Научная статья

УДК 004.021, 004.032.24, 004.051, 004.272
DOI 10.25205/1818-7900-2025-23-1-5-18

# Разработка программных систем автоматизированного проектирования и исполнения программ для эффективной реализации численных алгоритмов на основе концепции *Q*-детерминанта

**Валентина Николаевна Алеева**

Южно-Уральский государственный университет (НИУ)
Челябинск, Россия

allevavn@susu.ru, https://orcid.org/0000-0002-4717-0045

*Аннотация*

Статья посвящена решению актуальной проблемы разработки программных систем автоматизированного проектирования и исполнения программ для эффективной реализации численных алгоритмов. Теоретической основой описанного исследования является концепция *Q*-детерминанта, которая предлагает унифицированное представление численных алгоритмов в форме *Q*-детерминантов. *Q*-детерминант дает возможность выразить и оценить ресурс параллелизма алгоритма, а также показать способ его параллельного исполнения. В статье рассматриваются общие принципы организации программных систем автоматизированного проектирования и исполнения программ для эффективной реализации численных алгоритмов. Реализация этих принципов продемонстрирована на примерах создания конкретных программных систем. Кроме того, проведено сравнение подходов к реализации разработанных программных систем.

*Ключевые слова*

*Q*-детерминант алгоритма, *Q*-эффективная реализация алгоритма, ресурс параллелизма алгоритма, *Q*-эффективная программа, автоматизированное проектирование эффективных программ

*Благодарности*

Как выпускник математического (теперь механико-математического) факультета Новосибирского государственного университета, выражаю благодарность факультету за данные мне знания.

*Для цитирования*

*Алеева В. Н.* Разработка программных систем автоматизированного проектирования и исполнения программ для эффективной реализации численных алгоритмов на основе концепции *Q*-детерминанта // Вестник НГУ. Серия: Информационные технологии. 2025. Т. 23, № 1. С. 5–18. DOI 10.25205/1818-7900-2025-23-1-5-18

© Алеева В. Н., 2025

# Development of Software Systems for Computer-aided Design and Program Execution for the Effective Implementation of Numerical Algorithms Based on the Concept of *Q*-determinant

**Valentina N. Aleeva**

South Ural State University
Chelyabinsk, Russian Federation

allevavn@susu.ru, https://orcid.org/0000-0002-4717-0045

*Abstract*

The paper is devoted to solving the actual problem of developing software systems for computer-aided design and program execution for the effective implementation of numerical algorithms. The theoretical basis of our research is the concept of *Q*-determinant. This concept suggests a unified representation of numerical algorithms in the form of *Q*-determinants. The *Q*-determinant makes it possible to express and evaluate the parallelism resource of the algorithm, as well as show the method of its parallel execution. In the paper we consider the general principles of software systems organizing for computer-aided design and program execution for the effective implementation of numerical algorithms. We demonstrate the realization of these principles by the examples of the creation of specific software systems. In addition, we compare the approaches to the implementation of the considered software systems.

*Keywords*

*Q*-determinant of algorithm, *Q*-effective implementation of algorithm, parallelism resource of algorithm, *Q*-effective program, computer-aided design of effective programs

*Acknowledgements*

As I a graduate of the mathematical (now mechanics and mathematics) faculty of Novosibirsk State University, I would like to express my gratitude to the faculty for the knowledge given to me.

*For citation*

Aleeva V. N. Development of software systems for computer-aided design and program execution for the effective implementation of numerical algorithms based on the concept of *Q*-determinant. *Vestnik NSU. Series: Information Technologies*, 2025, vol. 23, no. 1, pp. 5–18 (in Russ.) DOI 10.25205/1818-7900-2025-23-1-5-18

## Введение

Одним из подходов к распараллеливанию численных алгоритмов является концепция *Q*-детерминанта. Концепция и ее применение для решения проблемы эффективной реализации численных алгоритмов, численных методов и алгоритмических проблем на параллельных вычислительных системах (ПВС) описаны в работах [1–9]. Эффективные программы, разработанные с помощью концепции *Q*-детерминанта, называются *Q*-эффективными. Они выполняют *Q*-эффективные реализации алгоритмов, полностью использующие ресурсы параллелизма алгоритмов.

Для создания *Q*-эффективных программ был специально разработан метод проектирования [4]. Его может применять любой разработчик при проектировании для любого численного алгоритма программы, использующей ресурс параллелизма алгоритма полностью. Уметь создавать эффективные программы очень важно, но не менее важно иметь возможность автоматизированного проектирования эффективных программ.

Цель данного исследования заключается в том, чтобы показать применение концепции *Q*-детерминанта для создания единых для численных алгоритмов программных систем автоматизированного проектирования и исполнения *Q*-эффективных программ, при реализации которых могут использоваться различные подходы. Для достижения цели решаются следующие задачи.

1. Разработка общих принципов организации программных систем.
2. Сравнение подходов к реализации разработанных программных систем.

