанию 4 для `exp` и 2 для `expm1`, однако позволяет их изменять при необходимости. Версия экспоненты по умолчанию в VecLibm использует аппроксимацию полиномом большой степени и адаптирована для `LMUL`=4, в то время как версия экспоненты с комбинированным подходом (таблица+полином) показывает более быстрые результаты на `LMUL`=1. Мы приводим результаты в наиболее выгодной конфигурации.

		SLEEF	VecLibm		Вариации из раздела "Алгоритм"						
			default	tbl64	Базовая	Вар. 1	Вар. 2	Вар. 3	Вар. 4	Вар. 5	Вар. 6
expd	array	263	75	456	83	90	115	141	143	162	175
	latency	260	47	440	72	78	102	128	131	153	167
	throughput	259	50	440	76	80	104	130	133	156	169
expf	array	162			66	83	90	126	129	140	170
	latency	155	-	-	56	71	79	115	118	130	162
	throughput	155			58	73	81	118	120	132	164
expm1d	array	453	422					166	169	181	200
	latency	445	409		-	-	-	162	165	177	193
	throughput	440	409					162	165	178	195
expm1f	array	404						155	164	170	199
	latency	401	-		-	-	-	148	157	165	193
	throughput	393						149	159	166	195

Рис. 2: Время работы функций `exp(x)` и `expm1(x)` для чисел в двойной («d») и, если применимо, одинарной («f») точности. Усредненное время одного векторного вызова в тактах, деленное на значение `LMUL`.

Можно видеть, что в случае нашей реализации время работы функций растет с увеличением точности вычислений. Для функции `exp` вариация 2 близка по точности к условию 0.501 ulp и занимает на 25% меньше времени, чем вариация 3, в связи с чем имеет смысл рассматривать ее как оптимальное решение с точки зрения баланса точности и производительности. Для функции `expm1` часть алгоритма с правильным вычитанием единицы занимает существенную часть времени работы, однако ее упрощение приводит к ошибке в 2 ulp из-за чувствительности функции к ошибкам вычислений.

Наша реализация превосходит библиотеку SLEEF по производительности. Медленная и точная вариация 5 функции `exp` в $\sim$1.5 раза быстрее SLEEF, при этом показывает более хорошую точность. Оптимальная с точки зрения скорости работы вариация 2 быстрее в 2-2.5 раза. Реализация `expm1` быстрее в 2-2.7 раз в зависимости от точности.

Рассмотрим результаты сравнения с библиотекой VecLibm. Большой потенциал для оптимизации под архитектуру RISC-V дает возможность объединять логические регистры, варьируя параметр `LMUL` и задействуя параллельные вычислительные блоки. Реализация функции `exp` библиотеки VecLibm показывает ускорение $\sim$8 раз при `LMUL`=4 относительно `LMUL`=1. В то же время, по всей видимости, неадаптированная под определенный `LMUL` реализация `expm1` показывает результаты, сопоставимые со SLEEF. В представленных на рисунке 2 результатах экспериментов мы выбрали наиболее выгодный для VecLibm режим работы, однако в вычислительных кодах работу математических функций нельзя рассматривать изолированно от остальных вычислений. Действительно, при векторизации вычислительного цикла может возникнуть ситуация, в которой использование оптимального для VecLibm `LMUL`=4 окажется невыгодным из-за ограниченного числа регистров. Возникшая в такой ситуации нехватка регистров будет вызвана их нерациональным распределением и приведет к деградации суммарного времени работы векторизуемого цикла. В некоторых алгоритмах этот эффект может быть нивелирован за счет предвычисления большого количества экспонент и сохранения результатов в достаточно большие массивы, однако это не всегда возможно из-за особенностей алгоритма. Кроме того, использование дополнительной памяти также может привести к существенному замедлению вместо ожидаемого ускорения. Эти соображения позволяют сделать следующий вывод: в математических библиотеках для RISC-V процессоров целесообразно поддерживать оптимизированные реализации функций для всех доступных значений `LMUL`. В настоящий момент предлагаемая нами реализация в 2-3 раза обгоняет VecLibm в режиме с `LMUL`=1 для обеих функций (`exp` и `expm1`), но отстает в 2-4 раза в случае функции `exp` и `LMUL`=4. Дальнейшая оптимизация кода для последнего случая является одним из направлений дальнейшей работы.

6. Заключение

Поиск компромисса между точностью реализации и производительностью для стандартных математических функций не является тривиальной задачей. Ограничение на ошибку 0.5 ulp требует вычислений в расширенной точности, что вызывает большие затруднения в случае векторной реализации. Мы ориентируемся на ограничение 0.501 ulp, однако такая точность требует дополнительных вычислительных затрат и

не требуется для ряда приложений. Оптимальным решением в данной ситуации является поддержка нескольких отличающихся по точности версий, как это сделано, в частности, в библиотеке VecLibm. Выбор одной из версий удобно осуществлять установкой соответствующих опций компиляции.

Разработанные нами реализации функций `exp(x)` и `expm1(x)` не отстают по точности от сторонних библиотек векторных математических функций, а в ряде случаев их опережают. С точки зрения производительности, большим потенциалом для оптимизации векторного кода под архитектуру RISC-V является возможность изменять и адаптировать реализацию под параметр `LMUL`, что демонстрирует библиотека VecLibm, показывая ускорение ~ 8 раз при изменении `LMUL` для функции `exp`. В дальнейшем мы планируем работу по адаптации под различные `LMUL`, при этом поддерживая эффективность каждой версии. Так или иначе, в базовом режиме с `LMUL`=1 наша реализация показывает ускорение 1.5–2.5 раза относительно рассмотренных в статье сторонних библиотек. В этом плане разрабатываемая нами библиотека обладает большим потенциалом и может быть полезна при разработке высокопроизводительных программ.

Список литературы

[1] Muller J.M. Elementary functions and approximate computing // Proceedings of the IEEE. 2020. V. 108. №. 12. P. 2136–2149.

[2] Liu W., Lombardi F., Shulte M. A retrospective and prospective view of approximate computing [point of view] // Proceedings of the IEEE. 2020. V. 108. №. 3. P. 394–399. https://doi.org/10.1109/JPROC.2020.2975695.

[3] Volder J.E. The CORDIC trigonometric computing technique // IRE Transactions on electronic computers. 1959. №. 3. P. 330–334.

[4] Remes E. Sur un procédé convergent d'approximations successives pour déterminer les polynômes d'approximation // CR Acad. Sci. Paris. 1934. V. 198. P. 2063–2065.

[5] Brisebarre N., Chevillard S. Efficient polynomial L-approximations // 18th IEEE Symposium on Computer Arithmetic (ARITH'07). IEEE, 2007. P. 169–176.

[6] Cody W., Waite W. Software Manual for the Elementary Functions. Prentice-Hall, 1980. https://doi.org/10.1137/1024023.

[7] Kornerup P., Muller J.M. Extending the Range of the Cody and Waite Range Reduction Method. 2005.

[8] Tang P.T.P. Table-driven implementation of the exponential function in IEEE floating-point arithmetic // ACM Transactions on Mathematical Software (TOMS). 1989. V. 15. №. 2. P. 144–157. https://doi.org/10.1145/63522.214389.

[9] Tang P.T.P. Table-driven implementation of the logarithm function in IEEE floating-point arithmetic // ACM Transactions on Mathematical Software (TOMS). 1990. V. 16. №. 4. P. 378–400.

[10] Muller J-M. Elementary functions: Algorithms and Implementation. 3rd Edition. Birkhäuser Boston, 2016. 283 p. https://doi.org/10.1007/978-1-4899-7983-4.

[11] Brisebarre N., et al. Correctly-rounded evaluation of a function: why, how, and at what cost? // HAL preprint. 2025.

[12] The GNU MPFR Library. https://www.mpfr.org/ (accessed: 30.04.2025).

[13] Daramy C. et al. CR-LIBM: A correctly rounded elementary function library // Advanced Signal Processing Algorithms, Architectures, and Implementations XIII. SPIE, 2003. V. 5205. P. 458–464.

[14] SLEEF Vectorized Math Library. https://sleef.org/ (accessed: 30.04.2025).

[15] Tang P.T.P. An Open-Source RISC-V Vector Math Library // 2024 IEEE 31st Symposium on Computer Arithmetic (ARITH). IEEE, 2024. P. 60–67.

[16] The VecLibm Library. https://github.com/rivosinc/veclibm (accessed: 30.04. 2025).

[17] IEEE Computer Society Microprocessor Standards Committee. IEEE standard for floating-point arithmetic. IEEE, 2019. 82 p.

[18] The GNU C Library. https://www.gnu.org/software/libc/manual/html_node/index.html (accessed: 30.04.2025).

[19] Sollya. https://www.sollya.org/ (accessed: 30.04.2025).

[20] Estrin G. Organization of computer systems: the fixed plus variable structure computer // Papers presented at the May 3-5, 1960, western joint IRE-AIEE-ACM computer conference, May 3-5, 1960, San Francisco, California, USA. ACM, 1960. P. 33–40. https://doi.org/10.1145/1460361.1460365.

[21] Dekker T.J. A floating-point technique for extending the available precision // Numerische Mathematik. 1971. V. 18. №. 3. P. 224–242. https://doi.org/10.1007/BF01397083.

[22] Daramy C., et al. CR-LIBM: The evaluation of the exponential. [Research Report] LIP RR-2003-37, Laboratoire de l'informatique du parallélisme. 2003. P. 2–37.

[23] Linnainmaa S. Software for doubled-precision floating-point computations // ACM Transactions on Mathematical Software (TOMS). 1981. V. 7. №. 3. P. 272–283. https://doi.org/10.1145/355958.355960.

[24] Lefèvre V., Muller J.M., Tisserand A. Toward correctly rounded transcendentals // IEEE Transactions on Computers. 1998. V. 47. №. 11. P. 1235–1243. https://doi.org/10.1109/12.736435.

[25] Ziv A. Fast evaluation of elementary mathematical functions with correctly rounded last bit // ACM Transactions on Mathematical Software (TOMS). 1991. V. 17. №. 3. P. 410–423. https://doi.org/10.1145/114697.116813.

[26] Tang P.T.P. Table-driven implementation of the expm1 function in IEEE floating-point arithmetic // ACM Transactions on Mathematical Software (TOMS). 1992. V. 18. №. 2. P. 211222. https://doi.org/10.1145/146847.146928.

[27] Volokitin V.D. et al. Improved vectorization of OpenCV algorithms for RISC-V CPUs // Lobachevskii Journal of Mathematics. 2024. V. 45. №. 1. P. 130–142. https://doi.org/10.1134/S1995080224010530.

[28] Pirova A. et al. Performance optimization of BLAS algorithms with band matrices for RISC-V processors // Future Generation Computer Systems. 2025. P. 107936. https://doi.org/10.1016/j.future.2025.107936.

Геометрический многосеточный метод для неразнесенных аппроксимаций уравнений динамики атмосферы на сетке кубическая сфера[1]

Г.С. Гойман[1], В.В. Шашкин[1,2,3]

[1]Институт вычислительной математики им. Г.И. Марчука РАН,
[2]Московский физико-технический институт,
[3]Гидрометцентр России

В работе представлен геометрический многосеточный решатель для СЛАУ, возникающих при пространственной дискретизации уравнений динамики атмосферы на сетке кубическая сфера без разнесения переменных. Ключевой проблемой применения неразнесенных сеток является рассогласование решения между узлами сетки, что в частности приводит к потери эффективности стандартных многосеточных алгоритмов. Для преодоления этого ограничения предложены специализированные операторы перехода между сетками, учитывающие четно-нечетное рассогласование решения. Эти операторы позволяют сохранить эффективность стандартного многосеточного метода без модификации остальных компонент алгоритма.

Предложенный метод протестирован в рамках решения модельных уравнений на сетке кубическая сфера с использованием конечно-разностных аппроксимаций, удовлетворяющих свойству суммирования по частям, второго и четвертого порядка аппроксимации. Проведено исследование скорости сходимости метода а также замеры параллельной эффективности. Предложенный метод демонстрирует высокую скорость сходимости, сравнимую со стандартным геометрическим многосеточным методом для разнесенных сеток. Параллельная реализация алгоритма показала эффективную масштабируемость на суперкомпьютерной системе Cray XC40.

Ключевые слова: Многосеточный метод, неразнесенная сетка, параллельная эффективность, моделирование атмосферы.

1. Введение

В моделях динамики атмосферы параллельная эффективность и точность решений в большей степени определяется выбором пространственной аппроксимации, тогда как метод интегрирования по времени влияет на устойчивость и вычислительную эффективность [19] модели. Полунеявные [26] и неявно-явные (Implicit-Explicit, IMEX) [3]

[1]Исследование выполнено в Институте вычислительной математики им. Г.И. Марчука РАН за счет гранта Российского научного фонда № 24-71-00123 (https://rscf.ru/project/24-71-00123/).

схемы интегрирования являются широко распространенными подходами для решения проблемы жесткости уравнений динамики атмосферы. В рамках этих подходов члены уравнений ответственные за описание динамики быстрых волновых процессов (такие как гравитационные и акустические волны) интегрируются неявно, а более медленные процессы (процессы переноса, эффекты вращения Земли) с использованием явной аппроксимации. Применение таких методов приводит к необходимости решения нелинейной системы уравнений на каждом шаге по времени. Эффективное решение такой системы является сложной задачей, затраты на которую могут перекрывать выигрыш от использования больших шагов интегрирования по времени. Основные вычислительные затраты при использовании неявного метода связаны с необходимостью решения последовательности систем линейных алгебраических уравнений (СЛАУ) с большими разреженными матрицами на каждом шаге по времени модели.

Многосеточные методы [6, 9, 37] представляют собой один из наиболее эффективных подходов к решению эллиптических задач, обладая оптимальной вычислительной сложностью, линейно зависящей от количества неизвестных, и минимизируя объем глобальных коммуникаций в распределенных вычислениях. В области численного моделирования динамики атмосферы данные методы продемонстрировали высокую эффективность как при использовании в качестве самостоятельных решателей, так и в роли предобуславливателей [7, 10, 11, 13, 18, 20, 29, 36].

Эффективность геометрических многосеточных методов существенно зависит от учёта особенностей расчетной сетки и используемой пространственной аппроксимации. Более точное описание динамики волновых процессов и перехода к геострофически сбалансированным состояниям, а также меньшие погрешности при вычислении производных исторически обусловили широкое применение разнесенных сеток (в частности, C-сетки [2]) в глобальных моделях атмосферы.

Однако современные требования к повышению горизонтального разрешения моделей потребовали перехода от традиционных регулярных широтно-долготных сеток к сеткам с квазиравномерным разрешением на сфере [34]. Этот переход стал важным фактором развития моделей атмосферной динамики нового поколения, стимулируя разработку новых численных методов и алгоритмов.

Использование сферических сеток на основе криволинейных неортогональных систем координат, таких как сетка кубическая сфера [24, 28], существенно осложняет реализацию схем с разнесением переменных. Основные сложности связаны с построением консервативных и устойчивых аппроксимаций высокого порядка, а также с вычислительной эффективность этих схем [12, 27, 30]. Схемы, использующие неразнесенную сетку, позволяют избежать этих проблем, подвержены проблеме четно-нечетного (шахматного) рассогласования при решении уравнений гидродинамики. Данный эффект возникает вследствие взаимодействия дискретных операторов градиента и дивергенции в уравнениях импульса и неразрывности, что приводит к появлению паразитных осцилляций на масштабе сетки.

В моделях динамики атмосферы для подавления таких осцилляций обычно применяется дополнительная фильтрация на основе бигармонического оператора [14]. Однако, хотя этот подход и позволяет контролировать проблему рассогласования решений, он не устраняет их причину — наличие дополнительных стационарных мод дискретного оператора Лапласа. Как будет показано далее, наличие этих стационарных мод существенно снижает эффективность стандартных геометрических

многосеточных методов при решении систем уравнений, возникающих в результате применения неявных аппроксимаций по времени.

В данной работе представлен геометрический многосеточный алгоритм, специально разработанный для решения систем линейных уравнений, возникающих при аппроксимации уравнений динамики атмосферы на неразненной сетке. Основная идея метода заключается в использовании специализированных операторов перехода между сетками, которые учитывают характерное для таких схем четно-нечетное рассогласование решений, сохраняя при этом стандартные компоненты многосеточного метода. Эффективность предложенного подхода подтверждена комплексным анализом, включающим исследование модельных задач и серию численных экспериментов на сетке кубическая сфера. Полученные результаты демонстрируют высокую скорость сходимости алгоритма, сравнимую со стандартным геометрическим многосеточным методом в случае использования разнесенной сетки.

Структура статьи организована следующим образом. В разделе 2 приводится основная идея предложенного метода в рамках модельных одномерных и двумерных уравнений мелкой воды в периодической области. Раздел 3 посвящен формулировке модельной задачи на сетке кубическая сферы с детальным описанием реализации предложенного многосеточного метода. В разделе 4 приведены результаты численных экспериментов, демонстрирующие скорость сходимости и параллельную эффективность алгоритма. Раздел 5 содержит основные выводы работы и перспективные направления развития

2. Анализ модельной задачи

2.1. Одномерная модельная задача

Для демонстрации сути неявно-явного подхода для интегрирования по времени в атмосферных моделях и изложения основной идеи предлагаемого многосеточного метода рассмотрим упрощенную одномерную модельную задачу. Рассмотрим линейные уравнения мелкой воды для скорости u и отклонение уровня жидкости h на периодическом отрезке $x \in [0, 2\pi)$:

$$\frac{\partial u}{\partial t} + c_{adv}\frac{\partial u}{\partial x} = -g\frac{\partial h}{\partial x}, \tag{1}$$

$$\frac{\partial h}{\partial t} + c_{adv}\frac{\partial h}{\partial x} = -H\frac{\partial u}{\partial x}. \tag{2}$$

Здесь c_{adv} — постоянная фоновая скорость течения, g — ускорение свободного падения, H — средняя высота уровня жидкости. Данная система описывает перенос со скоростью c_{adv} и распространение гравитационных волн с фазовой скоростью $c_{grav} = \sqrt{gH}$.

При условии $c_{adv} \ll c_{grav}$, максимально допустимый шаг по времени для явных схем ограничен фазовой скоростью гравитационных волн. Следуя логике неявно-явных методов, это ограничение можно ослабить, применяя неявную аппроксимацию к членам, отвечающим за динамику гравитационных волн. Например, можно

воспользоваться комбинацией схемы чехарда и метода трапеций:

$$\frac{u^{n+1}-u^{n-1}}{2\Delta t}+c_{adv}\frac{\partial u}{\partial x}^{n}=-\frac{g}{2}\left(\frac{\partial h}{\partial x}^{n+1}+\frac{\partial h}{\partial x}^{n-1}\right), \tag{3}$$

$$\frac{h^{n+1}-h^{n-1}}{2\Delta t}+c_{adv}\frac{\partial h}{\partial x}^{n}=-\frac{H}{2}\left(\frac{\partial u}{\partial x}^{n+1}+\frac{\partial u}{\partial x}^{n-1}\right), \tag{4}$$

где n — номер шага по времени, Δt — размер шага интегрирования по времени. Дифференцируя уравнение (3) и подставляя в (4), получаем эллиптическое уравнение:

$$\left(I-gH\Delta t^2\frac{\partial^2}{\partial x^2}\right)h^{n+1}=q, \tag{5}$$

где I — тождественный оператор, а правая часть q содержит комбинацию членов с временных шагов n и $n-1$.

Введем равномерную сетку $x_i=(i-1)\Delta x$, $i\in[1,N_x]$, $\Delta x=(b-a)/N_x$, где N_x — четное число. Аппроксимируя уравнения (3), (4) с использованием центрально-разностных схем второго порядка, получаем систему линейных уравнений:

$$h_i-\frac{\Delta t^2 gH}{4\Delta x^2}\left(h_{i+2}-2h_i+h_{i-2}\right)=q_i,\quad i\in[1,N_x], \tag{6}$$

где h_i — значение сеточной функции в узле x_i (индекс временного слоя $n+1$ опущен для краткости) с периодическими граничными условиями $h_{N_x+k}=h_k$. Отметим, что использование неразнесенной сетки приводит к “широкому шаблону” аппроксимации второй производной.

Для построения геометрического многосеточного метода решения уравнения (6), требуются следующие ключевые компоненты:

- Сглаживатель — алгоритм, эффективно устраняющий высокочастотные компоненты ошибки решения.
- Последовательность грубых сеток, позволяющая ускорить сходимость за счет уменьшения числа степеней свободы и улучшения обусловленности задачи на каждом уровне.
- Операторы перехода между сетками (сгрубления и продолжения) для переноса невязок и поправок решения между уровнями.
- Матрицы системы на грубых сетках, аппроксимирующие исходную задачу.
- Решатель на самой грубой сетке.

Среди перечисленных компонент процедура сглаживания играет ключевую роль, поэтому мы начнем с анализа процедуры сглаживания для системы (6).

Собственные значения матрицы системы имеют вид:

$$\lambda=1+\sigma^2\sin^2 k\Delta x\equiv 1+\sigma^2\sin^2\theta, \tag{7}$$

где k — волновое число, $\theta=k\Delta x\in[-\pi,\pi)$, $\sigma=c_{grav}\Delta t/\Delta x$ число Куранта для фазовой скорости гравитационных волн. В отличие от случая применения стандартных трехточечных аппроксимаций второй производной с узким шаблоном, зависимость, $\lambda\in[1,1+\sigma^2]$ не является монотонной. Эта немонотонность возникает из-за наличия в ядре матрицы аппроксимации оператора Лапласа дополнительного коротковолнового собственного вектора (двухшаговая мода), что обусловлено рассогласованием

нечетных и четных степеней свободы. С точки зрения многосеточных методов это означает, что дискретный оператор Лапласа не является h-эллиптическим [5, 37], а мера h-эллиптичности всей системы стремится к нулю при $\sigma \gg 1$. Это осложняет разработку эффективных сглаживающих процедур, основанных на стандартных геометрических определениях гладкости сеточных функций.

Рассмотрим применение итераций метода Якоби со взвешиванием в качестве сглаживателя:

$$h_i^{(p+1)} = h_i^{(p)} + \frac{\omega}{1+\sigma^2/2}\left(q_i - h_i^{(p)} + \frac{\sigma^2}{4}\left(h_{i+2}^{(p)} - 2h_i^{(p)} + h_{i-2}^{(p)}\right)\right), \tag{8}$$

где (p) — индекс итерации, ω — параметр взвешивания. Коэффициент сглаживания μ характеризует эффективность сглаживателя в уменьшении амплитуды высокочастотных компонент ошибки и определяется как максимальный коэффициент усиления высокочастотных мод (то есть мод с $\theta \in [-\pi, \pi) \setminus [-\pi/2, \pi/2)$) после применения одной итерации метода. Оптимальное значения ω должно выбираться из соображений минимизации $\mu(\omega)$:

$$\mu(\omega) = \max_{\theta\in[-\pi,\pi)\setminus[-\pi/2,\pi/2)} \left|1 - \frac{\omega}{1+\sigma^2/2}\left(1+\sigma^2\sin^2\theta\right)\right|, \tag{9}$$

что даёт:

$$\omega_{opt} = \arg\min_{\omega} \mu(\omega) = 1, \quad \mu(\omega_{opt}) = \frac{\sigma^2}{\sigma^2+2} \xrightarrow{\sigma\to\infty} 1. \tag{10}$$

Таким образом, применение итераций метода Якоби со взвешиванием не позволяют получить эффективную процедуру сглаживания ошибки. Однако, рассмотрение гладкости ошибки отдельно для четных и нечетных узлов сетки дает иную картину. После переупорядочивания неизвестных $(h_1, h_2, \ldots, h_N)^T \to (h_1, h_3, \ldots, h_{N-1}, h_2, h_4, \ldots h_N)^T$ матрица системы приобретает блочно-диагональную структуру с двумя независимыми блоками, спектры которых описываются выражением:

$$\tilde{\lambda} = 1 + \sigma^2 \sin^2\frac{\theta}{2} \tag{11}$$

Оптимизация коэффициента сглаживания для высокочастотной части спектра (11) дает:

$$\omega_{opt} = \arg\min_{\omega} \max_{\theta\in[-\pi,\pi)\setminus[-\pi/2,\pi/2)} \left|1 - \frac{\omega}{1+\sigma^2/2}\left(1+\sigma^2\sin^2\frac{\theta}{2}\right)\right| = \frac{2\sigma^2+4}{3\sigma^2+4}, \tag{12}$$

$$\mu(\omega_{opt}) = \frac{\sigma^2}{3\sigma^2+4} \xrightarrow{\sigma\to\infty} \frac{1}{3}. \tag{13}$$

Следовательно, хотя взвешенные итерации Якоби (8) с $\omega = \frac{2\sigma^2+4}{3\sigma^2+4}$ не сглаживают ошибку в классическом смысле, они эффективно сглаживают нечетные и четные компоненты ошибки по отдельности. На рисунке 1 показано поведение компонент ошибки после применения нескольких итераций метода Якоби. Верхняя панель демонстрирует компоненты погрешности во всех узлах сетки, тогда как нижние панели отображают погрешности отдельно для нечетных и четных узлов.

Таким образом, для сохранения эффективности многосеточного метода необходимо использовать операторы перехода между сетками, учитывающие, что геометрическая гладкость решения наблюдается отдельно для четных и нечетных узлов. Такой

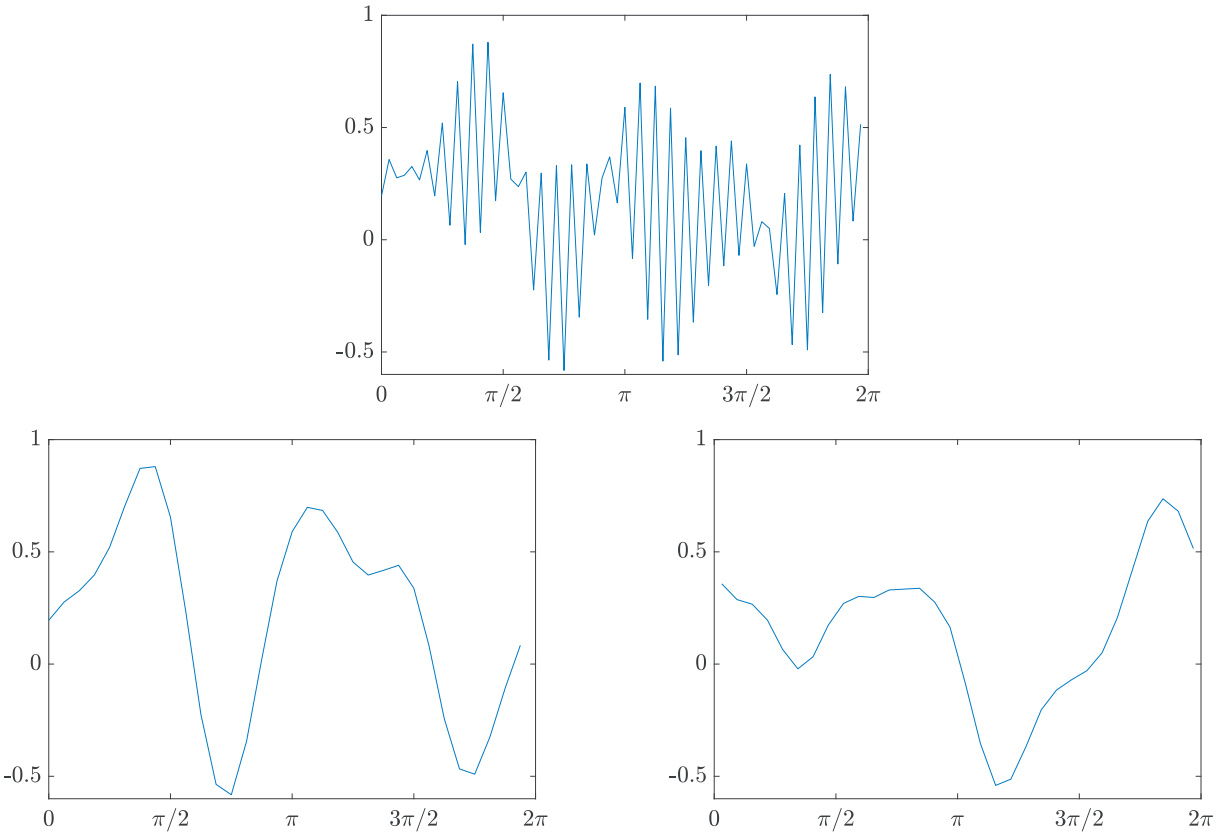


Рис. 1: Компоненты ошибки после применения нескольких итераций метода Якоби (8). На верхнем графике показаны компоненты погрешности во всех узлах сетки, на нижнем левом графике — в нечетных узлах, на нижнем правом графике — в четных узлах

подход обеспечит характеристики сходимости, аналогичные стандартному методу. Предлагается следующий подход для построения таких операторов. При стандартной схеме сгрубления сетки $\Delta x \to 2\Delta x$, узлы как на подробной, так и на грубой сетках разделяются на четные и нечетные подгруппы. Поскольку ошибка является гладкой внутри каждой подгруппы, оператор сгрубления должен использовать только точки из той же подгруппы при переносе информации на грубую сетку. Схематичное изображение возможного шаблона зависимостей между значениями в узлах сеток представлено на рисунке 2.

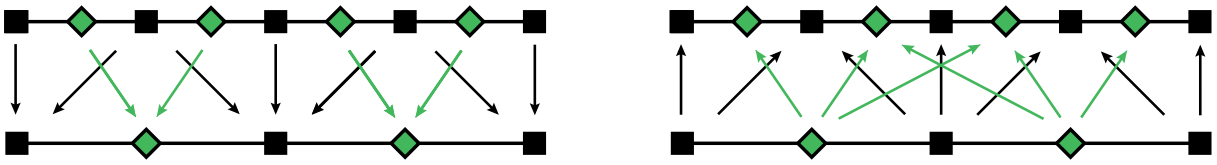


Рис. 2: Схематическое представление взаимосвязи между точками подробной и грубой сеток.

Как видно из рисунка 2, для построения оператора сгрубления можно использовать комбинацию известных стандартных операторов сгрубления: для нечетных точек использовать оператор полного взвешивания (full-weighting), а для четных узлов использовать осреднение по двум ближайшим (четным) точкам. Для оператора продолжения можно воспользоваться линейной интерполяцией. В таком случае мат-

рица оператора сгрубления R задаётся как:

$$(R\psi)_i = \begin{cases} \frac{1}{4}\left(\psi_{2i-3} + 2\psi_{2i-1} + \psi_{2i+1}\right), & \text{if } i \text{ is odd,} \\ \frac{1}{2}\left(\psi_{2i} + \psi_{2i+2}\right), & \text{otherwise.} \end{cases} \tag{14}$$

Выражение для матрицы оператора продолжения P:

$$(P\psi)_i = \begin{cases} \psi_{\frac{i+1}{2}}, & \text{if } (i-1) \bmod 4 = 0, \\ \frac{1}{2}\left(\psi_{\frac{i-1}{2}} + \psi_{\frac{i+3}{2}}\right), & \text{if } (i+1) \bmod 4 = 0, \\ \frac{1}{4}\left(3\psi_{\frac{i+2}{2}} + \psi_{\frac{i-2}{2}}\right), & \text{if } (i+2) \bmod 4 = 0, \\ \frac{1}{4}\left(3\psi_{\frac{i}{2}} + \psi_{\frac{i+4}{2}}\right), & \text{if } i \bmod 4 = 0. \end{cases} \tag{15}$$

2.2. Обобщение на двумерный случай

Представленный подход естественным образом обобщается на двумерный случай для логически прямоугольных сеток. Двумерный аналог системы уравнений (6) имеет вид:

$$h_{i,j} - \Delta t^2 g H \left(\frac{h_{i+2,j} - 2h_{i,j} + h_{i-2,j}}{4\Delta x^2} + \frac{h_{i,j+2} - 2h_{i,j} + h_{i,j-2}}{4\Delta y^2} \right) = q_{i,j}, \tag{16}$$
$$i \in [1, N_x],\ j \in [1, N_y],$$

где Δx, Δy — шаги сетки вдоль направлений, а N_x, N_y количество точек в каждом направлении. Такая аппроксимация приводит к аналогичному разделению решения на четыре независимые подсистемы в зависимости от четности индексов по обоим направлениям. При применении итераций сглаживания геометрическая гладкость наблюдается только внутри каждой подгруппы точек. На рисунке 3 изображено разделение узлов сетки на четыре независимые группы.

Рис. 3: Разделение точек на четыре несвязанные группы для двумерной подробной (левая панель) и грубой (правая панель) сетки.

Для построения операторов перехода между сетками в двумерном случае необходимо обеспечить, чтобы эти операторы использовали только точки из одной подгруппы. Такие операторы могут быть построены используя тензорное произведения одномерных операторов. Двумерный оператор сгрубления строится как:

$$R_{xy} = R_x \otimes R_y,$$

где R_x, R_y — одномерные операторы сгрубления. Аналогично, оператор продолжения P_{xy} определяется как:

$$P_{xy} = P_x \otimes P_y,$$

где P_x и P_y — одномерные операторы продолжения.

3. Модельное уравнение и компоненты метода

Для исследования свойств предложенного метода рассматривается решение модельного эллиптического уравнения на сфере, аналогичного уравнениям, возникающим при полунеявной и явно-неявной аппроксимации уравнений мелкой воды на сфере:

$$w - \nabla \cdot (\gamma \nabla w) = f, \tag{17}$$

где w — искомая величина, f — правая часть уравнения, $\nabla\cdot$ и ∇ обозначают операторы дивергенции и градиента на сфере соответственно, а γ — параметр, определяемый числом Куранта для фазовой скорости гравитационных волн.

Данное модельное уравнение репрезентативно для задач, встречающихся в атмосферном моделировании. Для трехмерных гидростатических уравнений динамики атмосферы использование неявных аппроксимаций приводит к эллиптическому уравнению, которое может быть сведено к набору независимых двумерных уравнений, аналогичных (17). В случае негидростатических уравнений вертикальная компонента матрицы системы может быть использована как эффективный предобуславливатель для полной эллиптической задачи. Таким образом, сходимость решателя в основном определяется свойствами горизонтальной части уравнения, которая структурно аналогична (17).

С точки зрения параллельной реализации, атмосферные модели обычно используют декомпозицию области по MPI-процессам только в горизонтальном направлении из-за вертикальной структуры зависимостей данных в подсеточных параметризациях физических процессов.

3.1. Сферическая сетка и пространственная аппроксимация

Для пространственной дискретизации используется сетка типа равноугольная кубическая сфера [24, 28]. Сетка строится путем проекции граней куба на вписанную сферу, где для каждой грани проекция задается соотношением:

$$\vec{r} = \frac{1}{\sqrt{1 + \tan^2\alpha + \tan^2\beta}} \begin{pmatrix} \tan\alpha & \tan\beta & 1 \end{pmatrix}^T, \tag{18}$$

где $\vec{r}$ — радиус-вектор в декартовых координатах, $\alpha,\ \beta \in [-\pi/4,\ \pi/4]$ угловые координаты на грани куба.

Вводится криволинейная система координат с ковариантными базисными векторами и метрическим тензором:

$$\vec{a}_\alpha = \frac{\partial \vec{r}}{\partial \alpha}, \quad \vec{a}_\beta = \frac{\partial \vec{r}}{\partial \beta}, \quad Q_{lm} = \vec{a}_l \cdot \vec{a}_m,\ l, m \in \{\alpha, \beta\}. \tag{19}$$

Операторы градиента и дивергенции определяются как:

$$\nabla\psi = \frac{\partial \psi}{\partial \alpha}\vec{a}^\alpha + \frac{\partial \psi}{\partial \beta}\vec{a}^\beta, \quad \nabla \cdot \vec{v} = \frac{1}{J}\left(\frac{\partial J v^\alpha}{\partial \alpha} + \frac{\partial J v^\beta}{\partial \beta}\right), \tag{20}$$

где $J = \sqrt{\det Q}$ — якобиан преобразования координат, $\vec{a}^\alpha$, $\vec{a}^\beta$ — контравариантные базисные векторы, v^α, v^β — контравариантные компоненты $\vec{v}$.

Координаты узлов сетки на каждой панели задаются формулой:

$$(\alpha_i, \beta_j) = \left(-\frac{\pi}{4} + (i-1)\Delta, -\frac{\pi}{4} + (j-1)\Delta\right), \quad i, j \in [1, N+1], \quad \Delta = \frac{\pi}{2N}, \tag{21}$$

где N — количество узлов по каждому координатному направлению.

Пространственная дискретизация основана на методе конечных разностей, удовлетворяющих свойству суммированием по частям (summation-by-parts finite differences, SBP FD) [8, 15, 21, 35]. В работе используются операторы градиента и дивергенции второго и четвертого порядков точности. Используемый подход обеспечивает симметричность и отрицательную полуопределенность дискретного оператора Лапласа. Подробное описание используемой дискретизации приведено в работе [32].

Дискретизация тестовой задачи (17) приводит к СЛАУ:

$$(E - \gamma DG)\mathrm{w} = \mathrm{f}, \tag{22}$$

где E — единичная матрица, w, f — сеточные функции.

3.2. Компоненты геометрического многосеточного метода

В данном разделе описываются компоненты геометрического многосеточного метода, используемого в численных экспериментах. Матрицы и соответствующие величины на двух последовательных уровнях многосеточного метода обозначаются с использованием подстрочного индекса c (грубая сетка) и f (подробная сетка).

Построение грубых сеток. Предполагается, что количество узлов вдоль каждого ребра грани куба на самом детальном уровне задается как $N = 2^L k + 1$, где L — общее количество уровней многосеточной иерархии, а k — положительное целое число. На уровне l количество узлов вдоль ребра определяется как $N_l = 2^{L-l+1}k + 1$, а координаты узлов вычисляются согласно формуле (21) с $N = N_l$.

Операторы перехода между сетками. Операторы сгрубления (restriction) и продолжения (prolongation) строятся на основе подхода, описанного в разделе 2.

Оператор сгрубления задается формулой:

$$\mathrm{q}_c = \mathrm{J}_c^{-1}\mathrm{R}\mathrm{J}_f\mathrm{q}_f, \tag{23}$$

где $\mathrm{J}_f \in \mathbb{R}^{6N_f^2 \times 6N_f^2}$ and $\mathrm{J}_c \in \mathbb{R}^{6N_c^2 \times 6N_c^2}$ — диагональные матрицы с значениями якобиана отображения на подробной и грубой сетках соответственно, $\mathrm{R} = E_6 \otimes R \otimes R \in \in \mathbb{R}^{6N_c^2 \times 6N_f^2}$, $E_6 \in \mathbb{R}^{6\times 6}$ — единичная матрица, а R — матрица одномерного оператора сгрубления (14).

Оператор продолжения $\mathrm{P} \in \mathbb{R}^{6N_f^2 \times 6N_c^2}$ определяется как:

$$\mathrm{P} = E_6 \otimes P \otimes P \tag{24}$$

где P задается формулой (15) с модификацией для компонент $i = 2$ и $i = N_f - 1$, где используется интерполяция по ближайшим соседям вместо линейной интерполяции.

Матрицы системы на грубых сетках. Матрицы операторов на грубых сетках строятся путем прямой дискретизации исходного оператора на соответствующем уровне [37], что предпочтительнее метода Галёркина из-за меньших вычислительных затрат и меньшего количества межпроцессорных обменов.

Сглаживающие итерации и решатель на грубой сетке. Используются трехчленные итерации Чебышева, применяемые как для сглаживания, так и для решения на грубой сетке. Алгоритм для СЛАУ $Ax = b$ может быть записан как:

$$\theta = \frac{1}{2}(\lambda_{max} + \lambda_*),\ \delta = \frac{1}{2}(\lambda_{max} - \lambda_*),\ \sigma = \frac{\theta}{\delta},\ \rho_0 = \frac{1}{\sigma} \tag{25}$$

$$r^0 = b - Ax^0, \quad d^0 = \frac{1}{\theta}r^0 \tag{26}$$

$$x^{k+1} = x^k + d^k, \tag{27}$$

$$r^{k+1} = r^k - Ad^k,\ \rho_{k+1} = \frac{1}{2\sigma - \rho_k} \tag{28}$$

$$d^{k+1} = \rho_{k+1}\rho_k d^k + \frac{2\rho_{k+1}}{\delta}r^{k+1} \tag{29}$$

где параметр $\lambda_* \in [\lambda_{min}, \lambda_{max}]$ определяет часть спектра $[\lambda_*, \lambda_{max}]$ для которой оптимизируется скорость сходимости. Для решения системы на самой грубой сетке $\lambda_* = \lambda_{min}$ (в нашем случае $\lambda_{min} = 1$), в то время как для итераций сглаживания λ_* выбирается как максимальное собственное значение, соответствующее собственным векторам, разрешаемым на грубой сетке [1]. Для модельных задач значение параметра λ_* можно определить аналитически. В нашем случае $\lambda_* = \kappa\lambda_{max}$ подбирается эмпирически путем оптимизации сходимости многосеточного метода. Максимальные собственные значения матриц систем предварительно вычисляются с использованием алгоритма степенного метода с динамическим ускорением [4].

3.3. Параллельная реализация

Предложенный алгоритм реализован и протестирован в рамках модели динамики атмосферы нового поколения, разрабатываемой в Институте вычислительной математики им. Г.И. Марчука РАН и Гидрометцентре России [33]. Основные концепции и принципы проектирования программного комплекса модели подробно описаны в работе[33], тогда как аспекты параллельной реализации модели рассмотрены в работе [31]. Организация кода для реализации линейных решателей, включая описанный в данной работе многосеточный метод, представлена в публикации [11].

Модель реализована на языке Fortran с использованием технологии Message Passing Interface (MPI) для параллельных вычислений. Параллелизация достигается за счет двумерного разбиения вычислительной области на логически прямоугольные подобласти, распределяемые между отдельными MPI-процессами. Все операторы модели вычисляются параллельно с применением стандартного подхода, использующего гало-зоны и гало-обмены данными между соседними процессами.

4. Численные эксперименты

4.1. Исследование сходимости

В данном разделе исследуется скорость сходимости предложенного подхода в зависимости от нескольких параметров: порядка аппроксимации пространственной схемы (рассмотрены схемы второго и четвертого), количества итераций предсглаживания (ν_{pre}) и постсглаживания (ν_{post}), а также параметра γ/Δ^2 в уравнении (17).

Правая часть уравнений задавалась случайными векторами, умноженными на матрицу СЛАУ. Результаты усреднены по 100 запускам с нулевым начальным приближением. Тесты проводились на кубической сфере с $N = 256$ узлами.

Рассматриваются задачи с тремя различными значения γ: $\frac{\sqrt{\gamma}}{\Delta} = \{5,\ 10,\ 15\}$. Минимальное значение характерно для моделей с эйлеровым описанием адвекции, максимальное — для полулагранжевых алгоритмов. Количество многосеточных уровней выбиралось как $L = \{2, 3, 4\}$. Используется от 2 до 4 итераций пред- и постсглаживания и 4-8 итераций для решения системы на самой грубой сетке.

На рисунках 4 и 5 представлены графики сходимости нормы относительной невязки для SBP FD аппроксимаций второго и четвертого порядка. Результаты демонстрируют высокую скорость сходимости для обоих порядков аппроксимации, при этом скорость сходимости практически не зависит от значений параметра $\sqrt{\gamma}/\Delta$. Данный результат ожидаем для геометрических многосеточных методов, где увеличение числа уровней позволяет компенсировать ухудшение обусловленности матрицы при больших значениях $\sqrt{\gamma}/\Delta$.

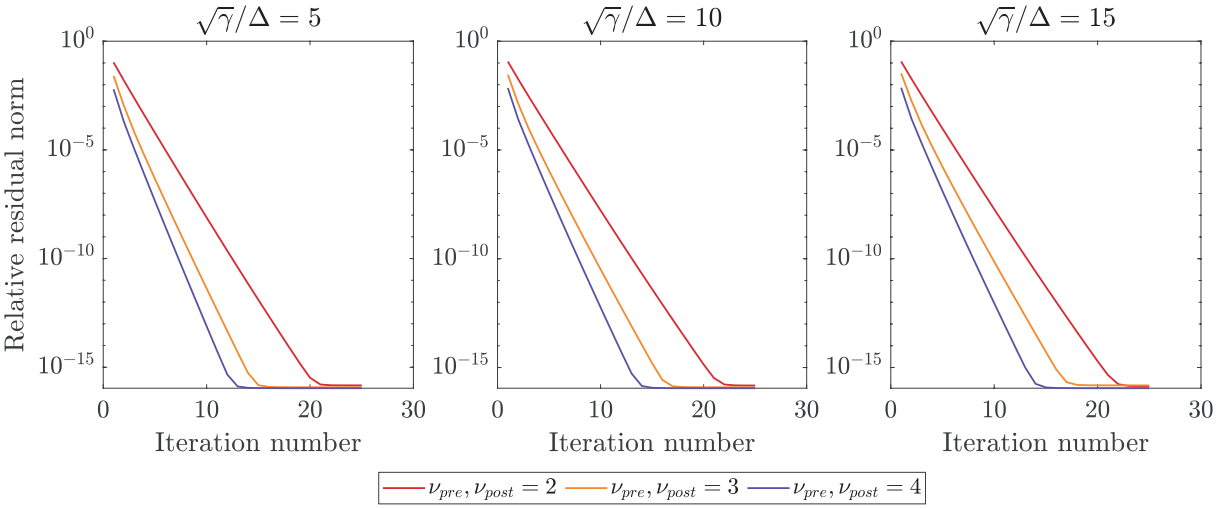


Рис. 4: Сходимость метода при различных значениях параметров $\sqrt{\gamma}/\Delta$, ν_{pre} и ν_{post} при использовании аппроксимации второго порядка точности.

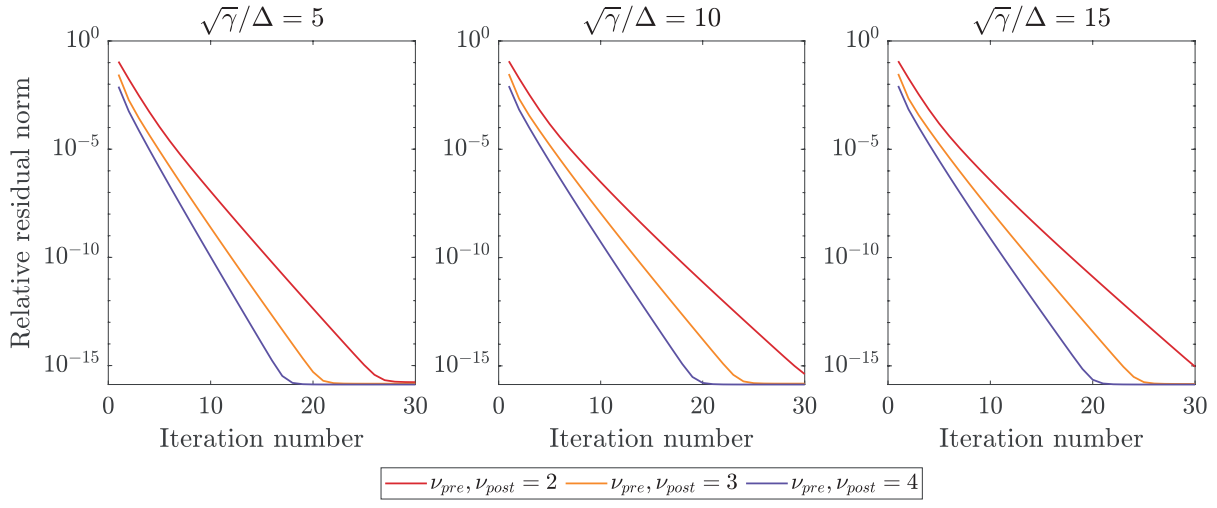


Рис. 5: Сходимость метода при различных значениях параметров $\sqrt{\gamma}/\Delta$, ν_{pre} и ν_{post} при использовании аппроксимации четвертого порядка точности.

Для сравнительного анализа на рисунке 6 представлены усредненные кривые сходимости разработанного метода для схемы второго порядка в сопоставлении с геометрическим многосеточным решателем [11], для разностной схемы второго порядка на сетке с разнесением переменных. В эксперименте использовалось значение параметра $\sqrt{\gamma}/\Delta$ и итерации Чебышева в качестве сглаживателя для обоих методов. Результаты демонстрируют сопоставимую скорость сходимости для обоих подходов, что подтверждает теоретические выводы, полученные при анализе одномерной модельной задачи в разделе 2.

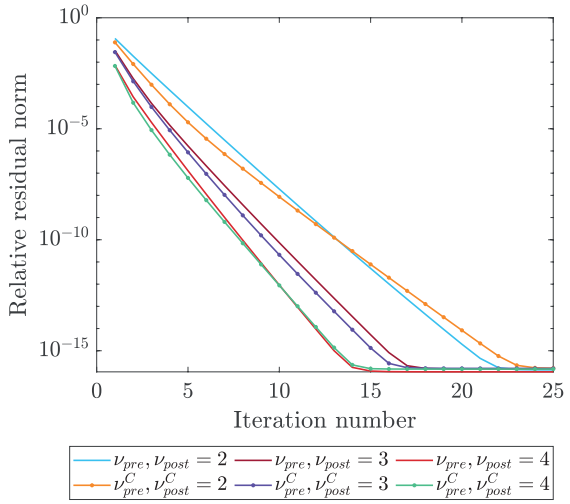


Рис. 6: Сравнение сходимости предложенного метода для аппроксимации на неразнесенной сетки со сходимостью стандартного метода [11] для аппроксимации на разнесенной сетке.

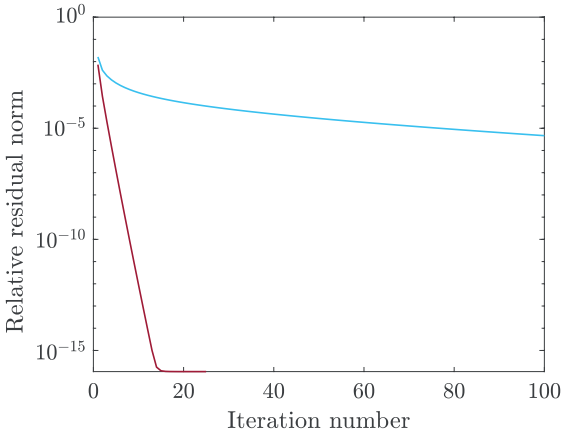


Рис. 7: Сравнение сходимости многосеточного решателя с использованием стандартных (голубая линия) и предложенных в данной работе (красная линия) операторов перехода.

Кроме этого были проведены замеры скорости сходимости при использовании стандартных операторов перехода между сетками, которые не учитывают эффектов рассогласования решения между соседними узлами сетки. Результаты для случая с $\sqrt{\gamma}/\Delta = 15$ и 4 итерациях пред- и постсглаживания приведены на рисунке 7. Из рисунка видно, что применение стандартных операторов перехода приводит к существенному замедлению скорости сходимости метода, что еще раз подтверждает необходимость применения специализированных операторов перехода между сетками.

4.2. Параллельная эффективность

Далее приводятся результаты исследования параллельной масштабируемости разработанного метода. В рамках тестирования решалась система уравнений (17) на сетках различного разрешения: $C288$ ($288 \times 288 \times 6$ узлов), $C576$ ($576 \times 576 \times 6$) и $C1152$ ($1152 \times 1152 \times 6$), где последний вариант соответствует горизонтальному разрешению около 9 км и содержит приблизительно $8 \cdot 10^6$ узлов. Для формирования правой части системы использовались случайные векторы, умноженные на матрицу системы. Измерения времени выполнения проводились для серии из 100 последовательных решений с нулевым начальным приближением, при этом критерием остановки служило уменьшение относительной нормы невязки в 10^{-5} раз. Рассматривается две конфигурации многосеточного метода: $L = 2$, $\nu_{pre} = \nu_{post} = 4$, $\nu_{coarse} = 8$ and $L = 3$, $\nu_{pre} = \nu_{post} = 4 = \nu_{coarse} = 4$.

Вычислительные эксперименты выполнялись на суперкомпьютерной системе Cray XC40, установленной в Главном вычислительном центре Росгидромета. Каждый вычислительный узел содержит два 18-ядерных процессора Intel Xeon E2697v4 с 45 МБ кэш-памяти третьего уровня. Использовался компилятор Intel Fortran версии 19.1.3.304.

На рисунках 8 и 9 представлены замеры времени решения при различных размерах стеки при использовании аппроксимации второго и четвертого порядка соответственно. Сплошные черные линии соответствуют идеальному линейному ускорению, а пунктирные линии соединяют результаты с одинаковым числом узлов сетки, приходящихся на MPI-процесс (слабая масштабируемость). Оба метода демонстрируют сверхлинейное ускорение, обусловленное более оптимальным использованием кэш-памяти при уменьшении числа узлов сетки на вычислительное ядро.

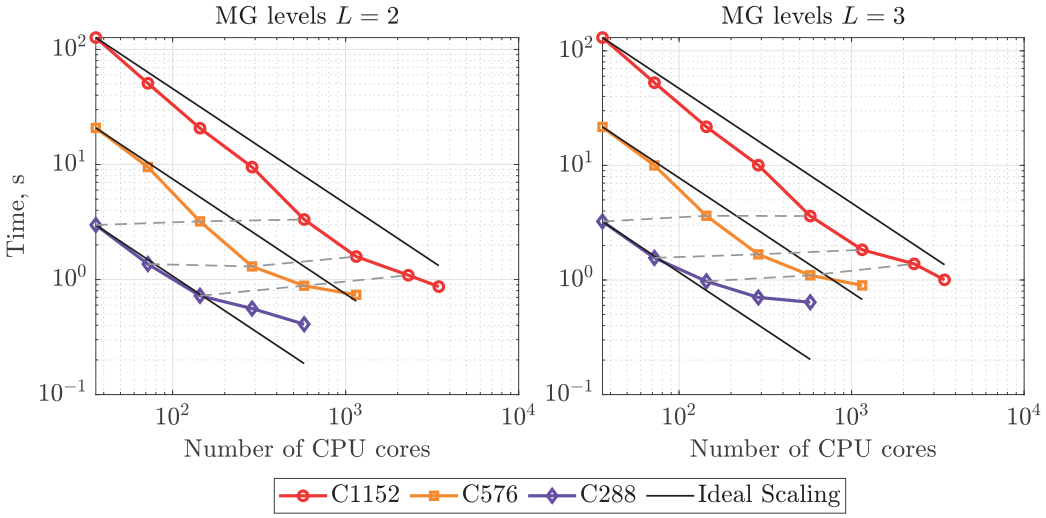


Рис. 8: Зависимость времени решения от количества MPI-процессов для сеток различного разрешения при использовании аппроксимации второго порядка точности.

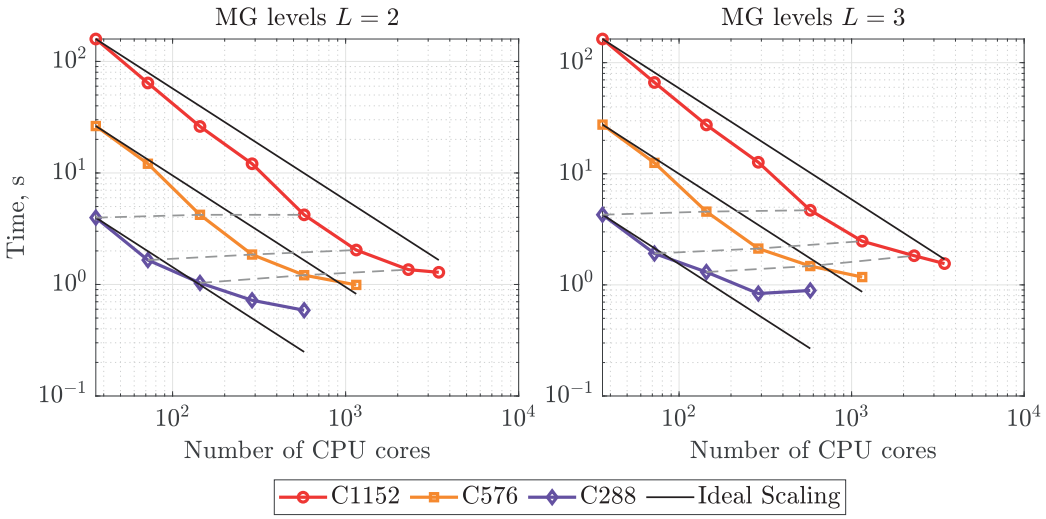


Рис. 9: Зависимость времени решения от количества MPI-процессов для сеток различного разрешения при использовании аппроксимации четвертого порядка точности.

В целом, конфигурации с меньшим количеством уровней демонстрируют более высокую эффективность, что объясняется снижением степени параллелизма на грубых сетках из-за уменьшения числа степеней свободы на вычислительное ядро.

Сравнение схем второго и четвертого порядков точности показывает схожую масштабируемость, с несколько более быстрым падением эффективности для схемы четвертого порядка, что объясняется более широким шаблоном схемы. Возможным направлением для компенсации этого эффекта является использование схем более низкого порядка для итераций сглаживания сглаживания на грубых уровнях сетки [17].

Для задачи на сетке $C288$ ускорение наблюдается при использовании до 576 MPI-процессов, что соответствует 864 узлам сетки на MPI-процесс. Эти результаты хорошо согласуются с предыдущими исследованиями масштабируемости многосеточного метода для разнесенных сеток [11]. Наблюдаемое насыщение масштабируемости обусловлено снижением эффективности на грубых уровнях многосеточного метода.

5. Заключение

В данной работе представлен геометрический многосеточный метод, разработанный для решения систем линейных алгебраических уравнений, возникающих при пространственной дискретизации на неразнесенных сетках уравнений динамики атмосферы. Ключевой особенностью метода является использование специализированных операторов перехода между сетками, учитывающих характерное для неразнесенных схем рассогласование решений в соседних узлах сетки.

На основе проведенных численных экспериментов в рамках модельного уравнения (17) на сетке кубическая сфера с тестовой задачей (17), показано, что разработанный метод обеспечивает высокую скорость сходимости для аппроксимаций как второго, так и четвертого порядка точности при использовании различных конфигураций задачи. Реализованный метод демонстрирует эффективную масштабируемость.

Перспективным направлением исследований является внедрение и тестирование разработанного метода в гидростатических и негидростатических моделях атмосферной динамики, создаваемых совместно Институтом вычислительной математики им. Г.И. Марчука РАН и Гидрометцентром России [33].

Предложенная методика имеет потенциал для применения в других областях, где использование неразнесенных сеток приводит к аналогичным проблемам рассогласования решений. В частности, метод может быть полезен при численном решении уравнений Навье-Стокса в задачах моделирования турбулентности. Важно отметить, что разработанный подход к построению операторов перехода между сетками не ограничивается прямоугольными сетками и может быть адаптирован для других типов дискретизации (треугольных, икосаэдрических и др.) путем учета структуры рассогласования решений на таких сетках.

Список литературы

[1] Adams M., Brezina M., Hu J., and Tuminaro R. Parallel multigrid smoothing: polynomial versus gauss–seidel // Journal of Computational Physics, 188(2):593–610, 2003.

[2] Arakawa A. and Lamb V. Computational design of the basic dynamical processes of the UCLA general circulation model, volume 17, pages 173–265. New York: Academic Press, 1977.

[3] Ascher U.M., Ruuth S.J., and Spiteri R.J. Implicit-explicit Runge-Kutta methods for time-dependent partial differential equations // Appl. Numer. Math., 25(2–3):151–167, November 1997.

[4] Austin C., Pollock S., and Zhu Y. Dynamically accelerating the power iteration with momentum // Numerical Linear Algebra with Applications, 31(6):e2584, 2024.

[5] Brandt A. Stages in developing multigrid solutions // Numerical Methods for Engineering, pages 23–44, 1980.

[6] Briggs W.L., Henson V.E., and McCormick S.F. A multigrid tutorial. SIAM, 2000.

[7] Buckeridge S. and Scheichl R. Parallel geometric multigrid for global weather prediction // Numerical Linear Algebra with Applications, 17(2-3):325–342, 2010.

[8] Del Rey Fernández D.C., Hicken J.E., and Zingg D.W. Review of summation-by-parts operators with simultaneous approximation terms for the numerical solution of partial differential equations // Computers & Fluids, 95:171–196, 2014.

[9] Fedorenko R.P. A relaxation method for solving elliptic difference equations // USSR Computational Mathematics and Mathematical Physics, 1(4):1092–1096, 1962.

[10] Gillard M., Szmelter J., and Cocetta F. Preconditioning elliptic operators in high-performance all-scale atmospheric models on unstructured meshes // Journal of Computational Physics, 520:113503, 2025.

[11] Goyman G. and Shashkin V. Implementation of elliptic solvers within ParCS parallel framework // Communications in Computer and Information Science, 1510:137–147, 2021.

[12] Goyman G.S. and Shashkin V.V. Horizontal approximation schemes for the staggered reduced latitude-longitude grid // Journal of Computational Physics, 434:110234, 2021.

[13] Heikes R.P., Randall D.A., and Konor C.S. Optimized icosahedral grids: Performance of finite-difference operators and multigrid solver // Monthly Weather Review, 141(12):4450–4469, 2013.

[14] Jablonowski C. and Williamson D.L. The pros and cons of diffusion, filters and fixers in atmospheric general circulation models // Numerical techniques for global atmospheric models, pp. 381–493, 2011.

[15] Kreiss H.-O. and Scherer G. Finite element and finite difference methods for hyperbolic partial differential equations // in Mathematical aspects of finite elements in partial differential equations, pages 195–212. Elsevier, 1974.

[16] Linden J., Steckel B., and Stüben K. Parallel multigrid solution of the navier-stokes equations on general 2d domains // Parallel Computing, 7(3):461–475, 1988.

[17] Liu C. and Henshaw W. Multigrid with nonstandard coarse-level operators and coarsening factors // Journal of Scientific Computing, 94(3):58, 2023.

[18] Maynard C., Melvin T., and Müller E.H. Multigrid preconditioners for the mixed finite element dynamical core of the lfric atmospheric model // Quarterly Journal of the Royal Meteorological Society, 146(733):3917–3936, 2020.

[19] Mengaldo G., Wyszogrodzki A., Diamantakis M., Lock S.-J., Giraldo F.X., and Wedi N.P. Current and emerging time-integration strategies in global numerical weather and climate prediction // Archives of Computational Methods in Engineering, 26:663–684, 2019.

[20] Müller E. and Scheichl R. Massively parallel solvers for elliptic partial differential equations in numerical weather and climate prediction // Quart. J. Roy. Met. Soc., 140(685):2608–2624, 2014.

[21] Olsson P. Summation by parts, projections, and stability I // Mathematics of Computation, 64(211):1035–1065, 1995.

[22] Oosterlee C.W. and Lorenz F.J.G. Multigrid methods for the stokes system // Computing in science & engineering, 8(6):34–43, 2006.

[23] Patankar S.V. and Spalding D.B. A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows // International Journal of Heat and Mass Transfer, 15(10):1787–1806, 1972.

[24] Rančić M., Purser R.J., and Mesinger F. A global shallow-water model using an expanded spherical cube: Gnomonic versus conformal coordinates // Quarterly Journal of the Royal Meteorological Society, 122(532):959–982, 1996.

[25] Rhie C.M. and Chow W.-L. Numerical study of the turbulent flow past an airfoil with trailing edge separation // AIAA journal, 21(11):1525–1532, 1983.

[26] Robert A. Integration of a spectral model of the atmosphere by the implicit method // in Procs. of WMO/IUGG Symp. on Numerical Weather Prediction, 1969.