## Обзор работ по теме исследования

Одним из наиболее развитых направлений исследований в области параллельных вычислений является направление, созданное В. В. Воеводиным. Это направление представляет известная научная школа Вл. В. Воеводина. Именно оно наиболее близко к направлению, представленному в данной статье. Из работ, связанных с направлением, созданным В. В. Воеводиным, отметим работу [10], где проводится очень важное и развитое исследование параллельной структуры алгоритмов и программ для их реализации на ПВС. Исследование параллельной структуры алгоритмов основано на описании и изучении информационных графов алгоритмов. Оно применяется в открытой энциклопедии AlgoWiki [11]. Этот проект сформировался под руководством Вл. В. Воеводина и Дж. Донгарры (Jack Dongarra). В энциклопедии описываются и изучаются графы конкретных алгоритмов. Это позволяет исследовать ресурсы параллелизма алгоритмов и реализовывать их эффективно. Однако в работах данного направления исследований программное обеспечение для автоматизированного исследования и использования ресурсов параллелизма алгоритмов не рассматривается.

В сфере высокопроизводительных вычислений одним из ведущих ученых мира является Джек Донгарра. В университете Теннесси для поддержки библиотеки матрично-векторных операций DPLASMA [12] научным коллективом с участием Дж. Донгарры развивается подход к конструированию и исполнению параллельных программ на основе машинно-независимого представления прикладного алгоритма в виде бесконтурного ориентированного графа. Это позволяет обеспечивать высокопроизводительную реализацию прикладных алгоритмов благодаря планированию вычислений и динамической поддержке исполнительной системы PaRSEC [13]. Дж. Донгарра занимается также вопросами автоматического распараллеливания алгоритмов [14].

Еще один подход к эффективной реализации алгоритмов на ПВС разрабатывается в Стэндфордском университете под руководством А. Айкена (A. Aiken). Подход заключается в том, что отдельно описываются прикладной алгоритм и способ его отображения на вычислительные ресурсы ПВС. Научным коллективом, разрабатывающим подход, создана система Legion [15], реализующая этот подход. Недостатком системы является отсутствие автоматизации в конструировании способа отображения прикладного алгоритма на вычислительные ресурсы.

Существуют и другие подходы по теме исследования, однако на практике ресурс параллелизма алгоритмов при реализации на ПВС часто используется не полностью. Можно сделать вывод, что в настоящее время нет широко применяемых решений для эффективной реализации численных алгоритмов, тем более в автоматизированном режиме. По мнению автора, перспективным для решения проблемы автоматизированного проектирования и исполнения эффективных программ является подход на основе унифицированного представлении алгоритма, показывающего ресурс параллелизма в полной мере. Например, подход, использующий концепцию *Q*-детерминанта.

## Концепция *Q*-детерминанта

Теоретической основой данного исследования является концепция *Q*-детерминанта. Наиболее полно она изложена в [6]. Приведем основные понятия, необходимые для понимания исследования.

Пусть $\alpha$ – численный алгоритм для решения алгоритмической проблемы $\vec{y} = F\left(N, B\right)$, где $N = \left\{n_1, \cdots, n_k\right\}$ – множество параметров размерности проблемы, $k \geq 1$, а $n_i$ равно любому натуральному числу для каждого $i \in \left\{1, \cdots, k\right\}$, либо $N = \varnothing$, $B$ – множество входных данных, $\vec{y} = (y_1, \cdots, y_m)$ – множество выходных данных, $y_i \notin B$ для каждого $i \in \left\{1, \cdots, m\right\}$. Целое число $m$ является либо константой, либо значением вычислимой параметрической функции от $N$ при условии, что $N \neq \varnothing$.

Пусть множество $Q$ состоит из унарных и бинарных операций. Предположим, что алгоритм использует операции из множества $Q$. Вводятся понятия выражений над $B$ и $Q$, их уровней вложенности и уровней вложенности операций. Выражения являются термами в стандартном смысле математической логики для обычной сигнатуры поля действительных чисел с пополнением операциями из множества $Q$.

Если $N=\{n_1,\cdots,n_k\}$, то через $\overline{N}=\{\overline{n}_1,\cdots,\overline{n}_k\}$ обозначим кортеж, где $\overline{n}_i$ – некоторое заданное значение параметра $n_i$ для каждого $i\in\{1,\cdots,k\}$, а через $\{\overline{N}\}$ множество всех кортежей $\overline{N}$. Определим понятия $Q$-термов и их значений.

**Определение 1.** Если $N=\varnothing$, то любое выражение $w$ над $B$ и $Q$ называется *безусловным Q-термом*. Пусть $N\neq\varnothing$ и $V$ – множество всех выражений над $B$ и $Q$. Предположим, что есть отображение $w:\overline{N}\to V\cup\varnothing$. Тогда отображение называется *безусловным Q-термом*.

Таким образом, для $N=\varnothing$ понятие безусловного $Q$-терма и выражения над $B$ и $Q$ совпадают. Если $N\neq\varnothing$, то для каждого $\overline{N}\in\{\overline{N}\}$ $w(\overline{N})$ либо некоторое выражение над $B$ и $Q$, либо $w\left(\overline{N}\right)=\varnothing$, что значит, что $w(\overline{N})$ не определено.

**Определение 2.** Если $N=\varnothing$, то нахождение значения *безусловного Q-терма w при любой интерпретации переменных B* равносильно нахождению значения выражения $w$. Если $N\neq\varnothing$ и $w(\overline{N})\neq\varnothing$, то $w(\overline{N})$ является выражением над $B$ и $Q$. Мы можем найти значение выражения $w(\overline{N})$. Конечно, мы опускаем значение $w\left(\overline{N}\right)=\varnothing$. Следовательно, нахождение значения *безусловного Q-терма w при любой интерпретации переменных B определено*.

**Определение 3.** Пусть $N=\varnothing$ и $w$ – безусловный $Q$-терм. Предположим, что выражение $w$ над $B$ и $Q$ имеет значение логического типа при любой интерпретации переменных $B$. В этом случае $Q$-терм называется *безусловным логическим Q-термом*. Пусть $N\neq\varnothing$ и $w$ – безусловный $Q$-терм. Если выражение $w(\overline{N})$ для любого $\overline{N}\in\{\overline{N}\}$ имеет значение логического типа при любой интерпретации переменных $B$, то безусловный $Q$-терм $w$ называется *безусловным логическим Q-термом*.

**Определение 4.** Пусть $u_1,\cdots,u_l$ – безусловные логические $Q$-термы, $w_1,\cdots,w_l$ – безусловные $Q$-термы. Введем обозначение $(\hat{u},\hat{w})=\{(u_i,w_i)\}_{i\in\{1,\cdots,l\}}$ и будем называть его *условным Q-термом длины l*.

Опишем нахождение значения условного $Q$-терма $(\hat{u},\hat{w})$ при интерпретации переменных $B$.

**Определение 5.** Пусть $N=\varnothing$. Находим значения выражений $u_i$, $w_i$ для $i\in\{1,\cdots,l\}$. При таком нахождении значений можно найти пару $u_{i_0}$, $w_{i_0}$ такую, что $u_{i_0}$ имеет значение *true*. Следовательно, можно найти значение $w_{i_0}$. После этого считаем, что $(\hat{u},\hat{w})$ имеет значение $w_{i_0}$. В противном случае считаем, что значение $(\hat{u},\hat{w})$ при данной интерпретации переменных $B$ не определено. В случае $N\neq\varnothing$ и $\overline{N}\in\{\overline{N}\}$ нахождение значения условного $Q$-терма выполняется аналогично.

**Определение 6.** Пусть $(\hat{u},\hat{w})=\{(u_i,w_i)\}_{i\in\{1,2,\cdots\}}$ – счетное множество пар безусловных $Q$-термов. Предположим, что $(\hat{u},\hat{w})=\{(u_i,w_i)\}_{i\in\{1,\cdots,l\}}$ является условным $Q$-термом длины $l$ для любого $l<\infty$. Тогда мы называем $(\hat{u},\hat{w})$ *условным бесконечным Q-термом*.

Опишем нахождение значения условного бесконечного $Q$-терма $(\hat{u},\hat{w})$ при интерпретации переменных $B$.

**Определение 7.** Пусть $N=\varnothing$. Находим значения выражений $u_i$, $w_i$ для $i\in\{1,2,\cdots\}$. В результате можно найти пару $u_{i_0}$, $w_{i_0}$ такую, что $u_{i_0}$ имеет значение *true*. Следовательно, можно найти значение $w_{i_0}$. После этого считаем, что $(\hat{u},\hat{w})$ имеет значение $w_{i_0}$. В противном случае считаем, что значение $(\hat{u},\hat{w})$ при данной интерпретации переменных $B$ не определено. В случае $N\neq\varnothing$ и $\overline{N}\in\{\overline{N}\}$ нахождение значения условного бесконечного $Q$-терма выполняется аналогично.

**Замечание 1.** Если не имеет значения, является ли $Q$-терм безусловным, условным или условным бесконечным, то мы называем его *Q-термом*.

**Определение 8.** Предположим, что алгоритм $\alpha$ состоит в том, что для каждого $i \in M$ где $M = \{1, ..., m\}$, нужно найти $y_i$, как значение $Q$-терма $f_i$. Тогда множество $Q$-термов $\{f_i \mid i \in M\}$ называется *Q-детерминантом алгоритма* $\alpha$, а система уравнений $y_i = f_i$ для всех $i \in M$ *представлением алгоритма* $\alpha$ *в форме Q-детерминанта*.

Пусть $U$, $C$ и $I$ образуют разбиение множества $M = \{1, ..., m\}$, удовлетворяющее условиям:

1. $U \cup C \cup I = M$;
2. $U \cap C = U \cap I = C \cap I = \varnothing$;
3. Одно или два подмножества $U$, $C$ и $I$ могут быть пустыми.

При этом $U$, $C$ и $I$ связаны с подмножествами множества $Q$-термов $\{f_i\}_{i \in M}$ следующим образом:

1. Для каждого $i \in U$ есть $Q$-терм $f_i$, который является безусловным, и $f_i = w^i$;
2. Для каждого $i \in C$ есть $Q$-терм $f_i$, который является условным, и $f_i = \left\{\left(u_j^i, w_j^i\right)\right\}_{j \in \{1,\cdots,l(i)\}}$,

где $l(i)$ – либо константа, либо значение вычислимой функции от $N$, если $N \neq \varnothing$;

3. Для каждого $i \in I$ есть $Q$-терм $f_i$, являющийся условным бесконечным, и $f_i = \left\{\left(u_j^i, w_j^i\right)\right\}_{j \in \{1,2,\cdots\}}$.

**Определение 9.** Пусть алгоритм $\alpha$ представлен в форме $Q$-детерминанта $y_i = f_i$ для всех $i \in M$. Процесс вычисления $Q$-термов $f_i$ для всех $i \in M$ называется *реализацией алгоритма* $\alpha$. Если реализация алгоритма $\alpha$ такова, что две или более операций выполняются одновременно, то она называется *параллельной реализацией*.