[27] Rodi W., Majumdar S., and Schönung B. Finite volume methods for two-dimensional incompressible flows with complex boundaries // Computer methods in applied mechanics and engineering, 75(1-3):369–392, 1989.

[28] Sadourny R. Conservative finite-difference approximations of the primitive equations on quasi-uniform spherical grids // Monthly Weather Review, 100(2):136–144, 1972.

[29] Sandbach S., Thuburn J., Vassilev D., and Duda M.G. A semi-implicit version of the mpas-atmosphere dynamical core // Monthly Weather Review, 143(9):3838–3855, 2015.

[30] Shashkin V., Goyman G., and Tretyak I. Summation-by-parts finite-difference method for linear shallow water equations on staggered curvilinear grids in closed domains. arXiv preprint arXiv:2501.07323, 2025.

[31] Shashkin V. and Goyman G. Parallel efficiency of time-integration strategies for the next generation global weather prediction model // in Russian Supercomputing Days, pages 285–296. Springer, 2020.

[32] Shashkin V.V., Goyman G.S., and Tolstykh M.A. Summation-by-parts finite-difference shallow water model on the cubed-sphere grid. Part I: Non-staggered grid // Journal of Computational Physics, 474:111797, 2023.

[33] Shashkin V.V., Goyman G.S., and Tretyak I.D. Development of the next-generation atmosphere dynamics model in russia: Current state and prospects // Lobachevskii Journal of Mathematics, 45(7):3159–3172, 2024.

[34] Staniforth A. and Thuburn J. Horizontal grids for global weather and climate prediction models: a review // Quart. J. Roy. Met. Soc., 138:1 — 26, 2012.

[35] Strand B. Summation by parts for finite difference approximations for d/dx // Journal of Computational Physics, 110(1):47–67, 1994.

[36] Tolstykh M., Goyman G., Fadeev R., and Shashkin V. Structure and algorithms of sl-av atmosphere model parallel program complex // Lobachevskii Journal of Mathematics, 39:587–595, 2018.

[37] Trottenberg U., Oosterlee C.W., and Schuller A. Multigrid. Academic Press, 2000.

[38] Varga R.S. Iterative analysis. New Jersey, 322, 1962.

Исследование масштабируемости параллельного алгоритма численного моделирования удержания плазмы в открытых магнитных ловушках методами имитационного моделирования[1]

И.Г. Черных, Д.В. Винс, В.А. Вшивков, М.А. Боронина

Институт вычислительной математики и математической геофизики СО РАН (ИВМиМГ СО РАН)

В этой статье рассматривается методика оценки масштабируемости параллельных алгоритмов с помощью имитационного моделирования. Вместо проведения дорогостоящих реальных экспериментов на суперкомпьютерах авторы предлагают использовать специальную имитационную модель. Методика заключается в изучении и разработке схемы коммуникаций и вычислений исследуемого алгоритма, создании на ее основе имитационной модели используя модель Акторов, настройке этой модели под конкретную архитектуру суперкомпьютера и исследовании его масштабируемости. В данной статье показано применение этой методики к алгоритму численного моделирования удержания плазмы в открытых магнитных ловушках. Получены данные, говорящие о 85% эффективности масштабирования алгоритма на сотни тысяч вычислительных ядер.

Ключевые слова: физика плазмы, параллельные алгоритмы, имитационное моделирование.

1. Введение

Современные и перспективные суперкомпьютерные системы демонстрируют экстремальный уровень параллельной обработки данных, гетерогенность архитектурных решений и иерархически распределённую организацию коммуникационных подсистем, что обусловливает значительную сложность разработки специализированного программного обеспечения, представляющую собой актуальную научно-техническую проблему.

Ключевым вызовом после реализации программного комплекса для решения вычислительно сложных задач становится определение точки насыщения масштабируемости — установление оптимального количества вычислительных узлов, при котором достигается максимальное ускорение вычислений (speedup) для заданной архитектуры и входных параметров, после чего добавление вычислительных ресурсов не приводит к значимому росту производительности (закон Амдала).

[1]Работа выполнена при поддержке Российского научного фонда (проект № 19-71-20026).

Экспериментальное решение данной проблемы требует проведения серии вычислительных экспериментов с вариацией количества задействованных узлов. Однако высокая себестоимость эксплуатации современных суперкомпьютерных систем, а также лимитированная доступность машинного времени в большинстве суперкомпьютерных центров существенно ограничивают возможность применения данного подхода на практике.

В контексте интегральной подхода к разработки параллельных программ, разработанного в ИВМиМГ СО РАН [1], предлагается решение указанной проблемы посредством применения методов мультиагентного моделирования. Данный подход позволяет имитировать серию вычислительных экспериментов, существенно сокращая требования к реальным ресурсам и временным затратам, при этом обеспечивая оценку масштабируемости исследуемых алгоритмов для современных и перспективных суперкомпьютерных архитектур.

Первоначально моделирование осуществлялось с использованием системы AGNES [2], однако выявленные ограничения данной платформы обусловили необходимость перехода к модели акторов для имитации вычислений (рис. 1) [3]. По аналогии с философией объектно-ориентированного программирования, где каждый примитив рассматривается как объект, модель акторов выделяет в качестве универсальной сущности понятие «актора». Актор является вычислительной сущностью, которая в ответ на полученное сообщение может одновременно: отправить конечное число сообщений другим акторам; создать конечное число новых акторов; выбрать поведение, которое будет использоваться при обработке следующего полученного сообщения. По сути, каждая вычислительная функция программы становится черным ящиком, в который приходит некоторый набор данных в виде сообщений и из которого выходит некоторый набор данных в виде сообщений. На рис. 1 схематично представлен такой подход. Время обработки данных разными черными ящиками и времена пересылок сообщений могут быть как синтетическими, так и данными из реальных запусков исследуемой программы на различном количестве вычислительных ядер.

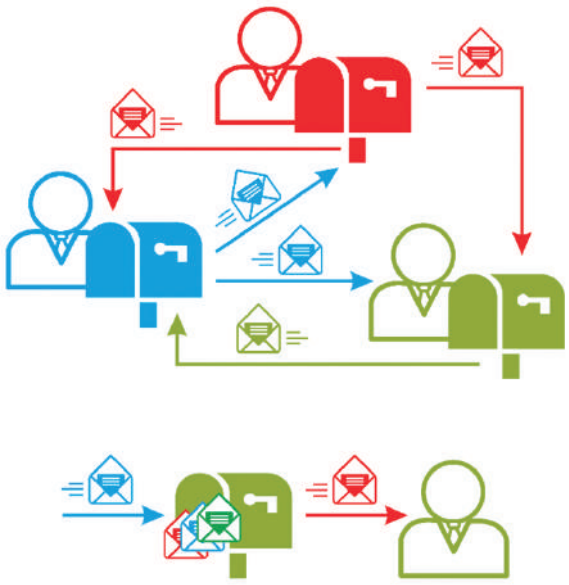


Рис. 1: Модель акторов для имитации вычислений

Как демонстрируют сравнительные исследования, данная модель обеспечивает:

- минимизацию коммуникационных накладных расходов между агентами;
- повышенную эффективность при масштабировании;
- улучшенные показатели отказоустойчивости.

Экспериментальная верификация подтвердила, что функциональный язык программирования Erlang демонстрирует оптимальные характеристики по следующим критериям:

- простота реализации моделей,
- эффективность масштабирования,
- устойчивость к отказам

что делает его предпочтительным выбором для задач имитационного моделирования масштабируемости алгоритмов [4, 5].

В данной работе будет проиллюстрировано применение такого подхода к исследованию масштабируемости алгоритма численного моделирования удержания плазмы в магнитной ловушке [6–8].

2. Общая схема исследования параллельного алгоритма

В параллельных вычислениях программа представляет собой набор одновременно работающих потоков (нитей), выполняющихся на отдельных вычислительных узлах. Эти потоки обмениваются данными через систему сообщений. Основными параметрами, определяющими работу каждой нити, являются длительность выполнения вычислений и время, затрачиваемое на передачу данных другим узлам. Для прогнозирования работы программы на суперкомпьютере строится специальная модель, которая включает создание схемы коммуникации между всеми нитями процесса (как показано на рис. 2) и разделение работы каждой нити на фазы вычислений и фазы обмена данными. Каждая нить представлена как последовательность вычислительных и коммуникационных этапов. Все взаимодействия между нитями формализованы и могут быть проанализированы. Модель позволяет оценить временные характеристики выполнения программы до её реального запуска на суперкомпьютере. Такой подход дает возможность заранее проанализировать поведение параллельной программы и оптимизировать её работу для конкретной вычислительной архитектуры.

Основным критерием эффективности вычислительного процесса при его исполнении на высокопроизводительных суперкомпьютерах — это его масштабируемость. Тут следует отметить, что для разных вычислительных алгоритмов может быть важен разный показатель масштабируемости. Например, для методов статистического моделирования [12] — это слабая масштабируемость, т.е. уменьшение времени исполнения при увеличении вычислительных узлов, а для сеточных методов численного моделирования [9, 10] — сильная масштабируемость, т.е. неизменность времени расчета алгоритма при увеличении расчетной области пропорционально количеству вычислительных узлов.

В исследовании рассматривается исключительно межузловая масштабируемость, так как в настоящее время количество ядер вычислительных элементов зафиксировано и намного меньше, чем потенциальное число вычислительных узлов высокопроизводительных суперкомпьютеров. Таким образом, все вычисления и обмены между

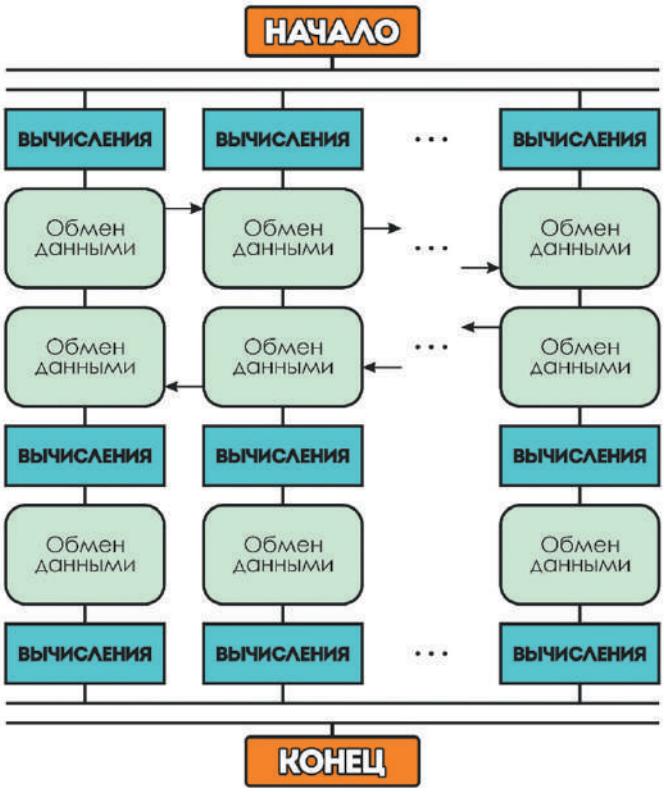


Рис. 2: Базовая схемы коммуникаций параллельного вычислительного процесса

вычислительными элементами одного вычислительного узла принимаются как величина, фиксированная и не оказывающая влияния на общую схему вычислений.

Процесс исследования масштабируемости параллельных алгоритмов можно условно разбить на несколько этапов:

- Создание упрощённой модели программы. На данном этапе происходит анализ того, как разные части программы (потоки) работают и общаются между собой. Затем группируем похожие потоки, которые делают одинаковые вычисления. Для каждой группы создаётся виртуальный “шаблон” (актор), который будет имитировать её поведение.
- Настройка виртуальных копий потоков. Каждый актор получает конкретные параметры: сколько времени занимают вычисления, как много данных нужно передавать и как быстро происходит обмен информацией.
- Запуск имитации. Акторы находят, с кем им нужно взаимодействовать и начинают работать по заданной схеме вычислений, т.е. выполняют вычисления и обмениваются сообщениями. Этот процесс продолжается, пока не выполнится заданное условие (выполнится заданное число циклов вычислений).
- Анализ результатов. На данном этапе происходит исследование полученных показателей, таких как общее время работы вычислительного процесса, время выполнения каждого актора, количество итераций цикла вычислений, совершенных каждым актором, задержки при передаче данных между акторами.

Описанный метод позволяет заранее узнать, как программа поведёт себя на реальном суперкомпьютере, помогает найти “узкие места” в работе программы, экономит время и деньги (не нужно арендовать реальный суперкомпьютер для тестов),

а также особенно полезен для разработки алгоритмов для суперкомпьютеров, которые только разрабатываются. Предлагаемый метод исследования вычислительных процессов уже был успешно использован при изучении возможности масштабирования нескольких вычислительных алгоритмов на большое число вычислительных ядер [13–16].

3. Исследование масштабируемости параллельного алгоритма динамики плазмы

3.1. Описание схемы вычислений для исследуемого алгоритма

Исходная физическая задача по исследованию динамики плазмы в магнитной ловушке описывается трехмерной гибридной моделью, где ионная компонента представляет собой набор частиц, а электронная — жидкость. В область ловушки с постоянным магнитным полем $\vec{B}_0$ и фоновой плазмой плотности n_0 в заданной точке инжектируются частицы, самосогласованное движение частиц в электромагнитных полях ловушки описывается кинетическим уравнением Власова для ионов:

$$\frac{\partial f_i}{\partial t} + \vec{v}\frac{\partial f_i}{\partial \vec{r}} + \frac{\vec{F}_i}{m_i}\frac{\partial f_i}{\partial \vec{v}} = 0,$$

где сила $\vec{F}_i$ учитывает трение между электронами и ионами. При этом плотности и скорости ионов определяются как интегралы

$$n_i(\vec{r}) = \int f_i(t, \vec{r}, \vec{v})\, d\vec{v}, \qquad \vec{V}_i(\vec{r}) = \frac{1}{n_i(\vec{r})}\int \vec{v} f_i(t, \vec{r}, \vec{v})\, d\vec{v},$$

Плазма считается квазинейтральной $n_i = n_e = n$, токи рассчитываются как $\vec{j} = e(n_i\vec{V}_i - n_e\vec{V}_e)$. Для описания электронов используются уравнений магнитной гидродинамики в безмассовом приближении:

$$-e\vec{E} - \frac{e}{c}[\vec{V}_e, \vec{B}] - \frac{\nabla p_e}{n_e} + \frac{m_e}{\tau_{ei}}(\vec{V}_i - \vec{V}_e) = 0,$$

где $p_e = n_e T_e$ — давление электронной компоненты, T_e — ее температура, при этом:

$$n_e\left(\frac{\partial T_e}{\partial t} + (\vec{V}_e\nabla)T_e\right) = (\gamma - 1)(Q_e - \operatorname{div}\vec{q}_e - p_e \operatorname{div}\vec{V}_e).$$

Электромагнитные поля описываются уравнениями Максвелла с малыми токами смещения:

$$\operatorname{rot}\vec{B} = \frac{4\pi}{c}\vec{j}, \qquad \operatorname{rot}\vec{E} = -\frac{1}{c}\frac{\partial \vec{B}}{\partial t}.$$

В начальный момент времени $\vec{B} = (0, 0, B_{0z})$, $\vec{E} = 0$, $T_i = T_0$. Задача решается с помощью метода частиц-в-ячейке (PIC-метод). На рис. 3 представлено общее описание PIC-метода, которое подробно описано в [6].

В параллельной реализации численной модели удержания плазмы в открытых магнитных ловушках используется гибридная декомпозиция расчетной области. Расчетная область делится равномерно на подобласти, за каждую подобласть отвечает

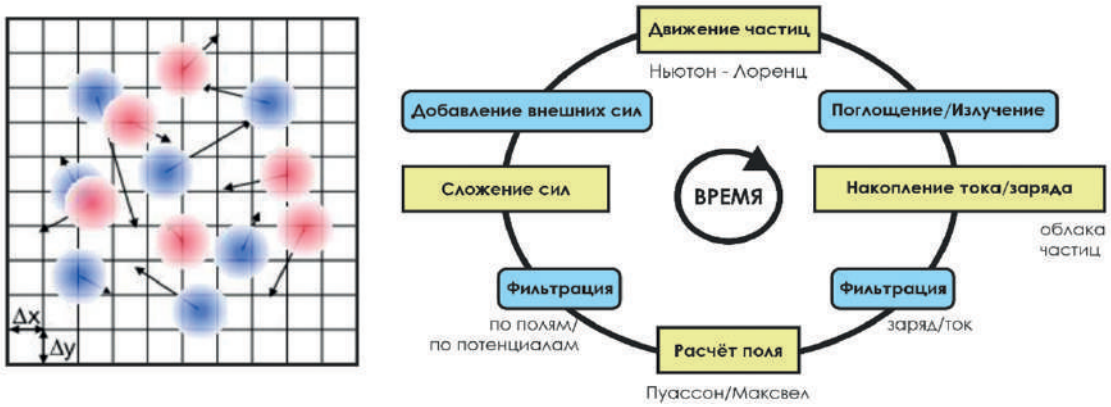


Рис. 3: Общее описание метода частиц-в-ячейках

группа процессов. Частицы в подобласти распределяются равномерно между процессами группы (рис. 4). Передача граничных узлов для подобласти узлов сетки происходит между процессами основной группы. Частицы могут быть перенаправлены в любой процесс из соседней группы.

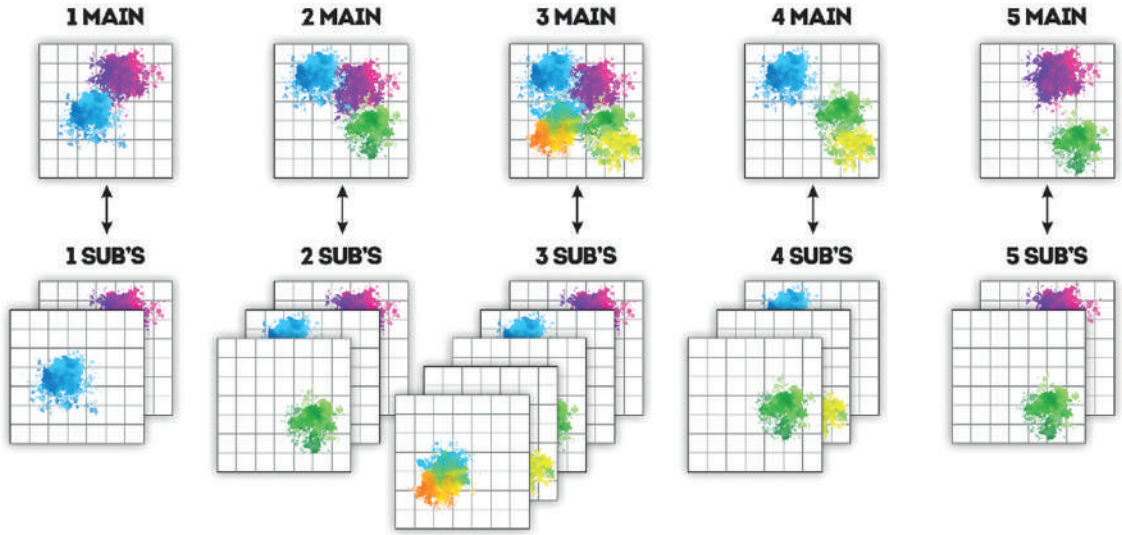


Рис. 4: Декомпозиция расчетной области в исследуемом алгоритме удержания плазмы в открытых магнитных ловушках

Каждый этап вычислительного цикла включает в себя лагранжеву стадию для вычисления скоростей и координат частиц и эйлерову стадию для вычисления на пространственной сетке.

В начале каждого шага вычислительного цикла основной процесс посылает электрическое и магнитное поле в подобласти через группу, используя MPI_BCAST.

Чтобы перейти к лагранжевой стадии, мы применяем билинейную интерполяцию, вычисляем в каждом процессе силу Лоренца, действующую на частицу в ее местоположении, и вычисляем скорости и координаты частиц. Если частица вылетает за пределы подобласти, то ее данные отправляются в один из процессов соседней группы. Использование локального ранжирования позволяет равномерно распределить частицы по процессам при отправке.

После этого каждый процесс вычисляет средние скорости ионов в ячейке и плотность, и мы переходим к этапу Эйлера. Основные процессы группы собирают данные о группе, используя MPI_REDUCE, а затем обмениваются граничными данными.

Далее вычисляются скорости электронов, параметры электрического поля, магнитного поля и температуры. После решения уравнения для каждого из перечисленных значений происходит обмен граничными узлами сетки между соседними процессорами основной группы.

Исходя из описания параллельной реализации исследуемого алгоритма была составлена схема вычислений (рис. 5), на основе которой подготовлена имитационная модель для исследования его масштабируемости.

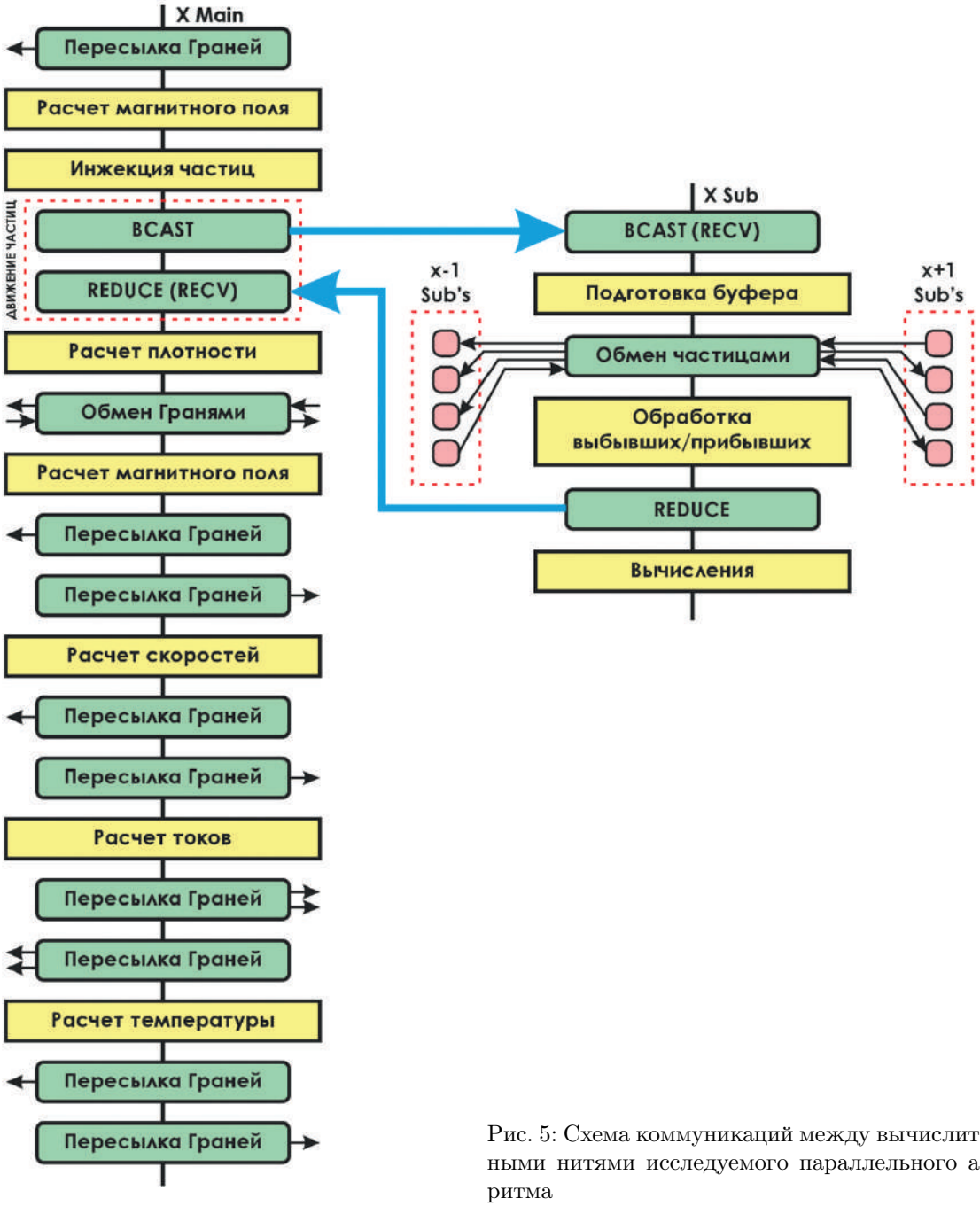


Рис. 5: Схема коммуникаций между вычислительными нитями исследуемого параллельного алгоритма

3.2. Исследование производительности алгоритма и оптимизация

Для того, чтобы провести исследование масштабируемости с использованием имитационного моделирования, необходимо набрать статистические данные по работе программы для различного количества MPI потоков. Таблица 1 показывает дан-

Таблица 1: Собираемые данные для имитационного моделирования

Процедура	Пересылки или другие подпроцедуры	Размер каждой пересылки
Магнитное поле1	Пересылка граней MPI_SEND — MPI_RECV	2* imp*lmp влево
Инжекция		
Движение частиц	MPI_BCAST	3*imp*imp*kmp_loc 3*imp*imp*kmp_loc 3*imp*imp*kmp_loc
	prepare_exchanges, подготовка для пересылок	Частицы, которые вылетают из своей части геометрии, попадают в буфер
	Пересылки частиц MPI_SEND, MPI_RECV	Размер разный
	Перераспределение по ячейкам	Заполнить пустые элементы массива из-под частиц, записать в конец принятые, процедура перевыделения памяти массивов
Плотности	MPI_Reduce	3*imp*imp*kmp_loc 3*imp*imp*kmp_loc imp*imp*kmp_loc
	Пересылки граней	6*Imp*imp туда, 6*Imp*imp обратно, 6*Imp*imp туда 6*Imp*imp обратно 2*Imp*imp туда 2*Imp*imp обратно
Магнитное поле2		
Токи	Пересылки граней	2*Imp*imp влево 2*Imp*imp вправо
Скорости электронов	Пересылки граней	2*Imp*imp влево 2*Imp*imp вправо
Электрическое поле	Пересылки граней Ex, Ey	2*Imp*imp вправо 2*Imp*imp вправо 2*Imp*imp влево 2*Imp*imp влево
	Пересылки Ez	Imp*imp влево Imp*imp влево
Температура	Пересылки граней	Imp*imp влево Imp*imp вправо

ные, собираемые для имитационного моделирования, где lmp, knp, imp — параметры групп данных.

Чтобы понимать источник падения производительности исследуемого алгоритма, мы исследуем различные показатели производительности программного кода, связанные с конкретной аппаратной архитектурой вычислительного комплекса. В нашем случае мы собирали данные с 8 до 32 MPI потоков для двухпроцессорного узла с 16 ядерными Intel Xeon 2697Av4 кластера НКС-1П ЦКП ССКЦ СО РАН. Оптимизация вычислительного кода проводилась с использованием Intel VTune [17] из пакета IntelOneAPI 2024.

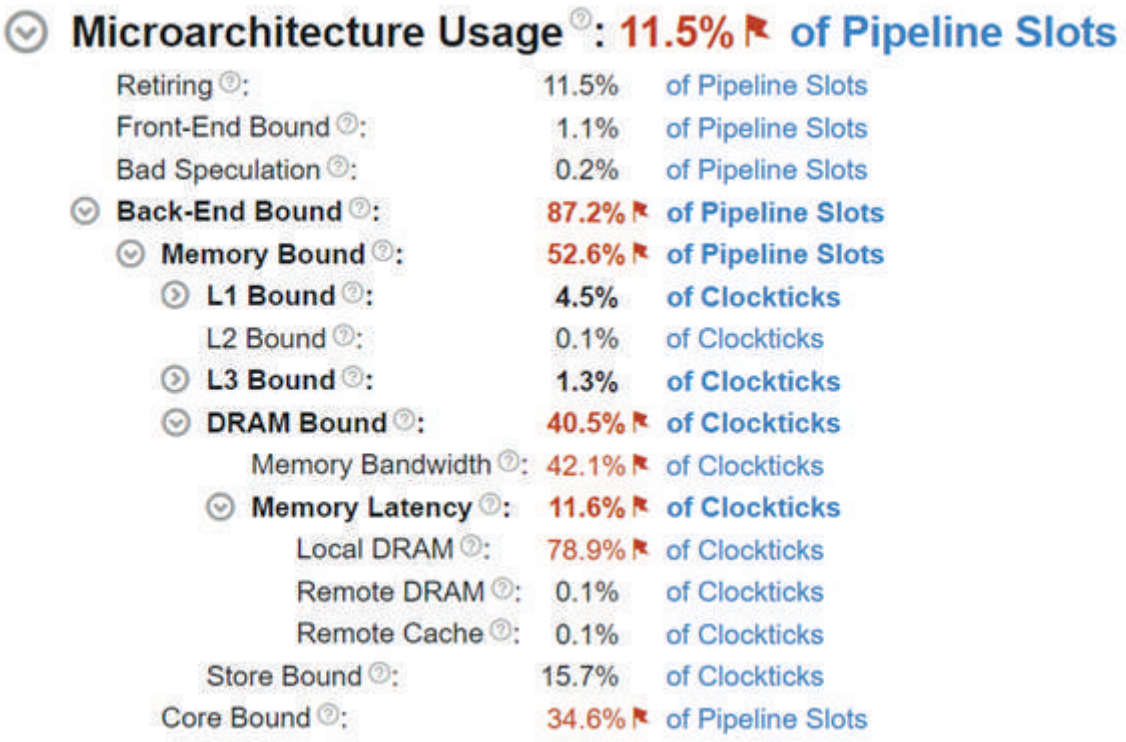


Рис. 6: Использование микроархитектуры

На рис. 6 показана метрика использования микроархитектуры. Показатель использования микроархитектуры — это ключевой показатель, который помогает оценить (в %), насколько эффективно код выполняется на текущей микроархитектуре. На использование микроархитектуры могут повлиять операции с памятью с длительной задержкой, с плавающей запятой или SIMD; неиспользуемые инструкции из-за неправильного прогнозирования ветвей. На рис. 7 показана метрика ограничения производительности кода из-за подсистемы оперативной памяти. Эта метрика показывает, как проблемы подсистемы памяти влияют на производительность. Memory Bound измеряет долю слотов, где конвейер может быть остановлен из-за инструкций по загрузке или сохранению по требованию. В нашем случае показатель $> 20\%$. Это означает, что из-за обращений к памяти, процессор простаивает в ожидании загрузки или сохранения данных. При этом нам удалось достичь неплохих показателей по использованию кэшей всех трех уровней за счет выравнивания данных и эффективного размера блоков данных, обрабатываемых за 1 такт процессора в случае использования SIMD инструкций. Что касается использования микроархитектуры, то из цифр

видно, что задержки, связанные с использованием памяти дают в целом невысокую эффективность использования вычислительных ядер. Это типичные результаты для кодов, связанных с решением задач физики плазмы. Однако, 34.6 процентов ошибок связаны с неэффективным использованием инструкций вычислительных ядер, при целевом показателе 10%. Это означает, что у нас есть хороший потенциал оптимизации кода.

Рис. 7: Ограничения, связанные с подсистемой памяти исследуемого параллельного алгоритма

Первоначальные тесты производительности показывали результаты до 30% хуже текущих. Мы подобрали правильные коэффициенты, представленные в таблице 1, чтобы данные для вычислений эффективно использовали кэш процессора, также разделили арифметические операции, чтобы облегчить компилятору Intel Fortran Compiler векторизовать код. К сожалению, у нас нет в доступе высокопроизводительных вычислительных систем, имеющих больше 50000 вычислительных ядер. Для исследования масштабируемости программы мы используем имитационное моделирование, позволяющее представить программный код в виде черных ящиков. Используя знания о задержках выполнения программы из-за памяти, сетевых устройств, мы можем построить модель исполнения программы на вычислительных системах с потенциально бесконечным количеством ядер. Следующая глава посвящена результатам такого моделирования.

3.3. Исследование масштабируемости параллельного алгоритма

Для имитации исполнения алгоритма численного моделирования динамики плазмы в открытых магнитных ловушках реализованы классы акторов PMain и PSub. Акторы класса PMain имитируют выполнения главного процесса группы — выполняет рассылку и сбор данных со всех процессов группы, а также обменивается данными с соседними акторами класса PMain. Акторы класса PSub имитируют движения частиц между группами и остальные вычисления в группе. Задержки при передаче сообщений между акторами в итоге имитируют соответствующие задержки при передаче сообщений в реальной вычислительной системе. Модель исполнения алгоритма выполняет заложенное заранее количество вычислительных циклов. Временные характеристики исполнения всех этапов вычислительных циклов взяты с результатов запуска реального вычислительного алгоритма на кластере НКС-1П ЦКП ССКЦ СО РАН.

Для демонстрации масштабируемости исследуемого алгоритма были произведены расчеты для различного числа вычислительных узлов (более 106 вычислительных ядер). Проведены исследования слабой масштабируемости вычислительного алгоритма, т.е. изменение времени выполнения задачи при увеличении параллельных

процессов (вычислительных ядер) с сохранением фиксированной вычислительной сложности в пересчёте на один процесс. Результаты данных исследований приведены на рис. 8.

Для идеального параллельного алгоритма время исполнения при этом должно меняться незначительно. Однако, стремительный рост нагрузки на систему обмена сообщениями приводит к тому, что после достижения определенного количества вычислительных ядер, их дальнейшее увеличение приведет к несопоставимому увеличению вычислительных и сетевых ресурсов, относительно выигрыша по времени. Поиск такого количества вычислительных ресурсов с помощью имитационного моделирования существенно снижает нагрузку как на разработчика вычислительных алгоритмов, так и на вычислительную инфраструктуру.

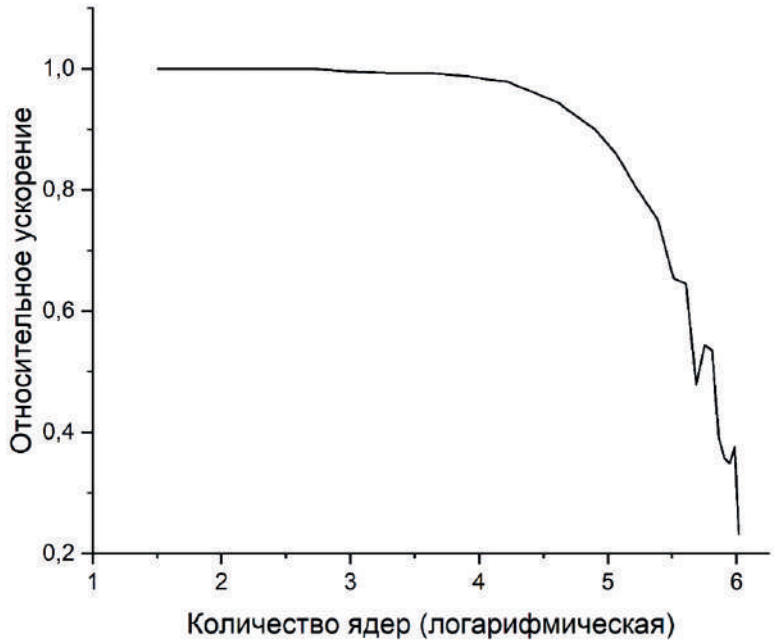


Рис. 8: Масштабируемость исследуемого алгоритма. Горизонтальная ось в логарифмическом масштабе

Из рисунка 8 можно сделать вывод, что использование в схеме вычислений главного процесса группы, который занимается рассылкой и агрегацией данных группы, и обменом данными с соседними областями существенно снижает издержки на коммуникации. Также, можно отметить, что алгоритм при исполнении на большем числе вычислительных ядер в целом показывает высокую масштабируемость, т.е. возможность существенно увеличить размер решаемой задачи.

4. Заключение

В рамках интегрального подхода к параллельному программированию разработан инновационная методика анализа масштабируемости, основанный на мультиагентной имитационной модели, принципах дискретного событийного моделирования и технологиях прогнозной аналитики. Предлагаемая методика позволяет осуществлять предиктивный анализ масштабируемости алгоритмов, минимизировать потребность в реальных вычислительных ресурсах, т.е. снизить материальные затраты на этапе

проектирования и оценивать перспективы использования алгоритмов в экзафлопсных системах. Данная методика расширяет инструментарий HPC -аналитики и позволяет избежать узких мест на этапе проектирования.

Исследован алгоритм численного моделирования динамики плазмы в аксиально-симметричных открытых магнитных ловушках. Особенностью данного алгоритма является локальность вычислительных операций, отсутствие глобальных коммуникационных операций и оптимизированная схема обмена данными. Исследована сильная масштабируемость данного алгоритма на конфигурациях до 1 млн. ядер. При расчетах задачи размерами, соответствующей 100 тысячам ядер, сохраняется эффективность 85%.

Список литературы

[1] Glinskiy B.M., Kulikov I.M., Chernykh I.G., Snytnikov A.V., Sapetina A.F., Weins D.V. The Integrated Approach to Solving Large-Size Physical Problems on Supercomputers // Supercomputing. RuSCDays 2017. CCIS. 2017. Vol. 793, pp. 278–289.

[2] Podkorytov D., Rodionov A., Choo H. Agent-based Simulation System AGNES for Networks Modeling: Review and Researching. Proc. of the 6th Int. Conference on Ubiquitous Information Management and Communication (ACM ICUIMC 2012), ISBN 978-1-4503-1172-4, pp. 115. ACM (2012) `https://doi.org/10.1145/2184751.2184883`.

[3] Weins D.V., Glinskiy B.M., Chernykh I.G. Analysis of Means of Simulation Modeling of Parallel Algorithms // RuSCDays 2018, CCIS 965, pp. 1–11, 2019.

[4] Cesarini F., Thompson S. Erlang Programming. O'Reilly Media, Inc., 2009. 498 p.

[5] Erlang Programming Language, http://www.erlang.org/, last accessed 2025/04/30.

[6] Kulikov I., Vshivkov V., Genrikh E., Weins D., Dudnikova G., Chernoshtanov I., Boronina M. Energy Efficiency of a New Parallel PIC Code for Numerical Simulation of Plasma Dynamics in Open Trap // Mathematics, 10, 3684, 2022.

[7] Boronina M.A., Chernoshtanov I.S., Chernykh I.G. et al. Three-Dimensional Model for Numerical Simulation of Beam-Plasma Dynamics in Open Magnetic Trap // Lobachevskii J. Math, 45, pp. 1–11, 2024.

[8] Chernoshtanov I.S., Chernykh I.G., Dudnikova G.I., Boronina M.A., Liseykina T.V., Vshivkov V.A. Effects observed in numerical simulation of high-beta plasma with hot ions in an axisymmetric mirror machine // Journal of Plasma Physics, 90(2):905900211, 2024.

[9] Kulikov I., Chernykh I., Glinskiy B., Weins D., Shmelev A. Astrophysics simulation on RSC massively parallel architecture // Proceedings — 2015 IEEE/ACM 15th International Symposium on Cluster, Cloud, and Grid Computing, CCGrid 2015. P. 1131–1134. ASTR Ref

[10] Glinskiy B.M., Kulikov I.M., Chernykh I.G., Snytnikov A.V., Sapetina A.F., Weins D.V. The Integrated Approach to Solving Large-Size Physical Problems on Supercomputers // RuSCDays 2017, CCIS 793, pp. 278-289, 2017

[11] Glinskiy B., Sapetina A., Martynov V., Weins D., Chernykh I. The Hybrid-Cluster Multilevel Approach to Solving the Elastic Wave Propagation Problem // Communications in Computer and Information Science, vol. 753, pp. 261-274, 2017.

[12] Glinsky B., Rodionov A., Marchenko M., Podkorytov D., Weins D. Scaling the Distributed Stochastic Simulation to Exaflop Supercomputers // Proceedings of the 14th IEEE International Conference on High Performance Computing and Communications (HPCC-2012). 2012. P. 1131–1136.

[13] Migov D.A., Weins D.V. Parallel Implementation and Simulation of Network Reliability Calculation by Monte-Carlo Method // Tomsk State University Journal of Control and Computer Science, 2019, Vol. 47, pp. 66-74.

[14] Боронина М.А., Куликов И.М., Черных И.Г., Винс Д.В. Использование комбинации схем Рое и Русанова для численного решения уравнений гидродинамики в задачах космической плазмы // Сибирский журнал индустриальной математики. — 2022. — Т. 25, № 4, стр. 14–26.

[15] Weins D., Vorobyev V., Chernykh I., Logashenko I. Development of simulation model of HPC system for Super Charm-Tau factory // Journal of Physics: Conference Series. — 2019. — V. 1336. — Article Number 012025.

[16] Particle-in-Cell Method. `https://warpx.readthedocs.io/en/22.11/theory/picsar_theory.html`, last accessed 2025/04/15.

[17] Intel VTune tool. `https://www.intel.com/content/www/us/en/developer/tools/oneapi/vtune-profiler.html`, last accessed 2025/04/30.

Многоуровневый параллелизм в программах, использующих параллельные библиотеки

В.А. Бахтин[1], Н.А. Катаев[1], А.С. Колганов[1], Д.А. Захаров[1], А.А. Смирнов[1], А.А. Малахов[2]

[1]ИПМ им. М.В. Келдыша РАН,
[2]Университет Неймарк

С ростом числа вычислительных ядер и использующих их потоков возрастают накладные расходы на их планирование, порождение и уничтожение, а объем вычислений, выполняемых каждым потоком в отдельности, сокращается. Многоуровневый параллелизм – способ обойти данную проблему. Его источником может стать распараллеливание функций, реализованных внутри библиотек, вызываемых из параллельной программы. Данный подход может потребовать дополнительной поддержки на уровне моделей параллельного программирования. В нашем исследовании мы предлагаем альтернативный способ реализации библиотеки времени выполнения OpenMP и исследуем его эффективность на примере библиотеки OpenBLAS и бенчмарка High Performance Linpack для Arm-систем.

Ключевые слова: Мультипроцессор, многоуровневый параллелизм, пользовательские потоки, OpenMP, TBB, OpenBLAS, HPL.

1. Введение

За прошедшее десятилетие наблюдается значительный рост числа процессорных ядер, входящих в состав современных вычислительных систем. Если раньше широкое распространение получали системы, использующие до десяти процессорных ядер, то сейчас число ядер в системах на общей памяти может насчитывать несколько сотен. Такой стремительный рост сложности вычислительных систем является вызовом для разработчиков прикладных программ и алгоритмов с точки зрения эффективного использования всех имеющихся вычислительных ресурсов. Невозможность постоянного уменьшения порций работ, выполняемых параллельно, из-за возникающих накладных расходов на порождение, уничтожение и планирование отдельной порции ведет к необходимости поиска дополнительных источников параллелизма. Вложенный параллелизм [1–5] — один из возможных путей решения.

OpenMP — общепризнанный стандарт для разработки параллельных программ для систем с общей памятью, поддерживает языки программирования C/C++ и Fortran, активно применяющиеся при разработке высокопроизводительных приложений. Кроме того, высокий уровень конструкций OpenMP, представляющих собой директивные расширения поддерживаемых языков, и инкрементальный способ распараллеливания уже имеющихся последовательных программ значительно снижает порог вхождения в область разработки параллельных программ.

OpenMP позволяет описать параллелизм явно на основе выделения потоков, при этом стандартная реализация системы поддержки времени выполнения (GCC, LLVM) полагается на использование потоков уровня операционной системы. Таким образом, при том, что OpenMP явно предусматривает возможность использования многоуровневого параллелизма, эффективность программ, использующих более одного уровня, без ручного подбора количества потоков на каждом уровне может приводить к деградации производительности из-за превышения количества создаваемых потоков уровня операционной системы над количеством физически доступных вычислительных ресурсов.

Некоторые реализации системы поддержки времени выполнения позволяют ограничить число одновременно исполняемых параллельных областей за счет реализации специального механизма блокировок [2]. Соответствующий режим исполнения OpenMP программ может быть задействован за счет использования дополнительных переменных окружения (например `KMP_COMPOSABILITY`), предоставляемых конкретной реализацией библиотеки.

Другим возможным решением в данной ситуации может быть добавление промежуточного уровня абстракции в виде потоков уровня пользователя, которые бы позволили отобразить несколько потоков на один поток уровня операционной системы и избавиться от несоответствия доступных физических ресурсов количеству запрашиваемых потоков уровня операционной системы. Одной из реализаций OpenMP, направленных на решение данной проблемы, является BOLT [6]. Данная реализация опирается на легковесную систему времени выполнения на основе пользовательских потоков Argobots [7]. Недостатком BOLT является деградация производительности при использовании подхода классического параллелизма[1] [8] по сравнению со стандартными реализациями OpenMP, что является следствием дополнительных расходов, возникающих во время планирования выполнения пользовательских потоков.

Использование большого числа даже пользовательских потоков требует реализации эффективных алгоритмов динамической балансировки нагрузки, мелкозернистых синхронизаций и планировщика, способного адаптироваться под архитектуру вычислительной системы и особенности конкретной программы. Достаточно гибкие средства планирования задач реализованы в библиотеке oneAPI Threading Building Blocks (TBB) [9] — высокоуровневой C++ библиотеке, которая предоставляет разработчику средства разработки параллельных программ, основанных на явном использовании параллелизма задач. Данная библиотека может являться одним из возможных кандидатов для реализации предложенного выше уровня абстракции между высокоуровневой моделью параллельного программирования и множеством потоков уровня операционной системы.

В данной работе мы рассматриваем влияние вложенного параллелизма на эффективность программ, использующих библиотеки для ускорения отдельных этапов вычислений. Внимание уделяется совместному использованию параллелизма, реализованного в прикладной программе пользователя, и параллельной реализации используемой библиотеки. Исследование направлено на рассмотрение реализаций библиотеки базовых подпрограмм линейной алгебры (BLAS), особое внимание уделяется реализации OpenBLAS [13].

[1]Подход к разработке параллельных приложений с использованием одного уровня параллелизма с использованием потоков уровня операционной системы.

Среди основных результатов данной работы можно выделить разработку прототипа системы компиляции, эффективно реализующей подмножество конструкций OpenMP, которые отображаются в вызовы библиотеки поддержки параллельного выполнения программы. Разработанная библиотека поддержки времени выполнения основана на применении промежуточного уровня абстракции в виде пользовательских потоков для оптимизации параллельного выполнения большого количества потоков, общее число которых может превышать физически доступные вычислительные ресурсы.

Статья организована следующим образом. В разделе 2 рассматривается библиотека BLAS и некоторые из ее существующих реализаций, обосновывается выбор в пользу реализации OpenBLAS, как предмета данного исследования. В разделе 3 рассматривается реализованный прототип системы компиляции, рассматриваются особенности реализации библиотеки времени выполнения в случае использования ее для параллельного выполнения библиотек. Внимание также уделяется хранению контекста, описывающего структуру выполняющегося пользовательского потока, в условиях оптимизаций обращений к памяти из разных потоков операционной системы, выполняемых стандартными компиляторами. В разделе 4 приводятся результаты экспериментов, предусматривающих использование библиотеки OpenBLAS в условиях вложенного параллелизма, а также исследуется потенциальный выигрыш, который можно достичь при выполнении бенчмарка High Performance Linpack (HPL). В разделе 5 кратко сформулированы результаты проведенного исследования.

2. Операции линейной алгебры

Пакет линейной алгебры (LAPACK), де-факто стандарт, который принят и поддержан большим сообществом пользователей, вычислительными центрами и поставщиками оборудования для высокопроизводительных вычислений. Этот пакет включает в себя самые современные численные алгоритмы для наиболее распространенных задач линейной алгебры, встречающихся в научных и инженерных приложениях: решение линейных уравнений, задач на собственные значения и др. Пакет LAPACK построен на базе BLAS.

Библиотека BLAS является одной из наиболее важных базовых программных библиотек в научных вычислениях. Эталонная реализация библиотеки реализована на Fortran [10]. В нее вошли процедуры, реализующие следующие преобразования:

- скалярное произведение двух векторов;
- копирование элементов одного вектора в другой;
- перестановка местами элементов двух векторов;
- умножение вектора на скаляр;
- векторные нормы (евклидова норма, сумма абсолютных значений элементов и максимальное абсолютное значение),
- и другие процедуры типа «вектор-вектор», «матрица-вектор», «матрица-матрица».

Помимо этой реализации, существуют оптимизированные варианты практически для каждой компьютерной архитектуры. Эти реализации в основном предоставляются производителями процессоров или аппаратного обеспечения для получения максимальной производительности на этих устройствах [11, 12]. В дополнение к реали-

зациям от поставщиков, существуют реализации с открытым исходным кодом, такие как OpenBLAS [13] и Automatically Tuned Linear Algebra Software (ATLAS) [14].

Преимуществами библиотеки OpenBLAS являются:

- библиотека активно развивается (последняя версия библиотеки 0.3.29 разработана в январе 2025 года);
- широкий спектр поддерживаемых архитектур: RISC-V, X64, SPARC, X86, MIPS, ARM, AArch64, POWER, PPC64, IBM System z;
- существует параллельная реализация библиотеки с помощью POSIX Threads и Op-enMP;
- библиотека полностью реализует интерфейсы BLAS и LAPACK, а также предоставляет дополнительное множество BLAS-like функций для расширения возможностей BLAS;
- BLAS-часть библиотеки содержит в себе набор базовых операций линейной алгебры, которые чаще всего используются в прикладных программах. Все функции оптимизированы как алгоритмически, так и на низком уровне, а на их основе строятся реализации более сложных алгоритмов, входящих в LAPACK;
- поддерживается большим сообществом, в составе которого не только независимые разработчики, но и крупные компании;
- наличие в библиотеке встроенных тестов, которые позволяют сравнивать различные реализации BLAS-библиотек по их эффективности. Некоторые тесты распараллелены с использованием OpenMP;
- один из основных модулей библиотеки, который используется для распараллеливания вычислений (blas_server_omp.c), состоит всего из 458 строк, что позволяет, в случае необходимости, адаптировать его в соответствии с возможностями, реализованными в разработанной версии компилятора.

Как уже отмечалось ранее, в составе OpenBLAS есть встроенные тесты, позволяющие оценить эффективность реализации различных функций библиотеки. Эти тесты, в основном, построены следующим образом. В последовательном цикле пе-

```
for(m = from; m <= to; m += step) {
  for(j = 0; j < m; j++) {
    for(i = 0; i < m * COMPSIZE; i++) {
      a[(long)i+(long)j*(long)m*COMPSIZE]
        = ((FLOAT)rand()/(FLOAT)RAND_MAX)-0.5;
      b[(long)i+(long)j*(long)m*COMPSIZE]
        = ((FLOAT)rand()/(FLOAT)RAND_MAX)-0.5;
      c[(long)i+(long)j*(long)m*COMPSIZE]
        = ((FLOAT)rand()/(FLOAT)RAND_MAX)-0.5;
    }
  }
  begin();
  SYMM (&side, &uplo, &m, &m, alpha, a, &m, b, &m, beta, c, &m );
  end();
  time1 = getsec();
  fprintf(stderr, "␣%10.2f␣MFlops\n",
     COMPSIZE*COMPSIZE*2.*(double)m*(double)m*(double)m/time1*1.e-6);
}
```

Рис. 1: Фрагмент программы, используемой при тестировании для процедуры SYMM

ребираются необходимые входные данные, задаются размеры матриц, определяется количество итераций алгоритма и т.п. Далее замеряется время работы, а также производительность программы (в MFlops). Такой поход, позволяет сравнить качество реализации функций библиотеки при решении различных задач на разных поддерживаемых библиотекой архитектурах.

На рис 1 представлен фрагмент программы, используемой при тестировании для процедуры SYMM, выполняющей преобразование вида $C := \alpha * A * B + \beta * C$ или $C = \alpha * B * A + \beta * C$, где α и β — скалярные переменные, матрица A — симметричная матрица размера ($m \times m$ или $n \times n$), а B и C — матрицы размера $m \times n$ (symm.c).

Все матрицы при тестировании считаются квадратными (размера $m \times m$). Размер матриц, на которых запускается процедура, перебирается в цикле. Сначала выполняется инициализация матриц A, B и C случайными значениями, а затем вызывается процедура SYMM, выполняющая необходимое преобразование. Замеряется время ее работы. Такое тестирование выполняется для матриц, использующих разные типы данных: `float`, `double`, `complex float`, `complex double`.

3. Прототип системы компиляции

Данное исследование предполагает разработку нового подхода, использующего абстракцию в виде пользовательских потоков, к реализации библиотеки времени выполнения для высокоуровневых моделей параллельного программирования. Таким образом, для достижения целей данного исследования был разработан транслятор подмножества спецификаций стандарта OpenMP, выполняющий конвертацию OpenMP-программы в программу на языке высокого уровня с вызовами библиотеки времени выполнения. Подмножество спецификаций было выбрано таким образом, чтобы его оказалось достаточно для описания параллелизма в исследуемых приложениях. Библиотека времени выполнения, контролирующая выполнение параллельной программы, также была разработана в рамках данного исследования.

Использование высокоуровневых спецификаций OpenMP для эффективного описания многоуровневого параллелизма вызывает отдельный интерес, так как способствует выработке рекомендаций по написанию и модификации большого числа уже написанных в модели OpenMP программ.

Библиотека времени выполнения обеспечивает промежуточный интерфейс для использования различных технологий параллельного программирования (OpenMP Task, OpenMP Taskloop, Argobots [7], Taskflow [15], TBB [9]), используемых для разных вариантов ее реализации [5, 8]. В данной работе основное внимание уделяется подходу, основанному на использовании возможностей TBB в сравнении со стандартными возможностями, предоставляемыми OpenMP для описания параллелизма уровня циклов.

Было реализовано два типа пользовательских потоков, различающихся в зависимости от выбора конструкций OpenMP их порождающих. Обработка директивы `omp parallel for` отличается от обработки директив `parallel` и `for`, указываемых по-отдельности, так как при совместном их указании компилятор и библиотека времени выполнения обладают большей информацией о параллельно выполняемых вычислениях и могут определять количество создаваемых легких потоков в зависимости от количества итераций в параллельном цикле. Если же директивы указаны

по-отдельности, то на момент порождения потоков — обработка директивы `parallel`, невозможно определить оптимальное число потоков для выполнения параллельной области, и приходится использовать заранее заданные настройки по умолчанию.

Для разработки транслятора были использованы существующие средства, предоставляемые инфраструктурой компиляторов LLVM [16] и имеющимися в ней фронтендами с языков высокого уровня.

Для упрощения компиляции и запуска программ данные этапы объединены под управлением отдельного инструмента — драйвера. Настройки компиляции и запуска задаются как через опции драйвера, так и через переменные окружения, которые позволяют управлять числом потоков уровня операционной системы, ограничениями на максимальное число порождаемых пользовательских потоков, используемыми алгоритмами динамического планирования и балансировки нагрузки, а также способами распределения потоков по NUMA-узлам вычислительной системы.

3.1. Особенности конвертации внутри библиотек

Для приложений, использующих OpenBLAS, специфично обращение к некоторым функциям OpenMP до начала выполнения функции `main()`, в момент загрузки динамической библиотеки реализующей функции OpenBLAS. Для успешного выполнения данных функций необходима предварительная инициализация библиотеки времени выполнения OpenMP. Для реализованных версий библиотеки данная инициализация выполняется специальной функцией инициализации, вызов которой вставляется в начало функции `main`.

Чтобы выполнить инициализацию до начала исполнения `main`, был использован механизм атрибутов функций для задания функций, вызываемых в момент загрузки динамической библиотеки: `__attribute__ ((constructor(101)))`. Кроме того, для данных функций были явно заданы приоритеты исполнения, чтобы обеспечить их вызов до начала исполнения функций OpenBLAS.

Аналогичные изменения были внесены для высвобождения ресурсов внутри соответствующих функций, помеченных атрибутом `__attribute__((destructor(101)))`.

3.2. Особенности хранения контекста пользовательских потоков

Для того, чтобы иметь возможность в любой точке программы идентифицировать пользовательский поток, который ее обрабатывает, необходимо сохранить информацию о потоке в глобальной области видимости. Так как одновременно могут выполняться несколько пользовательских потоков: каждый выполняется своим потоком операционной системы, то для хранения контекста выполнения потока был использован подход с применением thread-local-storage (TLS) (см. рис. 2).

Таким образом, перед началом выполнения пользовательского потока текущее значение контекста из TLS потока уровня операционной системы запоминается в локальной переменной, а после завершения потока в TLS восстанавливается информация о родительском потоке.

Стоит отметить, что при определенных обстоятельствах, которые могут проявляться из-за недетерминированности выполнения пользовательских потоков на по-

```
static thread_local context_t *this_context = nullptr;

context_t * get_context() {
  return this_context;
}
void set_context(context_t *context) {
  this_context = context;
}
```

Рис. 2: Сохранение контекста выполнения пользовательского потока в TLS потока уровня операционной системы

токах операционной системы, поток операционной системы, который выполнял родительский пользовательский поток до начала выполнения дочернего пользовательского потока, может смениться после завершения дочернего пользовательского потока. Например, в зависимости от особенностей выполнения потоков на рис. 3 поток операционной системы до начала выполнения внутреннего параллельного цикла мог иметь идентификатор `TID_1`, а после его завершения `TID_2`. Таким образом, TLS до и после выполнения внутреннего цикла — различны.

Особенность состоит в том, что если объявление TLS переменной видимо для стандартного компилятора в момент оптимизации кода для данного фрагмента, то компилятор может предположить, что данное значение осталось неизменным, так как в коде программы отсутствуют явные указания на возможность его изменения (стандартный компилятор не знает о пользовательских потоках и о том, что они могут менять принадлежность к потоку операционной системы, который их исполняет). Таким образом, компилятор может заменить в коде программы все использования `TID_2` на использования `TID_1` в целях оптимизации, что приведет к некорректной программе.

Чтобы обойти данную проблему, обращение к TLS переменным было вынесено в специальные функции, приведенные на рис. 2, а сами функции были вынесены в отдельную единицу компиляции. При этом важно, чтобы в момент сборки программы была отключена межмодульная оптимизация, и тела данных функций не были подставлены в точки вызова до и после момента исполнения внутреннего цикла на рис. 3.

```
#pragma omp parallel for
for (int i = 0; i < N; ++i) {
  // Some work in outer user-level thread.
  // Let us assume OS-level thread at this point is TID_1.
  #pragma omp parallel for
  for (int j = 0; j < N; ++j) {
    // Some work in inner user-level thread.
  }
  // Let us assume OS-level thread at this point is TID_2.
  // At this point TID_1 may differ from TID_2.
}
```

Рис. 3: Потенциальная возможность некорректной оптимизации кода стандартными компиляторами в случае использования TLS

4. Эксперименты

В рамках данного исследования эксперименты проводились на следующих вычислительных системах:

- сервер с процессорами Kunpeng 2.6 GHz (архитектура aarch64), включающий 192 ядра (8 NUMA узлов — 24 ядра на узел), 1 поток на ядро, 503GB, Ubuntu 18.04.6 LTS (далее — сервер Atlas),
- сервер, с процессорами Kunpeng 920 CPU, 2.6 GHz (архитектура aarch64), включающий 48 ядер (2 NUMA узла — 24 ядра на узел), 1 поток на ядро, 512 GB, CentOS Linux 8 (далее — сервер Kunpeng).

Чтобы обеспечить одинаковое программное окружение для запуска приложений был использован дистрибутив Miniconda [17], включающий в себя систему управления пакетами и средой выполнения приложений. Для компиляции приложений был использован компилятор GCC 13.2.0 и TBB 2021.11, при компиляции указывалась опция оптимизации -O3.

4.1. Библиотека OpenBLAS

Для исследования влияния многоуровневого параллелизма на эффективность параллельного выполнения программ, использующих библиотеку OpenBLAS, были разработаны несколько тестовых приложений, реализующих умножение различных матриц, решающих различные матричные уравнения. Разработанный пакет тестов позволяет задействовать и исследовать 2 уровня параллелизма — в основной программе и внутри библиотеки OpenBLAS. Указанный на рис. 1 фрагмент программы был переписан так, как показано на рис. 4.

При помощи процедуры `omp_set_num_threads(num_threads_in)` задается количество потоков, которые будут использоваться библиотекой OpenBLAS для выполнения умножения матриц. Параллельный цикл `for(m = from; m <= to; m += step)` запускает на выполнение множество задач, каждая из которых выполняет требуемое SYMM-преобразование для матриц соответствующего размера ($m \times m$). Количество и сложность решаемых задач задается при помощи параметров цикла `from`, `to` и `step`. Необходимые для работы программы данные инициализируются один раз, а затем копируются из соответствующих массивов перед выполнением преобразования каждый раз (вместо вызовов генератора случайных чисел) для исключения влияния используемых в процессе счета данных(чисел) на время выполнения программы.

Такой подход позволяет многократно запускать различные преобразования, задавая различное число потоков на внешнем (`num_threads_out`) и внутреннем (`num_threads_in`) уровнях, меняя стратегии распределения работ по потокам (спецификация `schedule`), варьируя количество выполняемых задач и изменяя их сложность.

В таблица 1 представлены результаты такого исследования теста SYMM на машине Atlas при использовании до 192 OpenMP-потоков (используется реализация OpenMP, предоставляемая компилятором GCC). В таблице для разных типов данных отдельно выделено наименьшее время и наибольшее ускорение относительно выполнения программы на одном потоке.

```
omp_set_num_threads(num_threads_in);
#pragma omp parallel for private(a,b,c,aa,bb,cc,i,j) \
        num_threads(num_threads_out) schedule(runtime)
for(m = from; m <= to; m += step) {
  a = a1[m - from];
  b = b1[m - from];
  c = c1[m - from];
  aa = a0[m - from];
  bb = b0[m - from];
  cc = c0[m - from];
  for(j = 0; j < m; j++) {
    for(i = 0; i < m * COMPSIZE; i++) {
      a[(long)i+(long)j*(long)m*COMPSIZE]
        = aa[(long)i+(long)j*(long)m*COMPSIZE];
      b[(long)i+(long)j*(long)m*COMPSIZE]
        = bb[(long)i+(long)j*(long)m*COMPSIZE];
      c[(long)i+(long)j*(long)m*COMPSIZE]
        = cc[(long)i+(long)j*(long)m*COMPSIZE];
    }
  }
}
time1 = omp_get_wtime();
#pragma omp parallel for private(a,b,c) \
        num_threads(num_threads_out) schedule(runtime)
for(m = from; m <= to; m += step) {
  a = a1[m - from];
  b = b1[m - from];
  c = c1[m - from];
  SYMM (&side, &uplo, &m, &m, alpha, a, &m, b, &m, beta, c, &m );
}
time1 = omp_get_wtime() - time1;
fprintf(stderr, "|␣␣␣␣␣␣␣␣␣␣␣␣%3d|␣␣␣␣␣␣␣␣␣␣␣␣%3d|%11.6f|\n",
        num_threads_out, num_threads_in, time1);
```

Рис. 4: Фрагмент параллельной программы, используемой при тестировании процедуры SYMM и задействующий два уровня параллелизма

Таблица 1: Время и ускорение выполнения теста SYMM на разном числе потоков для конфигурации: from=1400, to=1800, step=1, schedule=dynamic

Кол-во потоков	Тип данных							
	float		double		complex float		complex double	
	Время (с.)	Ускорение	Время (с.)	Ускорение	Время (с.)	Ускорение	Время (с.)	Ускорение
1x1	124.8	1.00	339.1835	1.00	517.19	1.00	1918.4	1.00
1x192	4.74	26.34	8.1486	41.62	16.83	30.73	33.61	57.08
6x32	2.20	56.69	4.5106	75.20	4.22	122.54	11.52	166.49
12x16	**1.44**	**86.79**	3.49	97.58	**3.78**	**136.87**	**11.23**	**170.85**
24x8	1.54	81.27	3.91	86.76	4.08	126.74	12.33	155.58
48x4	2.20	56.62	**3.47**	**97.69**	4.42	116.96	12.49	153.56
96x2	2.23	55.89	3.62	93.66	4.35	118.90	12.82	149.66
192x1	2.78	44.74	4.49	75.62	5.02	103.01	15.34	125.09

В таблице 2 представлены результаты для теста HEMM, решающего аналогичную задачу, в которой A — это эрмитова матрица. Элементы этой матрицы являются комплексными числами, и, будучи транспонирована, она равна комплексно сопряжённой.