**Определение 10.** Реализация алгоритма называется $Q$-эффективной, если выражения

$$W\left(\overline{N}\right) = \left\{w^i\left(\overline{N}\right)\left(i \in U\right); u_j^i\left(\overline{N}\right), w_j^i\left(\overline{N}\right)\left(i \in C, j \in \left\{1,\cdots,l\left(i\right)\right\}\right); u_j^i\left(\overline{N}\right), w_j^i\left(\overline{N}\right)\right.$$
$$\left.\left(i \in I, j \in \left\{1,2,\cdots\right\}\right)\right\}$$

вычисляются одновременно, при этом их операции выполняются по мере вычисления операндов.

При вычислении выражений выполняются следующие условия. Если для любых $i \in C \cup I$ и $i \in \{1,2,\cdots\}$ выражение $u_j^i\left(\overline{N}\right)$ имеет значение *false*, то вычисление соответствующего выражения $w_j^i\left(\overline{N}\right)$ завершается. Если для любых $i \in C \cup I$ и $i \in \{1,2,\cdots\}$ вычисление некоторой пары выражений $\left(u_j^i\left(\overline{N}\right), w_j^i\left(\overline{N}\right)\right)$ приводит к тому, что значение одного из двух выражений не определено, то вычисление другого выражения прекращается. Если для любого $i \in C \cup I$ вычисление некоторой пары выражений $\left(u_{j_0}^i\left(\overline{N}\right), w_{j_0}^i\left(\overline{N}\right)\right)$ приводит к нахождению их значений и $u_{j_0}^i\left(\overline{N}\right)$ равно *true*, то вычисление выражений $\left(u_j^i\left(\overline{N}\right), w_j^i\left(\overline{N}\right)\right)$ прекращается для любого $j \neq j_0$. Если $N \neq \varnothing$, то $\overline{N}$ в выражениях $W(\overline{N})$ не используется.

**Замечание 2.** Определение $Q$-эффективной реализации показывает, что это *наиболее параллельная реализация алгоритма*. Другими словами, $Q$-эффективная реализация полностью использует ресурс параллелизма алгоритма.

## Общие принципы организации программных систем

Программные системы автоматизированного проектирования и исполнения программ для эффективной реализации численных алгоритмов, разработанные на основе концепции $Q$-детерминанта, должны включать две подсистемы, решающие следующие задачи.

1. Получение *Q*-эффективных реализаций алгоритмов с помощью их представлений в форме Q-детерминантов.

2. Выполнение *Q*-эффективных реализаций алгоритмов.

Для формирования представлений алгоритмов в форме *Q*-детерминантов может применяться подсистема для генерации *Q*-детерминантов численных алгоритмов программной *Q*-системы [8], использующая для описания алгоритмов блок-схемы в формате JSON. При этом параметры размерности $N$ имеют фиксированные значения. Опишем структуру выходного файла этой подсистемы. Для каждого $i \in C$ условный *Q*-терм $f_i$ определяет $l(i)$ строк файла, где $l(i)$ – длина *Q*-терма. Каждая из этих строк содержит идентификатор выходной переменной, вычисляемой с использованием данного *Q*-терма, знак равенства и одну пару выражений $\left(u_j^i, w_j^i\right)$, если $N \neq \varnothing$ или $\left(u_j^i(\bar{N}), w_j^i(\bar{N})\right)$, если $N \neq \varnothing$ для некоторого $j \in \{1,\cdots,l(i)\}$. Здесь для каждого $j \in \{1,\cdots,l(i)\}$ *Q*-термы таковы, что $u_j^i$ является безусловным логическим *Q*-термом, а $w_j^i$ является безусловным *Q*-термом. Выражения $\left(u_j^i, w_j^i\right)$, если $N \neq \varnothing$ или $\left(u_j^i(\bar{N}), w_j^i(\bar{N})\right)$, если $N \neq \varnothing$, описываются в формате JSON и разделяются точкой с запятой. Описание условного бесконечного *Q*-терма аналогично описанию условного *Q*-терма, поскольку длина условного бесконечного *Q*-терма ограничена значением параметра $L$, соответствующего количеству используемых итераций алгоритма. Для каждого $i \in U$ безусловный *Q*-терм $f_i$ определяет одну строку файла. Безусловные *Q*-термы также описываются в формате JSON. В этом случае нет безусловного логического *Q*-терма, поэтому вместо него используется пробел. Таким образом, выходной файл подсистемы для генерации *Q*-детерминантов численных алгоритмов программной *Q*-системы содержит представление алгоритма в форме *Q*-детерминанта при фиксированных параметрах размерности $N$ и ограниченном числе итераций $L$. Для описания выражений в формате JSON применяются следующие обозначения: op – унарная или бинарная операция, od – операнд унарной операции, fO – первый операнд бинарной операции, sO – второй операнд бинарной операции, скобки { и } определяют порядок выполнения операций.

Приведем примеры представлений алгоритмов в форме *Q*-детерминантов, полученных с помощью *Q*-системы.