Таблица 2: Время и ускорение выполнения теста HEMM на разном числе потоков для конфигурации: from=1400, to=1800, step=1, schedule=dynamic

Кол-во потоков	Тип данных			
	complex float		complex double	
	Время (с.)	Ускорение	Время (с.)	Ускорение
1x1	516.26	1.00	1914.29	1.00
1x192	16.71	30.89	26.63	71.87
6x32	4.77	108.21	11.69	163.73
12x16	**4.01**	**128.74**	**11.4279**	**167.51**
24x8	4.13	125.05	11.91	160.70
48x4	4.27	121.02	12.0356	159.05
96x2	4.36	118.53	12.4746	153.46
192x1	4.88	105.83	15.0869	126.88

В таблице 3 представлены результаты для теста TRSM, который предназначен для решения левостороннего $A * X = \alpha * Y$ или правостороннего $X * A = \alpha * Y$ матричного уравнения для всевозможных вариантов представления треугольной матрицы A. Установкой параметров можно задать не только, с какой стороны в этом уравнении участвует матрица A, но и уточнить, является ли она нижней или верх-

Таблица 3: Время и ускорение выполнения теста TRSM на разном числе потоков для конфигурации: from=1000, to=1200, step=1, schedule=dynamic, side=L, uplo=U, trans=N, diag=U, loops=3

Кол-во потоков	Тип данных							
	float		double		complex float		complex double	
	Время (с.)	Ускорение	Время (с.)	Ускорение	Время (с.)	Ускорение	Время (с.)	Ускорение
1x1	40.23	1.00	89.27	1.00	140.60	1.00	468.58	1.00
1x192	11.43	3.52	18.56	4.81	19.13	7.35	22.76	20.58
6x32	4.64	8.66	2.71	32.96	2.89	48.66	4.94	94.88
12x16	1.83	21.93	1.59	56.10	1.75	80.13	4.29	109.31
24x8	1.0968	36.69	1.44	62.01	1.8	78.26	4.65	100.76
48x4	**0.78**	**51.26**	1.39	64.23	1.77	79.55	**4.12**	**113.86**
96x2	0.99	40.56	**1.28**	**69.77**	**1.74**	**80.70**	4.23	110.79
192x1	0.84	47.71	1.31	68.24	1.95	72.27	5.44	86.16

ней треугольной, транспонированная ли она или сопряжённая (для комплексных), единичная ли у неё диагональ или нет.

В таблице 4 показаны наилучшие времена выполнения различных тестов при использовании 192 потоков на машине Atlas, когда распараллелен только цикл, осуществляющий запуск множества задач, а параллелизм внутри OpenBLAS не используется (столбец «Внешний»), когда цикл по задачам выполняется последовательно, но используется параллелизм внутри OpenBLAS (столбец «Внутренний»), а также наилучший запуск, который использует 2 уровня параллелизма (столбец «Вложенный»). В таблице также показан прирост производительности, получаемый за счет использования вложенного параллелизма по отношению к одному уровню параллелизма (всегда выбирался внешний параллелизма, так как в проведенных экспериментах он показывает лучшее время по сравнению с внутренним параллелизмом).

Таблица 4: Наилучшее время (с.) выполнения на сервере Atlas тестов SYMM, HEMM, TRSM на 192 потоков с использованием разных уровней параллелизма для конфигураций тестов из таблиц 1, 2, 3 соответственно и типов данных float (f), double (d), complex float (cf), complex double (cd). Столбец «Ускорение» показывает ускорение, полученное от использования вложенного параллелизма, по отношению к лучшему варианту, использующему один уровень параллелизма

Тип данных	SYMM				HEMM				TRSM			
	Внешний (с.)	Внутренний (с.)	Вложенный (с.)	Ускорение	Внешний (с.)	Внутренний (с.)	Вложенный (с.)	Ускорение	Внешний (с.)	Внутренний (с.)	Вложенный (с.)	Ускорение
f	2.79	4.74	**1.44**	**51%**					0.84	11.43	**0.78**	**7%**
d	4.49	8.15	**3.47**	**23%**					1.31	18.56	**1.28**	**2%**
cf	5.02	16.83	**3.78**	**25%**	4.88	16.71	**4.01**	**18%**	1.95	19.13	**1.74**	**11%**
cd	15.34	33.61	**11.23**	**27%**	15.07	26.63	**11.43**	**24%**	5.44	22.76	**4.12**	**24%**

По полученным в результате тестирования временам, представленным в предыдущих таблицах, можно утверждать, что использование многоуровневого распараллеливания позволяет существенно ускорить выполнение тестов SYMM, HEMM, TRSM, по сравнению с использованием одного уровня параллелизма. Разработанные тесты позволяют управлять количеством задач, которые выполняются параллельно. Каждая задача выполняется для матриц различного размера и занимает разное время. Несмотря на использование в программах спецификации `schedule(dynamic)`, которая позволяет распределять задачи по потокам в процессе выполнения цикла (поток, раньше остальных завершивший выполнение своей задачи, получает следующую задачу на выполнение), для данных тестов не удается сбалансировать вычислительную нагрузку по потокам. Например, если выполняемых задач всего 200, а потоков — 192, то некоторым потокам приходится выполнять несколько задач, что приводит к итоговому замедлению выполнения программы. За счет включения внутреннего уровня параллелизма, удается ускорить выполнение задач, что позволяет улучшить балансировку нагрузки для каждого из представленных тестов.

4.2. High-Performance Linpack

Важным тестом, использующим библиотеку BLAS, является High-Performance Linpack (HPL). Данный тест используется для составления рейтинга самых высокопроизводительных систем мира — TOP500. Тест HPL устроен следующим образом. Генерируется линейная система уравнений порядка n, которая решается с помощью LU-разложения с частичным поворотом строк. Для работы программы требуются библиотеки MPI и BLAS. При помощи MPI осуществляется циклическое распределение данных (2-D блоки) и вычислений по процессам. Второй уровень распараллеливания возможен за счет подключения потоков при выполнении BLAS-функций.

На машине Kunpeng было проведено сравнение различных версий теста HPL при использовании различного числа MPI-процессов и потоков, порождаемых внутри этих процессов. При выполнении данного исследования использовалась библиотека OpenMPI и различные версии библиотеки BLAS (техническая возможность установки различных версий библиотеки на данную вычислительную систему определила выбор в пользу машины Kunpeng в дальнейших экспериментах):

- Исходная версия библиотеки OpenBLAS, скомпилированная в режиме OpenMP (далее OpenMP-версия).
- Версия библиотеки OpenBLAS, полученная при помощи компилятора, разработанного в рамках данного исследования и использующая реализацию библиотеки времени выполнения OpenMP на основе TBB (далее TBB-версия).
- Kunpeng Math Library, входящая в пакет Kunpeng BoostKit (далее KML-версия).

В таблицах 5, 6, 7 представлены времена выполнения различных версий теста HPL на разном числе процессов и потоков для линейных систем уравнений порядка 20000×20000. Использовались до 48 ядер системы Kunpeng. Для OpenMPI задавались различные варианты привязки процессов (а следовательно и потоков, порождаемых MPI-процессом) при помощи опций запуска: `-bind-to-none`, `-bind-to-numa`. В таблицах указаны наилучшие полученные времена в секундах.

Полученные результаты показывают:

1) Использование многоуровневого параллелизма дает небольшой эффект для теста HPL. Наилучшие времена выполнения теста получаются при использовании 24-MPI процессов, каждый из которых состоит из 2-х потоков.

Таблица 5: Время (с) выполнения OpenMP-версии теста HPL на разном числе процессов и потоков для системы 20000×20000.

No.	Number of OS-level threads							
MPI	1	2	4	8	16	24	32	48
1	536.33	269.96	137.20	70.42	37.21	26.14	22.85	19.02
2	276.78	139.32	70.72	36.76	19.84	14.68		
4	141.54	72.71	37.45	20.09				
8	75.89	38.63	20.10					
16	40.06	20.69						
24	27.54	**14.44**						
32	21.36							
48	15.20							

Таблица 6: Время (с) выполнения TBB-версии теста HPL на разном числе процессов и потоков для системы 20000×20000.

No.	Number of OS-level threads							
MPI	1	2	4	8	16	24	32	48
1	536.67	270.22	137.05	70.48	37.35	26.48	23.42	18.67
2	276.73	139.28	70.47	36.87	19.84	14.48		
4	141.54	72.60	37.41	19.71				
8	75.72	38.60	20.04					
16	40.05	20.61						
24	27.60	**14.58**						
32	21.16							
48	15.09							

Таблица 7: Время (с) выполнения KML-версии теста HPL на разном числе процессов и потоков для системы 20000×20000.

No.	Number of OS-level threads							
MPI	1	2	4	8	16	24	32	48
1	525.62	265.76	134.70	69.23	36.93	26.37	26.50	25.19
2	271.20	136.64	69.39	36.51	20.25	15.02		
4	139.51	71.20	36.97	19.86				
8	74.43	38.00	19.82					
16	39.45	20.40						
24	27.03	**14.22**						
32	21.04							
48	14.86							

2) Наилучшая конфигурация — 24 процесса по 2 потока, не позволяет в полной мере получить выигрыш от использования TBB-версии OpenBLAS по сравнению с OpenMP-версией. Одним из возможных объяснений плохого ускорения теста HPL при использовании потоков является то, что не все вычисления, выполняемые процессом, идут через OpenBLAS, часть работы выполняется MPI-процессом последовательно.
3) По эффективности выполнения TBB-версия теста HPL, полученная с использованием разработанного в рамках данного исследования компилятора, практически не уступает KML-версии.

5. Заключение

Проведенные в данной работе исследования подтверждают, что многоуровневый параллелизм может служить источником дополнительного ускорения программ в среднем на 20-25% по сравнению с временем, затрачиваемым на выполнение программ, задействующих только один уровень параллелизма. Однако, стандартные реализации общепризнанных высокоуровневых моделей параллельного программирования (таких как OpenMP) требуют кропотливой ручной настройки используемого чис-

ла потоков на каждом уровне параллелизма. Отсутствие данной настройки может негативно сказаться на эффективности выполнения программы, особенно в случае превышения запрашиваемого числа потоков операционной системы над числом доступных вычислительных ядер.

С другой стороны, наши исследования показывают возможность использования единого подхода к разработке параллельных программ с разной степенью параллелизма (классический и многоуровневый параллелизм) и реализующего потенциал многоядерных процессоров для повышения производительности вычислений. Данный подход предполагает использование дополнительной абстракции для описания параллелизма в виде пользовательских потоков, которые могут быть использованы для отображения параллельных фрагментов программы, число которых может превышать физически доступные ресурсы, на доступные потоки операционной системы, которые затем эффективно отображаются на ядра вычислительного процессора. Более того, использование механизма пользовательских потоков может быть скрыто за высоким уровнем используемой модели параллельного программирования.

Полученные результаты открывают путь к дальнейшим исследованиям, связанным с оптимизацией планирования пользовательских потоков и определением их числа, распределением их по NUMA-узлам вычислительной системы и реализацией примитивов синхронизации.

Список литературы

[1] Fiksman E., Malakhov A. Efficient Nested Parallelism On Large-Scale Systems (chapter 18), “High Performance Parallelism Pearls: Multicore and Many-core Programming Approaches” by Reinders, James, Jeffers, James, chapter 18, 2014, P. 307–318, Morgan Kaufmann

[2] Malakhov A.A., Liu D., Gorshkov A.V., Wilmarth T. Composable Multi-Threading and Multi-Processing for Numeric Libraries. SciPy, P. 18–24. https://proceedings.scipy.org/articles/Majora-4af1f417-003.pdf

[3] Malakhov A. Composable Multi-Threading for Python Libraries // in Proceedings of the 15th Python in Science Conference, P. 15–19, 2016. https://doi.org/10.25080/Majora-629e541a-002

[4] Malakhov A. Fusing Efficient Parallel For Loops with a Composable Task Scheduler, Hydra Conf, 2022. https://hydraconf.com/archive/2022/talks/50468720b63246e7a1389a8c100eba72/?referer%2Farchive%2F2022%2Fpersons%2F3d8ec0db499a4e19bae7724fa9c84bf8%2F

[5] Bakhtin V.A., Kataev N.A., Kocharmin M.D., Kolganov A.S., Smirnov A.A., Zaharov D.A. A Study of a Composable Approach to Parallel Programming for Many-Core Multiprocessors, Lecture Notes in Computer Science, Cham: Springer, 2025, Vol. 15406, P. 285–299. https://doi.org/10.1007/978-3-031-78459-0_21

[6] Iwasaki S., Amer A., Taura K., Seo S., Balaji P. BOLT: Optimizing OpenMP Parallel Regions with User-Level Threads // in Proc. The 28th International Conference on Parallel Architectures and Compilation Techniques (PACT ’19), Sept. 2019, 2019, P. 29–42. https://doi.org/10.1109/PACT.2019.00011

[7] Seo S., Amer A., Balaji P., et al. Argobots: A Lightweight Low-Level Threading and Tasking Framework // IEEE Transactions on Parallel and Distributed Sys-tems (TPDS), Oct. 2017 https://doi.org/10.1109/TPDS.2017.2766062

[8] Bakhtin V.A., Kataev N.A., Kolganov A.S., Smirnov A.A., Zaharov D.A. Exploring Composable Parallelism in Computational Modelling // Mathematical Models and Computer Simulations, 2024, Vol. 16, Issue Suppl 2, P. 216–224. https://doi.org/10.1134/S2070048224700923

[9] oneTBB https://github.com/oneapi-src/oneTBB

[10] Dongarra J., Du Croz J., Hammarling S., Hanson R. An extended set of FORTRAN Basic Linear Algebra Subprograms, ACM Trans. Math. Softw., 14, 1988, P. 1–17. https://doi.org/10.1145/42288.42291

[11] Intel®, Math Kernel Library (MKL). http://developer.intel.com/software/products/mkl/ (дата обращения 15.04.2025).

[12] Kunpeng Math Library (Kunpeng BoostKit 22.0.0) (MKL). https://support.huawei.com/enterprise/en/doc/EDOC1100283141/88ccc310/kml_blas-library-functions (дата обращения 15.04.2025).

[13] The OpenBLAS Team. http://xianyi.github.com/OpenBLAS (дата обращения 15.04.2025).

[14] Whaley R.C., Petitet A., Dongarra J.J. Automated empirical optimization of software and the ATLAS project, Parallel Computing, 27, 2001, P. 3–35. Also available as University of Tennessee LAPACK Working Note 147, UT-CS-00-448, 2000. https://doi.org/10.1016/S0167-8191(00)00087-9 www.netlib.org/lapack/lawns/lawn147.ps.

[15] Taskflow. A General-purpose Task-parallel Programming System. https://taskflow.github.io/ (дата обращения 15.04.2025).

[16] The LLVM Compiler Infrastructure. https://llvm.org/ (дата обращения 15.04.2025).

[17] Miniconda — Anaconda documentation. https://docs.anaconda.com/free/miniconda/index.html (дата обращения 15.04.2025).

Особенности обучения нейронной сети Бехлера–Парринелло на результатах квантовой молекулярно-динамической модели кристалла лития

А.Г. Потапов, А.В. Романов

Воронежский государственный университет

В последние пять лет наблюдается рост интереса к использованию нейронных сетей в моделировании материалов методом молекулярной динамики. Было реализовано несколько программных пакетов, позволяющих с помощью машинного обучения исключить проблему поиска классических межатомных потенциалов. В статье представлено исследование роли параметров нейронной сети Бехлера-Парринелло на аппроксимацию характеристик квантовой молекулярно-динамической модели кристалла лития. Показано, что выбор числа и формы функций Бехлера-Парринелло, количества нейронов в скрытых слоях и радиуса отсечки сложным образом влияет на качество машинного обучения.

Ключевые слова: нейросеть, межатомный потенциал, молекулярная динамика, функции Бехлера–Парринелло.

1. Введение

Расширение области применения композитных материалов ставит вопрос об усовершенствовании методов предсказания их свойств на основе математических моделей. Рост производительности современных вычислительных комплексов всё ещё не даёт возможности использовать для этих целей квантовую молекулярную динамику [1, 2]. Классическая молекулярная динамика, выполняющаяся на графических ускорителях, предоставляет возможности моделирования достаточно крупных систем. Однако достичь гибкости классическим методом не представляется возможным без внедрения нейронных сетей [3, 4]. Они позволяют сделать бесшовный переход между квантовым и классическим подходом путём обучения модели, способной масштабировать результаты квантово-химических исследований на системы из десятков и сотен тысяч атомов.

«Камнем преткновения» в вопросе использования нейронных сетей для обучения межатомных потенциалов долгое время было применение декартовых и сферических координат. Подобный подход не давал возможности учитывать трансляционную, вращательную симметрию системы, а также взаимную перестановку атомов одного элемента. Выходом из этой ситуации стала разработка специальных атомноцентрированных функций, не зависящих от конкретных координат и способных учесть особенности атомного окружения.

В качестве таких функций можно использовать, например, полиномы Чебышева [5, 6] или функции Бехлера–Парринелло [7, 8]. Ряд программ, разработанных

в последние несколько лет, в том числе Fortnet [9], реализует такую возможность. Однако, несмотря на успехи, достигнутые в части реализации программного кода и алгоритмов нейронных сетей, остаются вопросы в методике их применения к конкретным химическим системам. В частности, нейронные сети для обучения межатомных потенциалов содержат большое количество варьируемых параметров, влияние которых на успешность обучения остаётся туманным. Данная работа призвана пролить свет на некоторые из этих особенностей на примере программного пакета Fortnet и кристалла лития как модельной системы.

Представленная статья организована следующим образом. Во второй части будет рассмотрена функциональная форма каждой из пяти симметричных функций Бехлера–Парринелло, используемых в нейросети в качестве базисных наборов для аппроксимации. Будут даны описания параметров, входящих в состав данных функций. Третья часть посвящена описанию моделей, построенных с целью изучить влияние различных факторов, включающих выбор радиуса отсечки, числа и вида симметричных функций, а также числа нейронов в скрытых слоях, на качество обучения нейронной сети Бехлера-Парринелло, реализованной в программе Fortnet. В заключении будут представлены наиболее значимые выводы проведённых исследований.

2. Теоретическая часть

Перед тем, как использовать функции Бехлера–Парринелло на реальных задачах, необходимо ознакомиться с их формулами и базовыми свойствами. Их проверка на простых тестовых примерах [10] показала наличие важных особенностей, связанных, в том числе, с порядком получаемой величины, которые нужно будет учитывать при дальнейшей работе.

2.1. Функции Бехлера–Парринелло

Ниже в (2)–(6) приведено математическое описание каждой из симметричных функций. Формула (1) содержит функцию отсечки, использующуюся для определения атомов, входящих в атомное окружение заданного атома.

$$f_c(R_{ij}) = \begin{cases} 0{,}5 \cdot [\cos(\frac{\pi R_{ij}}{R_c}) + 1] & \text{для } R_{ij} \leq R_c, \\ 0 & \text{для } R_{ij} > R_c, \end{cases} \tag{1}$$

$$G_i^1 = \sum_j f_c(R_{ij}) \tag{2}$$

$$G_i^2 = \sum_j e^{-\eta(R_{ij}-R_s)^2} \cdot f_c(R_{ij}) \tag{3}$$

$$G_i^3 = \sum_j \cos(\kappa R_{ij}) \cdot f_c(R_{ij}) \tag{4}$$

$$G_i^4 = 2^{1-\zeta} \sum_{j,k\neq i}^{all} (1 + \lambda \cos\theta_{ijk})^\zeta e^{-\eta(R_{ij}^2+R_{ik}^2+R_{jk}^2)} f_c(R_{ij}) f_c(R_{ik}) f_c(R_{jk}) \tag{5}$$

$$G_i^5 = 2^{1-\zeta} \sum_{j,k\neq i}^{all} (1 + \lambda \cos\theta_{ijk})^\zeta e^{-\eta(R_{ij}^2+R_{ik}^2)} f_c(R_{ij}) f_c(R_{ik}) \tag{6}$$

Здесь η — ширина гауссиан, регулирующая эффективное радиальное расширение функций симметрии; R_s — сдвиг центра гауссиан на определённый радиус; κ — параметр, определяющий период функции косинуса; θ — угол между всеми комбинациями из трёх атомов, содержащих заданный атом; λ — параметр, отвечающий за ориентацию графика $G(\theta)$; ζ — параметр, регулирующий угловое разрешение, то есть плавность изменения значения функции G.

Функции G^1, G^2, G^3 относятся к группе радиальных и учитывают только взаимные межатомные расстояния. Функции G^4, G^5 являются радиально-угловыми и учитывают более сложные изменения атомных конфигураций. Поскольку функции внутри одной группы обладают довольно схожими свойствами, в сети Fortnet по умолчанию используются только две из них, относящихся к разным категориям: G^2 и G^5.

3. Практическая часть

В ходе исследования было проведено моделирование кристалла лития с ячейками на 27 и 64 атома в рамках квантовой молекулярной динамики. С помощью калькулятора ASE [11] / GPAW [12, 13] были рассчитаны траектории движения атомов моделируемых систем, а также получены их потенциальная и полная энергии в каждый момент времени. На основании полученных данных было проведено обучение нейросети Бехлера-Парринелло с помощью программы Fortnet с различными гиперпараметрами, в числе которых радиус отсечки, количество и вид радиальных и угловых симметричных функций и число нейронов в скрытых слоях. Все расчёты проводились на 12 ядрах процессора Intel(R) Xeon(R) CPU E5-2680 v3 2.50 GHz.

3.1. Влияние радиуса отсечки

Первым из рассматриваемых параметров является радиус отсечки, влияющий на выбор атомов, которые будут учитываться при рассмотрении ближайшего окружения некоторого конкретного атома. Данный параметр оказывает влияние на каждую из пяти симметричных функций, а потому к его выбору следует подходить с осторожностью уже на начальном этапе моделирования.

Как можно видеть на рисунках 1 и 2, для системы Li_{27} выбор недостаточно большого радиуса отсечки ($R_c = 8$ Å против $R_c = 20$ Å) приводит к появлению хорошо заметных всплесков на графике зависимости энергии от времени, что свидетельствует о недостаточной точности обучения.

Рис. 1: График зависимости $E(t)$ для $R_c = 20$ Å

Рис. 2: График зависимости $E(t)$ для $R_c = 8$ Å

В то же время, как можно видеть в сравнительной таблице (табл. 1), полученная закономерность характерна только для комбинации функций G^2 и G^5, выбранной в программе Fortnet по умолчанию. Использование альтернативной комбинации G^1 и G^4 приводит к противоположным результатам, для неё гораздо меньшее значение ошибки можно наблюдать при меньшем радиусе отсечки. Следовательно, разные функции по-разному работают на разных расстояниях.

Таблица 1: Влияние радиуса отсечки на качество обучения нейросети

Радиус отсечки, Å	**Количество эпох обучения**	**Значение MSE**	**Время обучения**
	Li_{27}; 2000 фс; 27/27 нейронов; 25g2+25g5		
8	4592	1.119942E-05	12 м. 34 с.
20	4283	5.951122E-06	12 м. 11 с.
	Li_{27}; 2000 фс; 27/27 нейронов; 12g1+12g4		
8	11496	2.313880E-05	23 м. 49 с.
20	20000	1.889200E-04	39 м. 55 с.
	Li_{64}; 250 фс; 32/32 нейрона; 25g2+25g5		
6	2918	4.877322E-06	5 м. 49 с.
12	1759	5.302363E-06	4 м. 31 с.
16	1848	3.759548E-06	4 м. 55 с.
16.5	2022	3.463652E-06	5 м. 13 с.
16.75	1782	3.270775E-06	4 м. 46 с.
>=17	1	NULL	0 м. 51 с.

Также в ходе экспериментов было установлено, что для 64 атомов наблюдается порог допустимого значения радиуса отсечки, выше которого значения функций после применения нормализации становятся настолько малыми, что попытка обучить нейросеть приводит к получению NULL-значений. Однако по мере приближения к установленному максимальному пределу комбинация G^2 и G^5 демонстрирует всё возрастающую точность.

3.2. Влияние количества симметричных функций

Следующим рассматриваемым параметром стало количество симметричных функций одного и того же вида с учётом того, что часть из них должна относиться к радиальным функциям, а часть — к радиально-угловым для полноценного учёта изменения как межатомных расстояний, так и валентных углов. Для системы Li_{27} меньшее число возмущений было зафиксировано при использовании 24 симметричных функций по сравнению с 50, однако для меньшего числа функций некоторые точки остались незахваченными (ср. рис. 3 и рис. 4).

В то же время система из 50 функций отлично показала себя для Li_{64}. Попытка увеличения их числа до 100 привела к тому, что визуально были зафиксированы некоторые ненужные осцилляции по сравнению с 50. Однако измерение среднеквадратичной ошибки (см. табл. 2) показывает, что именно комбинация из 100 функций для данной системы подходит больше всего.

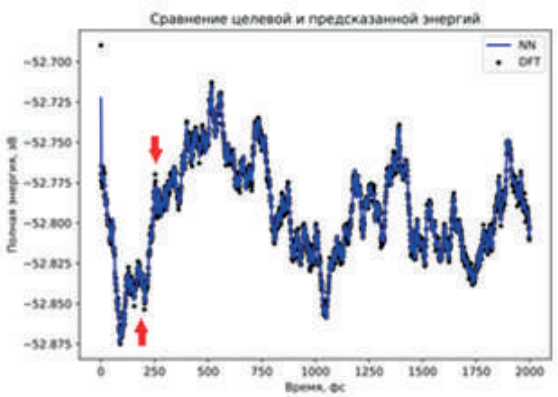


Рис. 3: График зависимости $E(t)$ для $12G^2 + 12G^5$

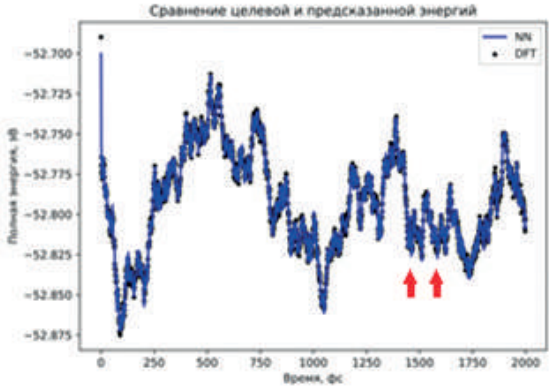


Рис. 4: График зависимости $E(t)$ для $25G^2 + 25G^5$

Таблица 2: Влияние числа симметричных функций на качество обучения нейросети

Симметричные функции	Количество эпох обучения	Значение MSE	Время обучения
Li_{27}; 2000 фс; 27/27 нейронов; радиус отсечки 12 Å			
6g2+6g5	16507	9.264668E-06	32 м. 52 с.
12g2+12g5	5173	5.467964E-06	11 м. 55 с.
25g2+25g5	4283	5.951122E-06	12 м. 12 с.
Li_{64}; 1000 фс; 32/32 нейрона; радиус отсечки 12 Å			
25g2+25g5	3029	4.479973E-06	25 м. 28 с.
50g2+50g5	2998	4.386149E-06	41 м. 45 с.
75g2+75g5	3567	4.747073E-06	62 м. 40 с.
Li_{64}; 2500 фс; 32/32 нейрона; радиус отсечки 12 Å			
25g2+25g5	3392	1.298785E-05	70 м. 21 с.
50g2+50g5	9341	1.543720E-05	301 м. 13 с.
75g2+75g5	8183	3.567265E-05	362 м. 40 с.

В целом, наблюдается необходимость увеличивать число симметричных функций при увеличении количества атомов и неизменном размере датасета.

3.3. Влияние числа нейронов скрытых слоёв

Анализ системы Li_{64} показал, что для временного интервала в 1000 фемтосекунд использование 32/32 нейронов является более оправданным, чем использование 64/64 нейронов, поскольку не вызывает лишних всплесков на графике аппроксимации энергии. Однако увеличение рассматриваемого интервала до 2500 фемтосекунд приводит к заметному изменению баланса. Сеть из меньшего числа нейронов всё ещё обучается быстрее, однако для получения большей точности возникает потребность откорректировать данный параметр в большую сторону (см. табл. 3).

Таблица 3: Влияние числа нейронов скрытых слоёв на качество обучения нейросети

Число нейронов	Количество эпох обучения	Значение MSE	Время обучения
Li_{27}; 2000 фс; 25g2+25g5; радиус отсечки 20 Å			
6/6	13676	4.774006E-05	13 м. 45 с.
13/13	5683	1.436364E-05	11 м. 01 с.
27/13	4221	5.811360E-06	10 м. 28 с.
27/27	4283	5.951122E-06	12 м. 12 с.
Li_{64}; 1000 фс; 25g2+25g5; радиус отсечки 12 Å			
32/32	3029	4.479973E-06	25 м. 28 с.
64/64	3916	4.796957E-06	86 м. 31 с.
Li_{64}; 2500 фс; 25g2+25g5; радиус отсечки 12 Å			
32/32	3392	1.298785E-05	70 м. 21 с.
64/64	4508	1.181917E-05	235 м. 16 с.

3.4. Вид симметричных функций

Вообще говоря, наиболее подходящей можно считать функцию, имеющую наименьший разброс при вычислении $E(G)$. Однако при одновременном использовании 50 симметричных функций изобразить график зависимости энергии от значений G невозможно ввиду слишком высокой размерности признакового пространства, в связи с чем оценка корректности подбора функций может быть осуществлена только на основе сравнения графиков $E(t)$ и значений численных метрик.

Использование комбинации угловых функций G^4 и G^5, в среднем, приводит к заметно более высокой точности аппроксимации, чем использование каждой из них в отдельности. К примеру, на рисунке 5 видно две точки, которые оказались захваченными нейросетью только при комбинировании этих двух функций (см. рис. 6).

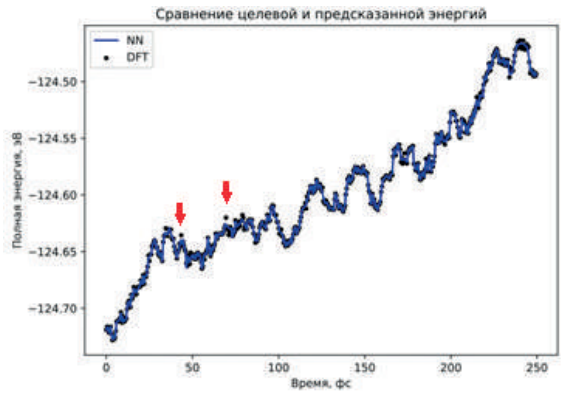


Рис. 5: График зависимости $E(t)$ для $25G^2 + 25G^4$

Рис. 6: График зависимости $E(t)$ для $25G^2 + 12G^4 + 12G^5$

При этом задание порога сходимости градиентов Threshold=1E-09 по сравнению с 1E-08 привело к меньшему значению среднеквадратичной ошибки, но потребовало значительно большего времени на обучение. Кроме того, более детальный анализ

показал, что комбинировать G^4 и G^5 следует не поровну, а в отношении 2 : 1 для получения большей точности.

Анализ поведения радиальных функций показал, что G^2 для выбранного набора данных подходит значительно лучше, чем G^1 или G^3. Сравнение результатов всех построенных моделей представлено в таблице 4.

Таблица 4: Влияние вида симметричных функций на качество обучения нейросети

Комбинация функций	**Количество эпох обучения**	**Значение MSE**	**Время обучения**
Li_{64}; 250 фс; 32/32 нейрона; 50 функций; R_c=12 Å			
12g1+12g2+12g4+12g5	3941	3.794943E-06	8 м. 40 с.
12g1+2g2+25g4	2076	3.309701E-06	5 м. 00 с.
12g1+12g2+25g5	4836	3.807081E-06	10 м. 28 с.
12g1+12g3+12g4+12g5	7045	5.471005E-06	14 м. 32 с.
12g1+12g3+25g4	6308	4.171834E-06	13 м. 40 с.
12g1+12g3+25g5	5533	6.690832E-06	11 м. 28 с.
12g2+12g3+12g4+12g5	3526	3.727510E-06	7 м. 48 с.
12g2+12g3+25g4	2999	3.992554E-06	6 м. 55 с.
12g2+12g3+25g5	3732	5.266695E-06	8 м. 12 с.
25g1+12g4+12g5	20000	1.310946E-05	40 м. 21 с.
25g1+25g4	19999	1.067284E-04	39 м. 51 с.
25g1+25g5	19999	1.858414E-04	39 м. 49 с.
25g2+12g4+12g5	1591	2.410607E-06	3 м. 56 с.
25g2+12g4+12g5	6452	6.871266E-07	13 м. 25 с.
25g2+12g4+12g5	50000	2.896467E-07	244 м. 48 с.
25g2+16g4+8g5	1660	1.687497E-06	4 м. 04 с.
25g2+18g4+12g5	1831	1.051187E-05	4 м. 40 с.
25g2+25g4	2207	2.294304E-06	5 м. 27 с.
25g2+25g5	2000	4.558489E-06	4 м. 43 с.
25g3+12g4+12g5	7231	7.038860E-06	15 м. 15 с.
25g3+25g4	19999	1.035068E-04	40 м. 03 с.
25g3+25g5	17115	5.151634E-06	35 м. 25 с.
8g1+8g2+8g3+12g4+12g5	3339	9.804373E-06	7 м. 25 с.
8g1+8g2+8g3+25g4	3339	1.130260E-05	7 м. 43 с.
8g1+8g2+8g3+25g5	4831	7.278316E-06	10 м. 31 с.

4. Заключение

На основе датасета для кристалла лития, полученного методом квантовой молекулярной динамики, показано, что с помощью нейронной сети возможно аппроксимировать значение потенциала системы в каждый момент времени, основываясь только на данных о взаимных межатомных расстояниях и валентных углах. Переводя декартовы координаты атомов в базисный набор симметричных функций Бехлера-Парринелло, программа учитывает трансляционную и вращательную симметрию по-

лучающихся атомных конфигураций, что делает возможным использование результатов обучения нейросетевой модели в классической молекулярной динамике.

Тем не менее, существует набор параметров, оказывающих непосредственное влияние на степень совпадения исходной и предсказанной энергий. В частности, большое значение имеет выбор радиуса отсечки. Для комбинации симметричных функций G^2 и G^5 наблюдается уменьшение значения среднеквадратичной ошибки в процессе его увеличения. Однако существует предельно допустимое значение, выше которого радиус отсечки задать невозможно, поскольку значения симметричных функций после нормализации становятся слишком малыми, в результате чего на выходе нейросети получаются NULL-значения.

В процессе увеличения числа атомов существенно увеличивается требуемое число функций Бехлера-Парринелло для описания более сложного характера изменения энергии с течением времени. В то же время увеличение размера датасета, выражающееся в удлинении рассматриваемого временного интервала, приводит к необходимости увеличить число нейронов в скрытых слоях нейронной сети.

На данный момент наиболее эффективным как для 27, так и для 64 атомов лития оказалось использование комбинации функций G^2, G^4 и G^5, причём для Li_{64} их следует взять в пропорции $25 : 16 : 8$. Наименее точные результаты нейросеть продемонстрировала при использовании комбинаций G^1 и G^4, G^1 и G^5, взятых в пропорции $1 : 1$.

В дальнейшем планируется расширить набор исследуемых систем, чтобы выявить новые закономерности в использовании различных параметров нейронной сети и функций Бехлера-Парринелло, а также использовать обученную нейросеть в моделировании методом классической молекулярной динамики для проверки применимости построенных моделей на практике.

Список литературы

[1] Marx D., Hutter J. Ab initio Molecular Dynamics: Basic Theory and Advanced Methods // Cambridge University Press. 2009. Vol. 63, No. 3. 567 p. https://doi.org/10.1017/CBO9780511609633.

[2] Kirchner B., di Dio P.J., Hutter J. Real-World Predictions from Ab Initio Molecular Dynamics Simulations // Topics in Current Chemistry. 2011. Vol. 307. P. 109–153. https://doi.org/10.1007/128_2011_195.

[3] Kyuhyun L., Dongsun Y., Wonseok J. et al. SIMPLE-NN: An efficient package for training and executing neural-network interatomic potentials // Computer Physics Communications. 2019. Vol. 242. P. 95–103. https://doi.org/10.1016/j.cpc.2019.04.014.

[4] Blank T.B., Brown S.D., Calhoun A.W. et al. Neural network models of potential energy surfaces // The Journal of Chemical Physics. 1995. Vol. 103, No. 10. P. 4129–4137. https://doi.org/10.1063/1.469597.

[5] Lindsey R.K., Fried L.E., Goldman N. ChIMES: A force matched potential with explicit three body interactions for molten carbon // Journal of Chemical Theory and Computation. 2017. Vol. 13, No. 12. 8 p. https://doi.org/10.1021/acs.jctc.7b00867.

[6] Lindsey R.K., Fried L.E., Goldman N. et al. Active learning for robust, high-complexity reactive atomistic simulations // The Journal of Chemical Physics. 2020. Vol. 153, No. 13. 16 p. https://doi.org/10.1063/5.0021965.

[7] Behler J. Constructing High-Dimensional Neural Network Potentials: A Tutorial Review // International Journal of Quantum Chemistry. 2015. Vol. 115, No. 16. 19 p. https://doi.org/10.1002/qua.24890.

[8] Behler J. Atom-centered symmetry functions for constructing high-dimensional neural network potentials // The Journal of Chemical Physics. 2011. Vol. 134. No. 3. 14 p. https://doi.org/10.1063/1.3553717.

[9] Van der Heide T., Kullgren J., Broqvist P. et al. Fortnet, a software package for training Behler Parrinello neural networks // Computer Physics Communications. 2023. Vol. 284. 13 p. https://doi.org/10.5281/zenodo.5969907.

[10] Потапов А.Г., Романов А.В. Моделирование межатомного потенциала по результатам квантовой-химического расчёта с использованием симметричных функций Бехлера-Парринелло // Труды молодых ученых факультета компьютерных наук ВГУ. Том Выпуск 4. Воронеж, 2024. С. 215–222.

[11] Larsen A.H., Mortensen J.J., Blomqvist J. et al. The Atomic Simulation Environment—A Python library for working with atoms // J. Phys.: Condens. 2017. Vol. 29, No. 27. 31 p. https://doi.org/10.1088/1361-648X/aa680e.

[12] Mortensen J.J., Larsen A.H., Kuisma M. et al. GPAW: An open Python package for electronic structure calculations featured // J. Chem. Phys. 2024. Vol. 160, No. 9. 42 p. https://doi.org/10.1063/5.0182685.

[13] Mortensen J.J., Hansen L.B. Jacobsen K.W. Real-space grid implementation of the projector augmented wave method // Phys. Rev. B. 2005. Vol. 71, No. 3. 11 p. https://doi.org/10.1103/PhysRevB.71.035109.

От «Трактата о числах» 1136 года до «Эпохи Келдыша»: «Ракетно-ядерный щит» и становление цифровой цивилизации

Т.А. Сушкевич

Институт прикладной математики им. М.В. Келдыша РАН

В современных исторических условиях тектонических турбулентных геополитических перемен в развитии человечества в 2025 году, когда на смену долго правящего миром либерального глобализма, доведшего планету до края угрозы третьей мировой ядерной войны, а население обрекли на вымирание и деградацию, стоят три консервативные исторические задачи для каждого государства с национальными интересами: обеспечение безопасности, суверенитета и консолидация народа. В России крайне важно возродить культурный код народа и напомнить о создании «Ракетно-ядерного щита» — достижениях отечественной науки и техники, которые заложили фундамент «цифровой» и «космической» цивилизации. В год 80-летия Великой Победы в Великой Отечественной войне актуально триумфальное достижение «Эпохи Келдыша» в покорении космоса: в разгар «холодной войны» 50 лет назад 17 июля 1975 года на орбите состоялась «стыковка» «Союза» и «Аполлона» и произошло первое в истории человечества символическое международное «рукопожатие» советских и американских космонавтов в космосе над рекой Эльбой, где за 30 лет до этого события в 1945 году состоялась встреча американских войск с частями Красной армии. Первая советско-американская Программа ЭПАС подготовлена и реализована по инициативе и под личную ответственность Президента АН СССР академика М.В. Келдыша со стороны СССР, а со стороны США — NASA (National Aeronautics and Space Administration) — Национального управления США по аэронавтике и исследованию космического пространства, Агентства, принадлежащего правительству США и занимающегося авиационными, космическими и астрофизическими исследованиями. Уникальные баллистические расчеты выполнялись в «Институте Келдыша» под руководством М.В. Келдыша и Д.Е. Охоцимского. В научной программе ЭПАС участвовала Т.А. Сушкевич: исследовалось воздействие на все среды 20-летней Вьетнамской войны 1955–1975 гг., где использовали «экологическое» и «климатическое» оружие и снаряды с напалмом, вызывающим масштабные пожары. Чрезвычайно актуально для современных войн.

Ключевые слова: М.В. Келдыш, Эпоха Келдыша, 50-летие ЭПАС, рукопожатие СССР-США в космосе, математика, ЭВМ, ракетно-ядерный щит, космические исследования, последствия войны, 80-летие Великой Победы, ИКИ, АН СССР, научное наследие

1. Введение

В 2025 году среди многочисленных масштабных торжественных юбилейных памятных мероприятий, посвященных 80-летию ВЕЛИКОЙ ПОБЕДЫ рабоче-крестьянской Красной армии СССР (РККА) и советского народа в Великой Отечественной войне 1941–1945 гг., пожалуй, два самых главных: Парад Победы на Красной площади в Москве 9 мая и триумф «Эпохи Келдыша» — 50-летие советско-американской Программы ЭПАС с ПЕРВЫМ международным «рукопожатием в космосе» над рекой Эльба. 80 лет назад на реке Эльба встретились ПОБЕДИТЕЛИ — советские и американские воинские части. В честь 30-летия окончания войны 15 июля 1975 года с космодрома «Байконур» успешно стартовал «Союз-19» с космонавтами Алексеем Леоновым и Валерием Кубасовым; вечером с космодрома на мысе Канаверал в космос на «Аполлон» отправились астронавты Томас Стаффорд, Вэнс Бранд, Дональд Слейтон. 17 июля в 19:12 «Союз-19» (НПО «Энергия») и «Аполлон» (Североамериканский Рокуэлл) успешно состыковались, создав первый в мире международный орбитальный комплекс — прообраз будущей МКС. Командиры кораблей Алексей Леонов и Томас Стаффорд первыми пожали руки.

По словам Л.И. Брежнева: «Советские и американские космонавты отправятся в космос для проведения первого в истории человечества крупного совместного научного эксперимента. Они знают, что из космоса наша планета выглядит ещё красивее. Она достаточно велика, чтобы мы могли мирно жить на ней, но слишком мала, чтобы ей угрожала ядерная война.»

В США официально назвали эту миссию «Аполлон»–«Союз» (англ. Apollo–Soyuz Test Project (ASTP)), а в СССР — ЭПАС (Экспериментальный полет «Союз»–«Аполлон»). О Программе ЭПАС — предшественнице Программы МКС — необходимо помнить, чтобы по достоинству оценить значимую роль учёных и лично академика М.В. Келдыша в снижении напряженности советско-американских отношений в годы разрядки «холодной войны» и в должной мере воспринимать роль международного научно-технического сотрудничества на МКС в настоящее время.

Обе стороны признали ASTP — ЭПАС политическим актом, направленным на достижение мира. Этот проект и это «рукопожатие» в космосе были символом разрядки между двумя сверхдержавами в период «холодной войны», которая началась фактически сразу после окончания второй мировой войны. Об этом известили весь мир Уинстон Черчилль и Гарри Трумэн. Свою Речь Черчилль, Фултон, штат Миссури, США, 5 марта 1946 г., начал с мирных слов «Я счастлив, что прибыл сегодня в Вестминстерский колледж и что вы присвоили мне ученую степень. . . », но значительную часть речи посвятил опустившемуся «железному занавесу». 12 марта 1947 г. президент США Трумэн выступил в сенате со своей знаменитой Доктриной, которая призывала сдерживать СССР на международной арене. Речь Трумэна ясно давала понять, что главная цель Соединённых Штатов — стать единственным экономическим и политическим центром мира.

Началась «гонка вооружений» и после американских «ядерных» ударов 6 и 9 августа 1945 г. по Японии возросли угрозы мировой ядерной войны. Советским ответом на этот вызов были успешные «Атомный» и «Космический» проекты и создание «Ракетно-ядерного щита», реализованные в кратчайшие сроки. Недруги просчитались — не ожидали такой мобилизации и успехов. В итоге СССР стал ПЕРВЫМ в покорении космоса и более 10 лет оставался лидером в космической гонке. ЭПАС положил начало разрядке и заключению Хельсинского Акта о безопасности.

Сейчас опять угрозы войны, потому важно изучать исторический опыт и великое наследие отечественной науки и техники, чтобы защитить и спасти советскую-русскую цивилизацию, которая единственная могла противостоять США и его союзникам. Более того, благодаря русским в середине 20-го века заложили основы такого научно-технологического прогресса, в итоге которого в 21 веке совершен взрывной скачок и планета живет в условиях «цифровой» и «космической» цивилизации. Вторая мировая война и «холодная война» явились триггерами и ведущими драйверами такого интенсивного развития науки, техники, технологий, которое никогда ранее не встречалось. Во главе этого процесса в СССР стояли три РУССКИХ ГЕНИЯ — «Три К» — самый эффективный коллектив: математик М.В. Келдыш, инженер-конструктор С.П. Королев, физик И.В. Курчатов, а ключевую роль стали играть «Математика как производительная сила» и ЭВМ. На ближайшее десятилетие определились приоритеты и в гражданских и в военных секторах: искусственный интеллект, суперкомпьютеры, квантовые вычисления, космос, Арктика.

Победа в войне и открытие космической эры — два самых значимых достижения СССР, которыми гордятся русские, но они навсегда повлияли на судьбы народов планеты Земля. **Значимая роль в этих достижениях принадлежит ОТЕЧЕСТВЕННОЙ НАУКЕ и прежде всего Академии наук СССР, которая расцвела и стала форпостом СССР во всем мире, когда 19 мая 1961 года в возрасте 50 лет Президентом АН СССР единогласно избрали М.В. Келдыша!**

«Эпоха Келдыша» продолжается: этим юбилеям посвящен старт 8 апреля 2025 года в 08:47 мск «Ракеты Победы» «Союз 2.1» с пилотируемым кораблем «Союз-МС-27» для доставки на МКС экипажа МКС-73. На борту космического корабля два российских космонавта и американский астронавт: Сергей Рыжиков и Алексей Зубрицкий от Роскосмоса, Джонатан Ким от NASA.

2. «Эпоха Келдыша»: международное сотрудничество СССР–США и 50-летний юбилей триумфа ЭПАС в космосе в 1975 году

Актуально напомнить о триумфальном достижении «Эпохи Келдыша» в покорении космоса: в разгар «холодной войны» весь мир с воодушевлением пристально наблюдал за ЭПАС-ASTP — эпохальным международным событием и забыли про войну. Ученые, инженеры, конструкторы и рабочие своими фантастическими достижениями фактически снимали напряженность международных отношений и демонстрировали возможности научно-технического сотрудничества в интересах мира на планете. Символично: благодаря ювелирным расчетам «стыковка» и «рукопожатие» на орбите в космосе произошли над рекой Эльба, где в 1945 году произошло «рукопожатие» советских и американских воинов, освобождавших Германию и Европу от фашистов.

Не допустимо перед отечеством и народом, когда недостоверно переписывают советскую историю и достижения и замалчивают М.В. Келдыша: Главный математик страны — ЕДИНСТВЕННЫЙ математик Трижды Герой Социалистического Труда (1956, 1961, 1971), Главный Теоретик космонавтики, Лучший Президент (1961–1975) за всю 300-летнюю историю Академии наук, основоположник «цифровой» и «космической» цивилизации. ВСПОМНИТЕ Мстислава Всеволодовича Келдыша — ВЕЛИЧАЙШЕГО РУССКОГО ГЕНИЯ — уникального и неповторимого за всю историю

Рис. 1: 80 лет назад встреча советских и американских войск на реке Эльба в 1945 г.

человечества, который для ОТЕЧЕСТВА и всей земной ЦИВИЛИЗАЦИИ сделал столько, как никто другой никогда: «чистый математик» написал формулки и при своей жизни реализовал МЕЧТУ ВСЕГО ЧЕЛОВЕЧЕСТВА о полетах в космос. Под руководством и под личную ответственность М.В. Келдыша осуществили полеты Первого спутника, Первого космонавта, Первой ДОС, Первых АМС на Луну, Венеру, Марс, в 1975 году, т.е. 50 лет назад, ПЕРВУЮ «стыковку» в космосе «Союз»-«Аполлон» и ПЕРВОЕ международное «рукопожатие» на орбите, а также создали «Ракетно-ядерный щит». СССР был впереди планеты всей!

На текущем историческом моменте тектонических турбулентных геополитических перемен в развитии человечества, начатых в 2025 году, когда на смену долго правящего миром либерального глобализма, доведшего планету до края угрозы третьей мировой ядерной войны, а человечество обрекли на вымирание и деградацию, поставлены три консервативные исторические задачи для каждого государства с национальными интересами: обеспечение безопасности, суверенитета и консолидация народа. Не случайно вышел Указ Президента Российской Федерации от 08.05.2024 № 314 “Об утверждении Основ государственной политики Российской Федерации и в области исторического просвещения”. Крайне важно возродить культурный код народа и напомнить о достижениях отечественной русской цивилизации, которой более тысячи лет. Наиболее актуальными успехами являются открытие космической эры и проект создания «Ракетно-ядерного щита», которые и ныне сдерживают агрессоров от разжигания горячей войны, после которой может исчезнуть жизнь на планете Земля.

Публикация напомнит о мировых достижениях отечественной науки НАШЕЙ ВЕЛИКОЙ ЦИВИЛИЗАЦИИ «СССР» — «золотом веке отечественной науки». Полезно знать для «исторического просвещения» в новых условиях «цифровой» и «кос-

мической» цивилизаций, фундаментальные основы которых были заложены под научным руководством М.В. Келдыша (10.02.1911—24.06.1978) — академика в 35 лет, как Л. Эйлер и А.Н. Колмогоров, и Лидера по «прикладной математике» (с 1946), Главного математика страны, ответственного за разработку ЭВМ и расчеты (с 1951), основателя ПЕРВОГО в мире Института прикладной математики АН СССР (1953), Руководителя «атомного» и «космического» проектов в составе «Три К» — М.В. Келдыш, С.П Королев (12.01.1907—14.01.1966), И.В. Курчатов (21.01.1903—07.02.1960), Научного руководителя проекта создания «Ракетно-ядерного щита» (с 1959) совместно с Д.Ф. Устиновым (30.10.1908 — 20.12.1984), А.Н. Косыгиным (21.02.1904–18.12.1980), С.Ф Ахромеевым (05.05.1923—24.08.1991), Главного Теоретика космонавтики, Президента АН СССР (1961–1075), Председателя МНТКС по КИ в статусе министра и генерала (1959–1978), выдающегося государственного и общественного деятеля. Все руководители проекта — Герои Социалистического Труда.

«Ракетно-ядерный щит» возможно было создать только тогда, когда математика достигла должного уровня и подготовили научные и инженерные кадры, в том числе математиков, программистов, кибернетиков, создали ЭВМ и покорили космос, чтобы оперативно обнаруживать старты ракет в трех средах. В рамках этого проекта впервые были разработаны системы искусственного интеллекта (ВЦ и ИПУ АН СССР). Советская наука оставила великое наследие и только СССР и США имели такие привилегии, что могли конкурировать между собой и с наукой всего мира. Но была большая разница в подготовке кадров: СССР готовил кадры, используя исторические традиции в образовании, науке, культуре, а в США во время и после второй мировой войны собирали специалистов со всего мира. Только в СССР возникло поколение «физики и лирики» — высокообразованные просвещенные культурные многогранно и гармонично развитые специалисты и интеллигенты из поколения «Дети войны». Ветераны-участники войны и «Дети войны» свою Родину сделали Великой Державой, которую не только боялись, но и уважали!

Автора интересует роль М.В. Келдыша, который стоял у истоков и отвечал за математику, ЭВМ, расчеты с 1946 и с 1951 гг., в условиях “холодной войны” и “гонки — конкуренции между СССР и США” по ключевым областям научно-технической революции в середине 20 века: развитие авиации, покорение атома, покорение космоса, разработка ЭВМ, старт “IT-технологий”.

После первых запусков спутников, первых пилотируемых полетов, АМС при ООН были созданы международные организации для регулирования космической деятельности. Поскольку М.В. Келдыш с 1939 года был чрезмерно засекреченным и был невыездным, зарубежные связи и контакты осуществляли Л.И. Седов и К.Я. Кондратьев. После избрания 19 мая 1961 года Президентом АН СССР М.В. Келдыш фактически инициировал и развивал международные связи по проблемам космоса под прикрытием Президента АН СССР, который с 1959 года был и Председателем МНТС по КИ при АН СССР. После смерти С.П. Королева в 1966 году и выступления с прощальной речью с трибуны мавзолея на похоронах С.П. Королева на Красной площади весь мир узнал, кто же скрывается за словами «Главный Теоретик космонавтики».

Впервые идея международного сотрудничества в космосе была высказана К.Э. Циолковским в 1920 году в научно-фантастической повести “Вне Земли”, где он излагал программу подготовки к космическому путешествию и его осуществлению. В этой публикации Циолковский высказал мысль о том, что целесообразнее

всего покорять и осваивать космос силами международного коллектива ученых и инженеров.

Через 40 лет С.П. Королев в статье "Творчество, воодушевлённое Октябрём", опубликованной в газете "Правда" 10 ноября 1960 года писал: "Нет сомнения в том, что не за горами и то время, когда могучие космические корабли весом во много десятков тонн, оснащенные всевозможной научной аппаратурой, с многочисленным экипажем, покинут Землю и, подобно древним аргонавтам, отправятся в далекий путь. Они отправятся в заоблачное путешествие, в многолетний космический рейс к Марсу, Венере и другим далеким мирам. Можно надеяться, что в этом благородном, исполинском деле будет все более расширяться международное сотрудничество ученых, проникнутых желанием трудиться на благо всего человечества, во имя мира и прогресса".

Первое соглашение о сотрудничестве в области мирного изучения космоса между Академией Наук СССР и НАСА было подписано в июне 1962 года.

А в январе 1971 года в Москве Президент АН СССР академик М.В. Келдыш и исполняющий обязанности директора НАСА доктор Дж. Лоу подписали документ о совместной деятельности в области космической физики, космической метеорологии, изучения природной среды, космической биологии и медицины.

24 мая 1972 г. во время визита в Москву президента США было подписано межправительственное соглашение между СССР и США о сотрудничестве в исследовании и использовании космического пространства в мирных целях, ставшее основанием для развертывания работ по программе совместного пилотируемого космического полета.

Первая попытка советско-американского сотрудничества в космосе датируется 1962 годом, когда в Академию Наук СССР пришло письмо из-за океана с предложением совместного проекта покорения Луны. Письмо это так и осталось без ответа — во-первых, в очередной раз испортились двусторонние отношения, во-вторых, советская космическая программа успешно двигалась вперёд и без помощи американцев. Да и американцы, уязвлённые тем, что уступили лидерство в космосе СССР, сосредоточились на покорении Луны в одиночку.

Вновь о сотрудничестве стороны задумались десять лет спустя. Американцы, уступив практически все «главные призы космической гонки», отыгрались на Луне, удовлетворив своё самолюбие. Первая советско-американская Программа ЭПАС была подготовлена и реализована по инициативе и при активном участии Президента АН СССР М.В. Келдыша со стороны СССР, а со стороны США — NASA (National Aeronautics and Space Administration) — Национального управления США по аэронавтике и исследованию космического пространства, Агентства, принадлежащего правительству США и занимающегося авиационными, космическими и астрофизическими исследованиями.

Инициатором проведения совместного полёта американского и советского пилотируемых космических кораблей со стыковкой на орбите выступило NASA. Эту идею высказал директор NASA Томас Пейн в начале 1970 года в ходе переписки с президентом Академии наук СССР М.В. Келдышем. Были образованы рабочие группы для согласования технических требований по обеспечению совместимости существующих на тот момент советского и американского кораблей — «Союза» и «Аполлона».

В 1966–1971 гг. Николай Петрович Каманин занимал должность помощника Главнокомандующего ВВС по космосу. Из дневника Н.П. Каманина **4 октября 1970 года:** ***«Мне позвонил Келдыш и, сообщив, что к нам едут шесть специалистов из США, попросил продумать процедуру их приема в ЦПК.***

Они будут вести переговоры с нашими специалистами по унификации стыковочных узлов советских и американских космических кораблей для обеспечения стыковки их в космосе с целью оказания помощи экипажу корабля, терпящего бедствие.»

ПЕРВАЯ встреча советских и американских специалистов по проблемам совместимости средств сближения и стыковки пилотируемых космических кораблей и станций проходила в Москве **26–28 октября 1970 года** — это был **Первый официальный визит делегации NASA в Советский Союз.** Делегации возглавляли: американскую — директор центра пилотируемых полетов им. Джонсона доктор Р. Гилрут, советскую — председатель Совета «Интеркосмос» при АН СССР академик Б.Н. Петров. Были образованы рабочие группы для выработки и согласования технических требований по обеспечению совместимости советских и американских кораблей.

В январе 1971 года исполняющий обязанности директора NASA доктор Дж. Лоу прибыл во главе американской делегации в Москву и на встрече с советской делегацией, возглавляемой Президентом АН СССР академиком М.В. Келдышем, предложил провести совместный пилотируемый испытательный космический полет. М.В. Келдыш выразил принципиальное согласие. Был подписан документ о совместной деятельности в области космической физики, космической метеорологии, изучения природной среды, космической биологии и медицины. Так создавался механизм советско-американского космического сотрудничества.

В июне 1971 года, в Хьюстоне и **в ноябре 1971 года**, в Москве состоялись очередные встречи специалистов АН СССР и NASA США. Делегации по-прежнему возглавляли Б.Н. Петров и Р. Гилрут.

В апреле 1972 года, в Москве американская делегация во главе с Дж. Лоу и советская делегация, возглавляемая исполняющим обязанности Президента АН СССР академиком В.А. Котельниковым, проанализировали работу, проделанную за прошедший период. Практическое начало экспериментальному проекту «Союз»–«Аполлон» было положено **6 апреля 1972 года** «Итоговым документом встречи представителей Академии наук СССР и NASA США по вопросу создания совместимых средств сближения и стыковки пилотируемых космических кораблей и станций СССР и США», в котором был сделан вывод о технической осуществимости и желаемости экспериментального полета с использованием существующих космических кораблей: советского — типа «Союз» и американского — типа «Аполлон».

Во время визита в Москву 24 мая 1972 года президент США Ричард Никсон и Председатель Совета министров СССР Алексей Николаевич Косыгин подписали исторический документ «Соглашение между Союзом Советских Социалистических Республик и Соединенными Штатами Америки о сотрудничестве в исследовании и использовании космического пространства в мирных целях», в котором была утверждена Программа ЭПАС — Экспериментальный полёт «Аполлон»—«Союз» (или более распространённое у нас название «Союз»—«Аполлон»). В третьей статье Соглашения записано: *«Стороны договорились о проведении работ по созданию совместимых средств сближения и стыковки советских и американских пилотируемых космических кораблей и станций с целью повышения безопасности полётов человека в космос и обеспечения возможности осуществления в дальнейшем совместных научных экспериментов. Первый экспериментальный полёт для испытания таких средств, предусматривающий стыковку советского космического корабля типа «Союз» и американского*

космического корабля типа «Аполлон» с взаимным переходом космонавтов, намечено провести в течение 1975 года». 1975 г. — год 30-летия Великой Победы в мировой войне, где СССР и США были союзниками!

Техническими директорами ЭПАС были назначены с советской стороны член-корреспондент АН СССР Константин Давыдович Бушуев, заместитель М.В. Келдыша — Председателя МНТС по КИ АН СССР, и с американской — Глинн Ланни, руководителями полёта соответственно — лётчик-космонавт СССР Алексей Станиславович Елисеев и Питер Франк.

Как писала газета «Правда», в Москве «**7 июля 1972 года** парафированием протокола завершились переговоры между заместителем Председателя Совета Министров СССР, Председателем Государственного комитета Совета Министров СССР по науке и технике В.А. Кириллиным и советником президента США по вопросам науки Э. Дэвида. Стороны определили условия и первоначальные области развития контактов в рамках соглашения между правительствами СССР и США о сотрудничестве в области науки и техники». В тот же день в Президиуме АН СССР Э. Дэвид был на приеме у Президента Академии наук СССР академика М.В. Келдыша. На встрече советских и американских специалистов в Хьюстоне **6–18 июля 1972 года** был намечен конкретный план полёта кораблей «Союз» и «Аполлон» в 1975 году: первым стартует корабль «Союз» с двумя космонавтами; примерно через 7,5 часа стартует корабль «Аполлон» с тремя астронавтами; через сутки (окончательный вариант — через двое суток) после старта корабля «Аполлон» производятся сближение и стыковка; длительность полёта кораблей в состыкованном состоянии — около двух суток.

В октябре 1972 года Ф. Хендлер (F. Handler) — Президент Национальной Академии наук США (англ. U.S. NAS — United States National Academy of Sciences — ведущая научная организация США, образована 3 марта 1863 года актом Конгресса, подписанным президентом Авраамом Линкольном) пригласил М.В. Келдыша и группу академиков (А.Н. Прохоров, И.М. Макаров, Г.И. Марчук, Ю.А. Овчинников) посетить Национальную Академию наук США и познакомиться с лучшими научными центрами. Это был **первый официальный визит делегации АН СССР в Америку** и проходил он на высоком уровне. М.В. Келдыш в США бывал и раньше, но в этот раз в сопровождении посла СССР Анатолия Федоровича Добрынина («дипломатическая дружба» связывала его с Генри Киссинджером) М.В. Келдыш был приглашен на прием у Президента США Р. Никсона, где присутствовали Ф. Хендлер, советник Президента США Г. Киссинджер (*G. Kissinger*) и Джеймс К. Флетчер (*James C. Fletcher*) — Глава администрации NASA с апреля 1971 по май 1977 гг., который был ответственным за ЭПАС со стороны США. Г. Киссинджер, известный американский государственный и политический деятель, исследователь международных отношений, в книге своих воспоминаний «Годы в Белом доме» одну из глав назвал «Американо-советские отношения как перманентная философская проблема». Суть этой философии в следующем. Исторический опыт США слабо подготовил американцев к ведению дел на постоянной основе со столь мощным противником, каким был Советский Союз. США взяли на себя историческую ответственность за поддержание равновесия сил, хотя были весьма плохо подготовлены к выполнению этой задачи. Поддержание равновесия сил — перманентная миссия, а отнюдь не приложение усилий в расчете на какой-то заранее предсказуемый срок. Киссинджер говорит об изменении природы силы, которое было вызвано появлением ядерного оружия. Можно добавить, что **межконтинентальные баллистические ракеты и космические аппараты тоже должны были изменить природу силы**. Понимание

важности космической компоненты силы постепенно приходило к политикам и в США, и в СССР.