**Пример 1.** На рис. 1 показано представление в форме *Q*-детерминанта алгоритма скалярного произведения векторов $\vec{A}$ и $\vec{B}$ длины 8 с использованием схемы сдваивания. *Q*-детерминант алгоритма состоит из одного безусловного *Q*-терма.

```
S= ;{"op":"+","f0":{"op":"+","f0":{"op":"+","f0":{"op":"*","f0":"A(1)","s0":"B(1)"},
                                                "s0":{"op":"*","f0":"A(2)","s0":"B(2)"}
                                    },
                                "s0":{"op":"+","f0":{"op":"*","f0":"A(3)","s0":"B(3)"},
                                                "s0":{"op":"*","f0":"A(4)","s0":"B(4)"}
                                    }
                        },
                    "s0":{"op":"+","f0":{"op":"+","f0":{"op":"*","f0":"A(5)","s0":"B(5)"},
                                                "s0":{"op":"*","f0":"A(6)","s0":"B(6)"}
                                    },
                                "s0":{"op":"+","f0":{"op":"*","f0":"A(7)","s0":"B(7)"},
                                                "s0":{"op":"*","f0":"A(8)","s0":"B(8)"}
                                    }
                        }
    }
```

*Рис. 1.* Представление в форме *Q*-детерминанта алгоритма скалярного произведения векторов длины 8
*Fig. 1.* Representation in the form of a *Q*-determinant of the algorithm for the scalar product of vectors of length 8

**Пример 2.** Предположим, что матрица $A$ размера $2 \times 2$ обратима (имеет ненулевой определитель). Пусть $\vec{X} = (X(1), X(2))^T$ и $\vec{B} = (A(1,3), A(2,3))^T$ – векторы-столбцы, $\overline{A} = \left[ A(i,j) \right]_{2\times 3}$ – расширенная матрица системы $A\vec{X} = \vec{B}$. Рис. 2 содержит представление в форме $Q$-детерминанта алгоритма решения системы линейных уравнений 2-го порядка методом Гаусса – Жордана, использующего в качестве ведущего элемента первый ненулевой элемент ведущей строки. $Q$-детерминант алгоритма состоит из двух условных $Q$-термов длины 2.

```
X(1)={"op":"!=","f0":"A(1,1)","s0":"0"};
     {"op":"-","f0":{"op":"/","f0":"A(1,3)",
                              "s0":"A(1,1)"},
               "s0":{"op":"*","f0":{"op":"/","f0":{"op":"-","f0":"A(2,3)",
                                                   "s0":{"op":"*","f0":{"op":"/","f0":"A(1,3)",
                                                                        "s0":"A(1,1)"},
                                                          "s0":"A(2,1)"}},
                                          "s0":{"op":"-","f0":"A(2,2)",
                                                   "s0":{"op":"*","f0":{"op":"/","f0":"A(1,2)",
                                                                        "s0":"A(1,1)"},
                                                          "s0":"A(2,1)"}}},
                              "s0":{"op":"/","f0":"A(1,2)",
                                          "s0":"A(1,1)"}}}
X(2)={"op":"!=","f0":"A(1,1)","s0":"0"};
     {"op":"/","f0":{"op":"-","f0":"A(2,3)",
                              "s0":{"op":"*","f0":{"op":"/","f0":"A(1,3)",
                                                   "s0":"A(1,1)"},
                                          "s0":"A(2,1)"}},
               "s0":{"op":"-","f0":"A(2,2)",
                              "s0":{"op":"*","f0":{"op":"/","f0":"A(1,2)",
                                                   "s0":"A(1,1)"},
                                          "s0":"A(2,1)"}}}
X(1)={"op":"==","f0":"A(1,1)","s0":"0"};
     {"op":"/","f0":{"op":"-","f0":"A(2,3)",
                              "s0":{"op":"*","f0":{"op":"/","f0":"A(1,3)",
                                                   "s0":"A(1,2)"},
                                          "s0":"A(2,2)"}},
               "s0":{"op":"-","f0":"A(2,1)",
                              "s0":{"op":"*","f0":{"op":"/","f0":"A(1,1)",
                                                   "s0":"A(1,2)"},
                                          "s0":"A(2,2)"}}}
X(2)={"op":"==","f0":"A(1,1)","s0":"0"};
     {"op":"-","f0":{"op":"/","f0":"A(1,3)",
                              "s0":"A(1,2)"},
               "s0":{"op":"*","f0":{"op":"/","f0":{"op":"-","f0":"A(2,3)",
                                                   "s0":{"op":"*","f0":{"op":"/","f0":"A(1,3)",
                                                                        "s0":"A(1,2)"},
                                                          "s0":"A(2,2)"}},
                                          "s0":{"op":"-","f0":"A(2,1)",
                                                   "s0":{"op":"*","f0":{"op":"/","f0":"A(1,1)",
                                                                        "s0":"A(1,2)"},
                                                          "s0":"A(2,2)"}}},
                              "s0":{"op":"/","f0":"A(1,1)",
                                          "s0":"A(1,2)"}}}
```

*Рис. 2.* Представление в форме $Q$-детерминанта алгоритма решения системы линейных уравнений второго порядка методом Гаусса – Жордана

*Fig. 2.* Representation in the form of a $Q$-determinant of the algorithm for solving a system of linear equations of the second order using the Gauss – Jordan method

На рис. 1 и рис. 2 строки файлов, содержащих представления алгоритмов в форме $Q$-детерминантов, размещены так, чтобы показать уровни вложенности операций.

## Сравнение подходов к реализации разработанных программных систем

Рассмотрим следующие программные системы автоматизированного проектирования и исполнения программ для эффективной реализации численных алгоритмов, разработанные в настоящее время:

1) на основе компилятора и виртуальной машины;
2) лексического анализатора.

В их разработку внесли вклад студенты кафедры системного программирования Южно-Уральского государственного университета А. В. Юферов и П. А. Манатин.