В июле 1973 года в Москве состоялась встреча делегации США во главе с профессором Ф. Хэндлером с руководством АН СССР. 17 июля М.В. Келдыш лично встречал делегацию в аэропорту. В 1974 году в Москву прибыл Джеймс К. Флетчер с визитом в АН СССР. М.В. Келдыш владел иностранными языками и при общении с высшим руководством США и др. обходился часто без переводчика, что располагало собеседников к непринужденному не официозному общению.

Что касается науки и техники, то **в середине 1973 года** Р. Никсон упразднил должность научного советника президента, как не приносящую реальной пользы, и передал его полномочия Национальному научному фонду (**NSF** — *National Science Foundation*) — независимому Агентству при правительстве США, отвечающему за развитие науки и технологий. NSF и в нынешнее время осуществляет свою миссию, предоставляя в основном временные гранты для поддержки творческой инициативы ученых. Р. Никсон упразднил Национальный совет по аэронавтике и космическому пространству, контролировавший всю соответствующую деятельность федеральных органов и действующий как часть канцелярии президента, его членами были вице-президент, министр обороны, министр иностранных дел, администратор NASA.

«Зеленый свет» для Программы ЭПАС открыл 37-й президент США Р. Никсон, который был вице-президентом ещё при президенте Дуайт Дейвиде Эйзенхауэре (*Eisenhower*), а завершалась Программа успешно в 1975 году при Джеральде Форде (*Gerald Rudolph Ford*) — 38-м президенте США (с 09 августа 1974 по 20 января 1977); тоже от республиканской партии. Л.И. Брежнев и Дж. Форд провожали космонавтов своих стран в этот полет и после стыковки обратились к экипажам кораблей с посланиями, в которых особо отмечалось, что «никогда ранее представители двух стран не жили и не работали в космосе совместно», подчеркивались историческая значимость этой миссии и демонстрация того, что «Соединенные Штаты и Советский Союз могут сотрудничать в столь важном деле».

В архиве Мемориального Музея-кабинета академика М.В. Келдыша, расположенного в историческом Главном здании Института прикладной математики им. М.В. Келдыша РАН, где в Актовом зале проходили заседания научно-технических советов, хранится убедительное свидетельство признания исключительной роли М.В. Келдыша в Программе ЭПАС — фотография старта корабля «Аполлон» с автографом руководителей НАСА (рис. 2).

«Академику Келдышу.
БЕЗ ВАШЕГО ОДАРЕННОГО ВООБРАЖЕНИЕМ РУКОВОДСТВА
«АПОЛЛО-СОЮЗ» БЫЛ БЫ НЕВОЗМОЖЕН.
С НАШИМ ВЕЛИЧАЙШИМ УВАЖЕНИЕМ
И ОГРОМНЫМ ПОЧТЕНИЕМ.
Джеймс Флэтчер и Джордж Лоу (Администраторы NASA),
15 июля 1975 года.»

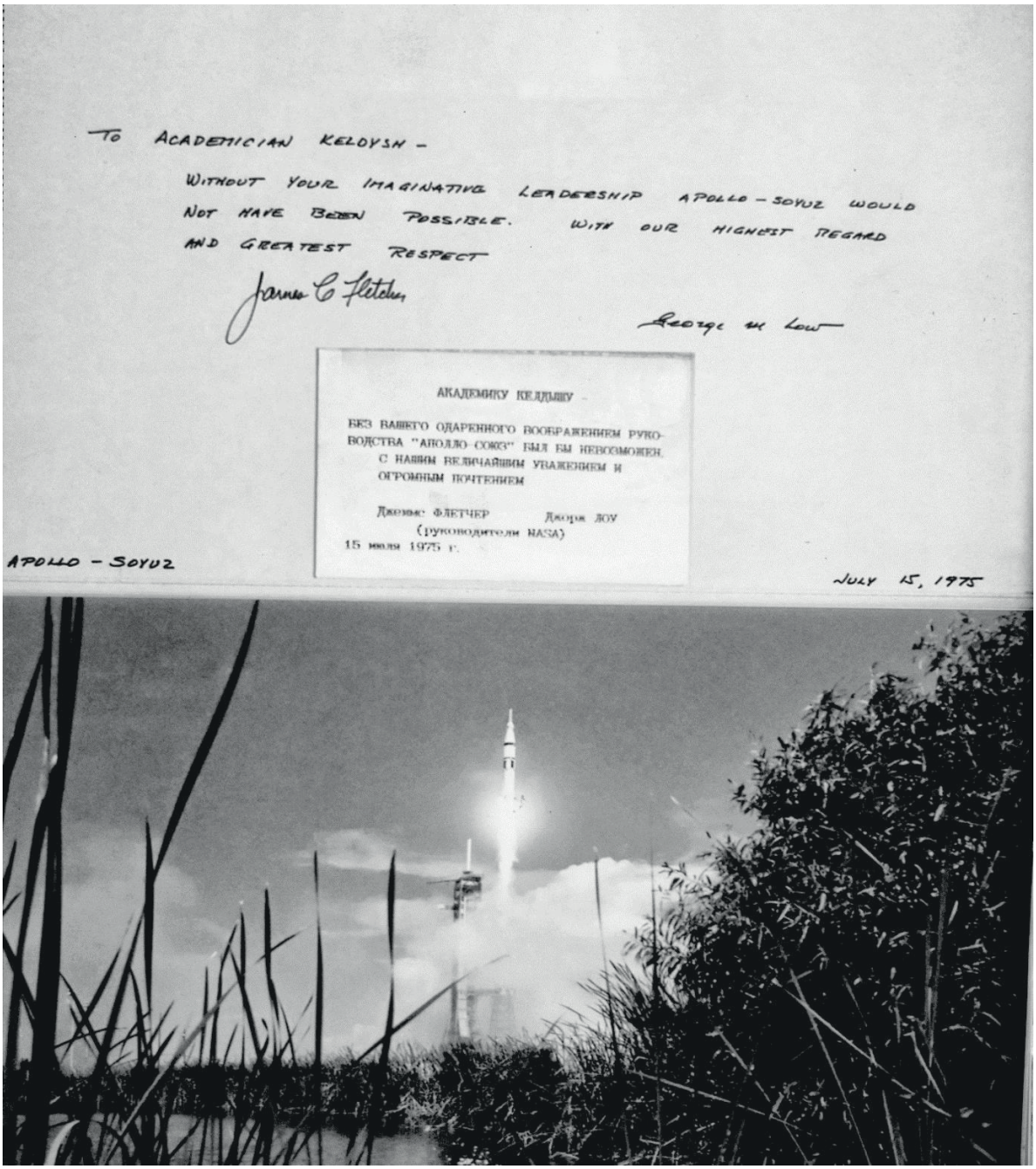


Рис. 2: Фото старта «Аполлона» с автографами администраторов НАСА. Музей М.В. Келдыша.

В Мемориальном музее-кабинете академика М.В. Келдыша сохранились уникальные документы, свидетельствующие о роли Президента АН СССР и Председателя МНТС по КИ при АН СССР в осуществлении триумфального эпохального космического проекта [1–6].

В ЭПАС участвовали коллектив сотрудников и Баллистический центр Института Келдыша, опыт которого был использован при создании и функционировании МКС. М.В. Келдыш лично участвовал в обсуждении многих технических вопросов ЭПАС в НПО «Энергия», руководил которым генеральный конструктор В.П. Глушко. Курировал работу лично Министр общего машиностроения С.А. Афанасьев. Уникальные баллистические расчеты без дублирования в других организациях выполнялись в

Рис. 3: Экипажи и автографы. Музей М.В. Келдыша.

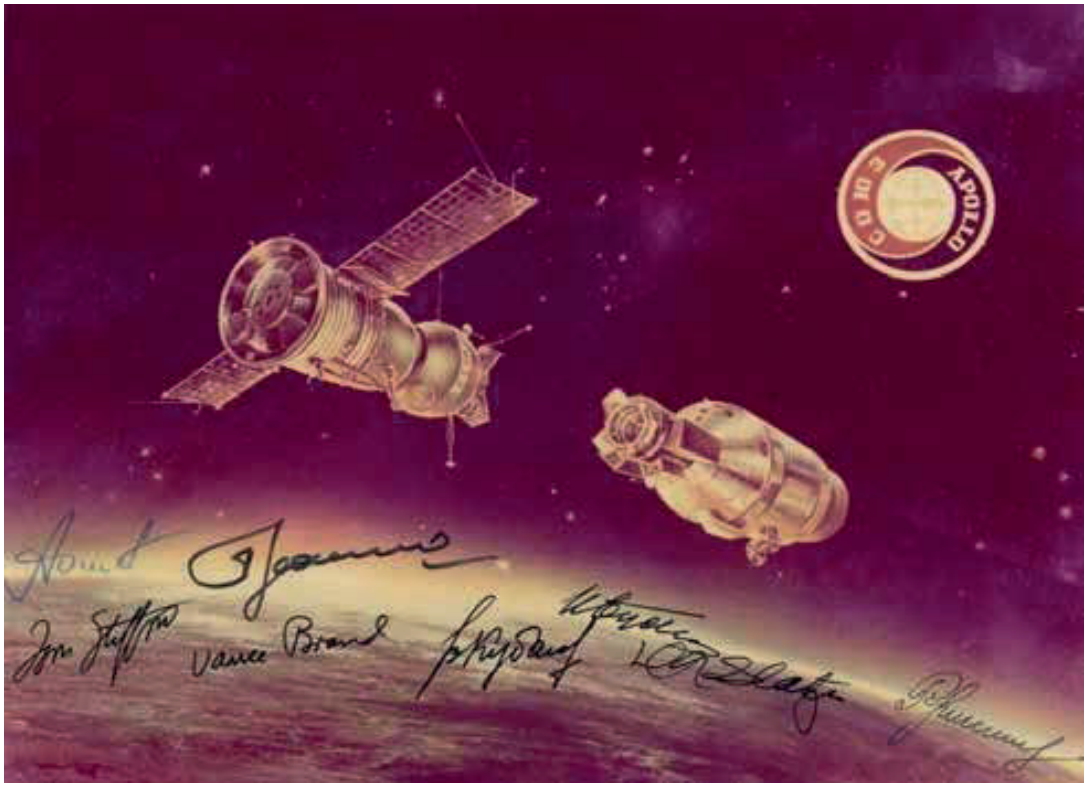


Рис. 4: «Союз» и «Аполлон» с автографами экипажей. Музей М.В. Келдыша.

«Институте Келдыша» под руководством М.В. Келдыша и Д.Е. Охоцимского. Процесс «стыковки» обеспечивал Институт проблем управления АН СССР во главе с академиком Б.Н. Петровым — научным руководителем советской части Программы ЭПАС. С 1966 г. Борис Николаевич работал на посту первого Председателя Совета по международному сотрудничеству в области исследования и использования космического пространства в мирных целях (Совет «Интеркосмос») при АН СССР.

Рис. 5: Свидетельство о первой международной стыковке в космосе. Музей М.В. Келдыша.

В 2025 году 60-летний юбилей отмечает Институт космических исследований РАН — Космический центр, созданный в 1965 году по личной инициативе М.В. Келдыша для организации и проведения в ом числе международного сотрудничества в космосе. В ЭПАС участвовали специалисты из секретных организаций, некоторые из которых на время выполнения проекта были зачислены сотрудниками ИКИ, чтобы иметь возможность открыто общаться с зарубежными коллегами. Рабочие встречи участников проекта проходили на территории ИКИ.

В научной программе ЭПАС участвовала автор публикации Т.А. Сушкевич [7] совместно с Институтом физики атмосферы АН СССР и Абастуманской обсерваторией, Грузия: проводилось исследование воздействия 20-летней (1955–1975) войны во Вьетнаме на окружающую среду и климат, где кроме «ковровых бомбежек» с широким применением напалма, выжигавших десятки гектаров земли, были апробированы сценарии «климатической» и «экологической» войны. В итоге в 1975 году был заключен Хельсинский Акт по безопасности и сотрудничеству в Европе. Активизировались международные и национальные программы и проекты по экологии, климату, устойчивому развитию. Чрезвычайно актуальная тематика современности.

Рис. 6: Президент АН СССР М.В. Келдыш принимает руководство НАСА в Президиуме.

Рис. 7: Президент АН СССР М.В. Келдыш принимает участников ЭПАС в Президиуме.

3. Связующая цепь времен достижений в математике на пути к покорению космоса

Космос начали покорять только после того, как МАТЕМАТИКА достигла должного уровня, подготовили кадры математиков и программистов, создали ЭВМ. Ключевую роль сыграли Академия наук, в которой аккумулировались мировые научные знания за всё время существования человечества, а также ведущие отечественные университеты с лучшей в мире системой образования, в которых готовили научные и инженерные кадры высшей квалификации. Существенным фактором успеха является ментальность русских, веками покоряющих огромные пространства и отражающих агрессию внешних врагов, что способствовало развитию способностей к точным, естественным, инженерным наукам.

Формат публикации не позволяет подробно отразить отечественные достижения в математике, потому остановимся на кратком перечислении наиболее значимых персон.

ВПЕРВЫЕ прорыв в космос свершился в ВЕЛИКОЙ ДЕРЖАВЕ — СССР, где подняли роль «истории знаний», наука и высшее и всеобщее образование — это приоритет, АН СССР — штаб науки, советский народ — народ-ПОБЕДИТЕЛЬ, созидатель, готовый на трудовые подвиги для реализации фантастического проекта! Это предсказывал Э.К. Циолковский.

ПЕРВЫЙ признанный в мире русский ученый М.В. Ломоносов (08.11.1711–04.04.1765), физик, химик, астроном, математик, специалист в области горного дела, географ, историк, филолог, поэт, адъюнкт физического класса с 08.01.1742, профессор химии с 25.07.1745, в середине 18 века заявил: «Математика — царица всех наук, но служанка физики».

Немецкий математик Карл Гаусс (30.04.1777–23.02.1855), член-корреспондент с 31.01.1802, почетный член 24.03.1824 Императорской Академии наук, в 19 веке часто говорил фразу: «Математика — царица наук, арифметика — царица математики», подчеркивая важность понятия «число» в математике, и оказался прав — «число» лежит в основе «цифровой» цивилизации.

К.Э. Циолковский (17.09.1857-19.09.1935), русский, российский и советский ученый-самоучка, математик, изобретатель, писатель, в т.ч. научной фантастики, физик, философ, астроном, космолог, разрабатывал теоретические вопросы космонавтики и философские проблемы освоения космоса; членом Академии наук не был. В 1879 году сдал экзамен на звание народного учителя и до 1921 года преподавал математику и физику в училищах Боровска и Калуги, параллельно пытался заинтересовать научное сообщество своими проектами аэропланов и цельнометаллического дирижабля, а впоследствии и ракетной техники. Основные научные труды по аэронавтике, ракетодинамике и космонавтике начинались с попытки использовать математический аппарат для решения фантастических задач. К.Э. Циолковский: «Математика — могучее орудие ума» и сбылось его пророчество: именно СССР открыл космическую эру!

В.И. Вернадский (12.03.1863–06.01.1945), геолог, минералог, кристаллограф, геохимик, историк науки, выдающийся ученый и блестящий популяризатор науки, адъюнкт с 04.03.1906, экстраординарный академик с 05.04.1908, ординарный академик с 03.03.1912 в физико-математическом отделении Императорской Академии наук, активный организатор в 1925 году и вице-президент Академии наук СССР в докладе

на тему «Мысли о современном значении истории знаний», прочитанном на первом заседании Комиссии по истории знаний (КИЗ) АН СССР 14.11.26, высказал много умных и полезных мыслей, актуальных и ныне. Важно помнить о преемственности в науке: «История науки является в такие моменты орудием достижения нового», говоря о переломных моментах или острых проблемах в истории государств.

М.В. Келдыш стоял во главе покорителей космоса: Главный МАТЕМАТИК страны, который отвечал за расчеты и ЭВМ; Главный Теоретик космонавтики и Председатель МНТС по КИ при АН СССР в чине министра и генерала — идеолог космических исследований в интересах фундаментальной и прикладной науки, а также народно-хозяйственного гражданского и военно-оборонительного назначения; Президент Академии наук СССР руководил всей наукой в «золотой век науки» — поднял статус «математика — производительная сила» и для покорения космоса М.В. Келдыш со своими единомышленниками впервые с помощью «формул» и ЭВМ создал «МАТЕМАТИЧЕСКИЙ КОСМОС»!

Речь идет об «отечественной науке» — её истории и перспективах в условиях новой научно-технической «цифровой и космической цивилизации», которую в год 100-летия со дня рождения М.В. Келдыша назвали «Эпоха Келдыша». Однако с тех пор «математика» последовательно расширяла сферу приложений и в 21-м веке «математика», «цифра», «компьютеры» и «искусственный интеллект» покорили все науки и население всей планеты. «Эпоха Келдыша» продолжается...

А как это начиналось?

1136 год — Первый научный труд в нашей стране, посвященный изучению чисел “Трактат о числах” появился в Новгороде, автор доместик Антониева монастыря Кирик Новгородец.

1701-1753 — Школа математических и навигацких наук (школа Пушкарского приказа) — математическая школа для дворянских и приказных детей, первое в Российском царстве артиллерийское, инженерное и морское училище, историческая предтеча и предшественник всей современной системы инженерно-технического образования современной России, основано в Москве 14 (25) января 1701 г. по указу Петра Первого для подготовки артиллеристов, инженеров, моряков армии и флота. Школа просуществовала до 1753 г.

1703 год — “Арифметика” Магницкого Леонтия Филипповича (1669–1739) подготовлена и издана по личному распоряжению царя Петра. Москва: Синодальная тип., январь 1703, 679 с.

1712–1723 гг. — Первая Военная инженерная школа. 14 января 1712 года Петр I издает Указ о создании в Москве Инженерной школы с отделением ее от школы Пушкарского Приказа, в которой учили математическим наукам. Говоря современным языком, Петр требует усиленного изучения математики учащимися инженерного класса школы пушкарского приказа, для чего при необходимости посылать их доучиваться математике в Сухареву башню.

Бернулли Иоганн (27.07.1667–01.01.1748), математик, почетный член РАН с 30.06.1725; отец Даниила Бернулли — великого математика своего времени, самого знаменитого представителя семейства Бернулли.

Бернулли Даниил (09.02.1700–17.03.1782), в РАН профессор по “физиологии” с 05.07.1725, профессор по “математике” с 30.06.1727, иностранный почетный член с 23.03.1733. В духе механистических воззрений XVII–XVIII вв. Даниил Бернулли на кафедре анатомии и физиологии Петербургской академии наук пытался с помощью

механикоматематических методов изучать тайны живой природы. В 1729 г. подготовил свой главный труд: монографию “Гидродинамика”, который был опубликован в 1738 г.

Леонард Эйлер (04.04.1707-07.09.1783), математик, механик, физик, в РАН адъюнкт по “физиологии” с 17.12.1726, профессор по “физике” с 01.01.1731, профессор по “высшей математике” с 15.06.1733, иностранный почетный член с 04.05.1742, вторично профессор с 26.04.1766; отразил свои интересы по “прикладной математике” в области кораблестроения в двухтомнике **“Морская наука, или Трактат о строительстве и управлении кораблями”**. СПб., 1749. Леонард Эйлер невероятен! «Идеальный математик» — так его и по сей день называют многие. Им написано около двух тысяч статей по различным наукам: по математическому анализу, дифференциальной геометрии, теории чисел, приближённым вычислениям, небесной механике, математической физике, оптике, баллистике, кораблестроению, теории музыки и другим областям. Его вклад в классическую и прикладную математику, в физику, картографию и другие разделы науки поистине бесценен.

1755 год — “Грамматика русского языка” главный труд М.В. Ломоносова (19.11.1711–15.04.1765) — при М.В. Ломоносове в РАН начали общаться, делать доклады и писать труды на русском языке. 25 января основан Императорский Московский университет (1755–1917).

Н.И. Лобачевский (01.12.1792–24.02.1856), математик, непризнанный гений, один из основоположников неевклидовой геометрии, выпускник, профессор, декан и выдающийся Ректор Казанского университета: первый математик — Ректор университета на посту был 18 лет (1827–1845). Не был членом Академии наук. Гений, которого при жизни никто не знал. В 2024 году в России по предложению Ректора МГУ В.А. Садовничего ввели «День математика» 1 декабря — день рождения Н.И. Лобачевского, чтобы увековечить его память.

М.В. Остроградский (24.09.1801–01.01.1862) работы по теории упругости, теории магнетизма, первый ординарный академик по “прикладной математике” с 21.12.1831, ординарный академик по “чистой математике” с 15.06.1855; признан основоположником «математической физики» в отечественной науке. Академик без российского диплома.

Н.Д. Брашман (14.06.1796–13.05.1866), чешский и российский математик и преподаватель, член-корреспондент РАН по математическому разряду Отделения физико-математических наук с 03.12.1855. С 1834 г. в Императорском Московском университете Н.Д. Брашман — ПЕРВЫЙ профессор математик: ординарный профессор кафедры чистой и прикладной математики физико-математического факультета (1850–1864), экстраординарный профессор (1834), ординарный профессор кафедры чистой и прикладной математики (1835–1850) физико-математического отделения философского факультета. Заслуженный профессор Московского университета (1859). Организовал научный кружок преподавателей математики и стал Основателем и первым президентом Московского математического общества (1864–1866) и журнала “Математический сборник” (1866). Оказал самое большое влияние на П.Л. Чебышева, которого познакомил с работами французского инженера Жана-Виктора Понселе.

П.Л. Чебышев (16.05.1821–26.11.1894), русский математик и механик, основоположник русской Петербургской математической школы; в Императорской Академии наук адъюнкт по “прикладной механике” Отделения физико-математических наук с 14.05.1853, экстраординарный академик по “прикладной математике” с 03.08.1856,

ординарный академик с 06.02.1859. Самый известный русский математик 19 века. Член 24 Академий мира. Изобретатель одной из первых вычислительных машин, опередившей по функциональности все существующие в те годы аппараты. “Арифмометр Чебышева” непрерывного действия практического применения не нашел, но сыграл важную роль в развитии “машинной математики” и зарождающейся тогда кибернетики. Сегодня это устройство хранится в музее искусств и ремесел во Франции.

А.Ф. Можайский (09.03.1825–20.03.1890), талантливый адмирал русского флота, изобретатель летательных аппаратов и конструктор одного из первых в мире и первого в России натурных самолетов; членом Академии наук не был. Пионер авиации: в мировую историю вошел как создатель первого летающего аппарата тяжелее воздуха, основные элементы которого присущи современным самолетам. Современная Военно-космическая академия в Санкт-Петербурге носит имя А.Ф. Можайского — одно из старейших высших военных учебных заведений России, ведет свою историю с 16 января 1712 года, когда Петр I подписал Указ о создании первой Инженерной школы; сыграла важнейшую роль в покорении космоса и подготовке кадров.

Н.Е. Жуковский (17.01.1847–17.03.1921), русский ученый, математик и механик — первый в России аэрогидродинамик, основоположник современной аэро- и гидромеханики и создатель аэродинамики как науки; талантливый и гениальный конструктор; как педагог и популяризатор науки много сделал для подготовки научных и инженерных кадров. Николая Жуковского справедливо называют «отцом русской авиации»; член-корреспондент физико-математического отделения (по разряду математическому) Санкт-Петербургской АН/РАН с 03.12.1894. В 1904 году Жуковский основал первый в мире аэродинамический институт в Качино под Москвой. С 1 декабря 1918 г. основатель и первый директор (1918–1921) Центрального аэрогидродинамического института (ЦАГИ), в котором с 1931 по 1946 гг. работал М.В. Келдыш.

А.Н. Крылов (15.08.1863–26.10.1945), математик, физик, специалист в области механики кораблестроения; член-корреспондент с 29.11.1914 по разряду физическому Физико-математического отделения, с 02.04.1916 первый ординарный академик по “математической физике” Отделения физико-математических наук. Первые лекции о приближенных вычислениях прочитаны в 1906 году и впервые изданы в 1911 году; первый Герой Социалистического Труда (1943) среди ученых за выдающиеся достижения в области математических наук.

С.А. Чаплыгин (05.04.1869–08.10.1942) — ученик и соратник Н.Е. Жуковского, после смерти которого в 1921 г. возглавил ЦАГИ; учитель М.В. Келдыша в ЦАГИ, специалист в области теоретической механики, гидро- и аэромеханики; член-корреспондент с 06.12.1924 по разряду математических наук (математика) и ПЕРВЫЙ академик с 12.01.1929 по специальности “аэро- и гидродинамика” Отделения физико-математических наук АН СССР. Первый Герой Социалистического Труда (1941) среди ученых АН СССР.

В.А. Стеклов (09.01.1864–30.05. 1926), член-корреспондент с 07.12.1902, адъюнкт по специальности “прикладная математика” с 06.11.1910, экстраординарный академик с 03.03.1912, ординарный академик с 01.07.1912. Владимир Андреевич первый математик — вице-президент с 31.05.1919 по 30.05.1926, основатель и первый директор (1921–1926) первого академического Физико-математического института, из которого после разделения в 1934 г. создали два мировых лидера: Физический институт им. П.Н. Лебедева АН СССР и Математический институт им. В.А. Стеклова АН СССР, где М.В. Келдыш с 1934 по 1953 гг. прошел главные этапы Главного математика и основателя первого в мире академического Института прикладной математики.

О.Ю. Шмидт (30.09.1891–07.09.1956), геофизик, математик, астроном, географ, путешественник; член-корреспондент с 01.02.1933 математика, астрономия, геофизика, академик с 01.06.1935, математика, география, Отделение математических и естественных наук; вице-президент АН СССР 28.02.1939–24.03.1942. Первый Герой Советского Союза (27.06.1937) среди всех ученых. С 1919 г. читал лекции по математике, в 1929 г. основал кафедру высшей алгебры и с 1930 г. проводил алгебраический семинар в Московском университете.

И.М. Виноградов (14.09.1891–20.03.1983) с 12.01.1929 академик по "математике" Отделения физико-математических наук; окончил Санкт-Петербургский университет (1914); директор Физико-математического института АН СССР (1932–1934), основатель и первый директор Математического института им. В.А. Стеклова АН СССР (1934–1941, 1944–1983) — рекордсмен среди директоров академических институтов; Дважды Герой Социалистического Труда (1945, 1971).

М.А. Лаврентьев (19.11.1900–15.10.1980), математик, механик, академик с 30.11. 1946 (в один день с М.В. Келдышем и И.Г. Петровским) по "математике" Отделения физико-математических наук; в 1951–1953 гг. М.А. Лаврентьев академик-секретарь Отделения физико-математических наук АН СССР; в 1955 г. избран в члены Президиума АН СССР; в 1955–1957 гг. вновь академик-секретарь Отделения физико-математических наук АН СССР; основатель и первый председатель Сибирского отделения АН СССР, вице-президент АН СССР 13.09.1957–27.11.1975; научный руководитель М.В. Келдыша — студента и по дипломной работе на физмате МГУ, в аспирантуре и по кандидатским диссертациям в МИАН; Герой Социалистического Труда в 1967 г. за выдающиеся заслуги в развитии науки и организации Сибирского отделения АН СССР, лауреат Ленинской премии в 1958 г. за работы по созданию артиллерийского атомного заряда.

И.Г. Петровский (18.01.1901–15.01.1973), математик; член-корреспондент с 29.09. 1943 и академик с 30.11.1946 по специальности "математика", Отделение физико-математических наук АН СССР. В мае 1951 г. Иван Георгиевич лично И.В. Сталиным назначен Ректором МГУ, проработал до конца жизни в 1973 г. Лучший Ректор за всю 270-летнюю историю МГУ! Первый Ректор вуза — Герой Социалистического Труда (13.03.1969).

А.Н. Колмогоров (25.04.1903-20.10.1987), крупнейший советский русский математик; академик с 29.01.1939, специальность «математика» Отделение математических и естественных наук АН СССР, признанный «Гений математики», Герой Социалистического Труда (24.04.1963).

А.М. Обухов (05.05.1918–03.12.1989), математик, геофизик; член-корреспондент с 23.10.1953 специальность «геофизика» Отделение физико-математических наук АН СССР; академик с 24.11.1970 специальность «физика атмосферы» Отделение океанологии, физики атмосферы и географии АН СССР; основал отечественную научную школу по физике атмосферы, признанную в мировой науке. Единственный признанный ученик А.Н. Колмогорова.

В.С. Владимиров (09.01.1923–03.11.2012), русский математик, Герой Социалистического Труда (1983), зам. директора МИАН 1986–1988, Директор МИАН (1988–1993) — в самый сложный момент в истории переходного периода СССР-Россия; член-корреспондент с 26.11.1968, академик с 24.11.1970 по «математике» Отделения математики. Для МФТИ создал учебники по математической физике.

4. Заключение

«Мстислав Всеволодович — это уникальное явление. Такого не было и не будет... Что поражало при общении с ним — это впечатление, что имеешь дело с ядерным реактором, который внешне интеллигентен, но главное в нём — это внутреннее существо. Это непрерывное горение, необычайный внутренний накал, огромное количество внутренней энергии — впечатление чего-то скрытого, могучего в этом человеке...» — академик О.Г. Газенко.

«Мстислав Всеволодович — рыцарь науки. Всё — для науки и ради науки. Я не знаю такого второго человека. Я абсолютно уверен, что за науку он фактически положил жизнь» — академик Г.К. Скрябин.

Связь времен и достижений: и М.В. Келдыш и АО Ракетно-космический центр «Прогресс» (входит в Госкорпорацию «Роскосмос») начинали свою деятельность с авиации и пересекались во время войны, а после войны они тесно сотрудничали в ракетно-космической отрасли. В 1941 году математик занял должность начальника отдела динамической прочности ЦАГИ, а в июне 1944 года стал заведующим отдела механики Математического института им. В.А. Стеклова АН СССР. В 1946 году М.В. Келдыш распростился с ЦАГИ — передним краем оборонных технологий после войны стали ракеты, а не самолеты, и после избрания в академики 30.11.1946 через пару дней 02.12.1946 был назначен на должность начальника Реактивного научно-исследовательского института (НИИ-1). Это был первый шаг на пути к открытию космической эры!

Не случайно «Ракета Победы» разработана и изготовлена на предприятии «Прогресс» — это символическая почетная миссия для всего коллектива. На ракету-носитель нанесена символика, посвященная 80-летию Победы в Великой Отечественной войне. А всё начиналось с советского Предприятия «Государственный авиационный завод № 1», который внес большой вклад в приближение Великой Победы. Каждый шестой самолёт, воевавший на фронтах войны в 1941–1945 гг., был изготовлен руками работников завода. В «Прогрессе» были созданы серии советских космических кораблей («Восток», «Восход», «Союз»), предназначенных для пилотируемых полетов по околоземным орбитам, в том числе и для полета ПЕРВОГО в мире космонавта Ю.А. Гагарина. **24 октября 2024 года Ракетно-космический центр отметил 130-летний юбилей.**

Важное, что актуально и в современном мире. Академик Б.Н. Петров, Герой Социалистического Труда (13.03.1969), первый Председатель Совета «Интеркосмос» при АН СССР: «Но как бы ни были интересны научно-технические результаты совместного полета, пожалуй, не менее важен и тот факт, что ученые, специалисты и космонавты двух стран смогли плодотворно объединить свои усилия, подготовить и осуществить этот сложный эксперимент. Многие коллективы, большое число инженеров, техников, рабочих участвовало в модернизации кораблей, в создании совместимых средств сближения и стыковки, в управлении полетом кораблей. Первый международный полет космических кораблей, во время которого была произведена стыковка кораблей различных стран и осуществлен переход космонавтов из одного корабля в другой, знаменует важный этап в развитии космонавтики, открыл новую страницу в международном сотрудничестве по исследованию и освоению космического пространства.

Путь, пройденный участниками программы «Союз–Аполлон» от замысла до блестящего успеха полета, был нелегок и непрост. Советским и американским специ-

алистам сообща пришлось решить немало проблем, преодолеть много технических трудностей. К тому же ведь не секрет, что у совместного полета были в США и свои недоброжелатели и даже откровенные противники. И если все это не помешало осуществлению проекта и полет был проведен, значит, действительно, взаимное стремление вместе работать во имя мира и блага людей, узнавать друг друга, делиться опытом и знаниями, находить общий язык оказались гораздо сильнее, чем предполагали те, кто сеял сомнения.»

«Эпоха Келдыша» продолжается... Уникальный опыт международного сотрудничества в ЭПАС и на МКС вполне можно использовать для совместного международного проекта «Марс», куда человек может долететь, но жить там опасно. Однако в век искусственного интеллекта и «умных роботов» покорить Марс можно с помощью «умных автоматов», объединив интеллект и инженерно-конструкторский опыт трех великих космических держав — Россия, США, Китай.

Особо следует напомнить: создание ядерного оружия и покорение космоса явились факторами мира и стабилизации мирового порядка — 80 лет не было мировых войн. В последние годы выросло поколение на планете, которое потеряло страх перед ядерной угрозой, — это ОПАСНО!

Список литературы

[1] Марчук Г.И., Алдошин С.М., Григорьев А.И., Козлов В.В. Эпоха М.В. Келдыша: вывод и уроки. 17 февраля 2011 г. http://www.ras.ru/news/shownews.aspx?id=6531c71e-d91f-44a2-bd7e-812a1405cffc;Eppokha_Keldysha.pdf

[2] Сушкевич Т.А. М.В. Келдыш — организатор международного сотрудничества в космосе и первой советско-американской Программы «Союз-Аполлон» (ЭПАС) // Современные проблемы дистанционного зондирования Земли из космоса. 2011. Т. 8. №. 4. С. 9-22. http://jr.rse.cosmos.ru/article.aspx?id=930

[3] Союз и Аполлон. Рассказывают советские ученые, инженеры и космонавты — участники совместных работ с американскими специалистами / Под ред. чл.-кор. АН СССР К.Д. Бушуева. М.: Политпросвещение, 1976. 271 с.

[4] РГАНТД Архивы России. История ЭПАС. https://dzen.ru/rgantd

[5] Летопись Московского университета. http://letopis.msu.ru/

[6] Информационная система «Архивы Российской академии наук». Персональный состав РАН. https://isaran.ru;https://www.isaran.ru/?q=ru/persostav

[7] Розенберг Г.В., Сандомирский А.Б., Сушкевич Т.А., Матешвили Ю.Д. Исследование стратификации аэрозоля в стратосфере по программе «Союз-Аполлон» // Изв. АН СССР. Серия Физика атмосферы и океана. 1980. Т. 16. № 4. С. 861–864.

Оценка применимости предобусловливателя AIPS в качестве альтернативы ILU(0) при гидродинамическом моделировании нефтегазовых месторождений на графических процессорах

А.С. Добровольцев, М.А. Сохатский, А.В. Юлдашев

Уфимский университет науки и технологий

Проведена адаптация алгоритмов предобусловливателя AIPS, основанного на аппроксимации обратной матрицы с использованием степенного разложения в ряд Неймана, к мелкоблочной структуре матриц, возникающих при численном моделировании фильтрации углеводородов в пористых средах. Проведено сравнение эффективности реализованного предобусловливателя и популярного предобусловливателя ILU(0) как в одноступенчатом варианте, так и на второй ступени специализированного двухступенчатого предобусловливателя CPR при итерационном решении СЛАУ с размерностью до 14 миллионов методом BiCGStab на GPU NVIDIA A100. В результате исследования было показано, что применение AIPS в качестве альтернативы ILU(0) может существенно сократить время решения СЛАУ: до 3,7 раза в режиме одноступенчатого предобусловливателя и до 1,5 раза на второй ступени CPR.

Ключевые слова: графические процессоры, итерационные методы, параллельные вычисления, предобусловливатели, разреженные матрицы, степенное разложение, трехдиагональные системы

1. Введение

Трехмерное гидродинамическое моделирование нефтегазовых месторождений — ресурсоемкая задача, вычислительным ядром которой является решение систем линейных алгебраических уравнений (СЛАУ). В целях ускорения расчетов в индустриальные гидродинамические симуляторы внедряются оптимизированные методы решения СЛАУ, а также параллельные вычислительные технологии. Например, гидродинамический симулятор «РН-КИМ» [1], используемый в университете в целях обучения и научных исследований, имеет многопоточную версию для рабочих станций с многоядерными процессорами (CPU), а также адаптирован для работы на кластерных системах. Одним из новшеств является возможность ускорения гидродинамических расчетов за счет использования графических процессоров (GPU) на этапе решения СЛАУ.

Ранее проведенные исследования [2, 3] показали возможность применения на GPU предобусловливателя AIPS, аппроксимирующего обратную матрицу на основе степенного разложения в ряд Неймана, в качестве альтернативы классическому алго-

ритму алгебраического многосеточного метода (AMG) [4], реализованному в библиотеке AmgX [5], на первой ступени специализированного двухступенчатого предобусловливателя CPR [6]. Благодаря нетрудоемкой процедуре построения, а также реализации ориентированного на архитектуру CUDA параллельного алгоритма пакетного решения линейных систем с трехдиагональными матрицами различной размерности, метод AIPS позволил снизить время процедуры итерационного решения СЛАУ с двухступенчатым предобусловливателем CPR на одном GPU, а также обеспечил приемлемую масштабируемость на вычислительной системе с несколькими GPU.

На текущем этапе исследований рассматривается применимость на GPU предобусловливателя AIPS в качестве альтернативы методу неполного LU-разложения без заполнения — ILU(0) [7], который широко используется как самостоятельный одноступенчатый предобусловливатель либо на второй ступени CPR [8]. Предпосылкой является то, что алгоритмы ILU(0) имеют низкий ресурс параллелизма и, несмотря на наличие параллельных модификаций [9, 10], в том числе реализованных в оптимизированных библиотеках, например, cuSPARSE [11], применение ILU(0) на GPU зачастую является узким местом в процедуре итерационного решения СЛАУ.

2. Теоретическая часть

Метод AIPS [2], аппроксимирующий обратную матрицу на основе степенного разложения в ряд Неймана, зарекомендовал себя в качестве масштабируемого предобусловливателя, который можно использовать на первой ступени двухступенчатого предобусловливателя CPR для решения СЛАУ с разреженной матрицей при гидродинамическом моделировании нефтегазовых месторождений на графических процессорах [3].

Применение AIPS в итерационном процессе предполагает выполнение на каждой итерации умножения на вектор матрицы M_N^{-1}, аппроксимирующей обратную матрицу по формуле

$$A^{-1} \approx M_N^{-1} = \left[E + \sum_{k=1}^{N} (-1)^k (P^{-1}R)^k \right] P^{-1}, \tag{1}$$

где A — матрица решаемой СЛАУ, $R = A - P$, E — единичная матрица, P — легко обратимая часть матрицы A, обеспечивающая выполнение условия $\rho(P^{-1}R) < 1$, N — количество используемых членов ряда Неймана. Процедуру применения данного предобусловливателя для приближённого решения СЛАУ $Ax = b$ целесообразно осуществлять без явного вычисления M_N^{-1} путём умножения на вектор правой части выражения (1):

$$x = A^{-1}b \approx M_N^{-1}b = \left[E + \sum_{k=1}^{N} (-1)^k (P^{-1}R)^k \right] P^{-1}b, \tag{2}$$

что требует выполнения многократных умножений на вектор матрицы P^{-1} либо решений систем с матрицей P ($N + 1$ раз), а также умножений на вектор матрицы R (N раз).

Характерными особенностями матриц, возникающих в процессе гидродинамического моделирования нефтегазовых месторождений, являются сильная разреженность, мелкоблочная структура (мелкозернистая блочность) с плотными квадрат-

ными блоками фиксированного размера (2×2, 3×3 или более в зависимости от используемой модели фильтрации) [12], а также наличие многочисленных «разрывов» в трехдиагональной части матрицы, обусловленных структурированным порядком нумерации ячеек при формировании СЛАУ в результате пространственной дискретизации уравнений фильтрации с использованием метода конечных объемов.

В наших предыдущих работах рассматривалось применение AIPS на первой ступени CPR для приближённого решения СЛАУ $A_p \delta x_p = r_p$. A_p — подматрица на давление, формируемая из исходной матрицы A с помощью алгоритма метода CPR в предположении о малом изменении насыщенностей в расчетных ячейках, что приводит к кратному снижению размерности матрицы и утрате мелкозернистой блочности, однако общая структура матрицы сохраняется. В качестве P нами было принято использовать трехдиагональную часть матрицы A_p, что обеспечивало соблюдение условия сходимости ряда Неймана $\rho(P^{-1}R) < 1$ и позволяло свести вычисление выражений вида $P^{-1}b$ к решению трехдиагональной системы уравнений.

Наличие многочисленных «разрывов» в трехдиагональной части предоставляет возможность её декомпозиции на множество независимых трехдиагональных матриц различной размерности и массивного распараллеливания процедуры решения системы с матрицей P. В работе [2] был предложен ориентированный на архитектуру CUDA параллельный алгоритм пакетного решения линейных систем с трехдиагональными матрицами различной размерности. Идея данного алгоритма заключается в параллельном применении метода прогонки для независимых частей трехдиагональной матрицы, которые предварительно сортируются, а элементы которых переставляются, чтобы обеспечить балансировку нагрузки и более производительный доступ к памяти GPU. Недавно проведенное сравнение [13] быстродействия данного алгоритма и альтернативных алгоритмов решения трехдиагональных систем, реализованных в cuSPARSE, подтвердило наличие преимущества: ускорение относительно альтернатив составило от 1,2 до 3,9 раза.

В данной работе проведена адаптация алгоритмов построения и применения предобусловливателя AIPS к мелкоблочной структуре матриц, возникающих при численном моделировании фильтрации углеводородов в пористых средах. Теперь в каче-

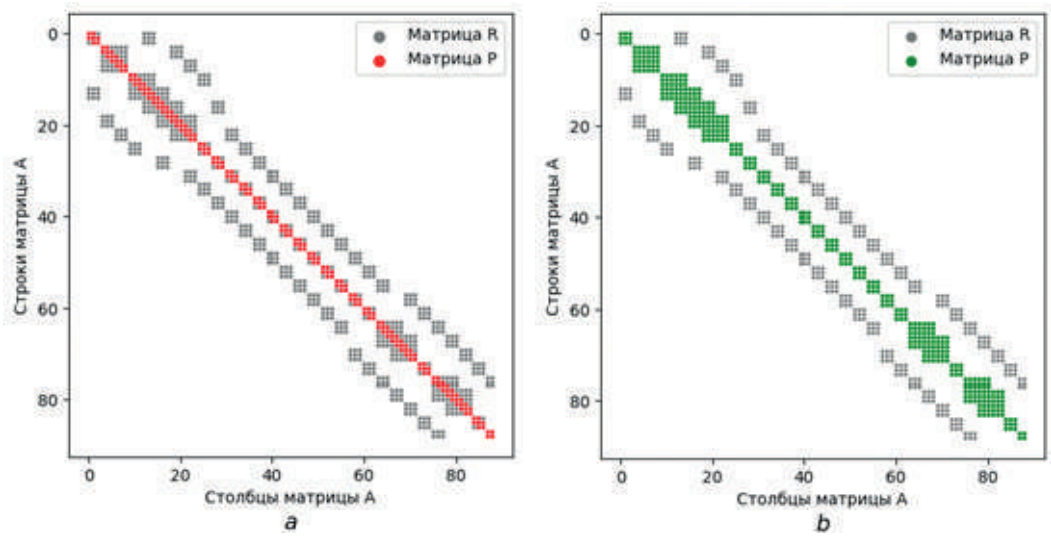


Рис. 1: Фрагменты с трехдиагональной (a) и мелкоблочной трехдиагональной (b) матрицей P

стве P можно использовать мелкоблочную трехдиагональную матрицу, элементами которой являются квадратные блоки фиксированного размера. В качестве иллюстрации на рисунке 1 приведены фрагменты матрицы, где видно расширение структуры матрицы P благодаря учету мелкозернистой блочности.

Соответствующим образом адаптирован ранее предложенный параллельный алгоритм пакетного решения линейных систем с трехдиагональными матрицами различной размерности: вместо скалярной прогонки задействован метод блочной прогонки [14, 15]. Это позволило апробировать AIPS в качестве самостоятельного предобусловливателя при решении СЛАУ, характерных для задачи гидродинамического моделирования нефтегазовых месторождений, а также на второй ступени предобусловливателя CPR, где, как правило, применяется ILU(0).

3. Практическая часть

В таблице 1 приведены характеристики тестовых матриц, полученных при решении уравнений трехфазной фильтрации, имеющих мелкоблочную структуру с размером блока 3×3. Использование при построении AIPS в качестве P мелкоблочной трехдиагональной части матрицы обеспечивает соблюдение условия сходимости ряда Неймана $\rho(P^{-1}R) < 1$.

Таблица 1: Характеристики тестовых матриц

Название матрицы	Размерность матрицы	Количество ненулевых элементов	Среднее количество ненулевых элементов в строке	Спектральный радиус $\rho(P^{-1}R)$
immn	2 304 102	42 859 314	18,601	0,918
krrv	4 320 921	85 471 137	19,781	0,981
mmnt	5 637 747	109 595 799	19,440	0,993
fdrv	6 610 263	118 221 633	17,885	0,806
kmms	8 630 895	167 332 329	19,388	0,995
lkms	13 665 705	254 102 823	18,594	0,996

Решение соответствующих СЛАУ на GPU проводилось итерационно стабилизированным методом бисопряженных градиентов (BiCGStab) [16] со следующим условием остановки итерационного процесса: достижение относительной невязкой величины $\varepsilon = 10^{-4}$. Численные результаты получены на одном из узлов вычислительного кластера УУНиТ, оснащенном 2×CPU Intel Gold 6326 и GPU NVIDIA A100 (ОС CentOS 7.9, CUDA Toolkit 11.8).

Проведена сравнительная оценка эффективности предобусловливателя AIPS с предобусловливателем ILU(0), реализованным в библиотеке cuSPARSE, с использованием формата хранения разреженных матриц Block compressed Sparse Row (BSR). В рамках тестирования для AIPS было принято количество членов ряда Неймана $N = 1$, которое в среднем оказалось оптимальным для набора тестовых матриц. Хотя увеличение N ускоряет сходимость, например, при $N = 10$ количество итераций для решения СЛАУ снижается до 4,2 раза, возрастающая сложность многократного выполнения операций на этапе применения предобусловливателя не окупается.

Таблица 2 содержит результаты применения методов AIPS и ILU(0) в качестве одноступенчатых предобусловливателей. Видно, что поддержка мелкозернистой блочности в AIPS обеспечивает убедительное преимущество данного предобусловливателя при проведении расчетов на GPU: максимальное ускорение относительно ILU(0) составило 3,66, в среднем — 2,34.

Таблица 2: Решение СЛАУ методом BiCGStab с различными предобусловливателями

Название матрицы	ILU(0)		AIPS		Ускорение при использовании AIPS относительно ILU(0)
	Количество итераций	Время расчета, мс	Количество итераций	Время расчета, мс	
immn	15,5	232,10	10,5	63,36	3,66
krrv	6,5	134,94	10,5	115,19	1,17
mmnt	6	157,55	5	82,58	1,91
fdrv	2	85,95	1,5	50,75	1,69
kmms	26,5	829,75	13	265,26	3,13
lkms	24	1 116,15	19,5	515,47	2,17

В таблице 3 приводится аналогичное сравнение, но теперь AIPS и ILU(0) выступают в роли второй ступени двухступенчатого предобусловливателя CPR, где на первой ступени используется классический алгоритм метода AMG, реализованный в библиотеке AmgX. Применение AIPS вместо ILU(0) на второй ступени CPR обеспечивает близкую скорость сходимости, но в то же время позволяет быстрее проводить решение СЛАУ: максимальное ускорение относительно ILU(0) составило 1,45, в среднем — 1,23.

Таблица 3: Решение СЛАУ методом BiCGStab с различными предобусловливателями на 2 ступени CPR

Название матрицы	CPR-AMG-ILU(0)		CPR-AMG-AIPS		Ускорение CPR-AMG-AIPS относительно CPR-AMG-ILU(0)
	Количество итераций	Время расчета, мс	Количество итераций	Время расчета, мс	
immn	2,5	99,21	2,5	74,51	1,33
krrv	2	124,30	2,5	122,73	1,01
mmnt	2	147,74	1,5	131,05	1,13
fdrv	1	135,04	1	110,80	1,22
kmms	3,5	288,66	2	199,17	1,45
lkms	2,5	325,75	2,5	287,12	1,13

На рисунке 2 обобщены результаты тестирования при решении СЛАУ с использованием AIPS и ILU(0) в качестве одноступенчатых предобусловливателей, а также на второй ступени двухступенчатого предобусловливателя CPR.

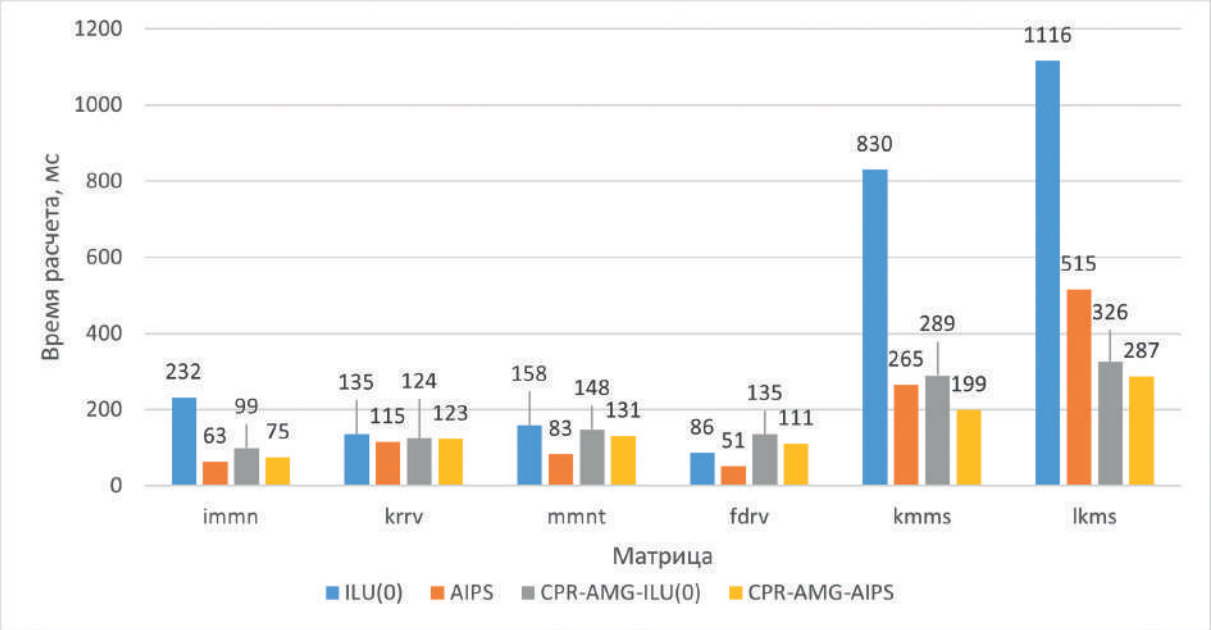


Рис. 2: Сравнение времени решения СЛАУ с различными предобусловливателями

4. Заключение

В работе рассмотрена применимость на GPU предобусловливателя AIPS в качестве альтернативы методу неполного LU-разложения без заполнения — ILU(0), при итерационном решении СЛАУ, характерных для задачи гидродинамического моделирования нефтегазовых месторождений. Алгоритмы построения и применения предобусловливателя AIPS адаптированы к мелкоблочной структуре матриц, что, главным образом, потребовало перехода к применению параллельного алгоритма на основе метода блочной прогонки при обработке трехдиагональной части матрицы СЛАУ.

В результате при решении набора СЛАУ с размерностью от 2 до 14 миллионов на GPU NVIDIA A100 получено существенное снижение времени решения СЛАУ за счет применения предобусловливателя AIPS: до 3,7 раза относительно ILU(0) в режиме одноступенчатого предобусловливателя, а также до 1,5 раза в качестве альтернативы ILU(0) на второй ступени CPR.

В настоящее время проводится исследование масштабируемости разработанного предобусловливателя AIPS с учетом мелкозернистой блочности в структуре матриц на многопроцессорных вычислительных системах.

Список литературы

[1] Гидродинамический симулятор РН-КИМ. `https://rn.digital/rnkim` (дата обращения: 01.05.2025).

[2] Юлдашев А.В., Репин Н.В., Спеле В.В. Параллельный предобусловливатель на основе степенного разложения обратной матрицы для решения разреженных

линейных систем на графических процессорах // Вычислительные методы и программирование, 2019. Т. 20. С. 444-456. https://doi.org/10.26089/NumMet.v20r439.

[3] Сайфуллин И.Ф., Шарипов Т.Р., Спеле В.В., Штангеев А.Л., Юлдашев А.В. Опыт использования графических процессоров для ускорения расчетов при гидродинамическом моделировании // Тезисы докладов научно-технической конференции «Цифровые технологии в добыче и переработке углеводородов: от моделей к практике» (6–9 октября 2020 г., г. Уфа). — М.: ЗАО «Изд-во «Нефтяное хозяйство»», 2020. — С. 83-84.

[4] Ruge J.W., Stuben K. Algebraic multigrid (AMG) // Multigrid Methods / S.F. McCormick (ed.) Philadelphia, PA, USA: SIAM, 1987. Vol. 3, p. 73–130. https://doi.org/10.1137/1.9781611971057.ch4.

[5] AmgX. https://developer.nvidia.com/amgx (дата обращения: 01.05.2025).

[6] Wallis J.R., Kendall R.P., Little T.E. Constrained residual acceleration of conjugate residual methods // Proc. SPE Reservoir Simulation Symposium. 1985. https://doi.org/10.2118/13536-MS.SPE1353.1985.415-428.

[7] Саад Ю. Итерационные методы для разреженных линейных систем. М.: Изд-во Моск. ун-та, 2013.

[8] Wang K., Liu H., Yan L., Luo J., Wu K., Li J., Chen F., Dong X., Chen Z. An adaptive preconditioning strategy to speed up parallel reservoir simulations // Society of Petroleum Engineers — SPE Reservoir Simulation Conference 2019. 2019. https://doi.org/10.2118/193872-MS.

[9] Li R., Saad Y. GPU-accelerated preconditioned iterative linear solvers // The Journal of Supercomputing. 2013. Vol. 63, No. 2. P. 443-466. https://doi.org/10.1007/s11227-012-0825-3.

[10] Naumov M. Parallel solution of sparse triangular linear systems in the preconditioned iterative methods on the GPU // NVIDIA Corp., Westford, MA, USA, Tech. Rep. NVR-2011, vol. 1, 2011.

[11] cuSPARSE. https://docs.nvidia.com/cuda/cusparse/ (дата обращения: 01.05.2025).

[12] Азиз Х., Сеттари Э. Математическое моделирование пластовых систем. М.: Институт компьютерных исследований, 2004.

[13] Добровольцев А.С., Сохатский М.А., Юлдашев А.В. Оценка быстродействия параллельного алгоритма пакетного решения линейных систем с трехдиагональными матрицами различной размерности на графических процессорах // Международная конференция «Суперкомпьютерные дни в России»: Труды конференции (Москва, 23-24 сентября 2024). — М: изд-во МГУ. — 2024. — С. 232-234.

[14] AlgoWiki: Открытая энциклопедия свойств алгоритмов. Блочная прогонка. http://algowiki-project.org/ru/Блочная_прогонка (дата обращения: 01.05.2025).

[15] László E., Giles M., Appleyard J. Manycore algorithms for batch scalar and block tridiagonal solvers // ACM Trans. Math. Softw. 2016. 42, N 4. https://doi.org/10.1145/2830568.

[16] AlgoWiki: Открытая энциклопедия свойств алгоритмов. Стабилизированный метод бисопряженных градиентов (BiCGStab). http://algowiki-project.org/ru/Стабилизированный_метод_бисопряженных_градиентов_(BiCGStab) (дата обращения: 01.05.2025).

Применение алгоритмов оптимизации параметров при решении систем уравнений на центральных процессорах и графических ускорителях

Б.И. Краснопольский[1], Р.М. Куприй[1,2]

[1]НИИ механики МГУ,
[2]Факультет ВМК МГУ

Использование итерационных методов решения систем линейных алгебраических уравнений зачастую сопряжено с проблемой подбора оптимальных настроечных параметров этих методов. В настоящей работе обсуждаются вопросы тестирования и расширения возможностей разработанного ранее алгоритма оптимизации параметров численных методов. Предложены и реализованы дополнительные целевые функции оптимизации на основе полного времени решения систем уравнений и оптимизации с ограничением на размер потребляемой оперативной памяти. Представлены результаты численных экспериментов, подтверждающие работоспособность и практическую важность предложенных алгоритмов при проведении расчетов на центральных процессорах и графических ускорителях.

Ключевые слова: системы линейных алгебраических уравнений, итерационные методы, многосеточные методы, генетический алгоритм, оптимизация

1. Введение

Решение систем линейных алгебраических уравнений (СЛАУ) является распространенной подзадачей в задачах математического моделирования. Ее сложность обусловлена как проблемой выбора метода решения СЛАУ, оптимального для характерных систем уравнений, возникающих в процессе расчетов, так и необходимостью эффективной реализации данных методов на имеющихся вычислительных устройствах. Отдельную сложность также может представлять определение понятия оптимальности метода для данной задачи. Это понятие, например, может включать минимизацию времени решения СЛАУ для заданной правой части с учетом или без учета времени предварительной подготовки данных, а также с учетом или без учета дополнительных ограничений на объем оперативной памяти, затрачиваемой алгоритмом в процессе работы. Имеющееся множество опубликованных итерационных методов решения систем уравнений, многие из которых также обладают своими настроечными параметрами, делает практически невозможным априорный выбор метода решения, близкого к оптимальному в каждом конкретном случае.

Упомянутые сложности, ожидаемо, привели к попыткам создания различных рекомендательных систем и алгоритмов оптимизации для выбора методов решения

СЛАУ. Среди встречающихся в литературе можно выделить четыре общих направления исследований: (1) предсказание оптимального метода из заданного набора методов для решения данной СЛАУ [6], (2) оптимизация параметров в процессе решения последовательности систем уравнений (например, в ходе расчета нестационарных постановок задач математической физики) [7], (3) предварительная оптимизация параметров методов для последующего решения большого числа однотипных систем уравнений [4] и (4) оптимизация программной реализации методов под конкретные вычислительные платформы [1, 2].

В работах [3–5] был предложен оптимизационный алгоритм для детальной оптимизации параметров численного метода под конкретную систему уравнений с заданной матрицей. Задачей являлась разработка такой процедуры оптимизации, которая позволила бы за ограниченное число испытаний подобрать конфигурацию методов, обеспечивающую минимизацию времени решения. Для этого был разработан алгоритм, основанный на комбинации эволюционной стратегии и предобученной нейросети, используемой в качестве фильтра для оператора случайных мутаций. Предложенный алгоритм был апробирован на задачах моделирования несжимаемых турбулентных течений и показал свою эффективность, позволив сократить общее время расчетов и автоматизировать процедуру вычислений, несмотря на дополнительные затраты времени на работу алгоритма оптимизации.

Упомянутый выше сценарий расчета предполагает целью оптимизации сокращение времени решения СЛАУ, когда в сценарии расчета время инициализации и предварительных вычислений для используемых методов (например, факторизация матриц для ILU-подобных методов или построение иерархии матриц многосеточного метода) считается пренебрежимо малым. Целью данной работы является апробация других сценариев в рамках разработанных процедур оптимизации, а именно: минимизация полного времени решения СЛАУ и оптимизация с ограничениями на объем потребляемой памяти. Дополнительно, демонстрируется практическая значимость использования данного алгоритма при проведении вычислений на различных вычислительных устройствах.

2. Оптимизационный алгоритм

Для детальной оптимизации параметров численных методов решения СЛАУ в серии работ [3–5] был разработан генетический эволюционный алгоритм, позволяющий проводить оптимизацию параметров под конкретные системы уравнений. Данный алгоритм представляет собой эволюционную стратегию вида $(1 + \lambda) - ES$ [8] с двумя типами мутаций. Алгоритм оптимизации представляет собой итерационный процесс, в котором на каждом новом поколении происходит клонирование исходного индивидуума (родителя), применение операторов мутаций, оценка эффективности полученных индивидуумов (векторов параметров алгоритма решения СЛАУ) путем прямого решения оптимизируемой системы уравнений, и отбор лучшего индивидуума. В качестве критерия сходимости для разработанного алгоритма оптимизации было предложено использовать оценку скорости убыли времени решения на нескольких подряд идущих поколениях.

Предложенный алгоритм использует два оператора мутаций – оператор мягких мутаций и оператор случайных мутаций. Оператор мягких мутаций отвечает за поиск оптимума в окрестностях текущей точки в пространстве параметров. Оператор

случайных мутаций применяется для поиска возможных кандидатов на локальные минимумы во всем пространстве поиска. Алгоритм допускает два сценария использования оператора случайных мутаций, когда новые вектора параметров генерируются полностью случайным образом, и когда предобученная нейросеть ранжирует большой набор случайным образом сгенерированных векторов параметров и предлагает из этого множества наиболее перспективных кандидатов для добавления в следующее поколение.

Более подробно детали оптимизационного алгоритма, особенности подготовки данных и обучения нейронной сети, и примеры применения реализованного в рамках библиотеки XAMG [9] алгоритма для оптимизации набора из 21 параметра алгебраического многосеточного метода решения СЛАУ изложены в [4]. Вопросы использования многостадийной оптимизации и оценка целесообразности использования непрерывных диапазонов оптимизируемых величин рассмотрены в [5].

3. Критерии оптимизации

В описанном выше алгоритме целевой критерий оптимизации отвечает за выбор вектора параметров, используемого для генерации следующего поколения индивидуумов. Для сформированного в начале расчета поколения вектора параметров выполняется тестовое решение СЛАУ с такими параметрами и собирается целевая информация для каждого из индивидуумов. В базовом сценарии оптимизации к такой информации относится время решения системы уравнений без учета времени на построение иерархии матриц многосеточного метода и каких-либо других ограничений.

Для удобства дальнейшего использования и расширения применимости оптимизационного алгоритма реализована возможность задания критерия оптимизации и задания дополнительных ограничений в виде предельного объема потребляемой алгоритмом памяти. Учет времени на инициализацию численного метода и проведение предварительных вычислений реализован тривиальным образом: к целевому времени решения СЛАУ прибавляется время инициализации. Ограничение на объем потребляемой памяти реализуется следующим образом. Путем чтения содержимого файла `/proc/self/status` в UNIX-подобной операционной системе и редукции содержимого поля `VmRSS` по всем вычислительным процессам осуществляется замер оперативной памяти, которую занимает данное приложение после вызова функции инициализации методов решения СЛАУ. Это показание используется как оценка объема памяти, занимаемой для хранения исходной системы уравнений и вспомогательных данных (например, иерархии матриц многосеточного метода). Если полученная величина превышает заданное пороговое значение, то целевое время для такого вектора параметров задается равным бесконечности и тестовое решение СЛАУ с этим набором параметров не производится. Таким образом, данный вектор исключается из числа кандидатов на включение в следующее поколение.

4. Параметры численных экспериментов

Приведенные в работе результаты тестирования были получены на рабочей станции с 8-ядерным центральным процессором Intel Core-i7 11700 (фактическая пропускная способность памяти порядка 43 ГБ/с) и графическим ускорителем NVidia 3060Ti

(фактическая пропускная способность памяти порядка 417 ГБ/с). Соотношение реальной пропускной способности шины памяти данных вычислительных устройств может быть использовано как ориентировочная оценка ожидаемого ускорения для решаемых задач, которое в данном случае составляет величину порядка 9.7 раза. Все приведенные ниже результаты получены при использовании 8 MPI процессов для вычислений на центральном процессоре и/или одного вычислительного процесса при расчетах на графическом ускорителе.

Для проведения численных экспериментов, представленных в работе, использовалась разрабатываемая библиотека XAMG [9] и встроенный в нее алгоритм оптимизации параметров численных методов [4]. В качестве основной комбинации методов решения СЛАУ использовался стабилизированный метод би-сопряженных градиентов с алгебраическим многосеточным предобуславливателем. Для пред- и постсглаживания при переходе между уровнями иерархии многосеточного метода использовался метод итераций Чебышева и прямой метод для решения СЛАУ на нижнем уровне иерархии. В качестве вектора правой части и начального приближения задавался единичный и нулевой вектор, соответственно. Сходимость оценивалась по величине относительной невязки; пороговое значение в расчетах задавалось равным $\varepsilon_{rel} = 10^{-8}$. Оптимизация проводилась для 13 параметров многосеточного метода. Перечень оптимизируемых параметров и значения для `baseline` конфигурации могут быть найдены в работе [4].

Для оптимизации параметров в данной работе использовался вариант оптимизационного алгоритма без нейросети. При оптимизации на каждом поколении строилось 20 индивидуумов, по 10 для операторов мягких и случайных мутаций. Всего в эволюционной стратегии строилось не более 15 поколений. При оптимизации в качестве начального приближения задавался начальный вектор параметров, соответствующий `baseline` конфигурации.

5. Результаты тестирования

5.1. Оптимизация полного времени решения СЛАУ

В зависимости от решаемой задачи, у исследователей могут возникать разные требования и ограничения при выборе метода решения СЛАУ. Так, например, при моделировании течений вязкой несжимаемой жидкости возникает необходимость многократного решения СЛАУ для давления, при том что матрица СЛАУ может оставаться неизменной в ходе всего расчета. В этом случае, время предварительных вычислений не представляет практической важности, и целесообразно минимизировать только время решения для заданного вектора правой части. Однако, если матрица СЛАУ претерпевает изменения в процессе расчета, то для общего сокращения времени расчета также необходимо минимизировать и время инициализации метода решения СЛАУ.

В настоящем разделе статьи на примере двух тестовых матриц проводится апробация алгоритма оптимизации параметров в случае оптимизации полного времени решения СЛАУ. В качестве тестовых используются две сгенерированные системы уравнений с матрицей для уравнения Пуассона в кубе с сеткой размером 100^3 и матрица для уравнения вида $-\nabla \cdot (\kappa \nabla u) = f$ в кубической области на сетке 100^3 с существенно неоднородным распределением коэффициента κ (`jumps`, [4, 11]).

Таблица 1: Сравнение времен решения СЛАУ для наборов параметров, оптимизированных под разные вычислительные устройства.

	Оптимизация времени всего расчета			Оптимизация этапа решения		
	“setup”, с.	“solve”, с.	“full”, с.	“setup”, с.	“solve”, с.	“full”, с.
`cube100`	0.45	0.59	**1.04**	1.66	**0.42**	2.08
`jumps100`	0.43	0.68	**1.11**	2.09	**0.40**	2.49

Для указанных тестовых матриц проведены две серии оптимизационных расчетов с заданными целевыми функциями минимизации времени общего решения СЛАУ и минимизации времени этапа решения СЛАУ при заранее построенной иерархии матриц многосеточного метода. Соответствующие результаты приведены в табл. 1. Полученные времена существенно различаются в зависимости от выбранного критерия оптимизации. Так, оптимизация общего времени решения СЛАУ позволяет получить конфигурацию численного метода, который может выполнить полный расчет в 2-2.5 раза быстрее, чем вариант, полученный для оптимизации этапа решения без учета времени инициализации. При этом, построение вектора параметров задачи с учетом оптимизации этапа решения СЛАУ позволяет выполнить расчет этого этапа в 1.5-1.7 раза быстрее, чем конфигурация, оптимизированная под весь расчет задачи. Полученные результаты демонстрируют обширную вариативность параметров многосеточных методов и важность выбора правильных критериев оптимизации в зависимости от поставленной перед исследователем задачи.

5.2. Оптимизация с ограничениями по памяти

Решение больших разреженных систем линейных алгебраических уравнений с использованием многосеточных методов может быть достаточно ресурсозатратным, и использовать большие объемы памяти для хранения иерархии матриц и вспомогательных буферов. В то же время, графические ускорители обладают ограниченным объемом оперативной памяти, в сравнении с обычно доступной памятью вычислительного сервера. В ряде случаев может представлять практическую значимость построение конфигурации численного метода, который при решении данной СЛАУ будет потреблять ограниченное количество оперативной памяти. На примере задачи `jumps` на сетке 100^3 ячеек проведены два тестовых расчета оптимизации времени, затрачиваемого на этап решения СЛАУ для предварительно выполненной инициализации численного метода. В первом случае расчет проводился без ограничений на размер потребляемой памяти. В этом случае объем потребляемой памяти для разных комбинаций методов в процессе оптимизации варьировался в диапазоне от 1350 до 2227 МБ. В результате оптимизации получена конфигурация методов, которая обеспечивает решение заданной СЛАУ за 0.48 секунды, потребляя при этом 1535 МБ оперативной памяти.

Второй расчет проведен с ограничением доступной оперативной памяти в размере 1400 МБ. Для таких условий оптимизации была получена комбинация параметров численного метода, обеспечивающая решение за 0.54 секунды при объеме памяти 1281 МБ. Введение дополнительного ограничения, ожидаемо, ухудшает качество оптимизации, но позволяет ограничить объем потребляемой памяти, что может быть актуальным в ряде случаев.

5.3. Оптимизация параметров под различные вычислительные устройства

В данном разделе рассматриваются оценки значимости оптимизации параметров под конкретную вычислительную платформу. Для этих целей используется СЛАУ для уравнения Пуассона, решаемого в единичном кубе на равномерной сетке размером 150^3. Для данной СЛАУ проведена процедура оптимизации параметров с целевой функцией минимизации времени решения для построенной иерархии матриц для центральных процессоров и для графических ускорителей. Полученные конфигурации параметров также были использованы для расчета на втором вычислительном устройстве. Результаты проведенного вычислительного эксперимента вместе с результатами для `baseline` конфигурации представлены в табл. 2.

Полученные результаты демонстрируют, в целом, ожидаемое поведение. Использование процедуры оптимизации позволяет существенно ускорить вычисления за счет оптимизации параметров под конкретное вычислительное устройство – сокращение времени составляет 1.4–1.8 раза. Оптимизация, проведенная под одно вычислительное устройство, дает некоторый выигрыш и на втором вычислительном устройстве, однако он заметно меньше, чем в случае целевой оптимизации алгоритма под данный тип вычислителя. Общее ускорение вычислений при сравнении оптимизированных комбинаций параметров для устройств составляет величину порядка 8. Это несколько меньше, чем ускорение для `baseline` конфигурации, но все также близко к соотношению пропускной способности памяти вычислительных устройств.

Таблица 2: Сравнение времен решения СЛАУ для наборов параметров, оптимизированных под разные вычислительные устройства.

	Расчет на CPU		Расчет на GPU	
	Время решения, с.	Число итераций	Время решения, с.	Число итераций
`baseline`	2.27	6	0.22	5
`cpu-based`	1.24	6	0.203	8
`gpu-based`	1.59	5	0.156	4

Вторая серия экспериментов направлена на более детальную оценку ускорения вычислений при переходе от центральных процессоров к графическим ускорителям. Рассмотрен набор из 16 тестовых матриц (3 сгенерированные матрицы и 13 матриц из SuiteSparse Matrix Collection [10]), и для соответствующих систем уравнений получены времена для `baseline` конфигурации и конфигураций, оптимизированных под центральные процессоры и графические ускорители. Результаты замеров представлены на рис. 1 в виде отношения

$$P_{baseline} = \frac{T^{\mathrm{CPU}}_{baseline}}{T^{\mathrm{GPU}}_{baseline}}, \quad P_{opt} = \frac{T^{\mathrm{CPU}}_{opt}}{T^{\mathrm{GPU}}_{opt}}$$

в зависимости от размера матрицы, а также на рис. 2 в виде метрик

$$S^{\mathrm{CPU}} = \frac{T^{\mathrm{CPU}}_{baseline}}{T^{\mathrm{CPU}}_{opt}}, \quad S^{\mathrm{GPU}} = \frac{T^{\mathrm{GPU}}_{baseline}}{T^{\mathrm{GPU}}_{opt}}.$$

Данные на рис. 1 позволяют оценить зависимость наблюдаемого ускорения от размера матрицы СЛАУ. За исключением двух выбросов, остальные данные укладываются в общую зависимость, при которой для матриц малого размера использование графического ускорителя обеспечивает ускорение расчета порядка 3 раз, и величина ускорения плавно возрастает до величины порядка 8 при увеличении размера матрицы до 100 МБ и более. Такое поведение объясняется ростом эффективности кэш-памяти процессора для матриц, размер которых сопоставим с размером матрицы СЛАУ (для процессора Intel Core-i7 11700 размер кэша L3 составляет 16 МБ). Для достаточно больших матриц эффективность переиспользования данных в кэш-памяти существенно снижается, и определяющей становится скорость доступа к данным в оперативной памяти.

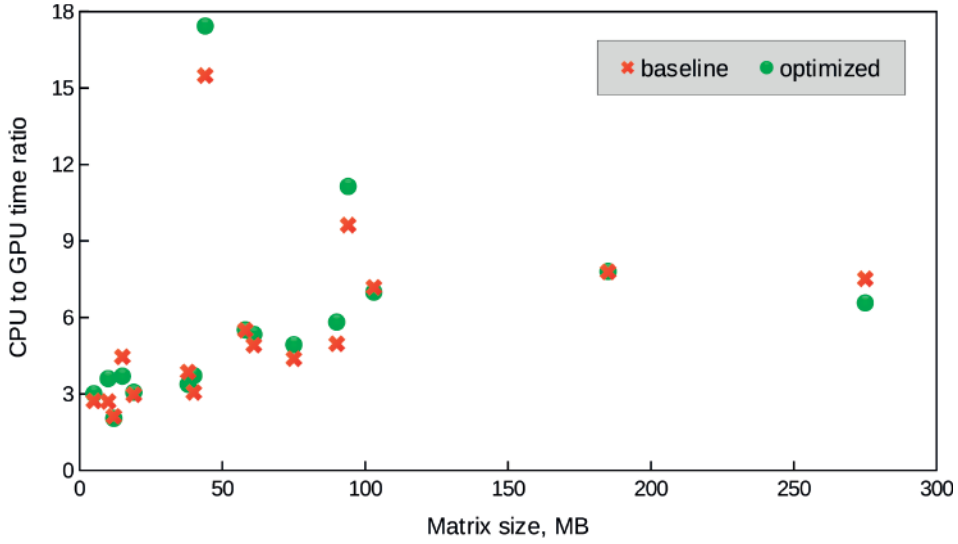


Рис. 1: Ускорение вычислений при переходе с центральных процессоров на графические ускорители для базовой конфигурации параметров и параметров, оптимизированных под конкретную систему уравнений.

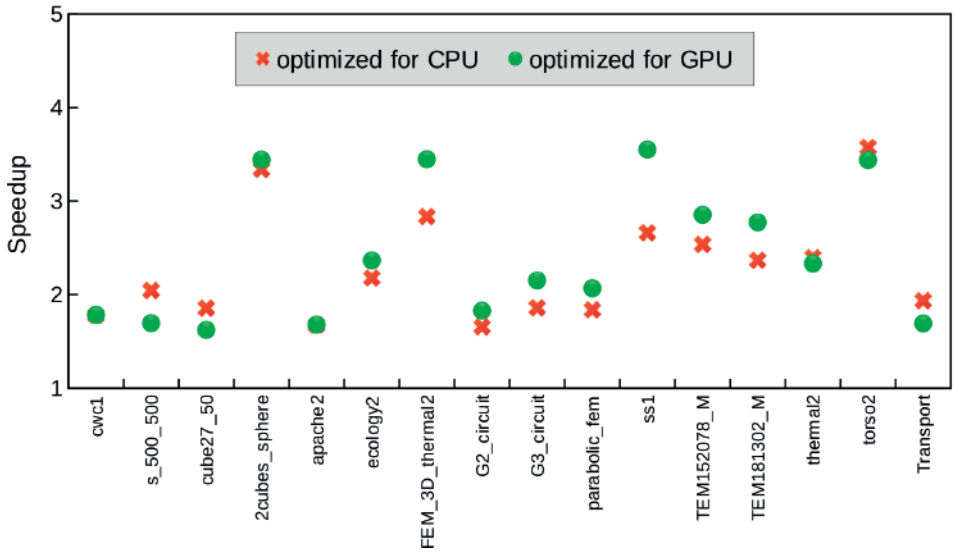


Рис. 2: Сокращение времени решения СЛАУ за счет оптимизации параметров численного метода на центральных процессорах и на графических ускорителях.

Для двух матриц, `G3_circut` и `parabolic_fem` из SuiteSparse Matrix Collection, наблюдается существенно большее ускорение при использовании графических ускорителей. Судя по всему, это обусловлено низкой степенью локализации ненулевых элементов в матрицах и низкой эффективностью распараллеливания для данных систем уравнений. В данном случае, при расчетах на центральном процессоре не использовались алгоритмы декомпозиции графов для оптимизации MPI обменов при распараллеливании вычислений, что для этих матриц привело к формированию больших объемов коммуникаций между вычислительными процессами при расчете операций умножения матрицы на вектор и, как следствие, низкой эффективности распараллеливания вычислений для данных матриц.

Метрики на рис. 2 позволяют оценить значимость оптимизации параметров численного метода при проведении расчетов на центральных процессорах и на графических ускорителях. Применение алгоритмов оптимизации под конкретную СЛАУ в обоих случаях является существенным способом ускорения вычислений, которое варьируется в диапазоне от 1.6 до 3.6 раза. Среднее ускорение для решения СЛАУ на центральных процессорах составляет величину порядка 2.28 раза. Для графических ускорителей средний эффект от оптимизации параметров оказывается чуть выше и достигает 2.42 раза. Данные результаты наглядно свидетельствуют о значимости алгоритмов оптимизации параметров для повышения эффективности методов решения систем уравнений.

6. Заключение

В работе рассмотрен вопрос расширения возможностей разработанного и реализованного в рамках библиотеки XAMG алгоритма оптимизации параметров численных методов решения систем линейных алгебраических уравнений. Предложена и реализована дополнительная целевая функция оптимизации, реализующая минимизацию общего времени решения системы уравнений, а также предусмотрен сценарий проведения оптимизации параметров с дополнительным ограничением на объем потребляемой оперативной памяти.

Проведено тестирование различных сценариев оптимизации и оценка значимости алгоритма при проведении расчетов на центральных процессорах и графических ускорителях. Показано, что для достижения наилучших результатов целесообразно использовать критерии оптимизации, сходные со сценарием использования алгоритма решения СЛАУ в вычислительном коде. Так, оптимизация времени полного решения СЛАУ не гарантирует получения оптимального времени выполнения только этапа решения, и наоборот. Схожая ситуация имеет место при оптимизации параметров под разные вычислительные устройства. Параметры, оптимизированные для вычислений на центральных процессорах, позволяют несколько снизить время и на графических ускорителях, но данная комбинация параметров оказывается далекой от оптимальной в сравнении с целевой оптимизацией расчета на графическом ускорителе.

Сравнение производительности оптимизированных комбинаций параметров на центральных процессорах и на графических ускорителях показало, что вопросы оптимизации параметров одинаково актуальны для обоих вычислительных систем. На использованном наборе из 16 тестовых матриц использование алгоритма оптимизации параметров позволяет снизить время решения системы уравнений при предварительно построенной иерархии матриц более, чем в 2 раза. Использование графических ускорителей позволяет сократить время расчетов как в случае базовой

комбинации параметров, так и в случае использования параметров, оптимизированных под конкретную систему уравнений и вычислительную платформу, в 3–8 раз. Фактическая величина ускорения зависит от размера матрицы и, как следствие, эффективности переиспользования кэш-памяти центрального процессора.

Список литературы

[1] Whaley R.C., Petitet A., Dongarra J.J. Automated empirical optimizations of software and the ATLAS project // Parallel Computing. 2001. Vol. 27. P. 3–35. https://doi.org/10.1016/S0167-8191(00)00087-9.

[2] Dinkelbach H.U., Bouhlal B.E., Vitay J., Hamker F. H. Auto-selection of an optimal sparse matrix format in the neuro-simulator ANNarchy // Frontiers in Neuroinformatics. 2022. Vol. 16, No 877945. https://doi.org/10.3389/fninf.2022.877945.

[3] Petrushov A., Krasnopolsky B. Advanced genetic algorithm in the problem of linear solver parameters optimization // Communications in Computer and Information Science. 2021. Vol. 1510. P. 297–309. https://doi.org/10.1007/978-3-030-92864-3_23.

[4] Petrushov A., Krasnopolsky B. Automated tuning for the parameters of linear solvers // Journal of Computational Physics. 2023. Vol. 494. No. 112533. https://doi.org/10.1016/j.jcp.2023.112533.

[5] Petrushov A.A., Krasnopolsky B.I. Tuning soft mutations of the evolution algorithm for optimizing the linear solver parameters // Lobachevskii Journal of Mathematics. 2023. Vol. 44. No. 8. P. 3148–3159. https://doi.org/10.1134/S1995080223080450.

[6] Jessup E., Motter P., Norris B., Sood K. Performance-Based Numerical Solver Selection in the Lighthouse Framework // SIAM Journal on Scientific Computing. 2016. Vol. 38. No. 5. P. S750–S771. https://doi.org/10.1137/15M1028406.

[7] Mishev I., Fedorova N., Terekhov S., Beckner B., Usadi A., Ray M., Diyankov O. Adaptive control for solver performance optimization in reservoir simulation // Proceedings of ECMOR XI — 11th European Conference on the Mathematics of Oil Recovery, 2008. https://doi.org/10.3997/2214-4609.20146368.

[8] Beyer H.-G., Schwefel H.-P. Evolution strategies — a comprehensive introduction // Natural Computing. 2002. Vol. 1. P. 3–52. https://doi.org/10.1023/A:1015059928466.

[9] Krasnopolsky B., Medvedev A. XAMG: A library for solving linear systems with multiple right-hand side vectors // SoftwareX. 2021. Vol. 14. No. 100695. https://doi.org/10.1016/j.softx.2021.100695.

[10] Davis T.A., Hu Y. The university of Florida sparse matrix collection // ACM Transactions on Mathematical Software. 2011. Vol. 38. № 1. P. 1–25.

[11] Yang, U.M. On long-range interpolation operators for aggressive coarsening // Numerical linear algebra with applications. Vol. 17. P. 453–472. https://doi.org/10.1002/nla.689.

Современные стандарты и тренды подготовки профессиональных кадров высшей квалификации в области информационных технологий

В.А. Сухомлин

МГУ имени М.В. Ломоносова,
Федеральный исследовательский центр
“Информатика и управление” РАН

Целью данной работы является анализ современных стандартов и трендов в подготовке ИТ-специалистов высшей квалификации (бакалавры, магистры, аспиранты, PhD) с акцентом на подготовку специалистов в области высокопроизводительных вычислений. Выполнен анализ современного состояния процессов международной стандартизации, включая: стандартизацию куррикулумов в системе ИТ-образования, создание отраслевых сводов знаний, стандартизацию квалификационных требований (компетенций/ навыков ИТ). Показан кризис системы ИТ-образования, ее неспособность обеспечивать потребности цифровой экономики высококвалифицированными ИТ-кадрами по многим актуальным технологическим направлениям. Рассмотрены основные требования к подготовке специалистов высшей квалификации в области суперкомпьютерных технологий, в частности, к математической, практической, общенаучной подготовке. С целью поиска путей выхода из кризиса анализируются тенденции, на основе которых возможно формирование инновационных решений, повышающих эффективность процессов подготовки ИТ-кадров. Определяется список лучших кейс-технологий, которые используются в мировой образовательной практике для подготовки IT-специалистов в области суперкомпьютерных технологий (HPC).

Ключевые слова: куррикулум, куррикулумная стандартизация, компьютинг, фреймворк, высокопроизводительные вычисления, суперкомпьютеры, компетенции и навыки, своды профессиональных знаний, онлайн-лаборатории, кэйс-технологии, кризис системы ИТ-образования.

1. Введение

В основе современных научных, промышленных и технологических вызовов лежит широкое использование компьютерных и информационных технологий (ИТ), для развития и применения которых требуются ИТ-специалисты высшей квалификации, владеющие набором языков и систем программирования, разнообразными фреймворками и технологическими платформами, суперкомпьютерными технологиями, методами и технологиями искусственного интеллекта (ИИ), аналитики больших данных,

системной инженерией, включая методы и средства моделирования и симуляции систем, создания цифровых двойников кибер-физических систем и метавселенных, а также владеющие многими другими инструментальными средствами и прикладными знаниями.

Особенно актуальной становится проблема подготовки высоко квалифицированных ИТ-специалистов, способных развивать и профессионально использовать суперкомпьютерные технологии, которые являются ключевым инструментом в таких областях, как:

- фундаментальные исследования (физика частиц, моделирование климата, исследования в биоинформатике)
- индустрия и экономика (оптимизация сложных производственных процессов, разработка новых видов материалов, финансовый анализ)
- оборона и безопасность (криптография, симуляция военных операций, моделирование и испытания военных систем нового поколения)
- медицина и фармацевтика (молекулярное моделирование и разработки новых лекарственных препаратов, персонализированная медицина) и многие другие.

Именно развитие суперкомпьютерных технологий является стратегически важной задачей для обеспечения технологического суверенитета и конкурентоспособности страны.

2. Образовательные стандарты в области ИТ

В сфере управления персоналом и подготовки профессиональных ИТ-кадров важную роль играют следующие три процесса международной стандартизации:

(а) стандартизация куррикулумов (учебно-методических материалов) по направлениям подготовки ИТ-кадров в системе образования

(б) создание сводов профессиональных знаний (Body of Knowlege — BoK) для актуальных доменов и видов деятельности области ИТ

(с) стандартизация квалификационных требований (компетенций / навыков / профилей профессиональных ролей) в области информационных и коммуникационных технологий (ИКТ, в данной работе ИКТ и ИТ будем считать синонимами).

При этом в последние годы характерным для этих процессов становится все более тесная интеграция, уровень которой в настоящее время позволяет рассматривать их как целостную систему процессов стандартизации методических основ для решения задач кадрового менеджмента и подготовки профессиональных кадров в области ИТ [1].

2.1. Куррикулумная стандартизация

Быстрое развитие ИТ-индустрии и расширение сфер ее применения требует постоянного обновления образовательных программ. При этом требования к уровню подготовки профессиональных кадров в области ИТ постоянно возрастают, что обусловлено масштабными и сложными задачами цифровой трансформации экономики.

Исторически, начиная с середины 60-х годов прошлого столетия, ведущую роль в формировании мировой системы подготовки ИТ-кадров, играли образовательные

стандарты в виде признанных на международном уровне программ учебных курсов или, так называемых, куррикулумов [2, 3].

Такие куррикулумы характеризуются тем, что:

- ориентированы на подготовку ИТ-профессионалов (бакалвров, магистров) по конкретным направлениям области ИТ (далее профилям)
- содержат описание свода знаний (СЗ) или BOK (Body of knowledge), соответствующих требованиям к профессиональным знаниям и умениям по конкретным профилям ИТ и определяющих содержание подготовки, т.е. чему учить учащихся по данному профилю
- определяют ядро свода знаний (core), содержащее описания минимально необходимых для конкретного профиля базовых знаний, которыми должны обладать все выпускники по данному профилю
- используют наборы дидактических параметров, связанных с элементами сводов знаний и определяющих рекомендуемые объемы учебной нагрузки, уровень когнитивности освоения профессиональных знаний и умений, требуемые социально -личностные характеристики
- как правило, в куррикулумах включается описание в качестве примеров образовательных программ и программ отдельных дисциплин ведущих университетов.

Процесс куррикулумной стандартизации можно разделить на три этапа:

1) Первый этап — 1968–2000 гг. (с принятия первого куррикулума по профилю computer science — CS1968 до принятия в 2000 г. новой концепции для области ИТ, получившей академическое название компьютинг (computing)). На данном этапе доминировали два направления подготовки — computer science и computer engineering, стандарты куррикулумов по которым обновлялись примерно один раз за десятилетие.

2) Второй этап — 2000–2020 гг., который определяется внедрением и развертыванием новой концепцией компьютинга, в которой дифференцированы следующие основные направления (профили) подготовки ИТ-профессионалов:

- Компьютерные науки (computer science — CS)
- Вычислительная техника (computer engineering — CE)
- Информационные системы (information systems — IS)
- Информационные технологии (information technology — IT)
- Программная инженерия (software engineering — SE) дополненные в последствии направлениями:
- Кибербезопасность (cybersecurity — Cybsec)
- Наука о данных (data science — DS)

На этом этапе обновление куррикулумов шло существенно более быстрыми темпами по сравнению с предыдущим этапом, а основным концептуальным и методическим документом куррикулумной стандартизации стал CC2005 [4], в котором определена архитектура системы куррикулумов, описаны важнейшие методологические положения, лежащие в основе куррикулумного подхода.

3) Третий этап куррикулумной стандартизации ИТ-образования начался с момента публикации (31 декабря 2020 г.) нового документа Computing Curricula 2020 (CC2020) [5], заявленного как преемник CC2005 и новый основной концептуальный и

методологический документ куррилумной стандартизации на следующее десятилетие. Как определено в CC2020, цель его разработки состояла в том, чтобы предоставить глобальное руководство в развивающейся среде компьютинга (ИТ), влияющее на программы бакалавриата в области ИТ во всем мире. Видение этого проекта заключалось в создании востребованного и надёжного набора руководящих принципов для использования (будущими) студентами, промышленностью, правительствами и образовательными учреждениями во всем мире с целью получения представления об ожиданиях выпускников компьютерных программ бакалавриата на следующее десятилетие. А миссия проекта CC2020 определялась так — создать всемирно признанный фреймворк для определения и сравнения программ бакалавриата в области компьютинга (computing или ИТ), которые отвечают растущим требованиям меняющегося технологического мира и были бы полезны для студентов, промышленности и академических кругов.

CC2020 — многоплановый документ, основные темы которого касаются таких вопросов как: уточнение понятия компьютинга, определение современной архитектуры компьютинга, состав ожидаемых направлений развития куррикулумной стандартизации, определение главной методологической концепции в разработке куррикулумов, основанной на комптентностно-базированном подходе («основное внимание должно уделяться тому, что студенты должны уметь выполнять, а не тому, чему должны учить преподаватели»), методы спецификации образовательного контента (BoK) и др.

На рис. 1 представлена текущая архитектура стандартов куррикулумов документа CC2020 [5].

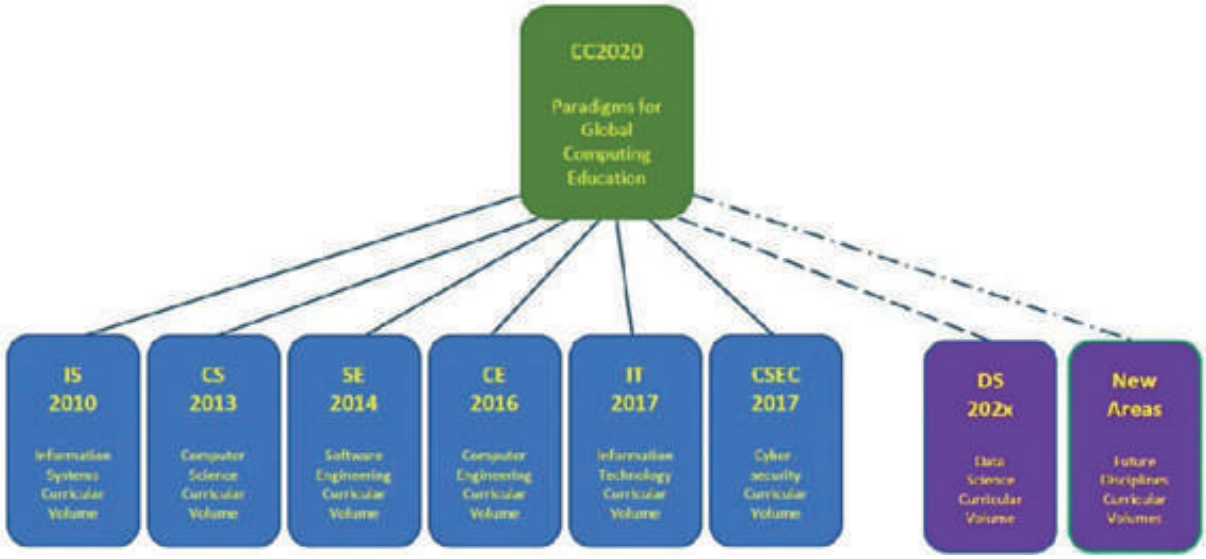


Рис. 1: Архитектура стандартов куррикулумов CC2020 [5]

Состав стандартов куррикулумов, действующих на текущий момент включает:

- Information Systems 2020 (IS2020)
- Computer Science Curricula 2023 (CS2023)
- Software Engineering Curricula 2014 (SE2014)
- Computer Engineering Curricula 2016 (CE2016)
- Information Technology Curricula 2017 (IT2017)

- Cybersecurity Curricula 2017 (CSEC2017)
- Data Science Computing Competenies 2021 (DSCC2021),
- Other emerging disciplines

В CC2020 определён набор направлений (профилей), которые в ближайшие годы могут стать самостоятельными направлениями подготовки и для которых уже ведётся разработка собственных куррикулумов. Этот набор называется «текущими куррикулумными областями» (Current curricular areas). В него входят следующие направления:

- искусственный и расширенный интеллект — ИИ (artificial and augmented intelligence (AI))
- облачные вычисления (cloud computing)
- умные города (smart cities)
- устойчивость (sustainability)
- параллельные вычисления (parallel computing)
- интернет вещей (Internet of thinks)
- граничные вычисления (edge computing)

Как отмечалось, основную часть куррикулумов составляет описание образовательного контента, (т.е. того, чему учить), называемого сводом или объемом знаний (Body of Knowledge — BoK). До 2016 г. традиционным способом представления образовательного контента являлось описание свода знаний в виде иерархической структуры, включающей такие элементы знаний, как, предметные области (areas), модули знаний (units), темы/подтемы (themes/subthemes). Такой подход к определению содержания обучения называется знание-ориентированным (knowledge-based learning — KBL). В подходе KBL уделяется также должное внимание определению результатов обучения (outcomes), которые связываются с соответствующими им дидактическими единицами свода знаний с помощью аппарата дидактических параметров, определяющих уровни когнитивности и другие аспекты результатов обучения.

Начиная с 2016 г. в куррикулумах нового поколения стало характерным применение компетентностного подхода, при котором своды знаний не определяются в явной форме, а задаются опосредованно через структурированные наборы требований к знаниям и умениям в форме компетенций, трактуемых в качестве результатов обучения. Такой подход именуется компетентностно-базированным (Competency-based learning — CBL). Описание ИТ-компетенций смещает акцент в куррикулумах с описания знаний на прагматику достижения конечного результата обучения, т.е. описание того, что выпускники могут делать в практических ситуациях. В подходе CBL более акцентированно и явно определяются цели обучение, также упрощается определение самого BOK, так как не требуется его детализация до уровня тем/подтем (такую работу придётся выполнять вузам, чтобы сконструировать учебные курсы, развивающие требуемые навыки/компетенции). Таким образом в подходе CBL образовательный контент определяется в виде иерархической структуры областей компетенций или навыков.

Именно подход CBL принимается в CC2020 как основной при разработке куррикулумов, а компетенция принимается за основу для выражения как цели обучения в ИТ-образовании, так и способности выполнять задачи на рабочем месте.

Понятие компетенция в CC2020 определяется следующим образом.

Компетенция = Знания + Навыки + Диспозиции в контексте задачи (Knowlege + Skills + Disposition -> Task)

Такая модель компетенции иллюстрируется на рис. 2.

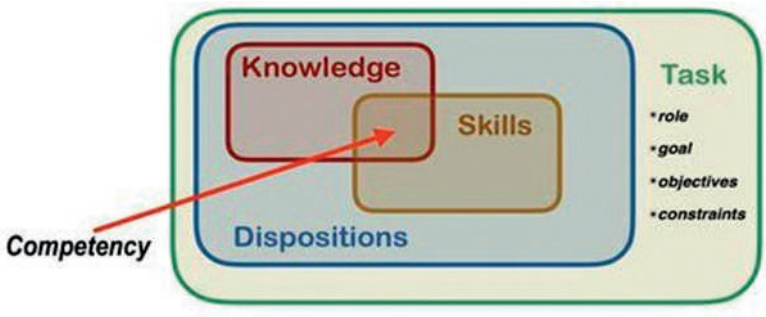


Рис. 2: Модель понятия компетенции ⟨K, S, D⟩ [5]

Таким образом для подготовки к карьере требуется, чтобы студенты развивали ряд качеств, организованных по трём параметрам: 1) знания, 2) навыки и 3) склонности (dispositions), где элементы компетенции интерпретируются следующим образом.

Знания — это компонент компетентности как фактического понимания того, что знает человек. Этот компонент статичен и инертен; на него нужно действовать со знанием дела, чтобы стать поведением.

Навыки дают возможность применять знания для активного выполнения задачи. Следовательно, навык выражает элемент знания, который используется с умением для определения аспекта компетенции «ноу-хау». В определении компетенции в CC2020 приняты уровни когнитивного процесса Блума [6], чтобы указать уровень навыков, ожидаемых при успешном выполнении задачи.

Диспозиции или предрасположенности — это привычные наклонности, которые представляют собой социально-эмоциональные склонности, пристрастия и отношения (например, надёжность, ответственность). Диспозиции определяют, склонен ли человек использовать свои навыки и как он это делает.

Задача — это конструкция, в контексте которой определяется умелое применение знаний и конкретизируются диспозиции.

Новое поколение стандартов куррикулумов в значительной мере следует рекомендациям CC2020. К ним относятся:

1) IS2020 — Модель компетенций для программ бакалавриата по информационным системам. Совместная программа ACM/AIS IS2020 (A Competency Model for Undergraduate Programs in Information Systems) [7]. IS2020 разработан в строгом соответствии с рекомендациями CC2020.

2) CCDSC2021 — Компетенции в области вычислений для учебных программ бакалавриата по науке о данных (CCDSC) (Computing Competencies for Undergraduate Data Science Curricula).

Целевая группа ACM по науке о данных. Январь 2021 г. [8]. В CCDSC2021 за основу принимается модель знаний (KBL), дополненная компетенциями для определения результатов обучения.

3) CS2023 — Учебные программы по компьютерным наукам 2023 г. ACM/IEEE-CS/AAAI (Computer Science Curricula) [9]. Куррикулум CS2023, представляющий собой комплексную ревизию предыдущей редакции (CS2013), также в основе своей применяет модель знаний, дополненная моделью компетенций в части определения способностей решения определённого набора задач.

CS2023 включает достаточно объёмную предметную область «Parallel and Distributed Computing», являющуюся фундаментом для освоения суперкомпьютерных технологий, которые рассматриваются как центральный корпус знаний и практик, пронизывающих почти все другие предметные области свода знаний этого стандарта. Параллельные и распределенные методы программирования формируют ядро высокопроизводительных вычислений (HPC), распределенных систем и почти каждого вычислительного приложения. Эта область знаний разделена на пять модулей знаний, а именно: базовые механизмы параллельных и распределенных программ и их выполнения; коммуникация (через каналы, память или общие хранилища данных), механизмы координации параллельных действий для достижения общих результатов, методы оценки с учётом спецификаций, параллельные и распределенные алгоритмы в различных прикладных областях.

Подводя некоторый итог куррикулумной стандартизации, следует отметить, что стандарты куррикулумов по указанным выше традиционным направлениям подготовки представляют собой тщательно разработанные фреймворки, чрезвычайно полезные для разработки учебных программ подготовки ИТ-кадров. Однако данный процесс явно отстаёт от текущих потребностей в развёртывании большого числа новых профилей в области ИТ, что объясняется сложностью самих фреймворков и сложностью организации процессов их разработки.

2.2. Создание профессиональных сводов знаний

Профессиональные (отраслевые) своды знаний (Body of Knowledge) или BoKs представляют собой описания знаний и умений, навыков/компетенций по конкретным направлениям ИТ или сферам деятельности, связанным с ИТ. За разработку и сопровождение таких BoKs отвечают авторитетные международные организации. Каждый BoK по существу есть евангелие по конкретному ИТ-направлению (например, программная инженерия, системная инженерия, кибербезопасность) или некоторому виду деятельности (например, управление проектами или управление деловыми отношениями). Основным содержанием BoK является описание структурированных объёмов знаний, которые используются профессионалами соответствующей дисциплины (считается примерно с трех-четырех летним стажем работы) в своей практике. Таким образом BoK определяет совокупности знаний в определённой области ИТ, которую работник должен освоить, чтобы его можно было рассматривать или сертифицировать как практикующего специалиста.

Совокупность таких BoKs составляют знаниевую основу области ИТ. Список из наиболее известных сводов знаний включает несколько десятков BoKs [10]. Примерами наиболее актуальных сводов знаний для подготовки ИТ-специалистов являются: SWEBOK V4 (Software Engineering BOK) [11], SEBOK (Systems Engineering BOK) [12], CyBOK (Cybersecurity BOK) [13].

При разработке новых куррикулумов и образовательных ресурсов отраслевые своды знаний выполняют роль базы знаний. Их исключительная полезность подвержена автором в его работах. В частности на основе CyBOK автором разработан

куррикулум «Кибербезопасность» [14–16], при этом определение свода знаний этого куррикулума было разработано на основе CyBOK, а также на основе модели навыков кибербезопасностис [17], в свою очередь разработанной на принципах современной концепции цифровых навыков, рассмотренных ниже. А на основе SEBOK и SWEBOK разработан учебник по системной и программной инженерии [18].

Сводов знаний, посвящённых суперкомпьютерным технологиям и стандартизованных на международном уровне ещё не разработано.

2.3. Стандартизация квалификационных требований

Стандартизация квалификационных требований (компетенций / навыков / профилей профессиональных ролей) в области информационных и коммуникационных технологий играет важную роль при разработке образовательных стандартов и ресурсов. Как показано в работе [10] этот процесс стандартизации тесно связан с рассмотренным выше процессом куррикулумной стандартизации. Существует множество систем (фреймворков) спецификаций и классификаций навыков и компетенций, ролей и т.п.

В сфере кадрового менеджмента наиболее известными и авторитетными фреймфорками являются:

- фреймворк навыков для информационного века (SFIA — Skills Framework for the Information Age)
- Европейский фреймворк компетенций (e-CF — European e-Competence Framework)
- The i Competency Dictionary, the Information Technology Promotion Agency (IPA) (iCD словарь i компетенций, разработанный Агентством по продвижению ИТ в Японии)
- профстандарты ИКТ

Анализ этих фреймворков и стандартов проводится в [19].

В последние годы доминирующую роль в системе кадрового менеджмента играет концепция навыков (skills). Под навыком в ней понимается комплекс характеристик исполнителя специфической части производственной деятельности (активности), необходимый для эффективного выполнения соответствующей работы на конкретном рабочем месте, благодаря тому, что такой исполнитель обладает необходимыми знаниями, ноу-хау, умениями, опытом, социально-личностными качествами. Более точно понятие навыка можно определить следующим образом.

Навык (skill) — совокупность качеств, необходимых для профессионального, эффективного выполнения некоторой роли в производственной деятельности или некоторой ее части. Навык определяется:

- **активностями** производственной деятельности (целевыми или операционными действиями в процессе реализации производственной деятельности),
- **знаниями**, необходимыми для выполнения активностей и являющимися ключевым элементом навыка, определяющим его содержание,
- **социально-личностными** качествами исполнителя навыка (называемые также мягкими навыками или диспозицией),
- **контекстом рабочего места** — часть в описании навыка, конкретизирующая дополнительные функциональные и нефункциональные требования к навыку (аспекты), связанные с жизненным циклом рабочего места.

Навыки представляют собой спецификации типовых модулей профессиональной деятельности и предназначены для описания ролей/подролей в производственных процессах.

Наиболее широко распространенным стандартом цифровых навыков является стандарт SFIA, определяющий навыки и компетенции, необходимые специалистам, которые проектируют, разрабатывают, внедряют, управляют и защищают данные и технологии, обеспечивающие работу цифрового мира [20]. Стандарты SFIA (последняя версия SFIA 9) определяют систему классификации и методику описания цифровых навыков области ИТ, соответствующих требованиям цифровой экономики. С помощью навыков системы SFIA, используемых в качестве строительных блоков, может быть описан обширный класс профессиональных ролей, связанных с областью ИКТ и ее приложениями.

Модель классификации ИТ-навыков в SFIA представляет собой трехуровневую иерархическую систему, на верхнем уровне которой навыки разбиваются на классы категорий, затем, на втором уровне, категории структурируются на подкатегории, которые в свою очередь выступают как совокупности близких по роду деятельности навыков, составляющих третий, самый нижний, уровень иерархии системы классификации. Всего в SFIA 9 определяется: 6 категорий навыков, 22 подкатегории и 148 индивидуальных навыков, причём общее описание навыка уточняется описанием каждого допустимого для него уровня исполнения (уровня ответственности).

В SFIA определены семь уровней ответственности, которые названы следующими глаголами в повелительном наклонении:

L=1 — следуй

L=2 — помогай

L=3 — применяй

L=4 — создавай возможности

L=5 — обеспечивай/советуй

L=6 — инициируй/влияй

L=7 — формулируй стратегию, вдохновляй и мобилизуй.

Семантика каждого уровня ответственности определяется по единому шаблону, содержащему следующие пять разделов, определяющие поведенческие факторы: Autonomy (Автономность), Influence (Влияние), Complexity (Сложность), Knowledge (Знание), Business skills (Бизнес навыки).

В качестве примера навыка в SFIA рассмотрим общее описание навыка «High-performance computing» (Высокопроизводительные вычисления) — HPCC, назначение которого: использование передовых компьютерных систем и специальных методов программирования для решения сложных вычислительных задач. Уровни ответственности для этого навыка: 5, 6, 7.

Основными задачами для навыка являются:

- использование суперкомпьютеров и методов параллельной обработки для решения сложных вычислительных задач с акцентом на разработку алгоритмов и систем параллельной обработки
- использование суперкомпьютеров исследовательских работах для выполнения объёмных вычислений, компьютерного моделирования, имитации и анализа больших данных.

Деятельность может включать, но не ограничиваться:

- проектирование и оптимизация параллельных алгоритмов
- управление и поддержка инфраструктур HPC
- разработка программного обеспечения для сред HPC
- проведение анализа производительности и оптимизация приложений HPC
- сотрудничество с исследователями для перевода научных проблем в решения HPC.

Технология HPC применяется в различных дисциплинах, включая: биологические науки и молекулярное моделирование, географические данные, разведку месторождений нефти и газа, моделирование климата и прогнозирование погоды, физическое моделирование, криптоанализ.

2.4. Выводы по процессам стандартизации

1) Современные артефакты рассмотренных процессов международной стандартизации в области кадрового менеджмента и подготовки профессиональных ИТ-кадров представляют собой достаточно развитую методическую основу для разработки актуальных образовательных стандартов и программ.

2) Состояние разработки новых стандартов куррикулумов по наиболее актуальным технологическим направлениям цифровой экономики, таким, как: Интернет вещей, блокчейн, суперкомпьютерные технологии, мобильная сетевая инфраструктура 5G-6G, облачные и граничные вычисления, цифровые двойники и BIM, умные города, иммерсивные технологии (включающие дополненную реальность (augmented reality — AR), виртуальную реальность (virtual reality - VR) и смешанную реальность (mixed reality — MR)), метавселенные, умный транспорт, умные материалы и др., следует признать неудовлетворительным. Это говорит о серьёзном кризисе системы ИТ-образования, ее невозможностью обеспечения потребностей цифровой экономики всем спектром высококвалифицированных ИТ-кадров, что ведёт к необходимости поиска путей повышения эффективности системы образования. В частности, такими путями могут быть:

- вовлечение в вузовские образовательные процессы специалистов-практиков из ведущих компаний цифровой экономики
- интеграция системы образования с корпоративными программами профессиональной подготовки
- освоение инновационных подходов в образовательной деятельности.

3. Особенности подготовки специалистов высшей квалификации в области суперкомпьютерных технологий

Подготовка специалистов высшей квалификации (магистров, аспирантов, исследователей) в области суперкомпьютерных технологий требует сочетания глубоких теоретических знаний и практических навыков, а также знаний в области приложений.

3.1. Математическая подготовка специалистов высшей квалификации в области суперкомпьютерных технологий

Основой подготовки специалистов высшей квалификации по суперкомпьютерным технологиям является фундаментальная математическая подготовка и подготовка по теоретическим основам компьютерных наук. Специалист в области суперкомпьютерных технологий должен владеть вычислительной математикой, методами линейной алгебры, параллельными алгоритмами, методами оптимизации и математического моделирования, а также методами математической статистики, дискретной математики и теории сложности.

В состав таких критически важных дисциплин и их основных разделов программ подготовки специалистов по суперкомпьютерным технологиям должны входить:

1) Вычислительная математика (Computational Mathematics)

- Численные методы (аппроксимация, интерполяция, численное интегрирование и дифференцирование)
- Методы решения СЛАУ (прямые и итерационные методы, разреженные матрицы)
- Оптимизация вычислений (минимизация ошибок, устойчивость алгоритмов)
- Численные методы решения дифференциальных уравнений (конечные разности, метод конечных элементов, спектральные методы)
- Разностные схемы

2) Линейная алгебра

- Матричные вычисления (разложения: LU, QR, SVD, собственные значения)
- Итерационные методы (метод сопряжённых градиентов, GMRES)
- Разреженные матрицы и их хранение (форматы CSR, CSC, ELLPACK)

3) Параллельные вычисления и распараллеливание алгоритмов

- Теория графов (распределение нагрузки, декомпозиция задач)
- Методы декомпозиции (domain decomposition, multigrid methods)
- Алгоритмы синхронизации (барьеры, атомарные операции, lock-free структуры)

4) Дискретная математика

- Теория сложности вычислений (классы P, NP, параллельные алгоритмы)
- Быстрые алгоритмы (БПФ, быстрое умножение матриц)
- Графовые алгоритмы (поиск кратчайших путей, максимальные потоки)

5) Теория вероятностей и статистика

- Стохастические методы (методы Монте-Карло, марковские процессы)
- Статистический анализ данных (обработка больших массивов)
- Машинное обучение и оптимизация (градиентные методы, SGD)

6) Математическое моделирование

- Дифференциальные уравнения в частных производных (УрЧП)
- Многомасштабное моделирование (молекулярная динамика, методы coarse-graining)
- Моделирование физических процессов (гидродинамика, квантовая механика)

7) Математические методы оптимизации

- Выпуклая оптимизация (линейное/нелинейное программирование)
- Глобальная оптимизация (генетические алгоритмы, swarm optimization)
- Оптимизация в распределённых системах (распределённый SGD, federated learning)

8) Теория информации и кодирования

- Сжатие данных (алгоритмы для больших массивов)
- Коды коррекции ошибок (в распределённых вычислениях)
- Криптография (защита данных в HPC-кластерах)

9) Формальные методы и верификация

- Математическая логика и теория автоматов (верификация параллельных алгоритмов)
- Теория типов (корректность программ)

3.2. Владение методами и инструментами параллельного и распределенного программирования

Специалист в области суперкомпьютерных технологий должен понимать архитектуру высокопроизводительных систем, владеть языками программирования, методами параллельного и распределенного программирования, методами и средствами хранения данных и организации вычислительных расчётов на суперкомпьютерных системах. В частности, актуальными темами здесь являются:

1) **Архитектуры HPC** (кластеры, GPU-ускорители, векторные процессоры, квантовые сопроцессоры)

2) **Системы хранения** (Lustre, GPFS, NVMe-over-Fabrics)

3) **Языки программирования**: C/C++, Fortran, Python (с ускорением через Numba/Cython)

4) **Параллельное и распределенное программирование**, включая:

- Парадигмы и инструментальные средства параллельного и распределенного программирования (MPI, OpenMP, CUDA, SYCL, OpenACC)
- Оптимизация кода (векторизация, cache-friendly подходы, уменьшение коммуникационных накладных расходов)
- Анализ масштабируемости (Amdahl's Law, Gustafson's Law, weak/strong scaling)

5) **Инструменты мониторинга и профилирования** (Intel VTune, Nsight, Arm Forge и др.)

6) **Инструменты управления очередями задач** (Slurm, PBS, LSF)

7) **Фреймворки поддержки суперкомпьютерных вычислений**: PETSc, Trilinos, FFTW, BLAS/LAPACK (MKL, cuBLAS)

8) **Реализация практических проектов на суперкомпьютерах**, например, на суперкомпьютерах «Ломоносов-2» или «Кристофари».

3.3. Управление данными и I/O-оптимизация в суперкомпьютерных технологиях

В высокопроизводительных вычислениях (HPC) управление данными и оптимизация ввода-вывода (I/O) играют ключевую роль, поскольку объёмы данных в научных симуляциях могут достигать петабайтов, и традиционные методы хранения и обработки становятся узким местом. Поэтому при подготовке профессионалов в области супервычислений становится актуальным изучение и освоение методов и инструментов в таких областях как:

1) **Эффективная работа с большими данными и их хранение, использование форматов с быстрым доступом** (HDF5, NetCDF, ADIOS2, распределенные файловые системы, многоуровневое хранение))

2) **Оптимизация ввода-вывода** (коллективные операции, буферизация и агрегация, staging- подходы, кэширование, сжатие))

3) **Визуализация данных во время выполнения моделирования** (In-situ и in-transit визуализация (ParaView, VisIt))

4) **Оптимизация обработки больших данных** (оптимизация доступа, кэширование, сжатие, ZFP, SZ)

5) **Промежуточная обработка** (выборка данных, фильтрация данных перед записью).

3.4. Междисциплинарность и мягкие навыки

Суперкомпьютерные технологии нашли широкое применение во многих научных и прикладных областях, поэтому специалисты по таким технологиями должны обладать прикладными знаниями в этих областях, чтобы понимать специалистов-предметников, эффективно участвовать в оптимизации вычислительных экспериментов. Такими областями, в которых суперкомпьютерные технологии нашли широкое применение, в частности являются:

- Вычислительная физика, вычислительная химия, биоинформатика (моделирование квантовых систем, молекулярная динамика)
- Климатология и прогнозирование (глобальные климатические модели)
- Машинное обучение и Big Data (обучение крупных нейросетей)
- Астрофизика и космология (N-тельные симуляции)
- Инженерные расчёты (CFD, прочность материалов).

Указанные выше домены, в которых используются суперкомпьютерные технологии, сами являются наукоемкими областями. Поэтому целенаправленная подготовка профессиональных исследователей в этих областях, владеющих суперкомпьютерными технологиями, может осуществляться на основе образовательных программ, построенных по формулам [5]:

- **Computing + X**, где «X» — некоторая прикладная для ИТ область (например, биология, астрономия, химия, экономика, лингвистика и др.). Дипломы в этой категории могут включать термин «информатика», например, медицинская информатика, юридическая информатика, биоинформатика или химическая информатика, и т.п.
- **X + Computing**, где «X» являются основными областями интересов, такими как физика, биология, медицина или другие приложения ИТ.

«X + Computing» отличается от «Computing + X» тем, что в первом базовая область — это не ИТ-направление (например, химия), тогда как во втором базовая область — это один из профилей ИТ.

Характерными для области суперкомпьютерных технологий мягкими навыками (Soft skills) или социально-личностными качествами являются научная коммуникация, навыки написания научных статей и заявок на вычислительные ресурсы, умение работать в распределенных командах.

3.5. Вывод

Подготовка специалистов в области суперкомпьютерных технологий должна сочетать:

- Фундаментальную математическую подготовку и подготовку по теоретическим основам компьютерных наук
- Профессиональный уровень программирования
- Получение практического опыта работы на реальных суперкомпьютерах
- Владение инструментами управления, мониторинга и оптимизации вычислительного процесса на суперкомпьютерных системах
- Знание одной или нескольких прикладных областей, в которых осуществляются научные симуляции с использованием суперкомпьютерных систем
- Стажировку в суперкомпьютерных центрах на реальных научных проектах
- Использование инновационных подходов в подготовке специалистов по суперкомпьютерным технологиям.

4. Внедрение в образовательную практику инноваций, соответствующих ключевым трендам в подготовке ИТ-специалистов

Как следует из сказанного выше, для своевременного обеспечения востребованными высококвалифицированными ИТ-кадрами современную цифровую экономику системе образования необходимо искать пути для оперативного развёртывания новых актуальных направлений подготовки и привлечения в учебные процессы экспертов-практиков.

Анализ возможных путей решения этой проблемы позволяет выявить следующие тенденции, на основе которых возможно формирование инновационных решений, повышающих эффективность процессов подготовки ИТ-кадров. К ним в первую очередь следует отнести:

1) **Интеграцию академического и корпоративного образования**

- совместные программы с ведущими ИТ-компаниями (например, Яндекс.Практикум, Huawei Academy)
- широкое использование наборов реальных кейсов
- включение в учебный процесс активных форм коллективной деятельностной практики типа хакатонов

2) **Гибкие и персонализированные траектории обучения**

- допущение обучения дисциплинам по выбору студентов с помощью технологий Micro-credentials и нанодипломов (Coursera, edX)
- использование методов адаптивного обучения на основе средств ИИ

3) **Дистанционные и гибридные форматы**

- онлайн-лаборатории
- VR-симуляторы (например, для обучения кибербезопасности)
- программы двойных дипломов и пр.

Учитывая фундаментальный характер подготовки ИТ-специалистов по суперкомпьютерным технологиям, весьма плодотворным и практичным подходом к повышению качества подготовки представляется развитие кейс-технологий в сочетании с дистанционными формами обучения.

Кейс-технология (от англ. «case» — случай) — интерактивная технология обучения, направленная на формирование у обучающихся знаний, умений, личностных качеств на основе анализа и решения реальной или смоделированной проблемной ситуации в контексте профессиональной деятельности, представленной в виде кейса [21]. При этом описание ситуации, содержащей проблему, представляется в виде, вызывающем дискуссию и активное обсуждение. Обучающимся предлагается проанализировать ситуацию, возможно, предварительно изучив дополнительные источники информации, и предложить возможные варианты решения и выбрать лучший из них.

Применение кейс-технологии в обучении позволяет реализовать проблемное обучение, оценить сформированность компетенций (способность работать в команде, способность к самоорганизации и самообразованию, способность осуществлять поиск, хранение, обработку и анализ информации из различных источников и баз данных, представлять ее в требуемом формате с использование информационных, компьютерных и сетевых технологий, способность учитывать современные тенденции развития техники и технологий в своей профессиональной деятельности и др.). Как правило, кейс-технология направлена на развитие междисциплинарных знаний и умений, способствует развитию метапредметных знаний и умений обучающихся, включая коммуникативные навыки, умение работать в команде, проявлять гибкость, улаживать конфликтов, умение убеждать и искать компромиссы и др.

В мировой образовательной практике для подготовки IT-специалистов в области суперкомпьютерных технологий (HPC) активно используются кейс-технологии, включая реальные задачи из науки и промышленности.

К наиболее продвинутым решениям применения кейс-технологий в HPC можно отнести:

1. **PRACE** (Partnership for Advanced Computing in Europe). PRACE предлагает образовательные программы и кейсы на основе реальных задач HPC, включая моделирование климата, квантовую физику и биоинформатику. Примеры кейсов:

- PRACE Case Studies — коллекция реальных задач с решениями[1]
- PATC (PRACE Advanced Training Centres) — курсы с практическими кейсами[2]

[1] `https://prace-ri.eu/training-support/case-studies/`

[2] `https://events.prace-ri.eu/courses`

2. **XSEDE** (Extreme Science and Engineering Discovery Environment, США). Программа предлагает кейсы для обучения работе с суперкомпьютерами, включая параллельные вычисления и оптимизацию кода. Примеры кейсов:

- XSEDE HPC Case Studies — учебные материалы и задачи[1]
- HPC University — ресурсы с практическими заданиями[2]

3. **NVIDIA Deep Learning Institute (DLI) + HPC**. NVIDIA предлагает кейсы по использованию GPU в HPC, включая задачи машинного обучения и физического моделирования. Примеры кейсов:

- NVIDIA DLI HPC Courses — практические лаборатории[3]
- CUDA Case Studies — оптимизация вычислений на GPU[4]

4. **Oak Ridge National Laboratory (ORNL) — HPC Training**. ORNL проводит школы (например, HPC Summer School) с реальными кейсами из энергетики и материаловедения[5] Примеры кейсов:

- ORNL Case Studies — разбор задач на суперкомпьютере Summit[6]

5. **Intel® HPC Academy**. Описание: Intel предоставляет кейсы по оптимизации кода для процессоров Xeon и ускорителей. Примеры:

- Intel HPC Toolkit Case Studies — примеры оптимизации[7]

6. **MIT Lincoln Laboratory — Supercomputing Challenges**. MIT использует кейс-методы в курсах по параллельным вычислениям. Пример:

- MIT Lincoln Lab HPC — задачи из оборонных и научных проектов[8]

7. **SURF** (Нидерланды) — **HPC Education** Голландский центр SURF предлагает кейсы по использованию суперкомпьютеров в исследованиях. Пример:

- SURF HPC Cases — учебные материалы[9]

8. Российские практики: МГУ, МФТИ, РАН Примеры:

- Суперкомпьютерный консорциум университетов России — кейсы на базе "Ломоносова" (МГУ)[10]
- МФТИ — HPC School — задачи по моделированию[11]

Таким образом к лучшим кейс-практикам можно отнести:

- реальные научные и инженерные кейсы (PRACE, ORNL)
- практические лаборатории на суперкомпьютерах (NVIDIA DLI, XSEDE)
- оптимизационные задачи (Intel, CUDA)

Для углублённого изучения суперкомпьютерных технологий безусловный интерес представляют открытые курсы (Coursera, edX) и материалы конференций (SC, ISC High Performance).

[1] `https://www.xsede.org/education-and-outreach`
[2] `https://hpcuniversity.org/`
[3] `https://www.nvidia.com/en-us/training/`
[4] `https://developer.nvidia.com/cuda-education`
[5] `https://www.olcf.ornl.gov/for-users/training/hpc-user-training/`
[6] `https://www.olcf.ornl.gov/case-studies/`
[7] `https://www.intel.com/content/www/us/en/developer/tools/oneapi/hpc-toolkit.html`
[8] `https://www.ll.mit.edu/r-d/high-performance-computing`
[9] `https://www.surf.nl/en/high-performance-computing`
[10] `https://hpc-russia.ru/`
[11] `https://mipt.ru/education/chairs/supercomputers/`

5. Заключение

Медленные темпы развития международной стандартизации в области ИТ-образования, быстрое устаревание знаний, нехватка квалифицированных преподавателей-практиков, разрыв между теорией и требованиями индустрии — основные причины кризиса а области ИТ-образования, ее неспособностью обеспечивать потребности цифровой экономики высококвалифицированными ИТ-кадрами по многим актуальным технологическим направлениям. Все это ведёт к необходимости к широкому внедрению инновационных решений, повышающих эффективность процессов подготовки ИТ-кадров. В работе выполнен анализ современных стандартов и трендов в подготовке ИТ-специалистов высшей квалификации (бакалавры, магистры, аспиранты, PhD), в частности, анализ современного состояния процессов международной стандартизации, включая: стандартизацию куррикулумов в системе ИТ-образования, создание отраслевых сводов знаний, стандартизацию квалификационных требований (компетенций/ навыков ИТ). Рассмотрены основные требования к подготовке специалистов высшей квалификации в области суперкомпьютерных технологий, в частности, к математической, практической, общенаучной подготовке. С целью поиска путей выхода из кризиса анализируются тенденции, на основе которых возможно формирование инновационных решений, повышающих эффективность процессов подготовки ИТ-кадров. Определяется список лучших кейс-технологий, которые используются в мировой образовательной практике для подготовки IT-специалистов в области суперкомпьютерных технологий (HPC).

Ключевым фактором в повышении эффективности системы подготовки ИТ-кадров высшей квалификации, особенно в области суперкомпьютерных технологий — стратегического направления в обеспечении государственной независимости, несомненно является тесное взаимодействие и партнёрство университетов, бизнеса и государства.

Список литературы

[1] Сухомлин В.А., Зубарева Е.В. Новый этап международной стандартизации ИТ-образования // Международный научный журнал «Современные информационные технологии и ИТ-образование», [S.l.], v. 17, n. 3, sep. 2021. http://sitito.cs.msu.ru/index.php/SITITO/article/view/794/

[2] Сухомлин В.А. Международные образовательные стандарты в области информационных технологий // Прикладная информатика, 2012, № 1(37), с. 33-54.

[3] Sukhomlin V., Zubareva E. Analytical Review of the Current Curriculum Standatds in Information Technologies. Communications in Computer and Information Science. Conference paper. First Online: 12 May 2020. pp 3-41, 13th International Conference, SITITO 2018, Moscow, Russia, November 29 — December 2, 2018, Revised Selected Papers. Conference proceedings© 2020. https://doi.org/10.1007/978-3-030-46895-8_1

[4] Computing Curricula 2005 (CC2005). Association for Computing Machinery and Computer Society of IEEE. [Электронный ресурс] https://www.acm.org/binaries/content/assets/education/curricula-recommendations/cc2005-march06final.pdf

[5] Computing Curricula 2020 (CC2020). Association for Computing Machinery and Computer Society of IEEE. [Электронный ресурс] https://www.acm.org/binaries/content/assets/education/curricula-recommendations/cc2020.pdf

[6] Anderson L.W. et al. A taxonomy for learning, teaching, and assessing: A revisions of Bloom's taxonomy of educational objectives, abridged edition, (White Plains, NY Longman, 2001)

[7] A Competency Model for Undergraduate Programs in Information Systems The Joint ACM/AIS IS2020. [Электронный ресурс] https://www.acm.org/binaries/content/assets/education/curricula-recommendations/is2020.pdf

[8] Computing Competencies for Undergraduate Data Science Curricula (CCDSC) ACM Data Science Task Force January 2021. [Электронный ресурс] https://www.acm.org/binaries/content/assets/education/curricula-recommendations/dstf_ccdsc2021.pdf

[9] CS2023. [Электронный ресурс] https://csed.acm.org/wp-content/uploads/2024/01/Body-of-Knowledge-v1-bookmarksv2.pdf

[10] [Электронный ресурс] https://sfia-online.org/en/tools-and-resources/bodies-of-knowledge/list-of-bodies-of-knowledge

[11] SWEBOK V4 2025 (Software Engineering BOK). [Электронный ресурс] https://ieeecs-media.computer.org/media/education/swebok/swebok-v4.pdf

[12] SEBoK v. 2.11, released 25 November 2024. [Электронный ресурс] https://sebokwiki.org/w/images/sebokwiki-farm!w/5/5c/Guide_to_the_Systems_Engineering_Body_of_Knowledge_v2.11.pdf

[13] CyBOK (Cybersecurity BOK) Version 2.1 June 5, 2019. [Электронный ресурс] https://www.academia.edu/40172072/CyBOK_Cyber_Security_Body_Of_Knowledge

[14] Сухомлин В.А. Куррикулум дисциплины «Кибербезопасность»: научное издание / В.А. Сухомлин, С.В. Лебедь, О.С. Белякова, А.С. Климина, М.С. Полянская. – Москва: Фонд «Лига интернет-медиа», 2022. – 402 с. https://doi.org/10.25559/f6676-8117-2920-j

[15] Сухомлин В.А. Архитектура и принципы разработки куррикулума для дисциплины "Кибербезопасность". Международный научный журнал «Современные информационные технологии и ИТ-образование», [S.l.], v. 16, n. 4, p. 927-939, dec. 2020. https://doi.org/10.25559/SITITO.16.202004.927-939

[16] Сухомлин В.А. Создание профиля "Кибербезопасность и искусственный интеллект". Международный научный журнал «Современные информационныетехнологии и ИТ-образование», [S.l.], v. 17, n. 3, sep. 2021.