Программная система на основе компилятора и виртуальной машины описана в работе [16], а на основе лексического анализатора в работе [17], кроме того, о ней сделан доклад на научной конференции «Параллельные вычислительные технологии (ПаВТ) 2024». Проанализируем и сравним основные решения по реализации систем.

В состав программной системы на основе компилятора и виртуальной машины входят:

1) компилятор – преобразователь представления алгоритма в форме *Q*-детерминанта в код программы для выполнения *Q*-эффективной реализации алгоритма;
2) виртуальная машина – исполнитель создаваемых компилятором программ.

Аналогично в состав программной системы на основе лексического анализатора входят:

1) программа для формирования представления *Q*-эффективной реализации численного алгоритма на базе лексического анализатора – далее лексический анализатор;
2) программа для выполнения представления *Q*-эффективной реализации численного алгоритма.

Таким образом, обе программные системы содержат две необходимые для таких систем подсистемы. При этом они применяют представления алгоритмов в форме *Q*-детерминантов, полученные с помощью программной *Q*-системы. Следовательно, разработанные программные системы используют общие принципы организации программных систем автоматизированного проектирования и исполнения программ для эффективной реализации численных алгоритмов, описанные выше. Далее кратко опишем реализацию каждой из разработанных программных систем.

Диаграмма размещения компонентов программной системы на основе компилятора и виртуальной машины представлена на рис. 3.

Поясним диаграмму. Подкомпонент «Обработчик *Q*-детерминанта алгоритма» компилятора, используя *Q*-детерминант алгоритма, строит в памяти дерево алгоритма – структуру, содержащую операции *Q*-детерминанта и связи между ними, а затем подкомпонент «Генератор программ» компилятора записывает в бинарном виде метаданные алгоритма и полученную с помощью дерева алгоритма очередь команд для выполнения *Q*-эффективной реализации алгоритма в файл программы. Файл метаданных алгоритма – текстовый файл, создаваемый пользователем, который используется компилятором для информации о входных данных алгоритма, их описания, а также текстового описания самого алгоритма.

Виртуальная машина загружает выбранную пользователем программу в память, читает входные данные алгоритма из файла, содержащего входные данные, получает поток команд из программы. Подкомпонент «Диспетчер ВМ» виртуальной машины обеспечивает распределение команд по потокам исполнения, в которых работают объекты компонента «Интерпретатор байт-кода». Команды выполняются по мере их готовности к выполнению. После выполнения всех команд программы виртуальная машина записывает полученные выходные данные в файл для выходных данных. При исполнении программы оперативная память используется непосредственно, что обеспечивает высокое быстродействие. Для разработки программного обеспечения компилятора и виртуальной машины применялось объектно-ориентированное программирование.

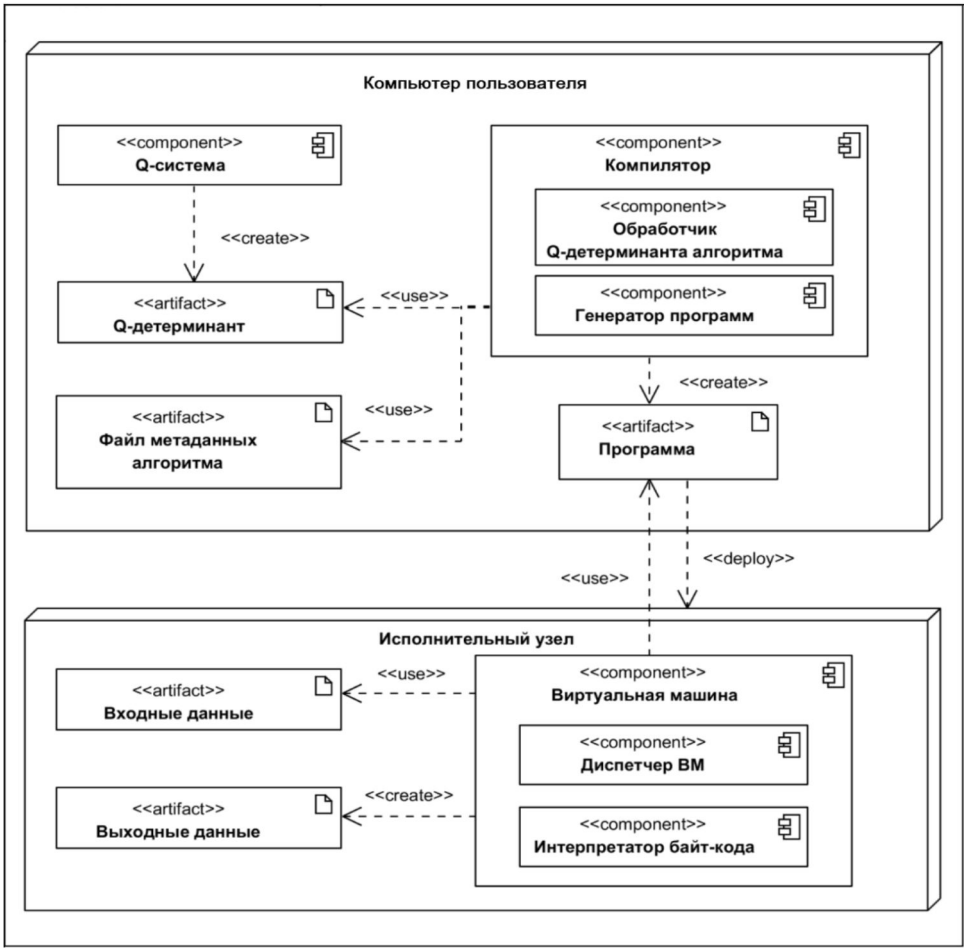


*Рис. 3.* Диаграмма размещения компонентов программной системы на основе компилятора и виртуальной машины
*Fig. 3.* Deployment diagram of the software system based on a compiler and a virtual machine

Программная система на основе компилятора и виртуальной машины при разработке была ориентирована на ПВС с общей памятью. Затем было разработано решение по ее модификации с целью возможности эксплуатации на ПВС с распределенной памятью. Это решение использует принцип MASTER-SLAVE [18], а также технологии OpenMP и MPI.