[17] Сухомлин В.А., Белякова О.С., Климина А.С., Полянская М.С., Русанов А.А. Фреймворк «Модель навыков кибербезопасности» — 2020. Научное издание.

[18] Сухомлин В.А., Романов В.Ю., Гапанович Д.А. Введение в модельно-ориентированную системную и программную инженерию (MBSSE): учебник / В.А. Сухомлин, В.Ю. Романов, Д.А. Гапанович. — Москва : Фонд «Лига интернет-медиа»; МАКС Пресс, 2024. — 672 с. https://doi.org/10.29003/m4300.978-5-317-07289-6.

[19] Сухомлин В.А. Система развития цифровых навыков ВМК МГУ & Базальт СПО. Методика классификации и описания требований к сотрудникам и содержанию образовательных программ в сфере информационных технологий / В.А. Сухомлин, Е.В. Зубарева, Д.Е. Намиот, А.В. Якушин. — М.: Базальт СПО; МАКС Пресс, 184 с. https://doi.org/10.29003/m2575.978-5-317-06336-8

[20] SFIA 9 Home. [Электронный ресурс] https://sfia-online.org/en/sfia-9

[21] Описание образовательной технологии «Кейс-технология». Материалы ИТМО https://edu.itmo.ru/files/95

Стратегии высокоуровневого синтеза для многокристальных реконфигурируемых вычислительных систем

А.И. Дордопуло[1], И.И. Левин[1,2], В.А. Гудков[1,2], А.А. Гуленок[1]

[1]ООО «НИЦ супер-ЭВМ и нейрокомпьютеров»
[2]Южный федеральный университет

В статье рассматриваются стратегии преобразования программ на языке С в конфигурационные файлы параллельно-конвейерных решений для многокристальных реконфигурируемых вычислительных систем (РВС), реализованные в программном комплексе высокоуровневого синтеза “Тесей”. В отличие от большинства известных средств высокоуровневого синтеза, для преобразования не требуется предварительной разметки кода программы, аппаратный ресурс синтезируемого решения не ограничен одним кристаллом ПЛИС, синхронизация информационных и управляющих сигналов для всех используемых ПЛИС и создание управляющей программы выполняются автоматически. Входная последовательная программа на языке С в автоматическом режиме преобразуется сначала в информационный граф, затем в кадровую структуру, которая с помощью формальных методов отображается на заданную пользователем конфигурацию РВС. Рассмотрены стратегии преобразования программ различных предметных областей (символьная обработка, линейная алгебра, цифровая обработка сигналов), реализованные в отдельном программном компоненте «Ариадна». Использование стратегий преобразования позволяет сократить время поиска рационального решения для заданной предметной области.

Ключевые слова: высокоуровневый синтез, HLS, трансляция программ, язык С, редукция производительности, реконфигурируемая вычислительная система, программирование многопроцессорных вычислительных систем

1. Введение

Высокоуровневый синтез (High Level Synthesis, HLS) — один из наиболее популярных современных подходов к проектированию специализированных вычислительных устройств в архитектуре программируемых логических интегральных схем (ПЛИС). HLS-компиляторы [1, 2], стремительно завоевывающие позиции среди средств автоматизированного схемотехнического проектирования, преобразуют программы на одном из языков высокого уровня в конфигурационные файлы специализированных аппаратных средств на языках описания аппаратуры HDL (Hardware description language). Автоматизированное преобразование последовательных программ в специализированную вычислительную архитектуру позволяет снизить требования к квалификации разработчиков и сократить время разработки, ведь именно скорость со-

здания конфигурационных файлов, а не эффективность кода и/или сокращение аппаратных затрат, является основным показателем большинства средств высокоуровневого синтеза.

Результатом работы HLS-компилятора является сложный функциональный блок (IP-ядро), представляющий собой цифровой автомат или специализированный процессор, выполняющий вычисления быстрее за счет прямой реализации информационных зависимостей и свойств архитектуры ПЛИС даже при существенной (по сравнению с микропроцессорами) разнице в тактовой частоте работы. Поэтому автоматизация создания IP-ядра HLS-компилятором облегчает портацию вычислений в архитектуру ПЛИС, но не гарантирует их эффективную реализацию, т.к. организация потоков данных возлагается на программиста, а средства масштабирования и синхронизации решения (хотя бы в пределах одного кристалла) отсутствуют. Для реконфигурируемых вычислительных систем (РВС) [3], содержащих множество ПЛИС, связанных пространственной коммутационной системой, настройка синхронизации и согласования одновременной работы множества IP-ядер является сложным и трудоемким процессом, сравнимым по сложности с разработкой нового схемотехнического проекта.

В статье представлены текущие результаты разработки программного комплекса высокоуровневого синтеза «Тесей» [4, 5] с автоматическим преобразованием последовательных программ на языке C в стандарте ISO/IEC 9899:1999 к доступному аппаратному ресурсу многокристальных РВС. Статья организована следующим образом. Во втором разделе представлен краткий обзор известных средств HLS и структура комплекса «Тесей». В третьем разделе описана методика синтеза стратегий новой программой «Ариадна». В четвертом разделе представлены разработанные стратегии преобразования программ различных предметных областей (символьная обработка, линейная алгебра, цифровая обработка сигналов). В заключении обобщаются полученные результаты и формулируются направления дальнейших исследований.

2. Сравнение существующих HLS-решений с комплексом «Тесей»

Современные инструменты высокоуровневого синтеза можно отнести к одной из следующих групп:

- академические, такие как DWARV [6], BAMBU [7], LEGUP [8], разрабатываемые научными коллективами для решения исследовательских задач;
- коммерческие, например, CatapultC, Vivado HLS [9], Vivado Vitis [10], предназначенные для широкого круга практических задач:
- инициативные исследовательские проекты: (CoDeveloper, CHiMPS, DK Design Suite) [10], направленные на решение частных задач.

Подробные обзоры возможностей, ограничений и отличительных характеристик HLS-компиляторов из этих групп представлены в работах [1, 2, 11–13]. Несмотря на различия в методах преобразования кода и форматах данных, HLS-компиляторы имеют общие ограничения [1, 2, 11–13]:

1) Синтез проекта выполняется для одного кристалла ПЛИС, размещенного на отладочной плате или однокристальной РВС.

2) Для создания эффективного проекта требуется предварительная разметка кода директивами (например, #pragma в Vivado), что требует глубоких, а подчас и экспертных знаний, о синтезируемом проекте.
3) Синхронизация и масштабирование IP-ядер в решении выполняется программистом самостоятельно.

Недостатком традиционных HLS-компиляторов является ориентация на однокристальные РВС, что упрощает синтез, но ограничивает масштабируемость. Задачи, требующие одновременной работы нескольких IP-ядер (например, обработка больших данных или нейросетевые модели), вынуждают разрабатывать сложные системы синхронизации вручную. Это увеличивает время разработки и повышает вероятность ошибок, особенно при работе с несколькими ПЛИС.

Синтез многокристальных решений существенно сложнее, поскольку требует поддержки и синхронизации большого количества вычислительных устройств для различных форм организации вычислений, а также анализа и выбора рациональных вариантов из множества возможных, поэтому по сложности сравним с автоматическим распараллеливанием.

В отличие от существующих HLS-решений, «Тесей» разработан для синтеза вычислений в многокристальных РВС, где критически важны автоматическая синхронизация множества вычислительных устройств между несколькими ПЛИС и поддержка сложных форм параллелизма (макроконвейер, конвейер в конвейере).

Рис. 1: Структура взаимодействия программ комплекса «Тесей»

Ключевые компоненты и схема работы комплекса «Тесей» представлены на рис. 1:

1) «Ангел» преобразует последовательный код на C в параллельную кадровую структуру на языке COLAMO.

2) «Кентавр» генерирует ресурсонезависимую программу на COLAMO, абстрагированную от аппаратных ограничений.
3) «Ариадна» анализирует возможные стратегии преобразования программы и оценивает возможность реализации специализированных форм организации вычислений (макроконвейер и др.), выбирая наиболее рациональные по критериям времени выполнения и занятым аппаратным ресурсам.
4) «Прокруст» адаптирует программу к архитектуре заданной РВС, подбирая параметры кадровых структур.
5) «Синид», при необходимости, редуцирует производительность кадровых структур, сокращая аппаратные затраты для реализации вычислений.

Исходная последовательная программа преобразуется «Ангелом» в информационный граф — конечный ориентированный ациклический граф, вершины (подграфы) которого соответствуют операциям над данными, а дуги отражают информационную зависимость между ними. Информационно-независимые вершины расположены в слоях, соответствующих ярусам графа алгоритма [12], а информационная зависимость подграфов из разных слоев отражается итерациями. Распределение подграфов по слоям и итерациям позволяет наглядно представить информационные зависимости между фрагментами задачи на различных уровнях иерархии. Кроме того, информационный граф инвариантен к различным формам описания вычислений в последовательной программе, поэтому разные по синтаксису и форме последовательные программы одной и той же задачи представляются одним и тем же информационным графом.

С помощью замены операционных вершин вычислительными устройствами, а дуг — связями коммутационной системы, информационный граф преобразуется в максимально параллельную *кадровую структуру* [5, 9], объединяющую вычислительную структуру задачи и правила организации потоков данных. Кадровая структура характеризуется количеством слоев и итераций, разрядностью и числом устройств в базовом подграфе, интервалом обработки данных, латентностью и частотой работы, которые вместе определяют параллелизм, время решения и занимаемый аппаратный ресурс. Для одной и той же прикладной задачи может быть синтезировано множество кадровых структур, обеспечивающих информационную эквивалентность вычислений, но различающихся вычислительными структурами, потоками данных и аппаратными затратами, что позволяет адаптировать кадровую структуру к доступному ресурсу различных РВС.

Для этого созданная «Ангелом» максимально параллельная кадровая структура преобразуется «Кентавром» в масштабируемую параллельно-конвейерную (ресурсонезависимую) форму, позволяющую с помощью значений параметров изменять параллелизм отдельных фрагментов задачи.

Расчет и согласованный подбор значений коэффициентов параллелизма всех фрагментов параллельно-конвейерной программы выполняется в соответствии с методикой [4], что позволяет адаптировать аппаратные затраты кадровой структуры к доступному ресурсу даже при нелинейной зависимости и найти рациональную кадровую структуру, обеспечивающую заданный уровень производительности.

Результатом работы комплекса «Тесей» является программа на языке высокого уровня COLAMO для доступного вычислительного ресурса с заданным (либо

наиболее близким к заданному) уровнем реальной производительности. Транслятор языка программирования COLAMO преобразует программу в конфигурационные файлы ПЛИС многокристальных РВС, синтезатор многокристальных решений Fire!Constructor выполняет автоматическую синхронизацию распределенных по разным кристаллам фрагментов решения, а синтезатор Xilinx Vivado создает загрузочные конфигурационные файлы (*.bit) для каждого кристалла.

3. Синтез стратегий преобразования: программа «Ариадна»

Программа синтеза стратегий преобразования кадровых структур «Ариадна» является новым компонентом комплекса «Тесей», предназначенным для реализаций стратегий преобразования для различных предметных областей. Входными данными для «Ариадны» является полученная от «Кентавра» параллельная программа на языке COLAMO с единичными коэффициентами масштабирования структурной реализации базового подграфа транслируемой задачи. Базовый подграф — основной подграф информационного графа, структурная (аппаратная) реализация которого позволяет выполнить все вычисления задачи. «Ариадна» оценивает аппаратные затраты на реализацию этой программы в РВС по всем компонентам ПЛИС (LUT, FlipFlop, DSP, BRAM и др.) и определяет критический аппаратный ресурс — компонент ПЛИС, который в наибольшей степени превышает доступный аппаратный ресурс для решаемой задачи. Кроме этого «Ариадна» анализирует интервал обработки данных, количество каналов распределенной памяти и латентность. Далее, в соответствии с методикой преобразования кадровых структур [4], для выбранного критического ресурса «Ариадна» определяет последовательность выполнения преобразований (или стратегию) входной кадровой структуры с учетом особенностей ее информационных зависимостей и базового подграфа.

При выборе преобразований «Ариадна» анализирует динамику нелинейного изменения аппаратных затрат $A_{CS} = (a_1^{CS}, a_2^{CS}, \ldots, a_n^{CS})$ кадровой структуры при изменении ее основных параметров быстродействия $P_{CS} = (p_1, p_2, \ldots, p_n)$ [4] с учетом информационных зависимостей внутри и между подграфами (фрагментами) задачи. К параметрам быстродействия кадровой структуры P_{CS} относятся L_i^{it} — число информационно-независимых подграфов в слое F_{ij}, It_i^{it} — число итераций в функции Φ_i, Ch_i^{it} — число внешних каналов кадровой структуры, Op_i^{it} — число устройств кадровой структуры FG^{it}, ρ_i^{it} — разрядность обрабатываемых данных, способ аппаратной реализации базового подграфа g_i и интервал обработки данных I. Каждый из этих параметров, в зависимости от предметной области, структуры и размерности решаемой задачи, может являться определяющим для критического аппаратного ресурса преобразуемой программы. Выбор первого изменяемого параметра быстродействия, как правило, определяет остальные преобразования стратегии (рис. 2).

На рис. 2 представлены различные режимы совместной работы «Прокруста» и «Ариадны» при синтезе стратегий преобразования кадровых структур: на рис. 2,*а* приведена схема синтеза одной, наиболее рациональной, стратегии с однократным запуском «Прокруста» для выполнения преобразований; схема на рис. 2,*б* соответствует синтезу множества стратегий для одной программы с запуском «Прокруста» для каждой из них, а схема на рис. 2,*в* соответствует поэтапному выбору наиболее

рациональной кадровой структуры с запуском «Прокруста» на каждом шаге выполнения преобразований.

На рис. 2 приняты следующие обозначения выполняемых преобразований: $K(R)$ — расчет и подбор количества каналов чтения из внешней памяти; L — расчет и подбор количества структурно реализованных информационно-независимых подграфов; I — расчет и подбор количества структурно реализованных информационно-зависимых подграфов; $K(W)$ — расчет и подбор количества каналов записи во внешнюю память; PIP — построение конвейера в конвейере; $PROC$ — процедурная реализация блоков и построение макроконвейера.

Приведенная на рис. 2,*а* схема соответствует наиболее часто встречающейся стратегии преобразования кадровой структуры, схема на рис. 2,*б* анализирует большинство наиболее перспективных стратегий, а схема на рис. 2,*в* реализует алгоритм перебора возможных вариантов с выбором наиболее рационального варианта преобразований на каждом шаге.

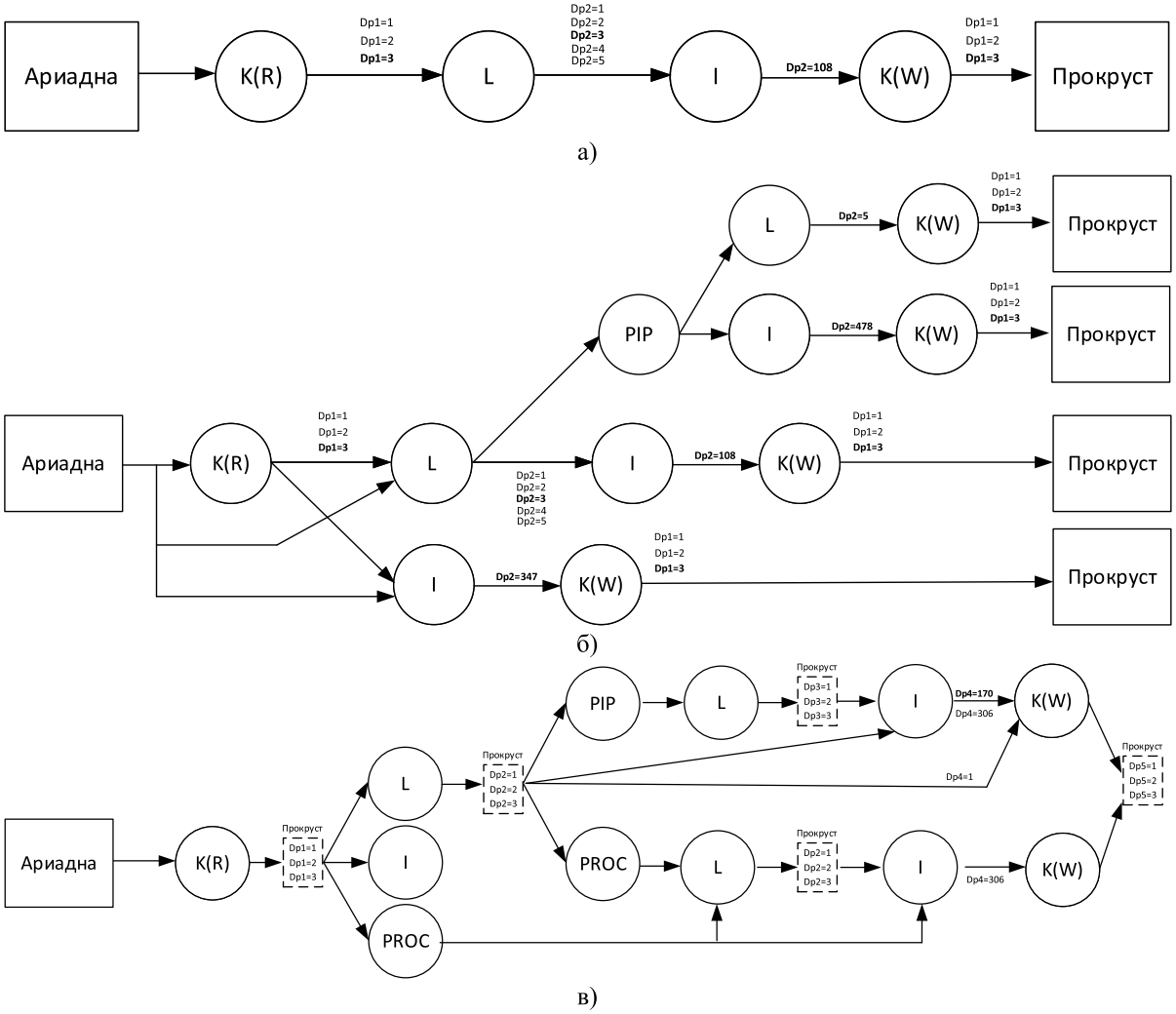


Рис. 2: Различные варианты синтеза стратегий преобразования кадровой структуры

Для выбранного критического аппаратного ресурса «Ариадна» определяет границы и рациональный шаг изменения параметров быстродействия кадровой структуры $P_{CS} = \{L_i^{it}, It_i^{it}, Ch_i^{it}, Op_i^{it}, \rho_i^{it}, g_i\}$, а также список возможных и/или необходи-

мых оптимизационных преобразований (конвейер в конвейере, макроконвейер, автоподстановка) для синтеза потенциально рациональной кадровой структуры. Таким образом, «Ариадна» определяет последовательность выполнения и основные параметры преобразований (рис. 2). «Прокруст» выполняет все преобразования кадровой структуры в соответствии с переданной ему стратегией как единую транзакцию, чтобы сократить накладные временные расходы на многократный вызов программы. Результатом реализации стратегии является создаваемая «Прокрустом» кадровая структура.

При наличии нескольких созданных «Ариадной» стратегий (рис. 2,*б* и 2,*в*) результирующие кадровые структуры помещаются в список, для всех элементов которого анализируются эффективность и время решения задачи, после чего выбирается наиболее рациональная кадровая структура, которая и является решением.

4. Стратегии преобразования программ для различных предметных областей

Одной из основных задач при формировании стратегии являются исключение заведомо неэффективных направлений движения в пространстве кадровых структур [5] и сокращение количества синтезируемых кадровых структур, каждая из которых должна удовлетворять заданным критериям эффективности — как правило, производительность не должна быть ниже 60% от пиковой.

Основная стратегия преобразования кадровой структуры (рис. 3) состоит в структурной реализации базового подграфа и последующем масштабировании сначала количества слоев до ограничений максимального числа каналов в системе, затем — итераций, влияющих на занимаемый аппаратный ресурс, потом — сокращение интервала обработки данных с помощью автоподстановки и/или других оптимизационных преобразований.

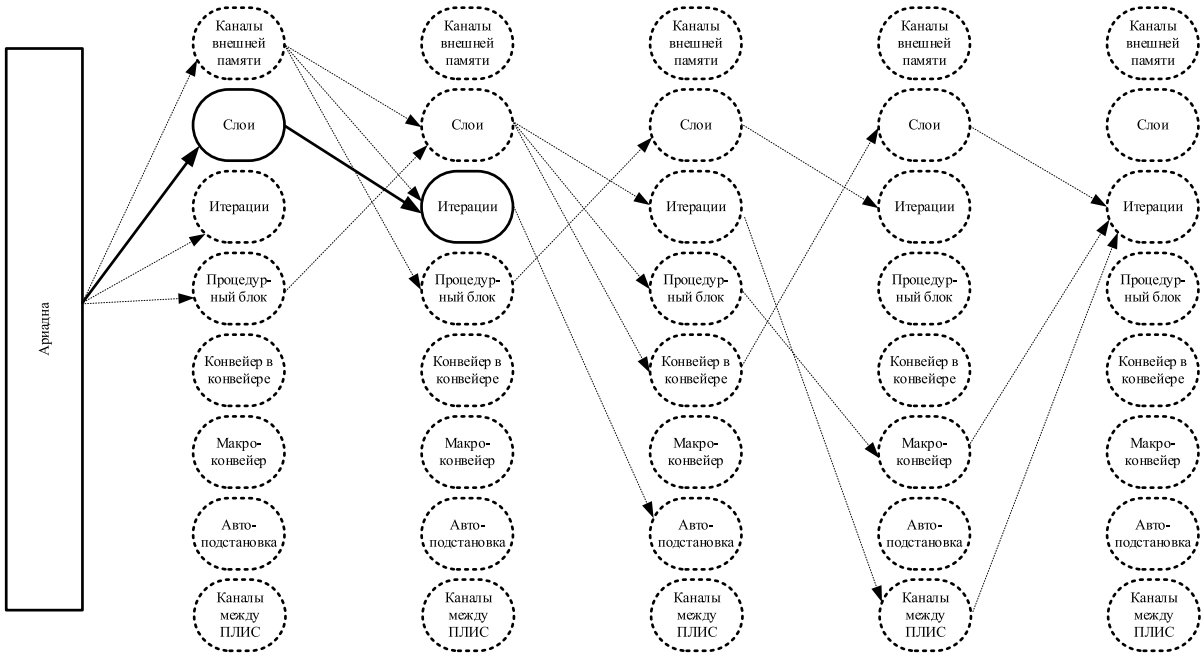


Рис. 3: Базовая стратегия преобразования кадровой структуры для задач символьной обработки

Эта стратегия, характерная для задач символьной обработки, не является непременно обязательной к выполнению, она может изменена в зависимости от типа и предметной области преобразуемой задачи, а ее выбор обусловлен наибольшей частотой встречаемости в задачах. Выбор такого порядка выполнения преобразований обусловлен следующими причинами:

- масштабирование (распараллеливание) по слоям линейно повышает производительность;
- распараллеливание по итерациям повышает производительность близко к линейной, но зачастую требует буферизации, которая в ряде случаев может достигать половины длины обрабатываемого потока данных;
- типовые методы сокращения интервала обработки данных — автоподстановка и замена переменных — существенно, в три-четыре раза увеличивают аппаратные затраты, а производительность при этом растет нелинейно.

Если в результате выполнения всех преобразований стратегии полученное решение удовлетворяет требованиям занимаемого ресурса и получен заданный уровень производительности, то цель считается достигнутой, и дальнейшие преобразования прекращаются. В противном случае, если заданный уровень производительности не достигнут, происходит изменение приоритетов выполнения преобразований и переход к следующей стратегии.

Если базовый подграф кадровой структуры входной параллельной программы поместился в доступный аппаратный ресурс, но не достигнут заданный уровень производительности (обычно 60%), необходимо снова определять стратегии. Порядок выполнения преобразований значительно зависит от типа и предметной области решаемой задачи: для задач линейной алгебры (рис. 4) и макроконвейерной обработки (рис. 5) стратегии будут различаться.

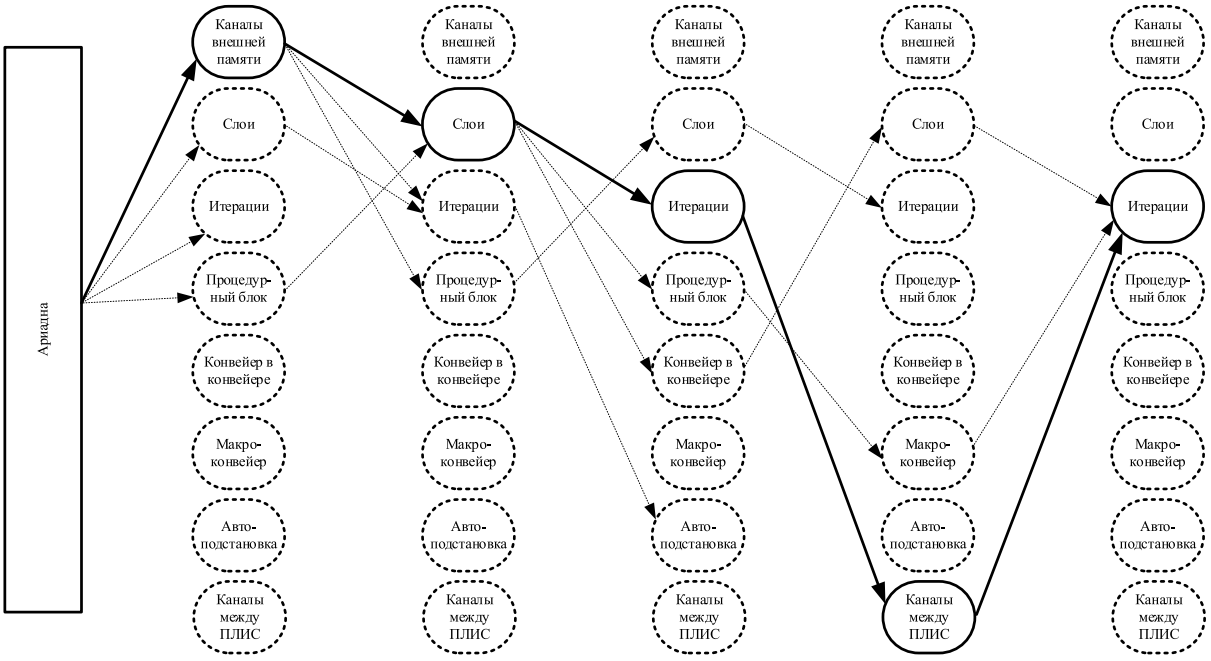


Рис. 4: Стратегия преобразования кадровой структуры для задач линейной алгебры

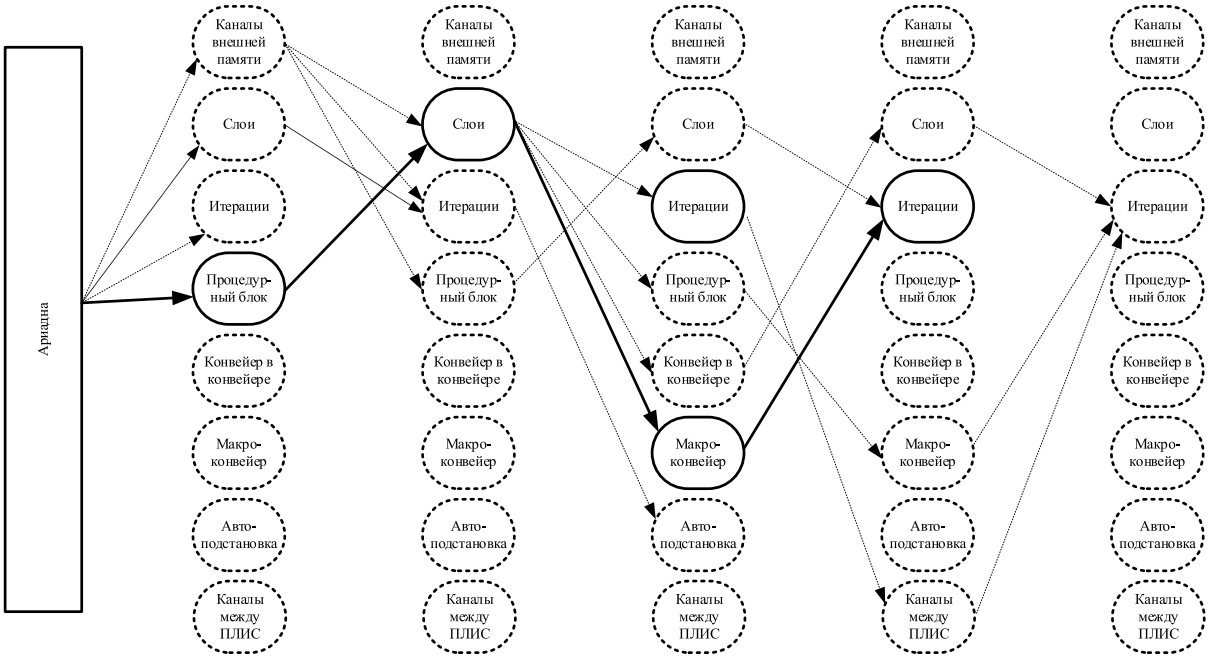


Рис. 5: Стратегия преобразования кадровой структуры при построении макроконвейера

Например, при реализации вложенных конвейеров сокращение интервала обработки данных становится даже более эффективным, чем распараллеливание по слоям. Причиной этого является линейный рост производительности решения практически без дополнительных аппаратных затрат (добавляется лишь цепочка регистров в цепи обратной связи). Поэтому для задач, содержащих такие фрагменты, первым преобразованием стратегии целесообразно выполнять оптимизацию интервала обработки данных.

При решении задач цифровой обработки сигналов, как ни странно, распараллеливание по слоям приводит к нелинейному росту производительности при увеличении аппаратного ресурса. Это связано с тем, что увеличение каналов обработки требует динамической коммутации, что усложняет не только коммутацию, но и адресацию памяти, что, в свою очередь, способствует разрывам информационных потоков. Поэтому порядок выполнения преобразований для таких задач целесообразно изменить по сравнению с базовой стратегией и первым преобразованием выполнить редукцию по числу каналов связи.

В ряде сложных задач полная реализация базового подграфа со всеми операционными вершинами требует огромного числа информационных каналов, что не позволяет применять распараллеливание по слоям. В этом случае даже при достаточности аппаратного ресурса для реализации базового подграфа его необходимо редуцировать, чтобы сократить количество каналов, которые являются критическим ресурсом.

Стратегия преобразования кадровой структуры входной параллельной программы не фиксирована и в значительной степени определяется информационной структурой задачи и типом информационных зависимостей. Так, для целого ряда задач символьной обработки характерно отсутствие итераций, но число слоев крайне велико, поэтому преобразования итерационной структуры для таких задач лишены смысла. При этом в задачах математической физики рациональная реализация итерационной составляющей кадровой структуры является залогом эффективности решения

всей задачи. Кроме этого, в ряде случаев аппаратного ресурса РВС может быть недостаточно для реализации даже базового подграфа, что требует применения редукционных преобразований, описанных в [5], к базовому подграфу, но позволяет впоследствии применять методы и алгоритмы масштабирования к редуцированной кадровой структуре для сокращения времени решения задачи (рис. 5). Похожая стратегия используется и при решении задач цифровой обработки сигналов — в частности, при реализации алгоритма БПФ с переходом к м-кадрам для сокращения количества входных информационных каналов с последующим масштабированием полученного решения для эффективного использования аппаратного ресурса. Если необходима редукция функционального подграфа задачи, «Ариадна» рассчитывает требуемый коэффициент редукции и вызывает вспомогательный компонент — программу преобразования фрагментов кадровой структуры на языке COLAMO в микро-кадры «Синид».

Разрядность как параметр быстродействия кадровой структуры, используемый для редукции производительности, меняется только для достаточно узкого класса задач, поэтому в текущей версии программы «Ариадна» редукция по разрядности не выполняется. Таким образом, в качестве первого преобразования кадровой структуры в стратегии может использоваться изменение любого из пяти параметров параллелизма (L, It, Ch, Op, I), что и определяет различие стратегий. Оценкой сверху для общего количества возможных стратегий тогда является $5! = 120$, т.е. их число достаточно невелико даже для полного перебора современными средствами вычислительной техники. На практике, в силу ортогональности преобразований кадровой структуры [5], количество возможных стратегий существенно меньше и едва ли превышает 20. Сравнительно небольшое общее число возможных стратегий преобразования кадровых структур позволяет существенно сократить время анализа кадровых структур и время синтеза гарантированно рационального решения прикладной задачи: если при полном переборе всех стратегий не был достигнут заданный уровень производительности, будет выбрано лучшее из проанализированных решений.

5. Заключение

За счет автоматизации синтеза проектов задач с высокой вычислительной сложностью для многокристальных РВС комплекс «Тесей» преодолевает ключевые ограничения современных HLS-инструментов. Комплекс средств высокоуровневого синтеза «Тесей», при схожей с Xilinx Vivado HLS и Vitis функциональности, обладает следующими основными отличиями:

- преобразование осуществляется на основе анализа информационных зависимостей входной программы, а не директив ручной разметки кода, аналогичных #pragma в Vivado HLS;
- высокоуровневый синтез и масштабирование его результатов выполняется сокращением параметров параллелизма кадровой структуры, а не распараллеливанием последовательной программы;
- произвольный доступ к памяти в последовательной программе преобразуется к регулярным операциям чтения-записи для потоков данных;
- повышение эффективности синтезируемого решения с помощью различных способов организации вычислений (макроконвейер, конвейер в конвейере);

- автоматическая адаптация прикладной задачи к архитектуре и конфигурации заданной пользователем РВС;
- автоматическая синхронизация информационных и управляющих сигналов для всех используемых кристаллов ПЛИС;
- поддержка задач разных прикладных областей.

Преимущества используемого в «Тесее» подхода состоят в следующем:

- автоматизация сложных этапов выбора стратегий «Ариадной» и автоматизация синхронизации фрагментов решения между кристаллами синтезатором Fire!Constructor сокращает время синтеза многокристальных проектов.
- поддержка многокристальных систем позволяет масштабировать результаты синтеза и решать задачи, недоступные для классических HLS-инструментов.
- автоматическая адаптация прикладной задачи к архитектуре и конфигурации заданной пользователем РВС позволяет оперативно выполнять портацию исходной последовательной программы на различные РВС.

В разработанном комплексе средств высокоуровневого синтеза «Тесей», в отличие от известных HLS-компиляторов, входная программа на языке С преобразуется автоматически, без ручной разметки кода или иных указаний пользователя, а результатом является рациональное решение задачи для многокристальной РВС с автоматической синхронизацией информационных и управляющих сигналов. Направления дальнейших исследований связаны с расширением классов и диапазонов решаемых задач и сокращением ряда требований к исходному тексту программ на С, в частности, поддержки косвенной адресации во входной программе и операторов динамического выделения памяти для массивов.

Список литературы

[1] Nane R. et al. A Survey and Evaluation of FPGA High-Level Synthesis Tools / IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 35, no. 10, pp. 1591-1604, Oct. 2016. https://doi.org/10.1109/TCAD.2015.2513673

[2] Numan M.W., Phillips B.J., Puddy G.S., Falkner K. Towards Automatic High-Level Code Deployment on Reconfigurable Platforms: A Survey of High-Level Synthesis Tools and Toolchains / IEEE Access, vol. 8, pp. 174692-174722, 2020. https://doi.org/10.1109/ACCESS.2020.3024098.

[3] Каляев, И.А. Реконфигурируемые вычислительные системы на основе ПЛИС : монография / И.А. Каляев, И.И. Левин. — Ростов-на-Дону: Издательство ЮНЦ РАН, 2022. — 474 с.

[4] Программный комплекс высокоуровневого синтеза конфигурационных файлов для многокристальных реконфигурируемых вычислительных систем / А.И. Дордопуло, И.И. Левин, В.А. Гудков, А.А. Гуленок // Параллельные вычислительные технологии (ПаВТ'2023) : Короткие статьи и описания плакатов. Материалы XVII всероссийской научной конференции с международным участием, Санкт-Петербург, 28–30 марта 2023 года. — Челябинск: Издательский центр ЮУрГУ, 2023. — С. 133-142. — EDN ZJZMNJ.

[5] Дордопуло А.И., Левин И.И., Гудков В.А., Гуленок А.А. Программные средства высоко-уровневого синтеза для многокристальных реконфигурируемых вычислительных систем // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2022. Т. 11, № 3. С. 5–21. https://doi.org/10.14529/cmse220301

[6] Nane R., Sima V.-M., Olivier B., Meeuws R., Yankova Y., Bertels K. DWARV 2.0: A CoSy-based C-to-VHDL Hardware Compiler. In FPL, pp. 619-622. 2012. https://doi.org/10.1109/FPL.2012.6339221

[7] Pilato C. and Ferrandi F. Bambu: A Modular Framework for the High Level Synthesis of Memory-intensive Applications. In FPL, pp. 1–4, 2013. https://doi.org/10.1109/FPL.2013.6645550

[8] Canis A., Choi J., Aldham M., Zhang V., Kammoona A., Anderson J.H., Brown S., Czajkowski T. LegUp: High-Level Synthesis for FPGA-based Processor / Accelerator Systems. In ACM FPGA, pp. 33-36. 2011. https://doi.org/10.1145/1950413.1950423

[9] Make Slow Software Run Fast with Vivado HLS https://www.xilinx.com/publications/xcellonline/run-fast-with-Vivado-HLS.pdf, (дата обращения 23.03.2023)

[10] Vitis Unified Software Platform Documentation. Application Acceleration Development. https://www.xilinx.com/support/documentation/sw_manuals/xilinx2019_2/ug1393-vitis-application-acceleration.pdf, (дата обращения 23.03.2023).

[11] Kamkin A.S., Chupilko M.M., Lebedev M.S., Smolov S.A., Gaydadjiev G. Comparison of High-Level Synthesis and Hardware Construction Tools. Trudy ISP RAN / Proc. ISP RAS, vol. 34, issue 5, 2022, pp. 7-22 https://doi.org/10.15514/ISPRAS-2022-34(5)-1

[12] Yun L., Kyle R., Yinan L., Dongbo M., Minh D., Deming C. (2012). High-Level Synthesis: Productivity, Performance, and Software Constraints. Journal of Electrical and Computer Engineering. 2012. https://doi.org/10.1155/2012/649057.

[13] Zhou Y., Gupta U., Dai S., Zhao R., Srivastava N., Jin H., Featherston J., Lai Y.-H., Liu G., Velasquez G.A., Wang W., and Zhang Z. 2018. Rosetta: A Realistic High-Level Synthesis Benchmark Suite for Software Programmable FPGAs. In Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA'18). Association for Computing Machinery, New York, NY, USA, 269–278. https://doi.org/10.1145/3174243.3174255

Суперкомпьютерное образование на примере анализа геномных данных и моделирования управления пространственно-нерегулярными кустовыми нефтегазовыми скважинами

Губайдуллин И.М.

Уфимский государственный нефтяной технический университет
Институт нефтехимии и катализа УФИЦ РАН

В работе излагаются основные принципы обучения технологии параллельных вычислений на основе распределенного педагогического подхода на примерах решения крупных промышленно-важных задач. Раскрываются тонкости использования существующих высокопроизводительных систем, а также организация мобильных научных групп с участием как студентов и аспирантов УГНТУ, аспирантов и научных сотрудников ИНК УФИЦ РАН. В УГНТУ читаются лекции «Параллельные вычисления» у бакалавров и «Современные высокопроизводительные вычисления» у магистров [1].

Суперкомпьютерное образование на примере решения промышленно-важных задач ведется в тесном контакте магистрантов, аспирантов и преподавателей на базовой кафедре технологии нефти и газа УГНТУ в ИНК УФИЦ РАН. Параллельная обработка крупных задач происходит и на уровне между студентами УГНТУ и аспирантами ИНК УФИЦ РАН. Каждый кандидат (доцент) является руководителем минимум двух аспирантов, а каждый аспирант консультирует минимум одного магистранта. Общее руководство решения всей производственной задачей ведут доктора наук (профессора).

В Институте нефтехимии и катализа УФИЦ РАН, в рамках аспирантуры по специальности 1.2.2. Математическое моделирование, численные методы и комплексы программ выполняются диссертационные работы, где используются технологии параллельных вычислений. Тематика диссертационных работ охватывает практически все региональные задачи в Республике Башкортостан. В частности, параллельные вычисления оказываются особенно ценными при анализе геномных данных и работе с петлевой изотермической амплификацией (LAMP), где приходится обрабатывать миллиарды нуклеотидов и сопоставлять их с исходным геномом [2]. Благодаря делению задач на тысячи независимых потоков можно одновременно проверять сотни или даже тысячи потенциальных праймеров, вычислять их термодинамические параметры и выявлять желаемые участки. Такая масштабируемая модель позволяет не только многократно ускорить поиск и фильтрацию кандидатов, но и гибко настраивать критерии отбора без заметных потерь в производительности. В результате проекты, которые раньше занимали часы или даже дни на центральных процессорах, на платформе графических карт сводятся к минутам — это обеспечивает возможность оперативной адаптации дизайна праймеров к новым штаммам или анализу нескольких геномов одновременно.

Второй крупной практической базовой задачи в суперкомпьютерном образовании является двухблочная математическая модель, описывающую массоперенос жидкости в среде с двойной пористостью с расщеплением по физическим процессам [3]. Модель состоит из блока с системой гиперболических уравнений относительно водонасыщенности на фоне фиксированных скоростей фильтрации и блока относительно пьезопроводности для определения давления в среде с двойной пористостью. Непосредственное использование такой системы для целей определения динамики необходимых переменных и построения неявной разностной схемы, требуемой для расчетов параболических уравнений с крупными шагами по времени, затруднительно. В связи с этим для исходных уравнений применяется метод расщепления по физическим процессам, а именно происходит разделение уравнений на пьезопроводную часть и относительно переноса насыщенностей.

Проведя аппроксимацию частных производных соответствующими конечными разностями, получим систему линейных алгебраических уравнений, которая решается методом параллельной прогонки. Программная реализация будет на языке C с использованием стандарта MPI. Программный код позволит проводить вычисления как на персональном компьютере, так и на вычислительном кластере.

Работа выполнена в рамках реализации проекта Российского научного фонда № 25-71-20005 «Разработка мортарно-согласованной численной технологии решения многомасштабных задач совместной фильтрации несмешивающихся флюидов в трещиновато-пористых коллекторах пластов сложного геологического и литологического строения».

Список литературы

[1] Линд Ю.Б., Губайдуллин И.М., Спивак С.И. Математическое моделирование и решение производственных задач на основе параллельных вычислений // Учебное пособие. Изд. РИО БашГУ. Уфа, 2011.

[2] Ахметзянова Л.У., Давлеткулов Т.М., Гарафутдинов Р.Р., Губайдуллин И.М. Применение алгоритма AXO-корасик для подбора праймеров для петлевой изотермической амплификации // Математическая биология и биоинформатика. 2022. Т. 17. № 2. С. 250-265.

[3] Uzyanbaev R.M., Bobreneva Yu.O., Poveshchenko Yu.A., Podryga V.O., Polyakov S.V., Rahimly P.I., Gubaydullin I.M. Numerical modeling of two-phase fluid filtration for carbonate reservoir in two-dimensional formulation // Mathematics. 2024. Т. 12. № 21. С. 3412.

Интеллектуальная система поддержки пользователей научных проектов на основе LLM с верификацией достоверности

Р.Б. Парчиев, И.А. Антонов, М.К. Исаченко

НИТУ МИСИС, Москва

Международный проект добровольных распределенных вычислений RakeSearch решает задачу нахождения свойств диагональных латинских квадратов. Однако сама предметная область проекта является достаточно узкой и сложной для понимания для большинства участников проекта. В данной работе представлена вопросно-ответная система на основе архитектуры RAG, которая позволит участникам проекта (в том числе, потенциальным) оперативно получать информацию об исследовании, сформулированную простым для понимания языком, а также приводится сравнительный анализ моделей, использованных в качестве компонента генерации данной системы.

Ключевые слова: RAG, LLM, добровольные распределенные вычисления, машинное обучение, генеративные нейронные сети

1. Введение

Добровольные распределенные вычисления (ДРВ) — это использование множества географически распределенных потребительских цифровых устройств для высокопроизводительных научных вычислений. Владельцы устройств участвуют в ДРВ, устанавливая программу, которая загружает и выполняет задания с серверов, обслуживаемых научными проектами. В настоящее время существует более 30 проектов ДРВ во многих научных областях и во многих учреждениях, а результаты исследований, проводимых с помощью ДРВ, публикуются в многочисленных статьях в журналах Nature, Science, PNAS, Physical Review, Biology, Bioinformatics и других [1]. Стать участником проекта ДРВ может любой пользователь со своим устройством. Такие участники оказывают организаторам проектов большую помощь и ускоряют процесс получения результата, поэтому имеет смысл популяризировать участие в добровольных распределенных вычислениях.

Исследования показывают, что очень многие участники проектов ДРВ мотивированы желанием помогать науке и пониманием своей причастности к научным открытиям [2, 3]. Поэтому один из способов популяризации проектов ДРВ — повышение доступности информации о проекте. Необходимо облегчить потенциальным участникам поиск информации об исследовании, основных концепциях предметной области и цели исследования.

Проект RakeSearch [4] занимается изучением диагональных латинских квадратов. По данной теме существует ряд научных исследований, в том числе, от организаторов проекта [5, 6], но предметная область является достаточно узкой и сложной для понимания, а прямой контакт с организаторами для уточнения информации по решаемым

задачам не всегда возможен. Поэтому хорошим вариантом доведения до пользователя информации о научном проекте является построение вопросно-ответной системы на основе архитектуры RAG, располагающей всеми необходимыми сведениями об исследовании и способной донести их до потенциального участника в доступном формате.

RAG (Retrieval-Augmented Generation) — это гибридная архитектура, разработанная для устранения ограничений генеративных моделей. RAG состоит из двух ключевых компонентов: механизма поиска, который извлекает релевантные документы или информацию из внешнего источника знаний, и модуля генерации, который обрабатывает эту информацию для генерации текста, похожего на человеческий. Такое сочетание позволяет моделям RAG не только генерировать текст, но и основывать свои ответы на реальных актуальных данных.

В процессе данной работы была разработана интеллектуальная система поддержки пользователей и потенциальных участников научного проекта ДРВ RakeSearch для поиска на основе LLM с верификацией достоверности.

2. Использованные методы

Прежде всего, опишем подробнее архитектуру RAG, на основе которой была построена вопросно-ответная система.

Как уже было сказано RAG расширяет возможностей больших языковых моделей за счет интеграции дополнительных механизмов поиска информации и состоит из компонентов поиска (retriever) и генерации (generator). Рассмотрим подробнее роль каждого компонента в общей работе системы:

- компонент поиска (retriever) отвечает за поиск и извлечение наиболее подходящей информации из внешних источников; он анализирует запрос и находит фрагменты данных, которые могут быть полезны для точного ответа;
- компонент генерации (generator) использует найденную информацию для создания ответа.

Преимущество RAG заключается в способности динамически использовать внешние знания, что позволяет такой архитектуре превосходить генеративные модели, такие как GPT-3, и системы, основанные на знаниях, такие как BERT, которые опираются на статические наборы данных. Помимо извлечения знаний, модели RAG отлично справляются с обновлением баз знаний. Поскольку модель извлекает внешние документы для каждого запроса, ей не требуется переобучение для включения последней информации. Такая гибкость делает модели RAG особенно подходящими для областей, где информация постоянно меняется, таких как медицинские исследования, финансовые новости и судебные разбирательства. Более того, исследования показали, что модели RAG достигают превосходных результатов в различных задачах, требующих больших знаний, включая реферирование документов и диалоги, основанные на знаниях.

Можно представить RAG-систему в виде схемы (рис. 1) [7].

Процесс работы такой системы устроен следующим образом:

1) Обработка запроса: система получает на вход запрос от пользователя. Это может быть вопрос, промпт или любая другая форма ввода, на которую языковая модель должна ответить.

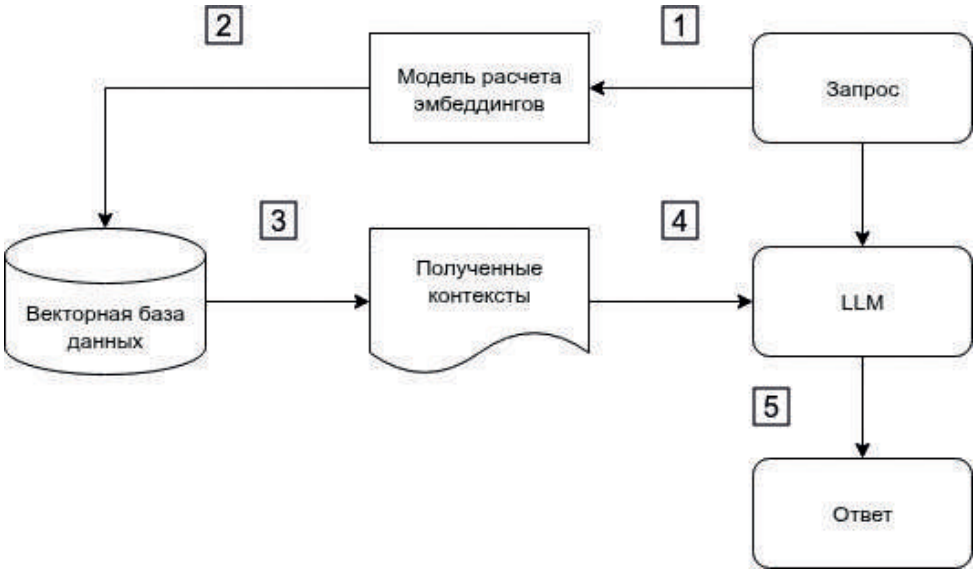


Рис. 1: Схема RAG-системы

2) Модель векторизации: запрос передается в модель векторизации, которая преобразует его в вектор — численное представление, которое система может обрабатывать и анализировать.
3) Поиск в векторной базе данных: вектор запроса используется для поиска в векторной базе данных. Эта база данных содержит заранее вычисленные векторы потенциальных контекстов, которые модель может использовать для генерации ответа. Система извлекает наиболее релевантные контексты, основываясь на том, насколько близки их векторы к вектору запроса. В качестве функции близости используется косинусное сходство.
4) Дополнение контекста: изначальный запрос дополняется контекстом из векторной базы данных, и вместе они передаются в большую языковую модель (LLM). Такой запрос содержит информацию, которая используется моделью для формирования более точного ответа.
5) Генерация ответа LLM: LLM учитывает как исходный запрос, так и дополнительный контекст для создания полного и релевантного ответа. Она синтезирует информацию из контекста, чтобы ответ основывался не только на предобученных знаниях, но и был дополнен конкретными деталями, полученными из извлеченных данных.
6) Итоговый ответ: в итоге LLM выводит ответ, который теперь дополнен внешними данными, что делает его более точным и детализированным.

Векторизация текста представляет собой преобразование данного текста в набор чисел (эмбеддинги). Происходит это с помощью модели, которая уже обучена на большом количестве текстов и способна рассчитывать эмбеддинги так, чтобы слова, часто упоминаемые вместе, имели векторы эмбеддингов, находящихся недалеко друг от друга в данном пространстве эмбеддингов.

Функция расстояния между векторами эмбеддингов в данной работе — косинусное сходство, рассчитываемое как:

$$S_C(A, B) = \frac{\langle A, B \rangle}{\|A\| \cdot \|B\|} = \frac{\sum_{i=1}^{n} A_i \cdot B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}}.$$

Для более грамотной подачи информации из документов в контекст языковой модели была реализована двухуровневая модель поиска. Она реализована следующим образом:

1) Используется векторный поиск для нахождения текстов, наиболее близких к запросу по смыслу.
2) Одновременно применяется алгоритм BM25 [8], чтобы учитывать точное совпадение ключевых слов.
3) Слияние результатов (Fusion): алгоритм объединяет результаты из двух поисков, чтобы получить наиболее релевантные части текста.
4) Дополнительная проверка релевантности: отобранные тексты проходят через модель классификации (cross-encoder), которая оценивает их по содержанию ответа на вопрос пользователя.

В результате топ-5 текстов с наивысшей релевантностью автоматически добавляются в запрос для языковой модели.

Важным элементом работы программы является проверка достоверности (фактчекинг). Она реализована следующим образом: прежде чем выдать пользователю ответ, модель повторно передает себе контекст и уточняет, насколько ответ является достоверным (отсутствие галлюцинаций модели и фактических ошибок). Наличие такой технологии очень важно, поскольку в работе с научными проектами и добровольными распределенными вычислениями необходимо, чтобы пользователь располагал достоверной информацией о том, на что будут потрачены его вычислительные ресурсы.

3. Исходные данные и методы их предобработки

Как уже было сказано, проект RakeSearch занимается исследованием свойств латинских квадратов, поэтому информация, которой располагает созданная RAG-система, взята из различных источников, посвященных данной задаче, в частности, опубликованных научных статей организаторов проекта или более общих трудов, содержащих основные понятия из области. Представим список используемых на данный момент материалов:

- Тужилин М.Э. Латинские квадраты и их применение в криптографии // Прикладная дискретная математика. — 2012. — № 3 (17). — С. 47–52.
- Фаруки Ш., Катре С.А., Гарг М. Псевдоортогональные латинские квадраты // Дискрет. матем. 2020. Том 32, вып. 3. С. 113–129. `https://doi.org/10.4213/dm1615`.
- Gallego Sagastume I. Generation of random latin squares step by step and graphically // XX Congreso Argentino de Ciencias de la Computación (Buenos Aires, 2014). — 2014.
- Ватутин Э.И. и др. Особенности использования взвешивающих эвристик в задаче поиска диагональных латинских квадратов // Известия Юго-Западного государственного университета. Серия: Управление, вычислительная техника, информатика. Медицинское приборостроение. — 2015. — № 3. — С. 18–30.
- Ватутин Э.И., Кочемазов С.Е., Заикин О.С. Алгоритмическая оптимизация программной реализации процедуры получения множества трансверсалей для латинских квадратов // Визуальная аналитика. — 2017. — С. 44–49.

- Ватутин Э.И. и др. Исследование свойств симметричных диагональных латинских квадратов. Работа над ошибками // Интеллектуальные и информационные системы (Интеллект–2017). Тула. — 2017. — С. 30–36.
- Ватутин Э.И. и др. О свойствах клик из диагональных латинских квадратов малой размерности на множестве бинарного отношения ортогональности // Интеллектуальные и информационные системы. Интеллект–2019. — 2019. — С. 17–23.
- Ватутин Э.И. и др. Исследование свойств обобщенных симметрий в диагональных латинских квадратах с использованием добровольных распределенных вычислений // Высокопроизводительные вычислительные системы и технологии. — 2019. — Т. 3. — № 2. — С. 39.
- Носов В.А., Панкратьев А.Е. О функциональном задании латинских квадратов // Интеллектуальные системы. — 2008. — Т. 12. — № 1–4. — С. 317–332.
- Вносов В. О построении классов латинских квадратов в булевой базе. [Электронный ресурс]. `http://intsys.msu.ru/staff/vnosov/latquadr.htm` (дата обращения: 05.01.2025).
- Малых А.Е. и др. Об историческом процессе развития теории латинских квадратов и некоторых их приложениях // Вестник Пермского университета. Серия: Математика. Механика. Информатика. — 2010. — № 4. — С. 95–104.
- Заикин О.С. и др. Применение высокопроизводительных вычислений для поиска троек взаимно частично ортогональных диагональных латинских квадратов порядка 10 // Вестник Южно-Уральского государственного университета. Серия: вычислительная математика и информатика. — 2016. — Т. 5. — № 3. — С. 54–68.
- Andersen L.D. Chapter on The history of latin squares. — 2007.
- Латинский квадрат. [Электронный ресурс] // Википедия: свободная энциклопедия. `https://ru.wikipedia.org/wiki/Латинский_квадрат` (дата обращения: 01.01.2025).
- Смирнов О. Латинский квадрат: вызываем демонов во имя математики. [Электронный ресурс] // Хабр. `https://habr.com/ru/companies/elbrusbootcamp/articles/593325/` (дата обращения: 02.01.2025).

Работа с данными для их использования в созданной RAG-системе выполнялась следующим образом:

Документы различных типов загружаются в программу (в данный момент в программе предусмотрена поддержка следующих типов: PDF, DOC(X), PPTX, XLSX, Markdown, TXT).

Текст каждого документа разбивается на чанки (кусочки меньшей длины) размером 462 символа с помощью метода рекурсивного разбиения по символам. Этот метод разбивает большой текст на более мелкие части рекурсивно, что означает, что если с первого раза разбить текст на куски нужного размера не получилось, он будет пытаться снова и снова, используя всё более мелкие разделители. Исходным разделителем является двойной перевод строки (разрыв между абзацами), дальше по приоритетности идут одиночный перевод строки, пробел и пустая строка (то есть, разделение идет по каждому символу).

Для каждого полученного чанка выполняется генерация описания с помощью LLM: на вход модели в промпте подается текст чанка, текст полного документа и указание для модели сгенерировать краткое объяснение значения данного фрагмента в контексте всего документа.

Результирующие документы для базы данных системы получаются путем конкатенации описания чанка и его текста для каждого чанка исходных документов.

Для полученных документов рассчитываются эмбеддинги и вносятся в векторную базу данных.

Важно сказать, что весомым преимуществом архитектуры RAG является возможность динамического обновления содержащейся в системе информации, то есть, в любой момент в нее можно добавить новые документы, и они сразу же будут учитываться во время работы системы.

4. Полученные результаты

Результатом данной работы является приложение с пользовательским интерфейсом, предоставляющее функционал вопросно-ответной системы поддержки пользователей научных проектов на основе LLM.

Для разработки был выбран язык программирования Python ввиду его простоты и привычности для авторов данной работы и инструмент Anaconda для удобного управления пакетами и виртуальным окружением.

В качестве средства разработки был выбран бесплатный редактор кода Microsoft Visual Studio Code, обладающий гибкими возможностями настройки и большим количеством расширений, приносящих новый функционал.

Перечислим основные технологии, использованные в данной работе:

1) Streamlit [9] — библиотека Python с открытым кодом для создания веб-приложений.
2) unstructured [10] — библиотека, предназначенная для предобработки документов различного формата (изображений, HTML, DOC, PDF и др.).
3) Langchain — фреймворк для работы с языковыми моделями.
4) FAISS [11] — библиотека, предоставляющая эффективные алгоритмы для быстрого поиска и кластеризации эмбеддингов.
5) Qwen2.5-32B-Instruct [12] — модель, использованная для генерации ответов.
6) intfloat/multilingual-e5-large-instruct [13] — модель, использованная для создания эмбеддингов.
7) DiTy/cross-encoder-russian-msmarco [14] — модель, использованная для ранжирования документов, близких к запросу пользователя.

В качестве данных для RAG-системы выступали файлы различных форматов (doc, pdf, pptx, rtf и др.), сами данные были не структурированы. Для работы с ними была использована библиотека unstructured. Это решение с открытым исходным кодом и подробной документацией, поддерживающая 26 расширений файлов. Эта библиотека позволяет извлекать таблицы и изображения с возможностью дополнительно их обрабатывать, а также может извлекать информацию из веб-страниц. Для большего удобства работы с ней она содержит готовые шаблоны и предоставляет возможность гибкой настройки.

Финальная архитектура решения представлена на рис. 2 [15]. Эта архитектура была представлена не так давно, и ее ключевой особенностью является гибридный поиск, о котором говорилось ранее. За счет него достигается сокращение количества неудачных поисков на 67% по сравнению с RAG-системами с обычным методом поиска по векторной базе данных. Также разработанная система обладает еще некоторыми преимуществами по сравнению с другими решениями, эти преимущества отражены в табл. 1.

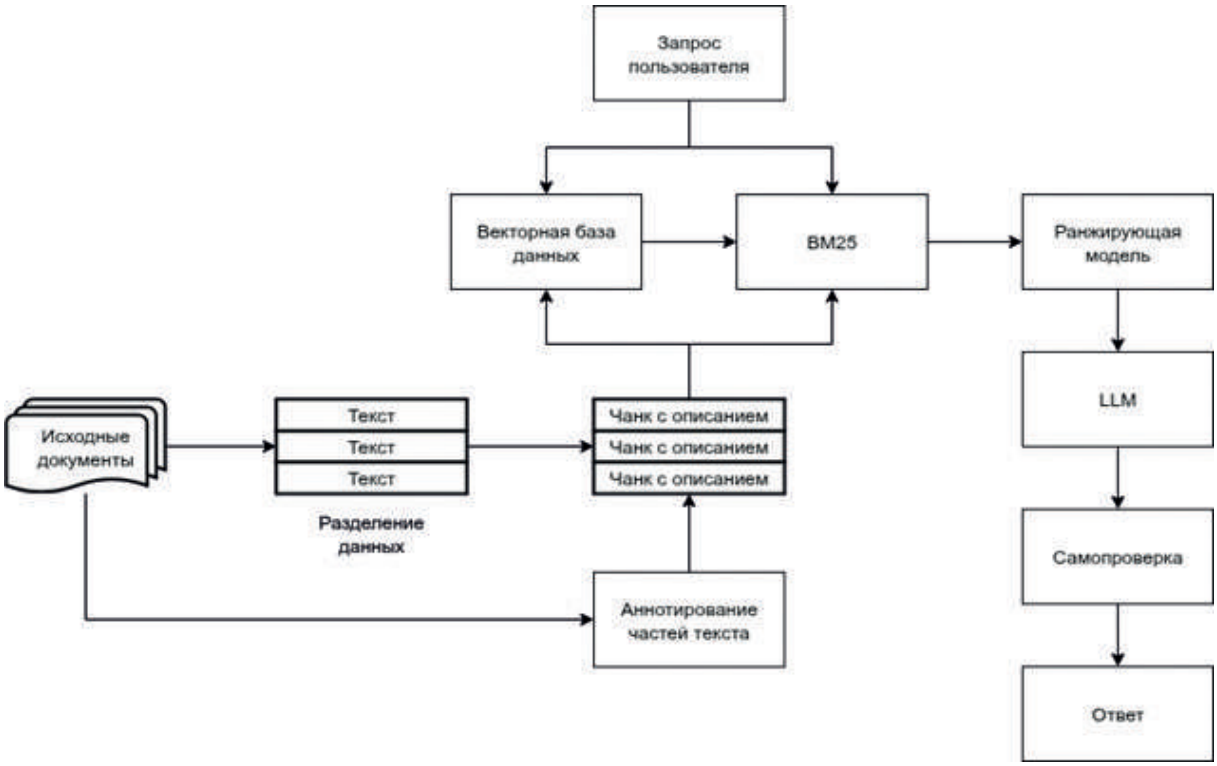


Рис. 2: Финальная архитектура приложения

Метрики разработанной RAG-системы с разными моделями в качестве ядра на бенчмарках представлены в табл. 2.