Узел MASTER распределяет задачи по узлам SLAVE и агрегирует результаты вычислений, в то время как подчиненные узлы SLAVE непосредственно производят вычисления. Технология MPI применяется для распределения задач по вычислительным узлам. Для распределения вычислений по потокам на одном узле используется технология OpenMP.

Узел MASTER принимает на вход двоичный файл программы, созданной компилятором, и входные данные. Если $Q$-детерминант алгоритма состоит из безусловных $Q$-термов, то узел MASTER определяет, какие из них могут быть вычислены, и передает их узлам SLAVE. Оставшиеся безусловные $Q$-термы передаются на исполнение по мере готовности. Если $Q$-детерминант алгоритма состоит из условных $Q$-термов, то узел MASTER определяет, какие пары условных $Q$-термов, состоящие из безусловного логического и безусловного $Q$-термов, могут быть вычислены, и передает их узлам SLAVE. Оставшиеся пары $Q$-термов передаются на исполнение по мере готовности. $Q$-термы из пары вычисляются параллельно, при этом, если логический $Q$-терм принимает значение *false*, вычисление второго $Q$-терма из пары прекращается, а на узел MASTER передается соответствующее сообщение.

Рис. 4 содержит диаграмму размещения компонентов программной системы на основе лексического анализатора. Дадим пояснения к диаграмме. Лексический анализатор выполняет считывание описания представления алгоритма в форме *Q*-детерминанта из файла формата JSON, полученного с помощью *Q*-системы, формирование представления *Q*-эффективной реализации алгоритма, вывод полученных результатов в виде директорий с файлами по определенным описанным далее правилам. В этих файлах описана булева функция, приведенная к конъюнктивной форме записи.

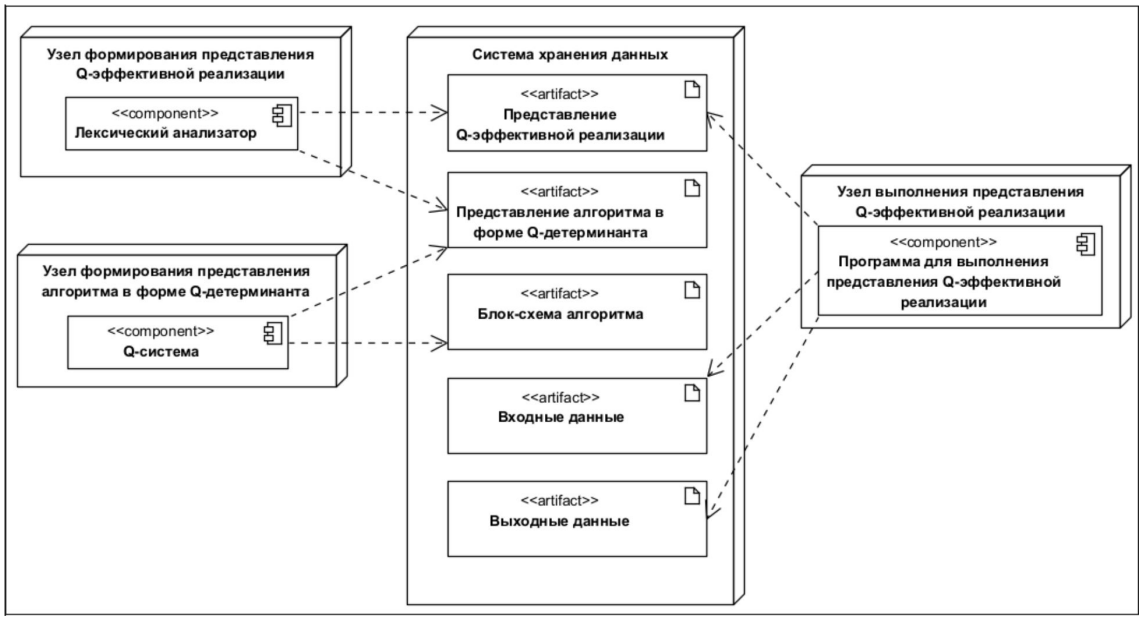


*Рис. 4.* Диаграмма размещения компонентов программной системы на основе лексического анализатора
*Fig. 4.* Deployment diagram of the software system based on a lexical analyzer

Лексический анализатор спроектирован в соответствии с парадигмой объектно-ориентированного программирования. Были созданы классы QString и QTerm. Принцип работы лексического анализатора заключается в следующем. В конструкторе класса QString исходная строка представления алгоритма в форме *Q*-детерминанта разбивается на две части: строку, содержащую логический *Q*-терм (при его наличии), и строку, содержащую безусловный *Q*-терм. Эти строки передаются конструктору класса QTerm для лексического анализа. Анализ происходит до тех пор, пока вложенные объекты QTerm не будут представлять собой выражения, содержащие одну операцию сравнения или одну арифметическую операцию. Каждый такой объект по окончании инициализации себя и вложенных объектов добавляет своему предку в словарь операций и множество данных значения своих соответствующих полей. Таким образом, корневой QTerm содержит результат лексического анализа соответствующего объекта QString.