Таблица 1: Сравнение разработанной системы с аналогами

Критерий	**Haystack**	**Langchain**	**Данный сервис**
Поддержка 26 форматов документов	Нет	Есть	Есть
Поддержка таблиц	Есть	Нет	Есть
Гибридный поиск	Нет	Есть	Есть
Интерфейс для добавления данных	Нет	Нет	Есть
Модуль самопроверки на галлюцинации	Нет	Нет	Есть, исследование подходов
Поддержка изображений	Есть	Нет	Нет, но планируется

Таблица 2: Метрики на бенчмарках

	Qwen/Qwen2.5-Math-7B-Instruct	**Qwen/Qwen2.5-7B-Instruct**
Ru RAG Test Dataset	0,76	0,92
Custom Benchmark	0,91	0,82

Для генерации ответа было рассмотрено 2 модели из семейства Qwen, так как

они являются открытыми и бесплатными, а также очень хорошо работают с русским языком. Для выбора модели было использовано два бенчмарка и измерена метрика BERTScore [16] — эта метрика оценивает сходство между двумя текстами, измеряя контекстное сходство на уровне слов (или токенов), а не просто по прямым совпадениям, как это делает традиционный BLEU или ROUGE. Это достигается путем вычисления косинусного сходства между эмбеддингами слов, полученными с помощью BERT или другой трансформерной модели. Первый бенчмарк – это датасет на русском языке Ru RAG Test Dataset [17], он содержит в себе следующие элементы:

1) Файлы — страницы русской Википедии (в формате TXT).
2) Pandas-датафрейм (сохраненный в формате Pickle) со столбцами: вопрос пользователя, правильный ответ (ожидаемый ответ от RAG), контекст (часть текста, в которой содержится правильный ответ) и название файла, в котором содержится правильный ответ.
3) Метрика замеряется между правильным ответом и фактическим ответом модели.

Второй бенчмарк — это собственный датасет для оценки качества модели на конкретной предметной области, то есть, информации о латинских квадратах. Он построен аналогично вышеупомянутому Ru RAG Test Dataset, но в нем используются материалы по теме латинских квадратов и проекта RakeSearch, а не общие статьи из Википедии. Из полученных метрик мы видим, что модель Qwen/Qwen2.5-7B-Instruct показывает лучшие результаты, поэтому именно она была использована в финальном решении.

Рассмотрим теперь внешний вид разработанного приложения.

Запуская приложение, пользователь попадает на главную страницу приложения, на которой расположено окно чата и описание сервиса (рис. 3).

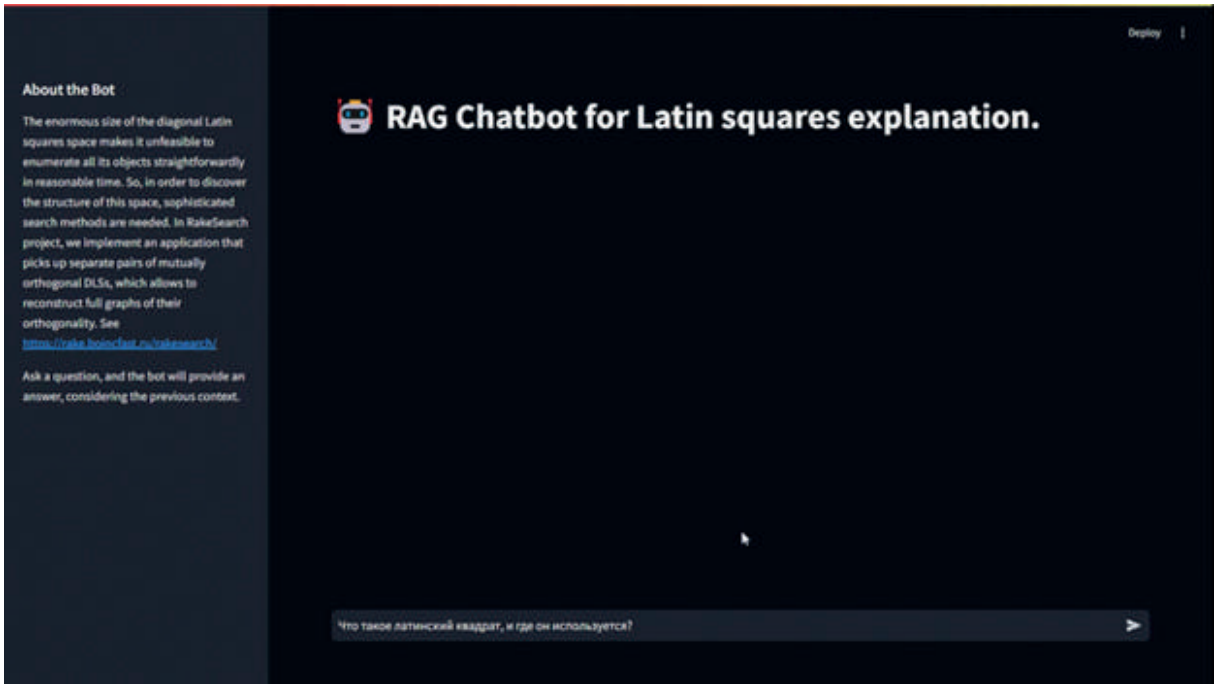


Рис. 3: Главное окно приложения

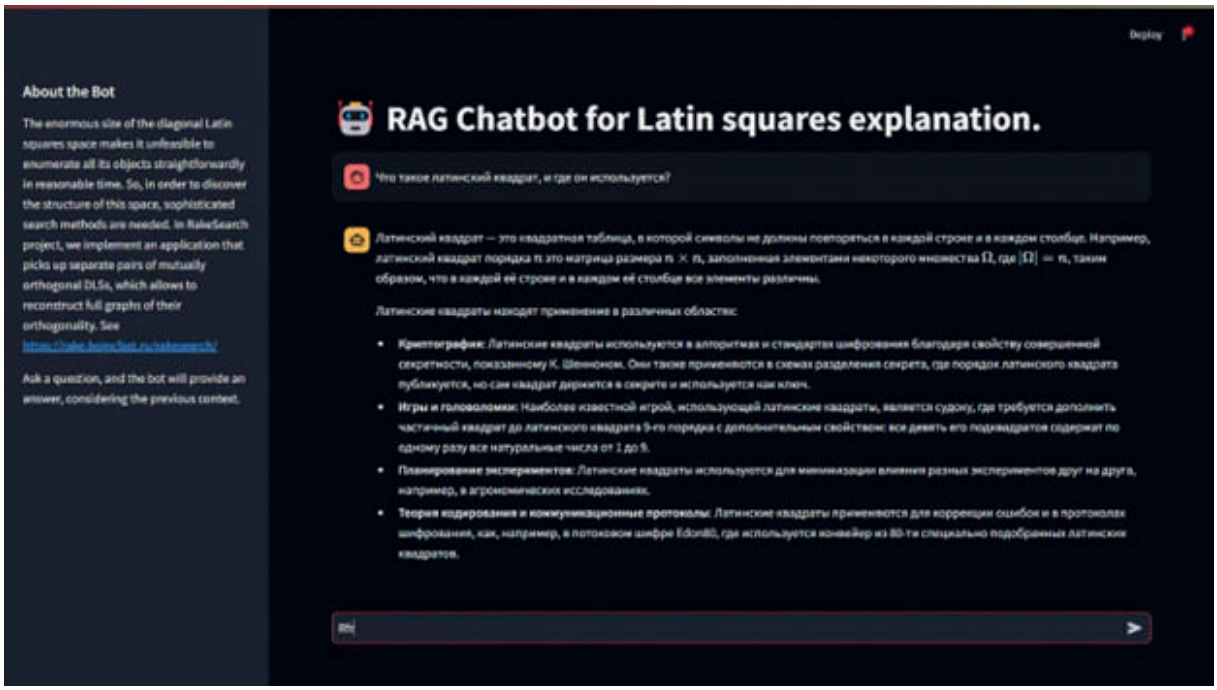


Рис. 4: Ответ системы на вопрос пользователя

При нажатии на кнопку отправки сообщения появляется значок загрузки, и сервис начинает генерировать ответ. По окончании генерации система выдает ответ (рис. 4).

После этого пользователь может по желанию сформулировать дополнительные вопросы системе и получить на них ответы.

Теперь приведем примеры вопросов различной сложности и отрывки (для компактности) ответов модели на них (табл. 3). Вопросы формулировались и задавались, в том числе, организаторами RakeSearch, которые являются экспертами в этой области исследований.

Таблица 3: Примеры ответов модели

Вопрос пользователя	**Отрывок ответа модели**	**Комментарий**
Что такое латинский квадрат, и где он используется?	Латинский квадрат — это таблица размером $n \times n$, заполненная n элементами множества M таким образом, что в каждой строке и в каждом столбце таблицы каждый элемент из M встречается в точности один раз. Например, латинский квадрат 3-го порядка может быть представлен следующим образом: ... Латинские квадраты широко используются в различных областях математики и кибернетики: ...	Легкий вопрос. Модель привела определение, формулы и области использования латинских квадратов.

Таблица 3 (продолжение)

Вопрос пользователя	**Отрывок ответа модели**	**Комментарий**
Какие особенности имеют латинские квадраты над абелевыми группами?	Латинские квадраты над абелевыми группами имеют несколько ключевых особенностей, которые описаны в предоставленных документах. Вот основные из них: 1. Задание латинского квадрата над абелевой группой: ... 2. Критерий правильности семейства функций: ... 3. Отсутствие циклов в графе существенной зависимости: ... 4. Простое достаточное условие построения правильных семейств: ...	Вопрос средней сложности. Модель перечисляет особенности латинских квадратов над абелевыми группами. Важно отметить, что модель успешно справляется с корректным отображением формул.
Что такое «правильное семейство функций» в контексте латинских квадратов?	В контексте латинских квадратов, «правильное семейство функций» определяется как набор функций, который удовлетворяет определенным условиям, обеспечивающим корректное построение латинского квадрата. Правильность семейства функций связана с отсутствием циклов в графе существенной зависимости и другими свойствами, которые гарантируют, что функции могут быть использованы для создания латинского квадрата. Вот ключевые моменты, извлеченные из документов: ...	Вопрос средней сложности. Модель нашла понятие правильного семейства функций и связала его с латинскими квадратами. Далее в ответе модель приводит 4 теоремы по теме и пример.
Как связаны латинские квадраты и абелевы группы?	Вот как эта связь описана в предоставленных документах: ... Таким образом, латинские квадраты и абелевы группы связаны через задание элементов квадрата с использованием функций, определенных на абелевых группах, и через критерии правильности и регулярности семейств функций.	Вопрос средней сложности. Модель нашла искомую связь, привела 4 варианта и подвела итог в конце ответа.
Как латинские квадраты применяются в научных исследованиях?	Латинские квадраты находят широкое применение в различных областях научных исследований. Вот несколько примеров их использования, основанных на предоставленных документах: 1. Планирование экспериментов: ... 2. Теория кодирования: ... 3. Защита информации:	Легкий вопрос. Модель приводит 5 примеров практического применения латинских квадратов.

Таблица 3 (окончание)

Вопрос пользователя	**Отрывок ответа модели**	**Комментарий**
В каких играх и головоломках используются латинские квадраты?	В документах упоминаются следующие игры и головоломки, в которых используются латинские квадраты: 1. Судоку: . . . 2. Футошики: . . . Задачи построения латинских квадратов: . . .	Легкий вопрос. Модель приводит 3 примера игр и головоломок.

Мы видим, что пользователь может задавать системе как вопросы по теоретической части предметной области, так и более практические вопросы, связанные с применением латинских квадратов, то есть, со смыслом самого исследования и значимости участия в нем. Другими словами, разработанная система может оказывать помощь и имеющим хорошую математическую базу пользователям, и энтузиастам, просто заинтересованным в помощи науке и практическом результате исследования. Виртуальная машина с развернутой на ней системой работала на протяжении 6 месяцев и была доступна пользователям и организаторам проекта RakeSearch. От организаторов была получена положительная обратная связь.

5. Заключение

В процессе работы было создано приложение с пользовательским интерфейсом, предоставляющее функционал вопросно-ответной системы на основе архитектуры RAG для международного проекта добровольных распределенных вычислений RakeSearch. Использование такой системы в проекте обосновано следующими преимуществами:

- повышение интереса участников через персонализированное взаимодействие и доступные объяснения научных целей проекта;
- мотивация участников благодаря понятным объяснениям сложных концепций;
- повышенная достоверность предоставляемых данных благодаря методам фактчекинга;
- конкурентоспособность системы благодаря гибридному поиску, поддержке множества форматов и модулю проверки на галлюцинации;
- возможность адаптирования системы для других научных проектов.

Результаты данной работы могут быть использованы не только в проекте RakeSearch, но и в любом другом научном проекте. Для этого достаточно заменить документы, используемые для формирования векторной базы данных, на подходящие для данного проекта ДРВ.

В качестве дальнейшей работы можно рассмотреть следующие направления:

- улучшение модуля самопроверки (например, с помощью алгоритмов Linear probing [18] и Local intrinsic dimensionality [19]);
- добавление поддержки работы с изображениями.

Список литературы

[1] Anderson D.P. BOINC: a platform for volunteer computing // Journal of Grid Computing. — 2020. — Т. 18. — № 1. — С. 99–122.

[2] Курочкин И.И. и др. Исследование способов привлечения добровольцев в проекты добровольных распределенных вычислений // Информационное общество: образование, наука, культура и технологии будущего. — 2022. — № 6. — С. 49–61.

[3] Kurochkin I.I. et al. Attraction of volunteers in projects of voluntary distributed computing // Problems of Information Technology. — 2019. — С. 60–69.

[4] RakeSearch // RakeSearch URL: https://rake.boincfast.ru/rakesearch/ (дата обращения: 01.02.2025).

[5] Manzyuk M., Nikitina N., Vatutin E. Start-up and the results of the volunteer computing project RakeSearch // Russian Supercomputing Days. — Cham : Springer International Publishing, 2019. — С. 725–734.

[6] Новиков А.О. и др. Исследование алгоритма Даббагяна-ву для построения нециклических пандиагональных латинских квадратов // Известия ЮФУ. Технические науки. — 2024. — № 3.

[7] Gupta S., Ranjan R., Singh S.N. A comprehensive survey of retrieval-augmented generation (rag): Evolution, current landscape and future directions // arXiv preprint arXiv:2410.12837. — 2024.

[8] Robertson S. et al. The probabilistic relevance framework: BM25 and beyond // Foundations and Trends in Information Retrieval. — 2009. — Т. 3. — № 4. — С. 333–389.

[9] A faster way to build and share data apps // Streamlit `https://streamlit.io/` (дата обращения: 01.02.2025).

[10] Unstructured // Langchain `https://python.langchain.com/docs/integrations/providers/unstructured/` (дата обращения: 19.12.2024).

[11] facebookresearch/faiss: A library for efficient similarity search and clustering of dense vectors // GitHub `https://github.com/facebookresearch/faiss` (дата обращения: 01.02.2025).

[12] Team Q. Qwen2 technical report // arXiv preprint arXiv:2407.10671. — 2024.

[13] Wang L. et al. Multilingual e5 text embeddings: A technical report // arXiv preprint arXiv:2402.05672. — 2024.

[14] DiTy/cross-encoder-russian-msmarco // Hugging Face `https://huggingface.co/DiTy/cross-encoder-russian-msmarco` (дата обращения: 02.01.2025).

[15] Introducing Contextual Retrieval // Anthropic `https://www.anthropic.com/news/contextual-retrieval` (дата обращения: 19.12.2024).

[16] Zhang T. et al. Bertscore: Evaluating text generation with bert // arXiv preprint arXiv:1904.09675. — 2019.

[17] slivka83/ru_rag_test_dataset // GitHub `https://github.com/slivka83/ru_rag_test_dataset` (дата обращения: 01.02.2025).

[18] CH-Wang S. et al. Do Androids Know They're Only Dreaming of Electric Sheep? //arXiv preprint arXiv:2312.17249. — 2023.

[19] Yin F., Srinivasa J., Chang K.W. Characterizing truthfulness in large language model generations with local intrinsic dimension // arXiv preprint arXiv:2402.18048. — 2024.

Аннотации стендовых докладов

High Resolution Road Traffic Simulation

D.S. Trusov, R.A. Rodriges Zalipynis

HSE University

Introduction. With each passing year, road transport systems continuously expand and structurally complicate, which leads to an increase in operational costs, complicates the timely detection and localization of pavement defects, complicates the analysis of congestion and accident cluster formation, hinders the optimization of traffic flows taking into account peak loads, affects the planning of interchange upgrades and the construction of new highways considering changing loads, and sets additional tasks for ensuring comprehensive traffic safety. To effectively solve these problems, high-resolution traffic modeling is proposed, providing detailed reproduction of the behavior of each vehicle, as well as detailed representation of the road segments themselves.

Existing Approaches and Proposed Solutions. In the work by Sigurdur F. Hafstein and co-authors, a high-resolution traffic model based on cellular automata is presented, adapted to the traffic rules on the freeways of North Rhine-Westphalia (Germany). The system integrates data from over 4,000 inductive loop detectors to simulate traffic flows with high resolution. The implementation includes a microscopic simulator and a macroscopic graphical interface for visualizing the current traffic state. The model efficiently processes incoming data and enables real-time monitoring of road conditions, as implemented on the portal `verkehr.nrw` [3].

In the work by Daniel Zehe and co-authors, a hybrid traffic simulation model was proposed that combines a highly detailed agent-based microscopic simulation with a low-resolution macroscopic flow model for specific road segments. This multi-resolution model improves simulation computational efficiency by 20% while maintaining accuracy within a 5% deviation compared to a purely microscopic model. The main challenge lies in ensuring data consistency at the boundaries between models with different levels of detail [2].

Data Sources. The most prevalent means of acquiring traffic information remains inductive loop detectors, which record events with high temporal accuracy and support the derivation of flow speed, density, occupancy and other aggregate indicators. Nevertheless, these detectors do not capture many salient characteristics of the road environment. To incorporate factors associated with pavement deterioration, temporary restrictions, accidents, maintenance activities and weather conditions, it is advisable to exploit high-resolution satellite imagery and video streams from publicly available surveillance cameras. These alternatives enhance modelling accuracy and obviate the need for detector deployment along every segment of the transport network [1, 4].

Data Storage and Visualization. Data from the aforementioned sources will be carefully systematized and thoroughly integrated into geospatial objects that form raster arrays with multiple layers of parameters, collectively reflecting the detailed state of the road network. This approach supports representations at a high spatial resolution, thereby enabling a faithful and precise reproduction of the complex structure and dynamic behavior of the transport system. For purposes of analysis and informed decision-making, the data will be visualized within GIS applications (e.g., QGIS [8]), allowing users to interactively select, monitor, and manage various layers such as traffic load, accident sites, maintenance zones, and pavement defects in real time [5, 8].

Modeling and Future Plans. An attractive idea is to run core simulations based on traffic simulation software, e.g. SUMO (Simulation of Urban Mobility) [11] or Treiber's Microsimulation of Road Traffic [12], where vehicle movement rules are predefined in the model. The output of these simulations, together with real detector and probe-vehicle data, can be used to train neural networks. The latter will be applied to predict short-term traffic flows, detect incidents and congestion, suggest optimal signal timings, recommend dynamic speed limits, plan efficient routes for public and emergency vehicles, and evaluate "what-if" scenarios quickly. Over time, continuous learning on live data will help improve both the simulation rules and the neural-network models for better traffic management.

Role of Supercomputing. High-resolution traffic modelling demands substantial computational resources. Acceleration is typically achieved by employing powerful GPU coprocessors, distributed computing systems, and advanced multithreaded architectures. Parallel processing enables the simultaneous handling of voluminous datasets and computationally intensive tasks, which is critical for effectively reducing response latency. A stringent requirement in such systems is the minimization of computational delays to ensure the timely delivery of results and to support prompt operational control [9].

Conclusions and Practical Applications. The results of traffic modeling will significantly improve the quality of transport infrastructure management by enabling timely identification of problem areas and forecasting traffic flow dynamics. The application of this model contributes to route optimization, congestion reduction, and accident decrease. Practically, it can be implemented in intelligent transportation systems for adaptive traffic signal control, planning road maintenance with minimal impact on traffic, as well as in navigation and road condition monitoring services. Moreover, the model serves as a foundation for developing measures to enhance the safety and environmental sustainability of transport systems.

References

[1] Yao Y., Xu J., Chen H. Road traffic monitoring from satellite images: a review // IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing. 2022.

[2] Zehe D., Grotzky D., Aydt H., Cai W., Knoll A. Traffic Simulation Performance Optimization through Multi-Resolution Modeling of Road Segments // Proceedings of the 2021 International Conference on Management of Data, 2021.

[3] Hafstein S.F., Chrobok R., Pottmeier A., Schreckenberg M., Mazur F.C. A High-Resolution Cellular Automata Traffic Simulation Model with Application in a Freeway Traffic Information System // Physics of Transport and Traffic, University Duisburg-Essen, 2010.

[4] Rodriges Zalipynis R.A. Convergence of Array DBMS and Cellular Automata: A Road Traffic Simulation Case // in Proceedings of the 2021 International Conference on Management of Data, Xi'an, China, 20–25 June 2021; pp. 2399–2403.

[5] Rodriges Zalipynis R.A. SimDB in Action: Road Trafic Simulations Completely Inside Array DBMS // Proc. VLDB Endow. 2022, 15, 3742–3745.

[6] Krajzewicz D., Erdmann J., Behrisch M. Recent development and applications of SUMO // International Journal on Advances in Systems and Measurements. 2012. Vol. 5, No. 3–4.

[7] Copernicus Open Access Hub. `https://scihub.copernicus.eu/`

[8] QGIS: A Free and Open Source Geographic Information System. `https://www.qgis.org/`

[9] Rodriges Zalipynis R.A. ChronosDB in Action: Manage, Process, and Visualize Big Geospatial Arrays in the Cloud // in Proceedings of the 2019 International Conference on Management of Data, Amsterdam, The Netherlands, 30 June–5 July 2019; pp. 1985–1988.

[10] Geotrellis: A geographic data processing engine for high performance applications. `https://geotrellis.io/`

[11] Behrisch M., Bieker L., Erdmann J., Krajzewicz D. SUMO – Simulation of Urban Mobility: An Overview // Proc. 2011 IEEE Intelligent Transportation Systems Conference, 2011.

[12] Treiber M., Kesting A. Microsimulation of Road Traffic // Springer Traffic Flow Dynamics, Springer, 2013.

Алгоритм распределения нагрузки на сетке с использованием заполняющих пространство кривых

С.А. Комаров[1], Г.С. Гойман[2,1]

Московский физико-технический институт[1],
Институт вычислительной математики им. Г.И. Марчука РАН[2]

Численный прогноз погоды и моделирование климата относятся к числу наиболее ресурсоемких вычислительных задач, требующих эффективного использования массивно-параллельных систем. Ключевым аспектом реализации численных алгоритмов в таких условиях становится распределение вычислительной нагрузки между процессами. Алгоритм разбиения должен минимизировать количество и объем межпроцессных коммуникаций, а также обеспечивать равномерную загрузку вычислительных узлов. Это особенно важно в случаях, когда различные точки расчетной области имеют неодинаковый вес, а система обладает гетерогенной архитектурой с узлами разной производительности.

В Институте вычислительной математики РАН им. Г.И. Марчука совместно с Гидрометцентром России ведется разработка новой модели атмосферы высокого разрешения [1]. Предполагается, что эта модель будет применяться для численного прогноза погоды с разрешением 3–5 км, что потребует выполнения расчетов с использованием порядка 10^5 CPU-ядер и/или систем с GPU-ускорителями.

Модель использует многоблочную логически-прямоугольную сетку типа кубическая сфера [2]. В текущей реализации применяется геометрический подход к распределению нагрузки, при котором шесть панелей сетки разбиваются на $N_p = 6 \cdot p \cdot q$ непересекающихся прямоугольников, где N_p — количество задействованных процессоров [1]. Однако этот метод имеет ряд ограничений: качество разбиения существенно зависит от величины N_p; алгоритм сложно адаптировать к случаю неравномерного веса расчетных точек или гетерогенной вычислительной системы; планируемое введение локальных областей с повышенным разрешением (сгущение сетки) дополнительно усложнит применение геометрического подхода.

В данной работе рассматривается альтернативный подход, основанный на применении заполняющих пространство кривых (Space-Filling Curves, ЗПК), в частности, кривых Гильберта и их вариациях. Заполняющие пространство кривые — это фрактальные кривые, которые задают биекцию между единичным отрезком и единичным квадратом (гиперкубом). Их дискретная аппроксимация позволяет ввести одномерную индексацию многомерных данных с сохранением свойства локальности: близкие по одномерному индексу точки остаются близкими в исходном пространстве (среднее отношение между одномерным расстоянием и двумерным (евклидовым) оказывается логарифмическим от N, где N — количество точек). Такой подход позволяет свести многомерную задачу распределения вычислительной нагрузки к одномерной, что упрощает балансировку нагрузки для случаев с неравномерными весами точек и гетерогенными вычислительными системами [3, 4].

Предложенный алгоритм включает в себя следующие основные этапы:

1) построение ЗПК верхнего уровня для задания обхода блоков многоблочной сетки;
2) независимое построение ЗПК для индексации узлов внутри каждого блока;
3) объединение индексов отдельных кривых с использованием кривой верхнего уровня и решение одномерной задачи распределения нагрузки.

Особенностью реализованного алгоритма является возможность применения многоблочных сеток с панелями различного разрешения, возможность применения балансировки нагрузки в случае различных весов расчетных сеток / различной производительности процессоров. Алгоритм не использует узкие особенности глобальной модели атмосферы, за счёт чего его возможно применять в различных сценариях, в том числе для гетерогенных вычислительных систем. Будут представлены результаты применения реализованного подхода для распределения нагрузки при использовании различных конфигураций сеток, анализ различных метрик качества разбиения предложенного алгоритма, а также сравнение с реализованным ранее геометрическим подходом.

Список литературы

[1] Shashkin V.V., Goyman G.S., and Tretyak I.D. Development of the next-generation atmosphere dynamics model in Russia: Current state and prospects // Lobachevskii Journal of Mathematics, 45(7):3159–3172, 2024.

[2] Rancic M., Purser R.J., and Mesinger F. A global shallow-water model using an expanded spherical cube: Gnomonic versus conformal coordinates // Quarterly Journal of the Royal Meteorological Society, 122(532):959–982, 1996.

[3] Pınar A., Aykanat C. Fast optimal load balancing algorithms for 1D partitioning // Journal of Parallel and Distributed Computing. – 2004. – Т. 64. – № 8. – С. 974-996.

[4] Pınar A., Tabak E.K., Aykanat C. One-dimensional partitioning for heterogeneous systems: Theory and practice // Journal of Parallel and Distributed Computing. – 2008. – Т. 68. – № 11. – С. 1473-1486.

Анализ производительности вычислительных устройств для задач квантовой химии

И.Э. Недомолкин[1], М.П. Коников[1], В.В. Стегайлов[2,1], А.В. Тимофеев[2,1], И.Д. Федоров[2,1]

[1]Национальный исследовательский университет «Высшая школа экономики»,
[2]Объединенный институт высоких температур РАН

Суперкомпьютеры важны для науки и инженерии, их применение растёт. Современные способы оценки производительности универсальны и базируются на общих математических задачах. Однако для большей точности лучше использовать научные пакеты при оценке производительности в специфических задачах, например, в квантовой химии.

Существует множество авторитетных рейтингов, основанных на различных бенчмарках, таких как NAS parallel benchmark [1], SPEC benchmark [2], HPC Challenge benchmark [3] и LINPACK [4].

На основе анализа статей и обсуждения с пользователями программ были выбраны программные пакеты Quantum Espresso [5][6] и CP2K [7][8]. Они являются достаточно популярными в научной сфере, распространяются бесплатно и имеют открытый исходный код.

Был проведен анализ производительности узлов суперкомпьютера “cHARISMa”, а также данных из открытых источников. Рис.1 и рис.2 демонстрируют производительность узлов суперкомпьютера при разном числе mpi потоков. Узел типа D наиболее производителен среди других узлов при решении квантово-химических задач. Нормировка на цену и потребляемую энергию пказала, что узел типа C оказался наиболее экономически эффективным.

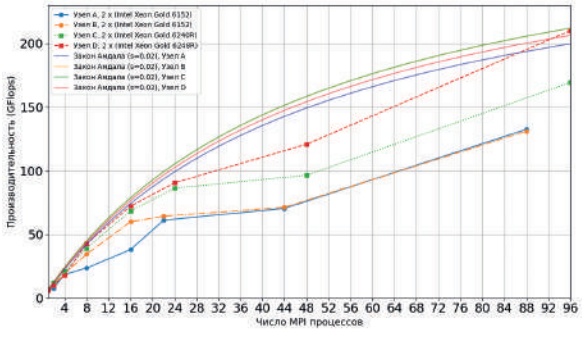


Рис. 1: Производительность в GFLops (CP2K)

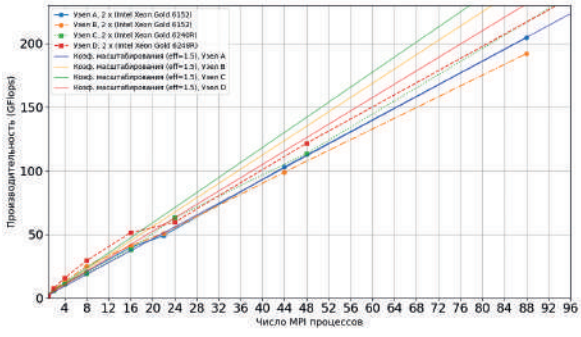


Рис. 2: Производительность в GFLops (QE)

На графиках рис.3 и рис.4 исследуется зависимость теоретической пиковой производительности от величины Баланса. Делается вывод, что при низком балансе (высокая пропускная способность относительно производительности) достигается лучшая эффективность. При высоком балансе нагрузка ограничивается памятью.

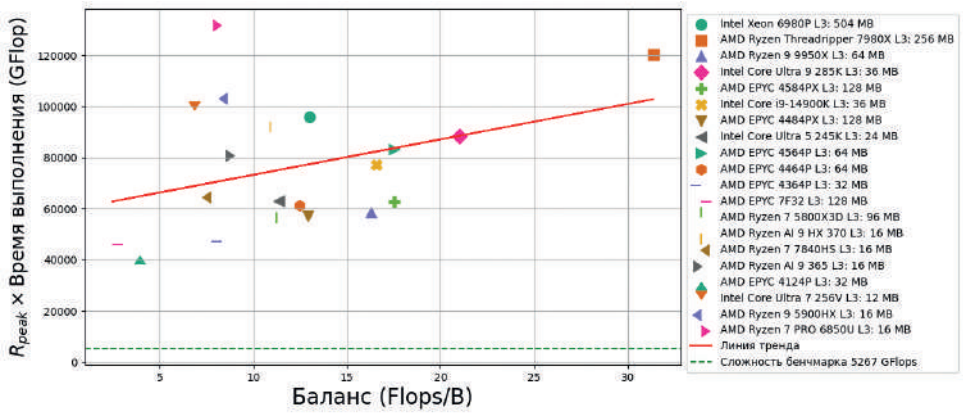


Рис. 3: Зависимость времени выполнения от приведённых параметров R_{peak} и Balance для CP2K

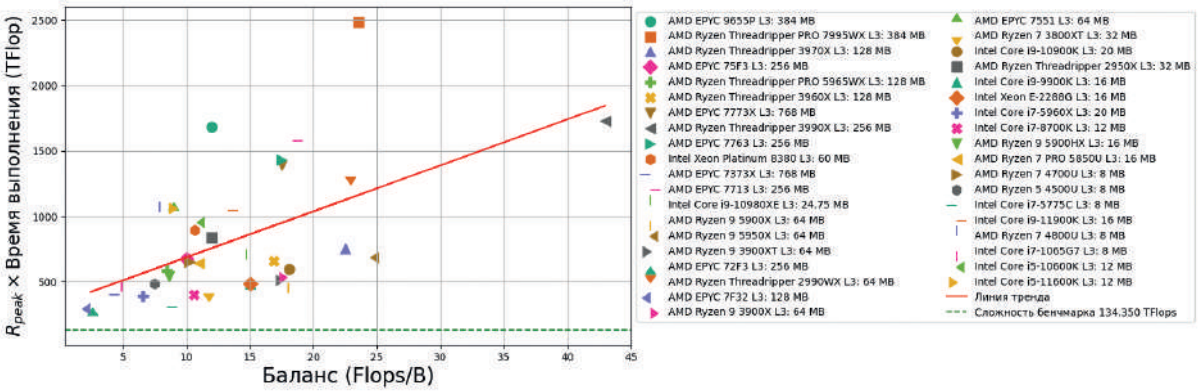


Рис. 4: Зависимость времени выполнения от приведённых параметров R_{peak} и Balance для QE

В будущем планируется произвести анализ узких мест в работе суперкомпьютера. Также уже проводится анализ программных пакетов с использованием профилировщиков для нахождения самых трудозатратных функций при работе пакета.

Благодарности: Создание и оптимизация бенчмарков, а также анализ результатов выполнены при финансовой поддержке Российского научного фонда (проект № 20-71-10127), `https://rscf.ru/project/20-71-10127/`. Тестирование узлов суперкомпьютера осуществлено в рамках Программы фундаментальных исследований НИУ ВШЭ с использованием суперкомпьютерного комплекса НИУ ВШЭ [9].

Список литературы

[1] Bailey D.H. et al. The NAS parallel benchmarks-summary and preliminary results // Proceedings of the 1991 ACM/IEEE Conference on Supercomputing, 1991. P. 158–165. (the NAS parallel benchmark suites)

[2] SPEC: Standard performance evaluation corporation. `http://www.spec.org`, 2005. (the SPEC benchmark)

[3] Luszczek P. et al. Introduction to the HPC challenge benchmark suite. – 2005. (the HPC Challenge benchmark)

[4] Dongarra J.J., Luszczek P., Petitet A. The LINPACK benchmark: past, present and future // Concurrency and Computation: practice and experience. – 2003. – Vol. 15. – No. 9. – P. 803-820. (LINPACK)

[5] Quantum ESPRESSO site, `https://www.quantum-espresso.org`

[6] User's Guide for QUANTUM ESPRESSO (v.7.2), `https://www.quantum-espresso.org/Doc/user_guide/`

[7] CP2K, `https://www.cp2k.org/`

[8] Thomas D.K. CP2K: An electronic structure and molecular dynamics software package Quickstep: Efficient and accurate electronic structure calculations // The Journal of Chemical Physics. 152.19 (2020), P. 194103.

[9] Kostenetskiy P.S., Chulkevich R.A., Kozyrev V.I. HPC Resources of the Higher School of Economics // Journal of Physics: Conference Series. 2021. Vol. 1740, No. 1. P. 012050. `https://doi.org/10.1088/1742-6596/1740/1/012050`.

Гибридные квантово-классические нейронные сети для анализа изображений магнитных плиток на наличие дефектов[1]

А.Д. Ивлев[1], А.В. Линев[1], М.В. Бастракова[1,2]

Нижегородский государственный университет им. Н.И. Лобачевского[1], Российский квантовый центр[2]

Введение. Задача классификации представляет интерес для различных приложений. В данной работе на примере прикладной задачи анализа дефектов магнитных плиток [1] мы предлагаем использовать подход квантового машинного обучения на основе гибридных квантово-классических нейронных сетей [2] для уточнения решения задачи классификации.

Описание структуры гибридной квантовой нейронной сети. В рамках нашей задачи рассматриваются два подхода: бинарная классификация, которая определяет наличие дефекта, и многоклассовая, которая при обнаружении дефекта определяет его тип из пяти приведённых.

Чёрно-белые изображения плиток приводятся к размеру 256×256. После чего они подаются на вход предобученной классической свёрточной нейронной сети. Результат работы сети представляет собой вероятности принадлежности изображения классам. Для получения более точного результата мы используем гибридную квантово-классическую нейронную сеть, полученную из оригинальной добавлением одного квантового слоя и одного линейного для постобработки выхода. Также будет произведено сравнение с моделью, где квантовый слой заменён линейным.

Таблица 1: Типы и параметры двух последних слоёв нейронных сетей

	Классическая сеть			Гибридная сеть		
	Тип слоя	Размерность выхода	Количество параметров	Тип слоя	Размерность выхода	Количество параметров
Бинарная	Линейный	4	12	Квантовый	4	12
	Линейный	2	10	Линейный	2	10
Многоклассовая	Линейный	8	56	Квантовый	8	30
	Линейный	6	54	Линейный	6	54

[1]Исследование выполнено при поддержке программы развития региональных научно-образовательных математических центров, соглашение № 075-02-2025-1727 с дополнительным соглашением № 075-02-2025-1727/1

Классические данные загружаются в квантовую часть угловым кодированием с помощью гейта Ry с множителем π, поэтому количество кубитов схемы соответствует количеству классов. Квантовый анзац представляет собой применение параметризованных гейтов Ry ко всем кубитам с последующим запутыванием с помощью гейтов CNOT.

Для бинарной классификации количество применений анзаца было выбрано равным шести, а для многоклассовой была предложена и построена более сложная свёрточная структура (см. Рисунок 1). После загрузки данных применяются три анзаца на все кубиты с циклическим запутыванием. Затем происходит свертка до трёх кубитов, к которым применяется ещё четыре анзаца с полным запутыванием. Соответственно, измеряется состояние не всей системы, а только трёх кубитов, и размерность выхода квантового слоя равна восьми. Данная свёрточная структура позволяет уменьшить количество параметров квантового слоя, что ускорит обучение.

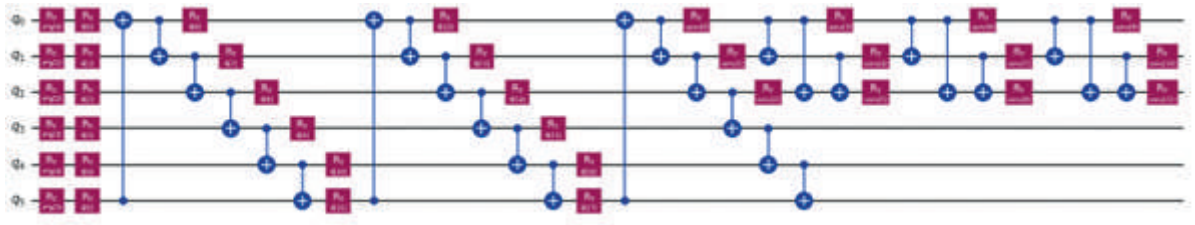


Рис. 1: Квантовая схема слоя для многоклассовой классификации

Методика проведения экспериментов. Программная реализация основана на библиотеке глубокого обучения Pytorch, библиотеке квантовых вычислений Qiskit и её расширения qiskit_machine_learning. Квантовая схема запускалась на симуляторе идеального квантового компьютера.

Тренировочная выборка 80% (1075), тестовая 20% (255). Оптимизатор: torch.optim.Adam. Функция потерь: torch.nn.CrossEntropyLoss. Количество эпох: 150 для бинарной и 75 для многоклассовой классификаций. Learning rate: $0.01 \cdot 2^{-i/15}$ и $0.1 \cdot 2^{-i/25}$, где i — номер эпохи, а деление целочисленное.

Результаты экспериментов. Метрики: точность и среднее значение функции потерь. В таблицах ниже в скобках указано количество изображений, принадлежащих классу в выборке.

Для бинарной классификации гибридная модель показала результат лучше, чем классическая при одинаковом количестве параметров. Точность обнаружения дефектов гибридной моделью больше на 10.811 процентов.

Таблица 2: Результаты бинарной классификации на тестовой выборке

	Классическая сеть		Гибридная сеть	
	loss	accuracy	loss	accuracy
Общие (255)	0.54243	0.92157	0.51279	0.95686
Defect (74)	0.57719	0.81081	0.53390	0.91892
Free (181)	0.52821	0.96685	0.50416	0.97238

Таблица 3: Результаты многоклассовой классификации на тестовой выборке

	Классическая сеть		Гибридная сеть	
	loss	accuracy	loss	accuracy
Общие (255)	1.21822	0.83922	1.2505	0.84314
Blowhole (22)	1.62685	0.45455	1.60642	0.40909
Break (16)	1.73914	0.3125	1.84287	0.3125
Crack (10)	1.14135	0.9	1.20086	0.9
Fray (6)	1.25636	0.66667	1.42245	0.66667
Uneven (20)	1.35088	0.7	1.40204	0.7
Free (181)	1.11083	0.95028	1.13518	0.96133

Для многоклассовой классификации гибридная модель показала результат сопоставимый с классической при меньшем количестве параметров. Небольшое улучшение достигнуто за счёт лучшего обнаружения плиток без дефектов, но упала точность обнаружения класса Blowhole.

Результаты показывают, что результаты работы квантово-классического решения лучше или сопоставимы классической модели при меньшем количестве обучаемых параметров. Также было обнаружено, что гибридные модели более устойчивы к переобучению. Полученные, при дополнительном сокращении параметров квантового слоя за счёт его свёрточной структуры, результаты схожи с результатами, представленными в статье [2]. На их основе будет продолжаться исследование свойств гибридных квантово-классических нейронных сетей и их применимость для прикладных задач в эпоху NISQ.

Список литературы

[1] Huang Y., Qiu C., Yuan K. Surface defect saliency of magnetic tile // The Visual Computer. 2020, Vol. 36, No. 1, P. 85-96, `https://doi.org/10.1007/s00371-018-1588-5`.

[2] Senokosov, A., Sedykh, A., Sagingalieva, A., Kyriacou, B., Melnikov, A. Quantum machine learning for image classification // Machine Learning: Science and Technology. – 2024. – Vol. 5. – No. 1. – P. 015040, `https://doi.org/10.1088/2632-2153/ad2aef`.

Исследование масштабируемости библиотеки XAMG при использовании GPU для решения систем уравнений

Р.М. Куприй[1,2], Б.И. Краснопольский[1], К.А. Жуков[2]

[1]Институт механики МГУ имени М.В. Ломоносова,
[2]Факультет ВМК МГУ имени М.В. Ломоносова

При численном моделировании различных физических процессов и явлений часто возникает необходимость нахождения решения систем линейных алгебраических уравнений (СЛАУ). Такая трудоёмкая задача может занимать вплоть до 95% от всего времени моделирования, поэтому особенно важна её эффективная реализация. Для решения могут применяться различные итерационные численные методы [1], где основная вычислительная нагрузка приходится на операцию умножения разреженной матрицы на вектор (SpMV). Производительность этой операции и самих методов ограничивается пропускной способностью шины памяти вычислительной системы [2].

Одним из перспективных подходов к ускорению таких вычислений является использование графических ускорителей (GPU), которые обладают значительно более высокой пропускной способностью памяти по сравнению с CPU. Однако эффективное применение GPU требует адаптации используемых алгоритмов, включая выбор формата хранения разреженных матриц и матрично-векторное умножение.

Выбор оптимального формата хранения особенно важен, поскольку он влияет на реализацию и эффективность алгоритма SpMV. Значительное количество форматов уже описаны в литературе, однако не все из них подходят для эффективной организации вычислений на GPU. В ходе работы был проведен сравнительный анализ различных форматов хранения разреженных матриц (CSR, Adaptive CSR, ELL, ELLR и Sliced ELL) с точки зрения их эффективности на GPU. Показано, что формат Adaptive CSR [3] является наиболее оптимальным для широкого класса матриц, обеспечивая в среднем на 5–10% лучшее время выполнения SpMV по сравнению с ELL-подобными форматами и сопоставимое время выполнения в сравнии с форматом CSR. Для матриц с ярко выраженным дисбалансом по числу ненулевых элементов в строках, Adaptive CSR демонстрирует ускорение в несколько раз в сравнении с другими форматами за счёт балансировки загрузки вычислительных ядер GPU.

Выбранные форматы и соответствующие вариации алгоритма SpMV были реализованы в разрабатываемой библиотеке численных методов XAMG [4, 5]. В ходе работы были проведены вычислительные эксперименты по оценке эффективности применения графических ускорителей при решении СЛАУ. Тестирование проводилось на наборе из нескольких сгенерированных матриц и выборке из коллекции SuiteSparse Matrix Collection [6]. Полученные результаты демонстрируют ускорение вычислений от 2 до 9 раз при использовании одного GPU по сравнению с CPU реализацией (рис. 1). Достигаемое ускорение зависит от размера матрицы. Так, для матриц, память для хранения которых сопоставима с размером L3 кэша системы, ускорение составляет около 2 раз. Затем, с увеличением размера матрицы растёт получаемое ускорение вычислений вплоть до 9 раз пока не выходит на плато.

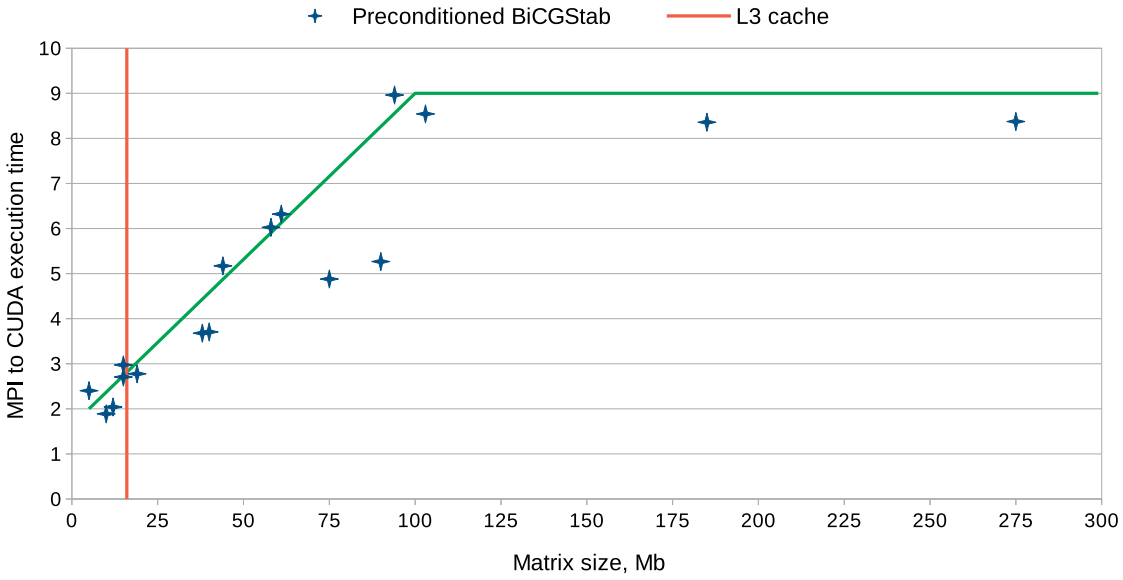


Рис. 1: Ускорение решения СЛАУ при использовании графического ускорителя вместо центрального процессора

Также в работе будут представлены результаты исследования масштабируемости вычислений с использованием нескольких серверных GPU. Исследования будут проведены на базе вычислительного кластера МГУ-270.

Список литературы

[1] Saad Y. Iterative methods for sparse linear systems. SIAM, 2nd edition. Philadelpha, PA. 2003. 520 p.

[2] Williams S., Waterman A., Patterson D. Roofline: An insightful visual performance model for multicore architectures // Communications of the ACM. 2009. Vol. 52. № 4. P. 65–76.

[3] Greathouse J.L., Daga M. Efficient Sparse Matrix-Vector Multiplication on GPUs Using the CSR Storage Format // Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. 2014, P. 769–780.

[4] Krasnopolsky B., Medvedev A. XAMG: A library for solving linear systems with multiple right-hand side vectors // SoftwareX. 2021. Vol. 14. № 100695. `https://doi.org/10.1016/j.softx.2021.100695`.

[5] Krasnopolsky B., Medvedev A. XAMG: Source code repository // `https://gitlab.com/xamg/xamg` (дата обращения: 21.01.2025).

[6] Davis T.A., Hu Y. The university of Florida sparse matrix collection // ACM Transactions on Mathematical Software. 2011. Vol. 38. № 1. P. 1–25.

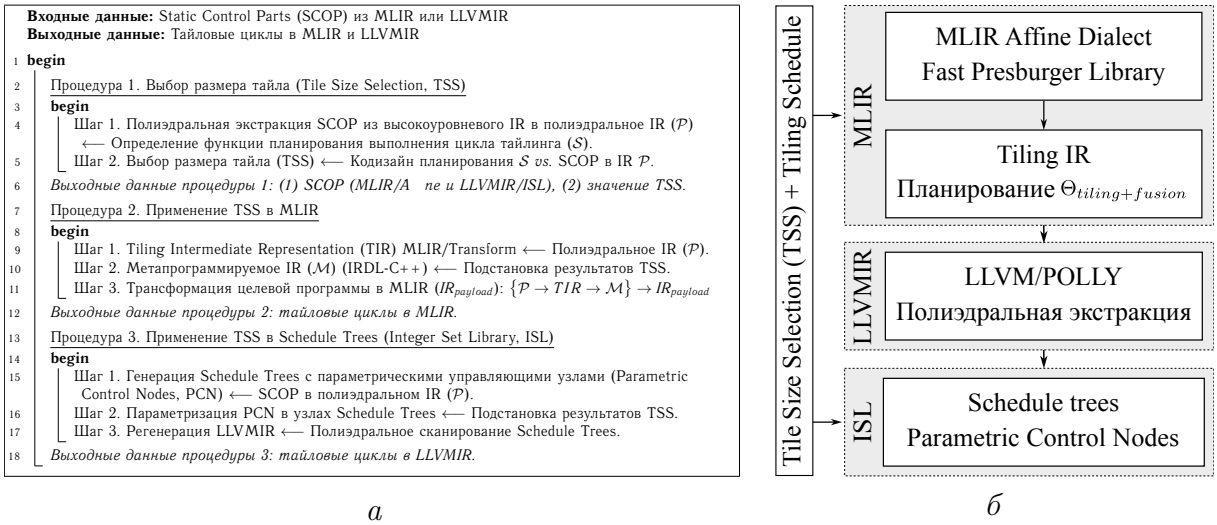
Метод применения размера тайла для эффективной полиэдральной компиляции

А.В. Левченко

Суперкомпьютерный центр СПбПУ

Актуальность работы обусловлена необходимостью разработки метода тайлинга циклов, позволяющего интегрировать алгоритмы выбора размера тайла (Tile Size Selection, TSS) [1] в промежуточные языки планирования (scheduling languages) компиляторов в HPC. Применение значений TSS в языках планирования позволит эффективно варьировать методы тайлинга в зависимости от целевой архитектуры без необходимости изменения многоуровневого промежуточного представления целевой вычислительной программы. Решению задачи интеграции TSS в полиэдральные представления посвящена данная работа.

Вклад данной работы. В работе предложен метод для согласованного применения TSS в двух полиэдральных представлениях (рис. 1). Так, в процедуре 2 (рис. 1,*а*) метод использует результаты работы [2] посредством разработки языка планирования для тайлинга (TIR) на основе предложенного в ней диалекта Transform (языка планирования MLIR). Затем в процедуре 3 (рис. 1,*а*) метод применяет TSS при трансформации деревьев расписания ISL [3]. Полезным аспектом данного подхода стала возможность выборочно применять тайлинг с указанием значений TSS для вычислений в полиэдральных представлениях разных уровней.



а *б*

Рис. 1: Макроструктура метода применения размера тайла: *а* — процедуры метода; *б* — библиотеки IR

Входными данными метода являются статические области графа потока управления (Static Control Parts, SCOP) с циклами. Процедура 1 (рис. 1,*а*) позволяет выбрать размер тайла на основе планирования цикла в полиэдральном представлении $\mathcal{P}$. В процедуре 2 разработан язык планирования (Tiling Intermediate Representation,

TIR) на основе диалекта MLIR/Transform. Предложена композиция IR (рисунок 1,*б*): полиэдральное IR ($\mathcal{P}$) определяет TIR, которое применяется для трансформации IR вычислительной программы ($IR_{payload}$). Процедура 3 позволяет внести планирование в узлы управления (PCN) в представлении Schedule Trees (рисунок 1,*б*) для отложенного тайлинга циклов, не трансформированных ранее.

Выходными данными метода являются тайловые циклы в MLIR/LLVMIR.

Экспериментальные запуски проводились для оценки пространственной локальности (рисунок 2,*а*) и для оценки ускорения тестовых полиэдральных программ, содержащих SCOP (рисунки 2,*в* и 2,*с*), на системе с AMD EPYC 7763 (64c/128t/l3 cache 256mb).

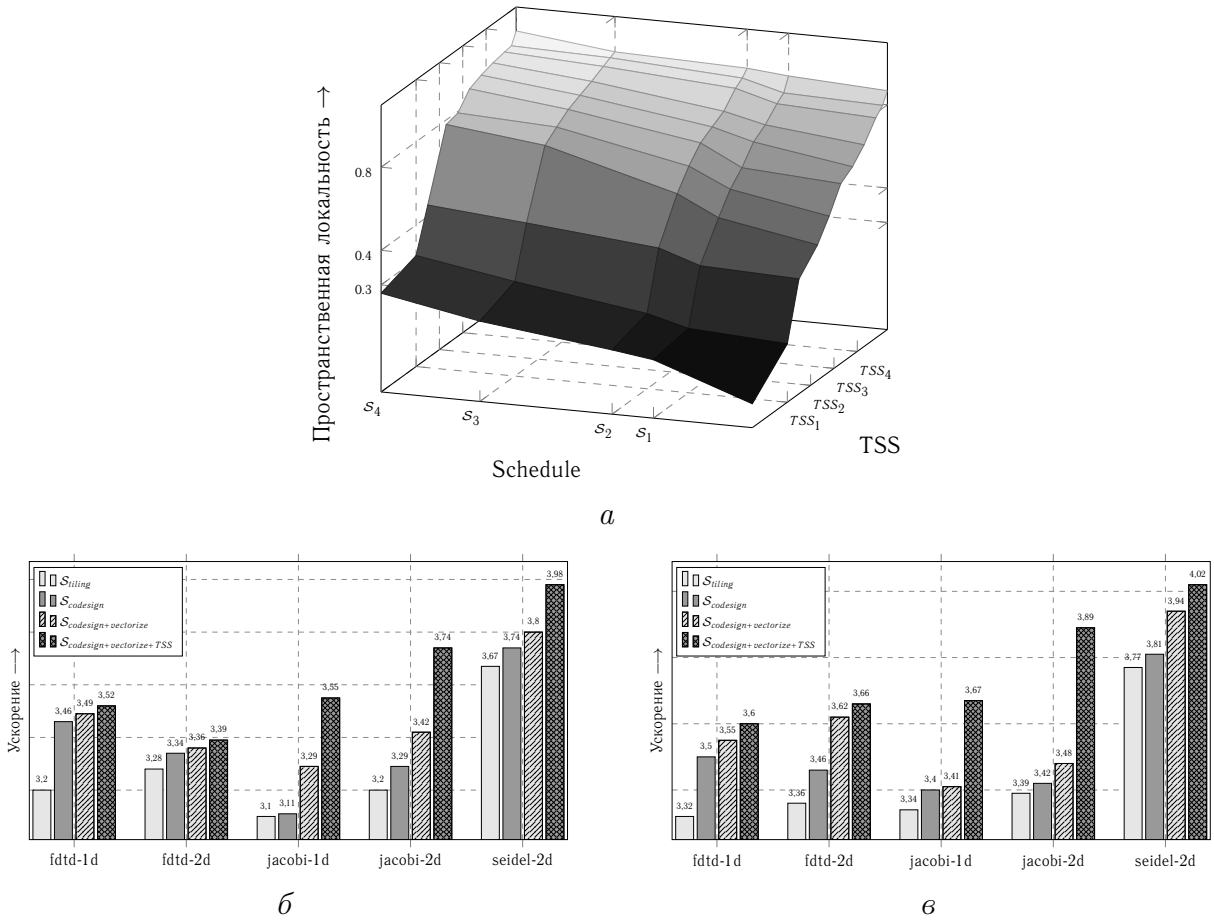


Рис. 2: Локальность и ускорение тайловых циклов относительно вариантов без тайлинга: *а* — Локальность; *б* — Ускорение в MLIR/Transform; *в* — Общее ускорение

Оценка пространственной локальности определяла тенденцию полиэдральной программы обращаться к регионам памяти, расположенным рядом с теми регионами, к которым доступ осуществлялся в течение заданного временного периода. По оси X отложены различные варианты изменения значений размеров тайла, полученных посредством применения TSS. По оси Y отложены различные варианты функций планирования ($\mathcal{S}_1 - \mathcal{S}_4$), применяемые для определения вариантов тайлинга. Здесь $\mathcal{S}_1$ — это минимальный тайлинг, который усложняется до варианта $\mathcal{S}_4$, обозначающего наиболее полный тайлинг с полиэдральным кодизайном и TSS. По оси Z отложено изменение пространственной локальности, возникшее в результате применения предлагаемого метода тайлинга. Показано, что пространственная локальность для набора

тестовых полиэдральных программ при использовании функций планирования тайлинга $\mathcal{S}_{1-4}$ и значений TSS_{1-4} улучшалась в основном благодаря TSS.

Оценка ускорения циклов, представленных в виде SCOP, проводилась с использованием полиэдральных тестовых программ fdtd-1d, fdtd-2d, jacobi-1d, jacobi-2d, seidel-2d отдельно для циклов в MLIR с использованием диалекта Transform (рисунок 2,*б*) и для полного конвейера компиляции, включающего Schedule Trees (рисунок 2,*в*). Наименьшее ускорение по сравнению с результатами неоптимизированной версии демонстрирует изолированный тайлинг циклов $\mathcal{S}_{tiling}$, не учитывающий влияние других трансформаций циклов. Так, $\mathcal{S}_{codesign}$ учитывает совместное применение тайлинга циклов и слияния циклов с векторизацией ($\mathcal{S}_{codesign+vectorize}$). Наиболее заметным результатом представляется ускорение посредством значений TSS ($\mathcal{S}_{codesign+vectorize+TSS}$), повлиявших на пространственную локальность.

Выводы. В работе показана (1) возможность применения TSS в полиэдральных IR в рамках единого конвейера (рисунок 1,*б*), (2) возможность инкапсуляции тайлинга с указанием TSS в виде операций диалекта MLIR/Transform (или Schedule Trees) отдельно от вычислительного IR, что даёт (3) возможность изменения планирования тайлинга и TSS в зависимости от целевой архитектуры без изменения вычислительного IR и (4) возможность выборочной/отложенной трансформации операций MLIR (вложенных деревьев Schedule Trees).

Список литературы

[1] Narasimhan K., Acharya A., Baid A., Bondhugula U. A Practical Tile Size Selection Model for Affine Loop Nests // Proceedings of the 35th ACM International Conference on Supercomputing. Virtual Event, USA. June, 2021. P. 27–39. https://doi.org/10.1145/3447818.3462213

[2] Lücke M.P., Zinenko O., Moses W.S., Steuwer M., Cohen A. The MLIR Transform Dialect: Your Compiler Is More Powerful Than You Think // Proceedings of the 23th ACM/IEEE International Symposium on Code Generation and Optimization. New York, USA. March, 2025. P. 241–254. https://doi.org/10.1145/3696443.3708922

[3] Verdoolaege S. *isl*: An Integer Set Library for the Polyhedral Model // Mathematical Software — ICMS 2010. Proceedings of the 3rd International Congress on Mathematical Software. Kobe, Japan. September, 2010. P. 299–302. https://doi.org/10.1007/978-3-642-15582-6_49

Моделирование течения в пограничном слое

П.А. Сонько, Е.В. Михальченко

Московский государственный университет имени М.В. Ломоносова

Данная работа посвящена численному моделированию ламинарного течения газа в пограничном слое над плоской пластиной с целью верификации теоретических моделей и подготовки к более сложным задачам, связанным с распространением пламени в условиях микрогравитации. Моделирование выполнено с использованием программного комплекса OpenFOAM, что позволило провести точную настройку расчётной сетки, граничных условий и параметров решателя.

В качестве базовой модели рассматривается течение несжимаемого газа вдоль пластины при различных скоростях набегающего потока: 100, 150 и 200 м/с. Расчёты проводились для двумерной области с заданной геометрией и вязкостью воздуха. Для каждого случая были получены профили скорости и толщина пограничного слоя, которые сравнивались с классическим теоретическим решением Блазиуса.

Результаты численного моделирования показали высокую степень согласия с аналитическими данными: значения толщины пограничного слоя, полученные в расчётах, практически совпадают с теоретическими, а форма профилей скорости соответствует ожидаемому поведению ламинарного потока. Это подтверждает корректность выбранной численной схемы и устойчивость модели в условиях, приближенных к реальным.

Работа имеет прикладное значение как первый этап в исследовании процессов горения полимерных материалов, таких как полиметилметакрилат (ПММА), в условиях невесомости [1]. В дальнейшем планируется учитывать теплопроводность, процессы пиролиза и химическую кинетику, что позволит перейти к моделированию распространения пламени и оценке пожароопасности на борту космических аппаратов [2].

Список литературы

[1] Karpov A.I., Korobeinichev O.P., Shaklein A.A., Bolkisev A.A., Kumar A., Shmakov A.G. Numerical study of horizontal flame spread over PMMA surface in still air. David L. Urban, Paul Ferkul, Sandra Olson, Gary A. Ruff, John Easton, James.

[2] T'ien S., Liao Y.-T.T., Li C., Fernandez-Pello C., Torero J.L., Legros G., Eigenbrod C., Smirnov N., Fujita O., Rouvreau S., Toth B., Jomaas G. Flame spread: Effects of microgravity and scale // Combustion and Flame, 2019. Vol. 199. P. 169–182.

Об опыте оптимизации базовых алгоритмов работы с ленточными матрицами в библиотеке OpenBLAS[1]

А.Ю. Пирова, А.А. Воденеева, К.И. Ковалев, А.В. Устинов, Е.А. Козинов, В.Д. Волокитин, А.В. Линев, И.Б. Мееров

Нижегородский государственный университет им. Н.И. Лобачевского

Основным вычислительным ядром программных комплексов для численного моделирования в физике, механике и других областях часто являются операции с плотными и разреженными матрицами, эффективность их реализации значительно влияет на общую эффективность и масштабируемость конечных приложений. С распространением новой архитектуры RISC-V стала актуальной задача оптимизации для нее базовых библиотек, в том числе, библиотек линейной алгебры, поскольку в настоящий момент компиляторы языка C++ не поддерживают автоматическую векторизацию циклов для набора векторных инструкций RVV 0.7.1, а генерация кода для набора инструкций RVV 1.0 не всегда приводит к оптимальному результату. В научных и инженерных приложениях широко используются открытые библиотеки OpenBLAS, Eigen, ATLAS, ARMAS, BLIS и др., оптимизированные под различные архитектуры за счет использования ассемблерных вставок и векторных инструкций. Среди них OpenBLAS наиболее полно использует возможности векторизации для процессоров RISC-V. Однако, массовое тестирование показывает, что ряд функций библиотеки имеет потенциал для дополнительной оптимизации, в том числе, функции BLAS для операций с ленточными матрицами.

Целью данной работы является оптимизация нескольких функций библиотеки OpenBLAS для работы с ленточными матрицами, предназначенная для процессоров архитектуры RISC-V. В докладе будут представлены результаты для четырех функций: матрично-векторного умножения ленточных матриц общего вида, симметричных и треугольных ленточных, а также решение СЛАУ с треугольной ленточной матрицей. Далее предлагаемые алгоритмы описаны на примере функции умножения ленточной матрицы общего вида на вектор (GBMV).

В библиотеке OpenBLAS ленточная матрица общего вида хранится в виде плотной прямоугольной матрицы, записанной по столбцам. Стандартный алгоритм умножения матрицы A на вектор x заключается в обходе матрицы A по столбцам и вызове векторизованных операций скалярного произведения векторов (DOT) или сложения векторов (AXPY), в зависимости от транспонированности матрицы. В OpenBLAS операции DOT и AXPY векторизованы для каждой поддерживаемой архитектуры, но данный подход не эффективен в случаях, когда в столбце матрицы недостаточно элементов для загрузки в векторный регистр.

Доклад продолжает исследования, начатые в работе [1]. Предлагается два алгоритма векторизации функции GBMV. Первый алгоритм адаптирован для матриц с узкой шириной ленты. В нем для выполнения умножения матрица разделяется на полосы шириной, равной количеству элементов, помещающихся в векторный регистр.

[1]Работа выполнена при поддержке программы академического лидерства ННГУ “Приоритет-2030”.

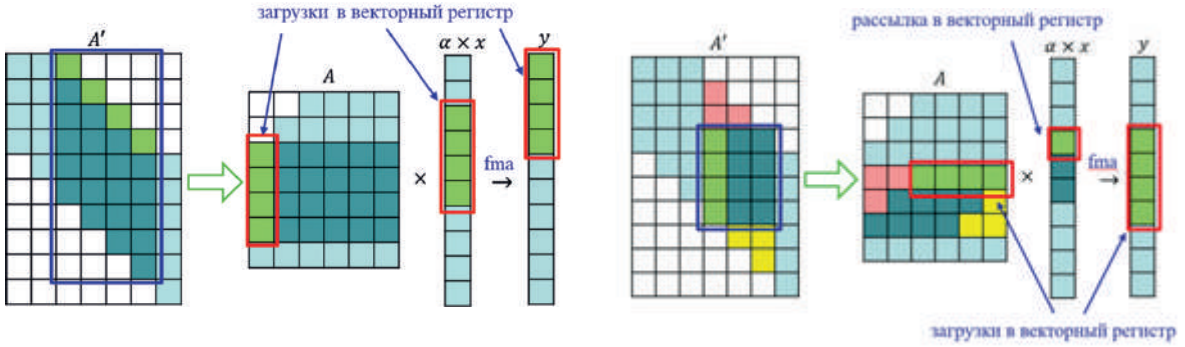


Рис. 1: Схема алгоритмов умножения ленточной матрицы общего вида на вектор. A' — исходная матрица, A — матрица в формате хранения OpenBLAS; *а* — алгоритм с обходом по диагоналям; *б* — алгоритм с выделением плотных блоков

В каждой полосе выполняется обход матрицы по диагоналям с умножением на соответствующий фрагмент вектора x (рис. 1,*а*). Подробнее реализация описана в [1]. Второй алгоритм подходит для матриц с произвольной шириной ленты и основан на выделении в матрице плотных блоков. Близкий к этому подход был предложен в [3]. Так, при обходе матрица разделяется на полосы ширины k, в каждой полосе выделяется плотный прямоугольный блок и два треугольных блока над и под ним (рис. 1,*б*). Элементы прямоугольного блока загружаются в векторные регистры и умножаются на фрагмент вектора x. Затем скалярными операциями вычисляется произведение треугольных блоков на фрагмент вектора x. Ширина полосы k определяется, исходя из размера векторного регистра и точности чисел с плавающей запятой. Описанные алгоритмы были адаптированы для функций умножения симметричной и треугольной ленточной матрицы. Реализация доступна по ссылке [2].

Вычислительные эксперименты проводились на платах двух видов: 1) плата Lichee Pi 4A, процессор T-Head TH1520 с поддержкой 128-bit RVV 0.7.1, 16 GB, ОС

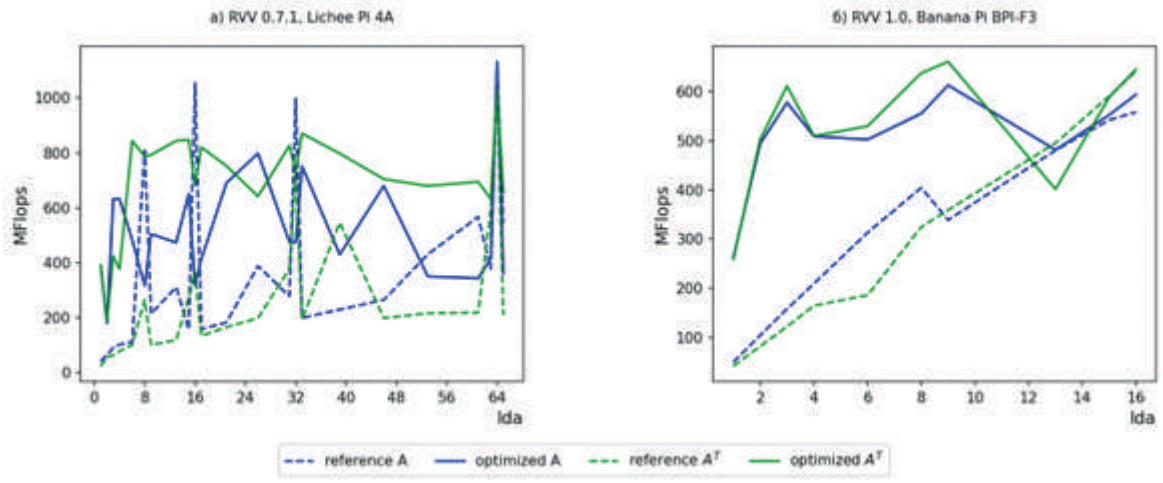


Рис. 2: Производительность функции DGBMV на процессорах RISC-V. *lda* — число диагоналей матрицы, reference — базовая реализация OpenBLAS, optimized — оптимизированная реализация. Размер матрицы 2,5 млн. × 2,5 млн. Вычисления в двойной точности.

Debian GNU/Linux 12, кросс-компилятор GCC 10.4.0; 2) плата Banana Pi BPI-F3, процессор SpacemiT Keystone K1 с поддержкой 256-bit RVV 1.0, 16 GB, ОС Bianbu 1.0.15, кросс-компилятор GCC RISC-V 14.2.0.

Для функции GBMV были реализованы оба оптимизированных алгоритма и эмпирически выбраны критерии переключения между ними в зависимости от числа диагоналей матрицы и типа процессора. Показано [1], что производительность GBMV зависит от числа диагоналей матрицы и не зависит от ее порядка. На рис. 2 сравнивается производительность исходной реализации функции DGBMV (GBMV в двойной точности) и комбинации оптимизированных алгоритмов при разном числе диагоналей. На процессоре Lichee Pi 4A (рис. 2,*а*) оптимизированный алгоритм опережает базовую реализацию в 2,8 раз в среднем для обычной матрицы и в 4,7 раз для транспонированной. На процессоре Banana Pi BPI-F3 (рис. 2,*б*) оптимизированный алгоритм опережает базовый на матрицах с узкой лентой, в среднем, в 3 раза для обычной матрицы, в 3,9 раз — для транспонированной матрицы. В дальнейшем планируется интегрировать оптимизированную версию в основной репозиторий OpenBLAS.

Список литературы

[1] Pirova A. et al. Performance optimization of BLAS algorithms with band matrices for RISC-V processors // arXiv preprint arXiv:2502.13839. – 2025.

[2] Код оптимизированных реализаций. `https://github.com/UNN-ITMM-Software/OpenBLAS/tree/band_matrix_RVV_improving`

[3] Dufrechou E. et al. Efficient symmetric band matrix-matrix multiplication on GPUs // High Performance Computing: First HPCLATAM-CLCAR Latin American Joint Conference, 2014. Proceedings 1. – Springer Berlin Heidelberg, 2014. – P. 1-12. `https://doi.org/10.1007/978-3-662-45483-1_1`

Обработка растровых данных на GPU в геоинформационных системах

К.И. Сулейманов, Р.А. Родригес Залепинос

Национальный исследовательский университет «Высшая школа экономики»

Введение. Растровая модель данных — это широко распространенный метод хранения географических данных. Чаще всего эта модель представляет собой структуру, напоминающую сетку, в которой хранятся значения с регулярными интервалами по всей площади растра. Растры особенно хорошо подходят для хранения непрерывных данных, таких как температура и высота над уровнем моря, но также могут содержать дискретные и категориальные данные, например, данные о землепользовании. Разрешение растра задается в линейных единицах (например, метрах) или угловых единицах (например, одной угловой секунде) и определяет протяженность вдоль одной стороны ячейки сетки. Растры высокого (или низкого) разрешения имеют сравнительно меньшее расстояние между ячейками сетки и больше ячеек, чем растры низкого (или высокого) разрешения, и требуют относительно большего объема памяти для хранения. Активные исследования в этой области направлены на улучшение схем сжатия и реализацию альтернативных форм ячеек (например, шестиугольников), а также на улучшение поддержки функций хранения и анализа растров с несколькими разрешениями [1].

Например, данные Sentinel — это спутниковые данные, предоставляемые программой Copernicus Европейского космического агентства [2]. Они включают многоспектральные снимки, радарные данные и данные о высотах, пригодные для мониторинга земной поверхности, сельского хозяйства, водных ресурсов и экологии [3].

Проблема обработки растровых данных заключается в их огромных объёмах. При увеличении разрешения растрового изображения (т.е. уменьшении размера пиксела) объём хранимых данных резко растёт — например, при двукратном уменьшении размера ячейки объём данных может увеличиться вчетверо, и скорость их обработки сильно снижается. С каждым годом появляются новые спутниковые миссии и БПЛА, генерирующие всё больше данных высокого разрешения, поэтому проблема быстрой обработки больших растров становится всё более актуальной и работа с растровыми данными выходит на первый план [3].

Для ускорения обработки растровых данных применяются графические процессоры (GPU) — специальные видеокарты, способные выполнять вычисления параллельно. GPU изначально предназначены для ускорения работы с графикой, но в последнее время их вычислительные мощности активно используется для общих вычислительных задач. В инструментах геопространственного анализа задача обработки растров выполняется GPU вместо центрального процессора: задача дробится на множество мелких подзадач, которые GPU выполняет параллельно с высокой скоростью, затем результаты собираются воедино [4, 5].

Цель работы — разработка универсальных GPU-ускоренных подходов для пакетной обработки геопривязанных растровых данных в ГИС, способные автоматически распараллеливать вычисления на видеокартах NVIDIA под CUDA и обеспечивать стабильное ускорение на любых объёмах. Ожидается достичь 5–10 кратного

сокращения времени анализа спутниковых данных и аэрофотоснимков (Sentinel-2, другие растры) и гибкая интеграция с популярными ГИС-платформами с целью масштабирования обработки растровых данных.

В качестве основы предлагается использовать библиотеку GDAL на языке C++ [9]. Например, реализация алгебры карт (Map Algebra), используя библиотеку для чтения и записи данных, выполняя математические операции над пикселями растров, обрезки (clipping) с использованием GDALWarpOperation с параметрами -cutline и -crop_to_cutline для выделения области интереса из растра по заданному векторному контуру или прямоугольной области и ресемплирования (resampling) с использованием функции GDALWarpOperation с настройкой параметров интерполяции и масштабирования для изменения пространственного разрешения растра с использованием различных методов интерполяции (например, ближайшего соседа, билинейной, кубической).

Существующие GPU-библиотеки и утилиты для обработки растров: cuCIM от NVIDIA — это библиотека для GPU-ускоренной загрузки и обработки больших файлов TIFF [7]. OpenCV имеет модуль CUDA с GPU-версиями фильтров и преобразований [8], библиотека CuPy в Python предоставляет NumPy — подобные массивы для вычислений на GPU. Библиотека Rasterio (на основе GDAL) непосредственно CUDA не использует, но с помощью Dask/CuPy можно распараллеливать обработку больших растров. Несмотря на наличие всех этих инструментов, они рассчитаны на отдельные задачи и не приспособлены для массивно-параллельной пакетной обработки геопривязанных растров в ГИС-сценариях. Аналогов полностью предполагающих решение для массивно-параллельной обработки данных ГИС нет.

Исследование [4] реализовало параллельную обработку растров в ГИС посредством CUDA C, обеспечив 7-кратное ускорение по сравнению с ArcGIS. Время выполнения CUDA линейно зависело от размера данных (подтверждено тестами до 8×64 ГБ). Например, обработка растра размером 1.96 ГБ заняла 2000 секунд, сохраняя линейное ускорение. Основной узкий момент — загрузка данных в RAM, но оптимизация GPU сократила общее время. Результаты подтверждают эффективность GPU для масштабируемых вычислений.

В работе [5] представлен открытый плагин QGIS на Python/PyCUDA для ускорения анализа рельефа посредством GPU. PyCUDA показал 3-кратное ускорение по сравнению с QGIS (C++): обработка 1.5 ГБ заняла 3:35 минут вместо 11:00 минут, 12 ГБ — 28 минут вместо 45. Основное преимущество — в вычислениях (GPU обрабатывал 1.5 ГБ за 2 сек), но ограничения ввода-вывода снижали эффект для больших данных.

Исследование [6] показывает применение данных Sentinel: авторы оценили долю городской растительности в дельте реки Чжуцзян (Китай) с помощью данных Sentinel-2 (NDVI, разрешение 10 м) и валидации по высокодетальным изображениям Google. Результаты показали корреляцию 0.97 между оценками Sentinel-2 и эталонными данными, а также выявили значительные расхождения с методом WUDAPT level-0 (100 м): в плотных урбанизированных зонах WUDAPT завышал городскую фракцию, а в промзонах — занижал. Подход обеспечивает детализацию, критичную для климатического моделирования и анализа городской среды [6].

Тензорные СУБД являются популярным инструментом для обработки растровых данных, однако в них до сих пор не используются GPU для ускорения различный операций [10–13].

Список литературы

[1] Rodriges Zalipynis R.A. Array DBMS: Past, Present, and (Near) Future. Proc. VLDB Endow. 2021, 14, 3186–3189

[2] Sentinel Data. Copernicus satellite missions. `https://sentivista.copernicus.eu`

[3] The ArcGIS Imagery Book: New View. New Vision. Redlands, CA: ESRI Press, 2016.

[4] Kirby S.P., Kostan W.B., Lembo A.J. Parallelizing Raster-Based Functions in GIS with CUDA C. Salisbury University. 2013. `https://faculty.salisbury.edu/~ajlembo/cudaposter.pdf`

[5] Fuerst A. et al. GIS based terrain analysis with GPU and CPU strategies. Salisbury University. 2016. `https://faculty.salisbury.edu/~ealu/reu/Projects_File/2016/Lembo.pdf`

[6] Wong M.M.F., Fung J.C.H., Yeung P.P.S. High-resolution calculation of the urban vegetation fraction in the Pearl River Delta from the Sentinel-2 NDVI for urban climate model parameterization. Geoscience Letters, 6, 1-10, 2019.

[7] NVIDIA RAPIDS cuCIM. `https://github.com/rapidsai/cucim`

[8] OpenCV (модуль CUDA). `https://opencv.org/platforms/cuda`

[9] GDAL – Geospatial Data Abstraction Library. `https://gdal.org`

[10] Rodriges Zalipynis R.A. BitFun: Fast Answers to Queries with Tunable Functions in Geospatial Array DBMS. Proc. VLDB Endow. 2020, 13, 2909–2912.

[11] Rodriges Zalipynis R.A. ChronosDB: Distributed, File Based, Geospatial Array DBMS. Proc. VLDB Endow. 2018, 11, 1247–1261.

[12] Rodriges Zalipynis R.A. ChronosDB in Action: Manage, Process, and Visualize Big Geospatial Arrays in the Cloud // in Proceedings of the 2019 International Conference on Management of Data, Amsterdam, The Netherlands, 30 June–5 July 2019; pp. 1985–1988.

[13] Baumann P., Misev D., Merticariu V., Huu B.P. Array databases: Concepts, standards, implementations // J. Big Data 2021, 8, 1–61.

Оценка влияния архитектуры операционной системы на производительность суперкомпьютера

А.В. Семанин, М.М. Никитин, Крылов Д.Е.

Севастопольский государственный университет

Многие современные суперкомпьютеры строятся как кластеры Beowulf [1]. Такие системы состоят из общедоступных аппаратных компонентов и используют Unix (Linux) в качестве операционной системы с организацией параллельных вычислений при помощи библиотек PVM или MPI. Однако, существует вариант построения суперкомпьютера на тех же аппаратных компонентах, но на операционной системе Plan 9 [2].

Plan 9 — это операционная система совместного использования ресурсов. Unix и ее архитектурные потомки, такие как Linux, являются системами удаленного доступа [3]. Системы совместного использования ресурсов позволяют получать доступ к ресурсам, таким как файлы и устройства ввода-вывода, независимо от того, где они находятся в мире. Системы удаленного доступа требуют знания идентификационных данных хоста, содержащего ресурс, прежде чем получить к нему доступ. Системы совместного использования ресурсов обеспечивают прозрачность местоположения. Главной особенностью Plan 9 является концепция пространства имен. Пространство имен состоит из файлов и каталогов, используемых группой из одного или нескольких процессов. Пространства имен создаются путем добавления серверов 9p и устройств ядра к процессу с помощью системного вызова mount. Пространства имен не являются глобальными, как в Unix; скорее, они устанавливаются процессом init и наследуются. Потомки init могут разорвать соединение с пространством имен init, изменить его и передать измененное пространство имен своим потомкам. Архитектура Plan 9 резко контрастирует с Linux и Unix, в которых ресурсы представлены всевозможными вещами: файлы — файловыми системами в ядре; сетевые стеки — системными вызовами сокетов, каналами межпроцессорного взаимодействия (IPC) и синтетическими файлами; процессы — системными вызовами и файловыми системами. В отличие от Plan 9, даже пространствам имен Linux не хватает базовой согласованности: в Linux существует почти дюжина различных типов пространств имен; пространства имен — это не столько объединяющий механизм в Linux, сколько дополнительный набор различных API.

Могут ли суперкомпьютеры с общими ресурсами быть быстрее суперкомпьютеров с удаленным доступом? Мы намерены провести сравнительное тестирование этих операционных систем на одинаковом «железе».

В качестве тестовой задачи нами была выбрана математическая модель HEOM [4], которая испытывает сложности на традиционных суперкомпьютерах и которая была реализована в PHI — Parallel Hierarchy Equations of Motion Integrator [5]. Многие системы, которые часто возникают в биологии и, в частности, в описании фотосинтезирующих организмов, требуют квантовой механики для точного представления. Теоретическое исследование динамики таких систем является чрезвычайно сложной задачей с точки зрения моделирования, а также с точки зрения вычислений.

HEOM — это набор связанных дифференциальных уравнений, число которых может легко превышать 100 000 (в зависимости от параметров системы). С помощью параллельного компьютера можно решать эти уравнения одновременно, заставив каждый процессор численно интегрировать только подмножество уравнений. Однако, связанная природа уравнений требует, чтобы процессоры передавали свои вычисленные обновления интеграции на каждом временном шаге (или несколько раз в течение временного шага для схем интеграции более высокого порядка). Чрезвычайная стоимость связи численного интегрирования исключает использование параллельных компьютеров с распределенной памятью до тех пор, пока скорости между узлами в сети не будут значительно улучшены. Таким образом, в настоящее время PHI в основном использует параллельные компьютеры с общей памятью, где пропускная способность межпроцессорной связи достаточна для систем разумного размера. Схема разбиения использует коммуникационную структуру HEOM, представленную на рис. 1. HEOM можно представить как d-мерный симплекс Паскаля [6] для системы, содержащей d состояний. Каждая вершина в симплексе Паскаля соответствует матрице dxd, которая имеет ассоциированное дифференциальное уравнение. Связь между вершинами на данном уровне симплекса происходит только с вершинами на уровне непосредственно выше и непосредственно ниже. PHI использует эту структуру для минимизации коммуникации и, таким образом, значительного улучшения времени численного интегрирования.

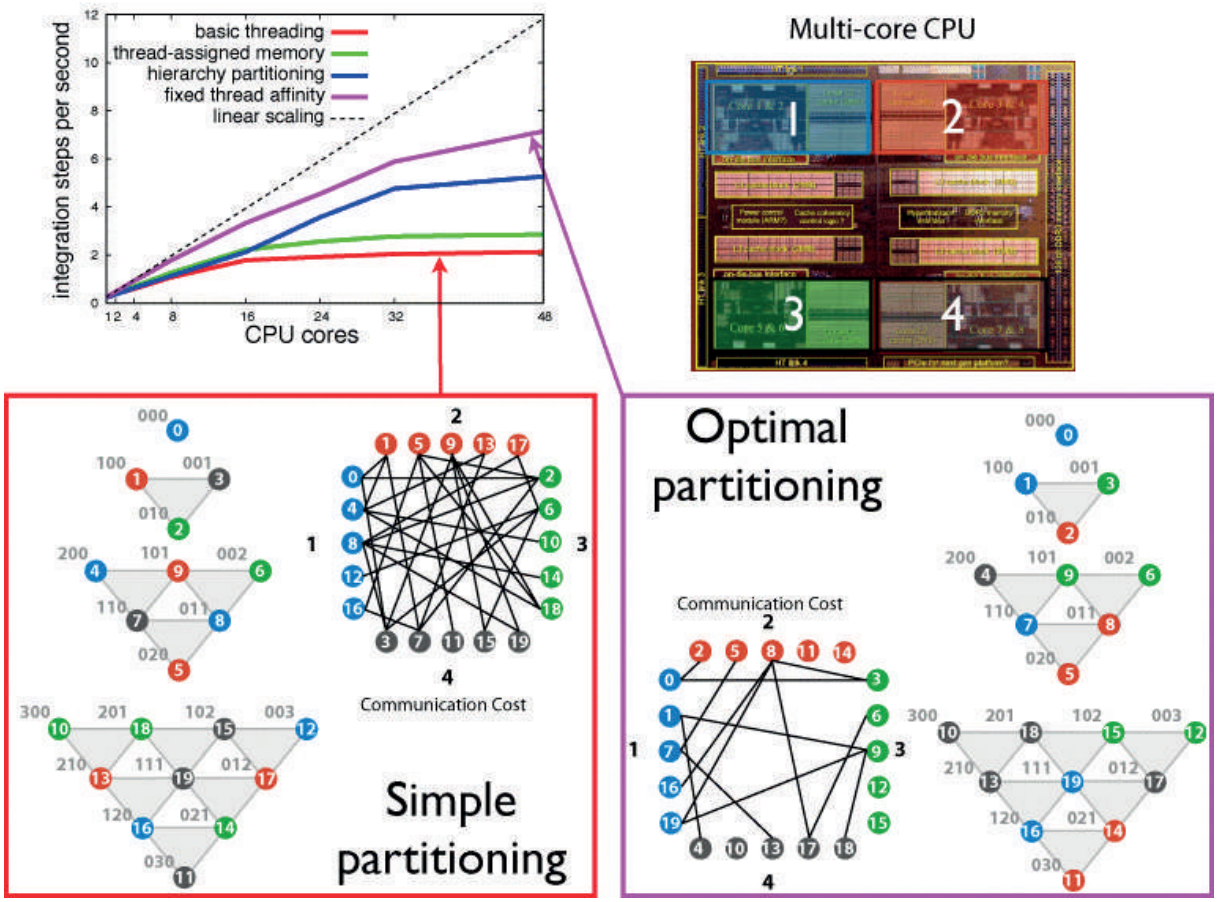


Рис. 1: Организация коммуникационной структуры HEOM

Предварительные результаты тестирования программы PHI на суперкомпьютере

с установленной на нем операционной системы Plan 9 показали увеличение его производительности по сравнению с установленной на него ранее OpenHPC и CentOS 8. Это вероятно обусловлено уменьшением «шума» операционной системы и уменьшением ее накладных расходов при передаче данных между процессорами.

Список литературы

[1] Кластер Beowulf `https://en.wikipedia.org/wiki/Beowulf_cluster`

[2] Pike R., Presotto D.L., Dorward S., Flandrena B., Thompson K., Trickey H., et al. Plan9 from Bell Labs. Computing Systems, 8(2), 221–254 (1995).

[3] Padlipsky M.A. The Elements of Networking Style: And Other Essays and Animadversions on the Art of Intercomputer Networking. Upper Saddle River, NJ, USA: Prentice-Hall c. (1985).

[4] `http://www.ks.uiuc.edu/Research/phi/about.html`

[5] Johan Strümpfer and Klaus Schulten // Journal of Chemical Theory and Computation, 8:2808-2816, 2012.

[6] Pascal's simplex `https://en.wikipedia.org/wiki/Pascal%27s_simplex`

Построение полноволновых сейсмограмм в трехмерных и одномерных моделях Земли[1]

О.А. Усольцева, В.М. Овчинников

Институт динамики геосфер имени академика М.А.Садовского РАН

Внутреннее ядро Земли из-за его малых размеров остается наиболее трудным для изучения элементом нашей планеты, единственного, в отношении которого так и не сложилось единой и непротиворечивой картины его описания — как в целом, так и отдельных свойств. Для сейсмологии основные проблемы связаны с исключением из измеряемых на поверхности Земли характеристик сейсмического волнового поля факторов, обусловленных особенностями вышележащих оболочек Земли — коры, мантии и внешнего ядра, а также очень неравномерного по объему зондирования различных областей внутреннего ядра. Большая неопределенность в физических параметрах ядра значительно усложняет интерпретацию экспериментальных сейсмических данных. Тем не менее, прогресс продолжается по многим направлениям, одним из которых является применение новых методов расчета полноволновых теоретических сейсмограмм для трехмерных моделей Земли. Такие сейсмограммы позволяют восполнить недостатки наблюдательных возможностей сейсмических сетей, а также повысить качество интерпретации натурных сейсмических данных.

Мы представляем результаты математического моделирования сейсмограмм с использованием алгоритма AXISEM3D [1], в котором с одной стороны есть возможность учесть неоднородности в мантии, коре и топографию границ, а с другой — проводить расчеты в одномерных глобальных моделях с высокой скоростью и экономией вычислительных ресурсов. Программа построения волновых полей AXISEM3D является усовершенствованным продолжением осесимметричного комбинированного аналитически-спектрально-элементного алгоритма AXISEM [2]. В AXISEM3D совмещены два численных подхода к построению сейсмограмм — спектрально-элементный и псевдоспектральный. Вычислительная эффективность такого гибридного метода для 3D моделей обусловлена хорошей горизонтальной гладкостью существующих трехмерных моделей Земли, а также наличием в задаче единственного источника. Учет рельефа существующих границ кора-мантия, 410 км, 660 км, мантия-ядро, а также эллиптичности Земли становится более эффективным в вычислительном плане в AxiSEM3D по сравнению с обычным спектрально-элементным методом за счет использования численного преобразования «перемаркировки частиц» [3], при котором элемент неправильной формы превращается в элемент шарового слоя.

При описании скоростных и плотностных характеристик 3D модели порядок Фурье-разложения варьируется в зависимости от сложности пространственной функции физического параметра, а также от рассматриваемого момента времени. В связи с этим при распараллеливании в AXISEM3D не выгодно равномерное распределение спектральных элементов по процессорам, как это было сделано в AXISEM. Для оптимизации вычислительной процедуры применяется стандартный многоуровневый рекурсивный алгоритм поиска оптимального распределения по процессорам, используемый для неструктурированных графов [4].

[1]Работа выполнена в рамках Госзадания № 125012200561-3.

Представляемый в работе алгоритм для построения глобальных сейсмограмм написан на языке Си++ с распараллеливанием через MPI интерфейс с использованием не блокируемых асинхронных связей, т.е. выполнение одних задач продолжается, когда результаты других задач передаются в фоновом режиме. Ранее в [5] были представлены авторские расчеты глобальных синтетических сейсмограмм для одномерных сферически-симметричных моделей Земли с помощью более ресурсоемких алгоритмов DSM [6] и Axisem [2], которые написаны на языке Фортран, а распараллеливание организовано с использованием интерфейса MPI с синхронными связями.

Оценены возможности алгоритма на вычислительном кластере, установленном в ИДГ РАН, со следующими характеристиками: 1 процессор Intel(R) Xeon(R) Silver 4210, 10 ядер, на каждом ядре 2 потока с частотой 2.2 ГГц, ОП 100 ГБ, OS Linux Mint 20.1. Для изучения особенностей структуры волнового поля расчеты проводились для 1D модели PREM [7] и 3D модели S362ANI [8], учитывающей анизотропию и вариации скорости поперечных волн в мантии и описывающей горизонтальные неоднородности с помощью 362 сферических сплайнов. При моделировании 30-минутных сейсмограмм с минимальным периодом 50 сек для 129 станций мировой сети при задании механизма источника с помощью тензора моментов затрачивается 0.2 п $\cdot$ ч (потоко-часа) и 1.6 п $\cdot$ ч для 1D и 3D моделей соответственно при распараллеливании на 4 потока и 0.9 п $\cdot$ ч (1D) и 4.6 п $\cdot$ ч (3D) при распараллеливании на 20 потоков. При вычислении более детальных сейсмограмм с минимальным периодом 10 сек для трехмерной модели Земли количество используемых потоко-часов возрастает в 30 раз при распараллеливании на 20 потоков, при этом на обмен сообщениями между процессами тратится около 30% времени.

С помощью представляемого алгоритма авторы планируют провести моделирование поздней коды землетрясений, а также фаз, связанных с ядром, для трехмерных моделей Земли и модифицированных одномерных моделей с особенностями в подошве и кромке внешнего ядра, которые при использовании подходов [2, 6] были или невозможны, или численно неустойчивы.

Список литературы

[1] Leng K., Nissen-Meyer T., van Driel M., Hosseini K., Al-Attar D. AxiSEM3D: broadband seismic wavefields in 3-D global earth models with undulating discontinuities // Geophys. J. Int. 2019. 217. 2125–2146. https://doi.org/10.1093/gji/ggz092

[2] Nissen-Meyer T., van Driel M., Stähler S.C., Hosseini K., Hempel S., Auer L., Colombi A., Fournier A. AxiSEM: Broad-band 3-D seismic wavefields in axisymmetric media // Solid Earth. 2014. V. 5. № 1. P. 425–445. https://doi.org/10.5194/se-5-425-2014

[3] Al-Attar D., Crawford O. Particle relabelling transformations in elastodynamics // Geophys. J. Int., 2016. 205(1), 575–593.

[4] Karypis G., Kumar V. A fast and high quality multilevel scheme for partitioning irregular graphs // SIAM J. Sci. Comput., 1998. 20(1), 359–392.

[5] Овчинников В.М., Усольцева О.А. Численное моделирование сейсмограмм для изучения структурных особенностей зоны перехода от внешнего к внутреннему ядру Земли // Суперкомпьютерные дни в России: Труды международной

конференции. – М.: Изд-во МГУ, 2017. с.757-758. http://russianscdays.org/Proceedings17

[6] Kawai K., Takeuchi N., Geller R.J. Complete synthetic seismograms up to 2 Hz for transversely isotropic spherically symmetric media // Geophys. J. Int., , 2006, 164, 411-424. https://doi.org/10.1111/j.1365-246X.2005.02829.x

[7] Dziewonski A.M., Anderson D.L. Preliminary reference Earth model // Physics of the Earth and Planetary Interiors. 1981, 25 (4): 297–356. https://doi.org/10.1016/0031-9201(81)90046-7

[8] Kustowski B., Ekstrom G., Dziewonski A.M. Anisotropic shear-wave velocity structure of the Earth's mantle: A global model // JGR. 2008. V. 113. B06306. https://doi.org/10.1029/2007JB005169

Распространение фронта пламени в канале с учётом обрыва реакции на стенке

Р.Д. Коробов, Е.В. Михальченко

Московский государственный университет имени М.В. Ломоносова

В работе исследуется распространение пламени в водородно-воздушной смеси с учётом гетерогенных эффектов — в частности, наличия обрыва цепных реакций на стенках. Основное внимание уделено влиянию поглощения атомарного водорода (H) поверхностью реактора на стабильность и скорость горения. Для математического моделирования использована система уравнений Навье–Стокса [1], с учетом многокомпонентной диффузии, теплопроводности и детальной химической кинетики [2]. Численный анализ уравнений математической модели выполнен с помощью бездиссипативного метода КАБАРЕ [3], обеспечивающего корректное воспроизведение структуры мелкомасштабных газодинамических течений и структуры пограничного слоя [4]. Вычисления проведены с использованием комплекса NRG с открытым исходным кодом [5].

Результаты численного моделирования распространения пламени в двухмерном канале, заполненном стехиометрической смесью водород-воздух, показали, что поглощение атомарного водорода на стенках канала приводит к обрыву цепной реакции на стенках канала, локальному торможению пламени и соответствующему уменьшению интегральной скорости распространения пламени по сравнению с расчетом без учета поглощения водорода на стенках. При этом в приграничных зонах наблюдаются локальные минимумы до 1.2 МДж/(м$^3 \cdot$ с). Это подчёркивает необходимость учёта гетерогенных процессов при проектировании микрореакторов и горелочных устройств.

Список литературы

[1] Law C.K. Combustion Physics. Cambridge Univ. Press, 2006.

[2] Keromnes A. et al. Kinetic modeling of hydrogen oxidation. Combust. Flame, 2013.

[3] Karabasov S.A., Goloviznin V.M. A CABARET scheme as a low-dispersive method for computational aeroacoustics // J. Comp. Phys., 2009, Vol. 228, pp. 7426–7451.

[4] Kiverin A.D., Yakovenko I.S. On the mechanism of flow evolution in shock-tube experiments // Phys. Lett. A, 2018, Vol. 382, pp. 309–314

[5] `https://github.com/yakovenko-ivan/NRG`

Расчет распространения динамических возмущений в сложных геометриях, описываемых многоблочными и наложенными структурированными сетками с использованием параллельных вычислений[1]

Н.А. Волков[1], И.Н. Агрелов[1], Н.И. Хохлов[1,2]

Московский физико-технический институт
(национальный исследовательский университет)[1],
Федеральный научный центр
Научно-исследовательский институт системных исследований
Национального исследовательского центра «Курчатовский институт»[2]

Нефтегазовый сектор является одним из ключевых для Российской экономики. Значительная часть запасов нефти и газа находится в арктическом регионе. На сегодняшний день в нем, согласно данным «Роснефти», добывается 10% нефти и 25% газа в мире [1]. Из них 80% расположены на российских территориях. Поэтому Арктика является одним из важных направлений развития России. Но Арктика до сих пор является слабо освоенным регионом. Суровый климат региона создает множество трудностей: долгая зима, толстый ледяной покров более 2 метров, ураганные ветры до 50 м/с и высокие волны до 10 м. Это накладывает определенные требования к используемым сооружениям и оборудованию. В частности, важен вопрос устойчивости ледостойких нефтедобывающих платформ.

Одним из важных направлений исследований является определение динамической прочности сооружений. Для этого в них можно численно моделировать распространение динамических возмущений. Одним из этапов моделирования является дискретизация расчетной области и создание сетки. Для этого используют два основных типа сеток: структурированные и неструктурированные. Каждый тип имеет свои преимущества и недостатки. В этой работе используются структурированные сетки. Расчетная сетка создавалась на основе платформы «ПА-Б» проекта Сахалин-2 [2]. Платформа разбивается на блоки из простых геометрий, которые стыкуются между собой (рис. 1). Итого, расчетная сетка состоит из большого числа отдельных сеток. Расчет проводится с использованием программного комплекса, разработанным на кафедре информатики и вычислительной математики и лаборатории прикладной вычислительной геофизики МФТИ и предназначенным для расчета волновых возмущений. В нем используется сеточно-характеристический метод [3] с разделением по пространственным переменным.

[1]Работа выполнена в рамках государственного задания НИЦ «Курчатовский институт» — НИИСИ по теме № FNEF-2024-0002 «Математическое моделирование многомасштабных динамических процессов и системы виртуального окружения» (1023032900401-5-1.2.1).

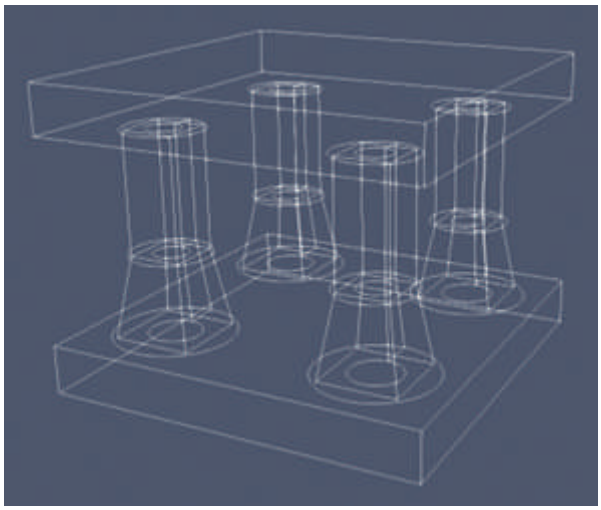


Рис. 1: Блочно-структурированная наложенная сетка платформы.

Для ускорения расчета предлагается использовать распараллеливание на MPI. Стандартно программой каждая сетка разбивается равномерно по количеству MPI процессов (рис. 2) [4]. Это приводит к измельчению сеток и необходимости обмена данными между процессами. В этой работе предлагается разбивать сетки по процессам крупнее особым образом [5]. Учитывая размер сеток, нужно распределить их по процессам так, чтобы все процессы работали примерно одинаковое время. Результат работы алгоритма показан на рис. 3.

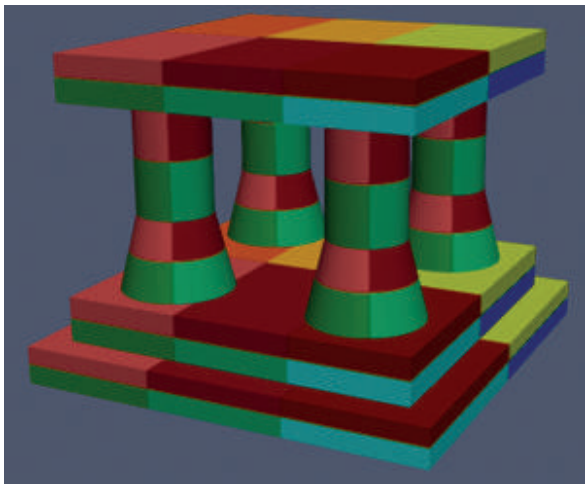


Рис. 2: Стандартная декомпозиция расчетной сетки по MPI процессам.

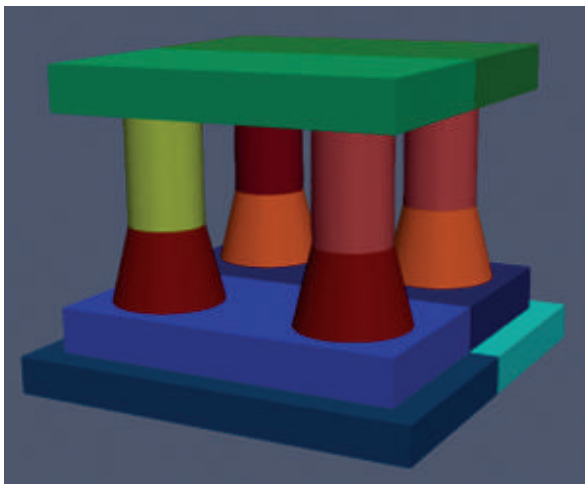


Рис. 3: Декомпозиция расчетной сетки по MPI процессам предложенным алгоритмом.

Были проведены расчеты на разном количестве MPI процессов с использованием двух способов декомпозиции. Вычисления проводились на машине с процессором Intel Xeon E5-2620 v2. Результаты расчета показаны на рис. 4. График ускорения приведен на рис. 5.

Далее планируется улучшить алгоритм декомпозиции. Сейчас он не учитывает, что некоторые сетки сшиваются между собой. Возможно доработка алгоритма ускорит расчет, если на одном процессе будут вычисляться значения в соседних сетках,

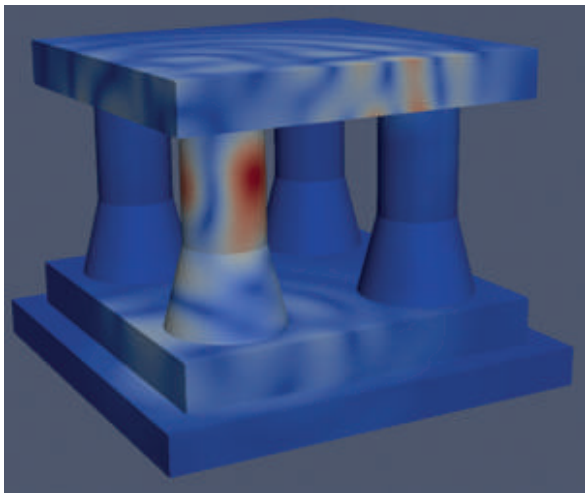


Рис. 4: Пример распространения волны в платформе.

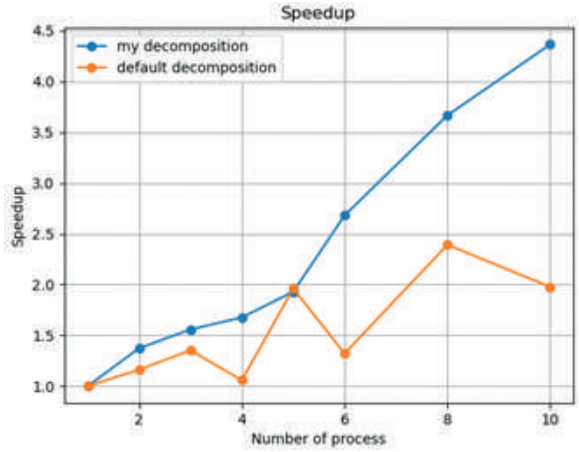


Рис. 5: Ускорение расчета для разных способов декомпозиции на разном количестве процессов.

так как в этом случае будет меньше пересылок данных между процессами. Также планируется провести более масштабные тесты ускорения на более подробной сетке с использованием большего числа процессов.

Список литературы

[1] Крупнейшая арктическая держава. 13.02.25 // Портал центрального диспетчерского управления топливно-энергетического комплекса [Электронный ресурс]. `https://www.cdu.ru/tek_russia/articles/7/1328/` (дата обращения: 24.02.25).

[2] Цуприк В.Г. Динамические ледовые нагрузки на сооружения континентального шельфа. Владивосток: Изд-во Дальневост. федерал. ун-та, 2023. `https://doi.org/10.24866/7444-5500-2`.

[3] Петров И.Б., Хохлов Н.И. Моделирование задач 3D сейсмики на высокопроизводительных вычислительных системах // Математическое моделирование. 2014. Т. 26, № 1. С. 83–95.

[4] Хохлов Н.И., Петров И.Б. Применение современных технологий для высокопроизводительных вычислительных систем для решения задач локальной и глобальной сейсмики // Суперкомпьютерные дни в России. Труды международной конференции. 2015. С. 380-391.

[5] Agrelov I.N., Khokhlov N.I., Stetsyuk V.O., Agibalov S.D. On an Algorithm for Decomposing Multi-Block Structured Meshes for Calculating Dynamic Wave Processes in Complex Structures on Supercomputers with Distributed Memory // Supercomputing Frontiers and Innovations, 2025. Vol. 11, no. 4. P. 54–65. `https://doi.org/10.14529/jsfi240405`

Реализация модели мелкой воды на сетке кубическая сфера на графических процессорах[1]

Д.А. Марханов[1,2], В.В. Шашкин[1,2,3], Г.С. Гойман[1,2,3]

Институт вычислительной математики им. Г.И. Марчука РАН[1],
Гидрометеорологический научно-исследовательский центр России[2],
Московский физико-технический институт[3]

Развитие графических процессоров (GPU) дало толчок к бурном росту в области высокопроизводительных вычислений. Кластера, использующие графические ускорители, считаются эффективной альтернативой суперкомпьютерам на основе классических процессоров (CPU), так как имеют более высокие соотношения цена-производительность и производительность-энергопотребление. Более того, семейства GPU-устройств становятся более специализированы под высокопроизводительные вычисления, тем самым демонстрируя рост вычислительных мощностей. Так, параллельные алгоритмы на GPU могут достигать ускорения x100 в сравнении с аналогичными алгоритмами на CPU, а кластера на основе GPU занимают лидирующие позиции в мировом рейтинге суперкомпьютеров TOP500. Поэтому адаптация численных методов и алгоритмов под GPU имеет первостепенную важность в задачах с большим объёмом сложных вычислений. Примерами таких задач являются моделирование климата и, в частности, расчет глобальной динамики атмосферы.

В данный момент в институте ИВМ РАН им Г.И. Марчука ведется разработка новой модели атмосферы высокого разрешения на сетке кубическая сфера. Планируется, что переход на сетку с разрешением 3–5 км даст возможность воспроизводить бета-мезомасштабные метеорологические явления, что, как ожидается, повысит качество прогнозов погоды, так как значительная часть опасных погодных явлений относится к этому классу. Для практических расчётов при таком разрешении потребуется около 10^5 CPU-ядер и/или GPU-кластер. Однако для использования последнего требуется специфическая реализация численных методов. В данной работе мы представляем первый шаг на пути к этой цели — реализация алгоритма численного решения уравнений мелкой воды на сетке кубическая сфера на GPU, исследование эффективности и масштабируемости.

[1]Поддержана Отделением Московского центра фундаментальной и прикладной математики в ИВМ РАН (Соглашение с Минобрнауки России № 075-15-2025-347)

Управление технологическими параметрами прямого лазерного выращивания тонких стенок с использованием нейронной сети

М.И. Банников, П.С. Родин, А.В. Дубров

Национальный Исследовательский Центр «Курчатовский институт»

Прямое лазерное выращивание (другое название — лазерное нанесение металла) — одна из технологий аддитивного производства, которая находит своё применение для изготовления сложных тонкостенных деталей [1]. Для эффективного применения этой технологии необходимо обеспечить подбор оптимального технологического режима, который зависит от конфигурации детали и может различаться для различных слоёв [2]. Модели машинного обучения (МО) могут использоваться для предсказания оптимальных параметров процесса и получения изделий с требуемыми свойствами [3]. Цель данной работы — разработка модели машинного обучения для онлайн-коррекции параметров процесса, обеспечивающей постоянную геометрию выращенных слоёв тонких стенок.

Для обучения моделей МО был подготовлен датасет с экспериментальными данными, включающий информацию о 37 стенках, выращенных по технологии лазерного нанесения материала с использованием проволоки. Во время выращивания стенок варьировались мощность лазера, количество слоёв в стенке, а также скорость сканирования и подачи проволоки. Для сравнения эффективности предсказания были выбраны следующие модели МО: градиентный бустинг (Catboost), полносвязная нейронная сеть и рекуррентная нейронная сеть с блоком долговременной краткосрочной памяти (LSTM). Архитектурой моделей предполагалось предсказание оптимальных параметров процесса для выращивания слоёв с требуемой высотой слоя и её соотношением с шириной. Таким образом, на вход подавались характеристики слоёв — высота, ширина и стандартное отклонение высоты, а выходом являлись мощность лазера, скорости сканирования и подачи материала. Для обучения модели рекуррентной нейронной сети датасет был модифицирован таким образом, чтобы при предсказании параметров процесса для текущего слоя, на вход подавалась также история с параметрами предыдущих слоёв. Были исследованы зависимости качества предсказания от количества слоёв в истории, порядкового номера предсказываемого слоя в стенке, а также гиперпараметров моделей. Использованные метрики качества предсказания представлены в таблице 1. Обнаружено, что среди трёх исследуемых моделей наивысшую точность предсказания демонстрирует LSTM. Дальнейшие исследования будут направлены на интеграцию модели МО для контроля геометрии слоёв тонких стенок в реальном времени.

Таблица 1: Суммарные ошибки предсказания трёх параметров процесса для исследуемых моделей

Модель МО	MSE	MAE	MAPE
LSTM	1.82	0.63	0.04
FCNN	13.24	1.57	0.17
Catboost	15.81	2	0.17

Список литературы

[1] Herzog T. et al. Process monitoring and machine learning for defect detection in laser-based metal additive manufacturing // Journal of Intelligent Manufacturing. – 2024. – Vol. 35. – №. 4. – P. 1407–1437.

[2] Cai Y. et al. Monitoring process stability in robotic wire-laser directed energy deposition based on multi-modal deep learning // Journal of Manufacturing Processes. – 2024. – Vol. 128. – P. 111–124.

[3] Paulus P. et al. Prediction of single track clad quality in laser metal deposition using dissimilar materials: Comparison of machine learning-based approaches // Journal of Laser Applications. – 2023. – Vol. 35. – № 4.

Учебно-методические комплексы для подготовки ИТ-специалистов в области суперкомпьютерных и квантовых вычислений

С.В. Борзунов, С.Д. Кургалин, А.В. Романов

ФГБОУ ВО "Воронежский государственный университет"

Подготовка специалистов в области интенсивно развивающихся направлений информационных технологий, а именно, с использованием перспективных платформ вычислений, формирует перед современной высшей школой новые вызовы [1]. Как известно, значительная сложность актуальных задач математического и компьютерного моделирования приводит к необходимости использования предельных в отношении производительности вычислительных систем. До самого последнего времени такие компьютерные системы строились на основе кластерных и суперкомпьютерных архитектур. Известные особенности программирования суперкомпьютеров – нетривиальные алгоритмы распараллеливания и архитектура, как правило, принципиально отличная от аппаратного обеспечения обычных рабочих станций, формируют особую значимость для успешного учебного процесса современной учебно-методической базы и уровня подготовки профессорско-преподавательского состава.

Новый этап развития информационных технологий наступил с открытием возможности вычислений общего назначения на основе законов квантовой теории, а именно, квантовых вычислений.

В учебные планы факультета компьютерных наук Воронежского государственного университета (ВГУ) в настоящее время включены следующие дисциплины: квантовые информационные системы, квантовые вычисления, квантовая теория информации, языки программирования квантовых компьютеров, линейная алгебра в квантовых вычислениях, алгоритмы коррекции ошибок. Успешное изучение этих дисциплин требует от обучающихся предварительного освоения базовых физико-математических курсов.

В течение многих лет на кафедре цифровых технологий ВГУ создаются новые учебно-методические комплексы и по суперкомпьютерным вычислениям, и по квантовым вычислениям, ориентированные на практику и на решение реальных задач. Методика преподавания дисциплин комплексов, их содержание и структура отрабатывались в течение длительного времени. Основная часть учебной литературы этих комплексов [2–8] представлена на рис. 1. Так, учебное пособие [2], входящее в учебно-методический комплекс по суперкомпьютерным вычислениям, за несколько лет, прошедших с момента его выпуска, показало свою востребованность для формирования основных теоретических представлений о способах программирования высокопроизводительных систем и, особенно, для получения практических навыков работы с полнофункциональной суперкомпьютерной системой (на базе Суперкомпьютерного центра ВГУ). Далее, в учебном пособии [5], вышедшем совсем недавно, в 2025 г., входящем в учебно-методический комплекс по квантовым вычислениям, детально

рассмотрены: квантовая модель вычислений, и для ее описания введено понятие элементарных носителей информации квантовой системы — кубитов; основные операции над кубитами; квантовые схемы; запутанные состояний; применение матрицы плотности для задач передачи и хранения информации. Большое внимание уделено важнейшим квантовым алгоритмам. Одной из ключевых особенностей пособия является значительное количество задач и упражнений широкого спектра сложности.

Рис. 1: Учебные пособия по перспективным платформам вычислений.

Учебные пособия комплексов изданы за рубежом [3, 6], где получили положительные отзывы и представлены в библиотеках ведущих мировых университетов.

Основываясь на многолетнем опыте преподавания соответствующих дисциплин в Воронежском государственном университете, следует подчеркнуть важность системного подхода для освоения обучающимися новых перспективных платформ вычислений [9].

Список литературы

[1] Садовничий В.А. Университеты как ключевой фактор в системе подготовки кадров для обеспечения технологического суверенитета России // Вестник Московского университета. Сер. 20. Педагогическое образование. 2024. Т. 22, № 1. С. 9–25.

[2] Борзунов С.В., Кургалин С.Д. Суперкомпьютерные вычисления: практический подход. СПб.: БХВ, 2019. 256 с. (Учебная литература для вузов).

[3] Kurgalin S., Borzunov S. A Practical Approach to High-Performance Computing. Springer, 2019. 206 p.

[4] Борзунов С.В., Кургалин С.Д., Флегель А.В. Практикум по параллельному программированию. СПб.: БХВ, 2017. 236 с. (Учебная литература для вузов).

[5] Борзунов С.В., Кургалин С.Д. Основы квантовых вычислений и квантовой теории информации. СПб.: БХВ-Петербург, 2025. 248 с. (Учебная литература для вузов).

[6] Kurgalin S., Borzunov S. Concise Guide to Quantum Computing: Algorithms, Exercises, and Implementations. Springer, 2021. (Series: Texts in Computer Science). 122 p.

[7] Борзунов С.В., Кургалин С.Д. Квантовые вычисления. СПб.: БХВ-Петербург, 2022. 144 с.

[8] Борзунов С.В., Кургалин С.Д. Основы квантовых вычислений (для программистов). Воронеж: Издательский дом ВГУ, 2020. 100 с.

[9] Кургалин С.Д., Борзунов С.В. Особенности подготовки специалистов в области информационных технологий // Вестник Воронежского государственного университета. Серия: Проблемы высшего образования. 2025. № 1. С. 33–37.

Формат представления данных о веществах в памяти графического ускорителя при моделировании многокомпонентных течений сплошной среды

К.В. Иванова[1,2], С.А. Краюхин[1]

Российский Федеральный Ядерный Центр «Всероссийский научно-исследовательский институт экспериментальной физики»[1],
Филиал Московского государственного университета имени М.В. Ломоносова в г. Сарове[2]

Вопрос о представлении данных о веществах в памяти вычислительного устройства актуален для лагранжево-эйлеровых (ALE) [1] газодинамических методик, в которых для выделения контактных границ применяется метод концентраций [2]. В соответствии с основами метода концентраций, для определения состояния каждого вещества в смешанной ячейке вводятся в рассмотрение концентрации, а также другие величины, характеризующие каждое вещество.

Простейшим вариантом представления концентраций в памяти вычислительного устройства является подход, когда для каждой ячейки хранятся значения концентраций всех веществ, присутствующих в задаче (далее полное представление значений концентраций). Однако существенным минусом такого подхода является то, что для каждой ячейки необходимо хранить информацию о каждом веществе, даже если оно отсутствует в ячейке. В конечном итоге в памяти хранится большое количество нулей, не используемых в расчетах. Учитывая, что количество смешанных ячеек как правило не превышает 10% от всего размера задачи, при таком подходе число нулей зачастую превалирует над объемом полезных данных.

Работа посвящена разработке формата хранения концентраций в памяти графического ускорителя, позволяющего более экономно использовать ресурсы вычислительного устройства в части оперативной памяти (далее «сжатый» формат). В качестве технологии для реализации вычислений на GPU была выбрана CUDA. Разработка формата проводилась для явной конечно-разностной методики, использующей регулярную счетную сетку [3]. При организации счета в большинстве kernel-функций использовался поточенный метод — одна нить соответствует одной ячейке или одному узлу.

При разработке формата предполагалось, что любое отклонение от регулярного доступа к памяти GPU может в той или иной степени снизить производительность, поэтому новый формат должен обеспечивать не только компактное хранение данных, но и минимальные потери в производительности при работе с новым форматом относительно полного представления.

Разработанный «сжатый» формат представлен тремя массивами:

- Массив позиций — целочисленный массив, размер которого всегда равен количеству ячеек в задаче. Данный массив используется для определения количества веществ в ячейке и для определения индекса в массивах номеров и значений.

- Массив номеров — целочисленный массив, в котором хранится идентификационный номер вещества. Размер массива зависит от количества смешанных ячеек и от числа веществ в них.
- Массив значений — массив, в котором хранятся значения концентраций веществ с номерами из соответствующего индекса массива номеров. В данной работе использовалось три массива значений: массовые и объемные концентрации, а также внутренние энергии веществ. Однако для удобства изложения будем рассматривать их как один. Концентрации в работе представлены вещественными числами двойной точности. Размер массива значений всегда совпадает с размером массива номеров.

Массив позиций рассчитывается по принципу вычисления префиксной суммы для последовательности, содержащей число веществ в каждой ячейке. Поэтому, если в процессе счета появляется смешанная ячейка, массив позиций необходимо пересчитать. Для исключения необходимости пересчета массива полностью было решено под массивы номеров и значений выделять памяти больше, чем требуется, и данный запас распределять между блоками нитей. Таким образом у каждого блока в массивах номеров и значений есть свой резерв для случаев, когда в ячейке появляется новое (для данной ячейки) вещество.

На рисунке 1 продемонстрирован вариант заполнения массивов и связь между ними на примере получения информации о веществах одной из ячеек счетной области.

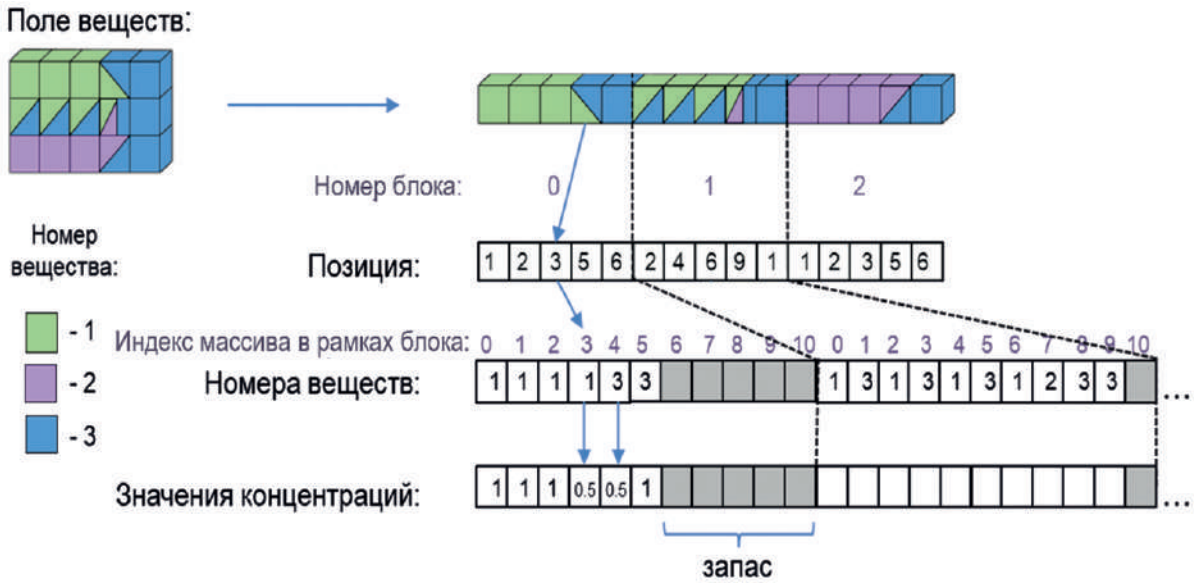


Рис. 1: Пример заполнения массивов и связь между ними.

В таблице 1 представлено сравнение объема используемой для хранения значений концентраций памяти для полного и «сжатого» формата. Преимуществом «сжатого» формата является то, что объем памяти не зависит от количества веществ в задаче. Принимая во внимание тот факт, что количество смешанных ячеек в задаче, как правило, не превышает 10% от всего размера задачи, можно сделать вывод о том, что в сравнении с полным представлением разработанный формат позволяет сократить от 13 до 48% памяти, необходимой для хранения значений концентраций, в случае от трех до пяти веществ в задаче.

Таблица 1: Зависимость объема памяти, требуемой для размещения значений концентраций, от числа веществ в задаче

Число веществ	Объем памяти, байт	
	Полное представление	«Сжатый» формат
1	$24 \cdot N_{\text{cell}}$	$60 \cdot N_{\text{cell}} + 28 \cdot N_{\text{mixed\_cell}}$
2	$48 \cdot N_{\text{cell}}$	
3	$72 \cdot N_{\text{cell}}$	
4	$96 \cdot N_{\text{cell}}$	
5	$120 \cdot N_{\text{cell}}$	

Формат обеспечивает независимость данных между разными блоками, исключает необходимость использования атомарных операций и прочих механизмов, оказывающих существенное негативное влияние на производительность. Расположение данных в массивах удовлетворяет требованиям выполнения слияния доступа при обращении к памяти GPU. Однако работа с форматом предполагает, что для получения информации требуется обращение к нескольким массивам. Это порождает нерегулярный доступ в память и оказывает негативное влияние на производительность. Сравнение двух вариантов представления данных в памяти GPU на примере тестовых расчетов показало замедление (до 5,1%) при использовании «сжатого» формата, что по мнению авторов является приемлемым.

Список литературы

[1] Hirt C.W., Amsden A.A., Cook J.L. An Arbitrary Lagrangian-Eulerian Computing Method for All Flow Speeds // Journal of Computational Physics, Vol. 14, No. 3, 1974. P. 227–253.

[2] Бахрах С.М., Глаголева Ю.Г., Самигулин М.С. и др. Расчет газодинамических течений на основе метода концентраций // ДАН. 1981. Т. 257, № 3. С. 566–569.

[3] Бахрах С.М., Величко С.В., Спиридонов В.Ф. и др. Методика ЛЭГАК-3D расчета трехмерных нестационарных течений многокомпонентной сплошной среды и принципы ее реализации на многопроцессорных ЭВМ с распределенной памятью // ВАНТ. Сер. Математическое моделирование физических процессов. 2004. Вып. 4. С. 41–50.

Содержание

Полные и короткие статьи

Аннотации стендовых докладов

Научное издание

СУПЕРКОМПЬЮТЕРНЫЕ ДНИ В РОССИИ
Труды международной конференции
29–30 сентября 2025 г. Москва

Издательство «МАКС Пресс»
Главный редактор: *Е. М. Бугачева*
Компьютерная верстка: *К. Е. Панкратьев*
Обложка: *А. В. Кононова*

Подписано в печать 19.09.2025 г. Формат 60x90 1/8.
Усл.печ.л. 33,5. Тираж 20 экз. Изд. № 120.

Издательство ООО “МАКС Пресс”. Лицензия ИД N 00510 от 01.12.99 г.
119992, ГСП-2, Москва, Ленинские горы, МГУ им. М.В. Ломоносова,
2-й учебный корпус, 527 к. Тел. 8(495) 939-3890/91. Тел./Факс 8(495) 939-3891.

Отпечатано в полном соответствии с качеством
предоставленных материалов в ООО «Фотоэксперт»
109316, г. Москва, Волгоградский проспект, д. 42,
корп. 5, эт. 1, пом. I, ком. 6.3-23Н

9 785317 074517

СУПЕРКОМПЬЮТЕРНЫЕ ДНИ В РОССИИ

Труды международной конференции

29–30 сентября 2025 г., Москва

Данный сборник содержит полные статьи на русском языке, короткие статьи и аннотации стендовых докладов, включенных в программу Международной конференции «Суперкомпьютерные дни в России».

Proceedings of the International Conference

September 29–30, 2025, Moscow

RUSSIAN SUPERCOMPUTING DAYS

Proceedings of the International Conference

September 29–30, 2025, Moscow, Russia

МАКС ПРЕСС