После лексического анализа строк лексический анализатор создает представление *Q*-эффективной реализации, группируя полученные объекты QString по некоторым правилам, чтобы иметь возможность разделить исходную задачу на несколько подзадач, которые можно выполнять на системах с распределенной памятью, а также использовать для распараллеливания задачи при выполнении на системах с общей памятью.

Представление *Q*-эффективной реализации алгоритма состоит из директорий. Каждая директория содержит файлы, описывающие вычисление одного из элементов выходных данных. Таким образом, количество директорий соответствует количеству элементов выходных данных. Действия над директориями могут проводиться параллельно на разных вычисли-

тельных устройствах. В таком случае можно обеспечить локальность данных при выполнении алгоритма.

Если выходная переменная вычисляется с помощью безусловного *Q*-терма, то соответствующая ей директория содержит один файл с одной строкой, которая представляет собой этот безусловный *Q*-терм. Если выходная переменная вычисляется с помощью условного *Q*-терма, то соответствующая ей директория содержит количество файлов, равное длине условного *Q*-терма. В этом случае каждый файл соответствует одной паре безусловных *Q*-термов условного *Q*-терма. Первая строка файла представляет собой второй *Q*-терм пары, а остальные строки – выражения с одной операцией сравнения, полученные лексическим анализатором из логического *Q*-терма пары. Строки каждого из этих файлов можно вычислять параллельно.

Программа для выполнения представления *Q*-эффективной реализации выполняет *Q*-эффективную реализацию алгоритма, используя ее представление, сформированное лексическим анализатором. Перед выполнением программа читает входные данные алгоритма из файла специального формата, содержащего входные данные, результат выполнения она записывает в файл для выходных данных.

Выполнение *Q*-эффективной реализации с помощью ее представления заключается в следующем. Если выходная переменная вычисляется с помощью безусловного *Q*-терма, то ее значение определяется путем вычисления безусловного *Q*-терма, содержащегося в файле, соответствующем выходной переменной. Если выходная переменная вычисляется с помощью условного *Q*-терма, то для определения ее значения вычисляются *Q*-термы и выражения, содержащиеся во всех строках файлов, соответствующих выходной переменной, и выходной переменной присваивается результат вычисления безусловного *Q*-терма файла, все остальные строки которого содержат выражения, имеющие значение *true*. Программа для выполнения представления *Q*-эффективной реализации может использоваться как для общей, так и для распределенной памяти ПВС.

Каждый из процессов построения представления *Q*-эффективной реализации и его выполнения может осуществляться на своей вычислительной системе.

Сравнивая решения, применяемые для реализации рассмотренных программных систем, приходим к выводу, что подходы к разработке систем существенно различаются, несмотря на то, что системы используют общие принципы организации программных систем автоматизированного проектирования и исполнения программ для эффективной реализации численных алгоритмов.

Следует отметить, что организация хранения данных алгоритма и представления *Q*-эффективной реализации алгоритма значительно влияет на быстродействие программной системы. Для организации хранения данных программной системы на основе компилятора и виртуальной машины используется оперативная память, а для организации хранения данных программной системы на основе лексического анализатора применяются файлы, поэтому быстродействие первой программной системы выше, чем второй. Это различие в быстродействии подтверждают экспериментальные исследования, приведенные в работах [16; 17].

Таким образом, при разных подходах к разработке программных систем быстродействие может быть разным. Кроме уже реализованных подходов, конечно, возможны и другие подходы.

## Заключение

Исследования, приведенные в данной статье, показывают, что разработка программных систем автоматизированного проектирования и исполнения программ для эффективной реализации численных алгоритмов на основе концепции *Q*-детерминанта возможна. При этом для реализации программных систем могут использоваться разные подходы, обеспечивающие разное быстродействие. Вместе с тем программные системы используют общие принципы организации.

Ценность программных систем автоматизированного проектирования и исполнения программ состоит в том, что нет необходимости в использовании труда высококвалифицированных программистов.

## Список литературы


1. **Алеева В. Н.** Анализ параллельных численных алгоритмов. Препринт № 590. Новосибирск: ВЦ СО АН СССР, 1985. 23 с.
2. **Алеева В. Н., Зотова П. С., Склезнев Д. С.** Расширение возможностей исследования ресурса параллелизма численных алгоритмов с помощью программной Q-системы // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2021. Т. 10, № 2. С. 66–81. DOI: 10.14529/cmse210205
3. **Алеева В. Н., Шатов М. Б.** Применение концепции Q-детерминанта для эффективной реализации численных алгоритмов на примере метода сопряженных градиентов для решения систем линейных уравнений // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2021. Т. 10, № 3. С. 56–71. DOI: 10.14529/cmse210304
4. **Aleeva V.** Designing a Parallel Programs on the Base of the Conception of Q-Determinant // Supercomputing. RuSCDays 2018. Communications in Computer and Information Science. 2019. Vol. 965. P. 565–577. DOI: 10.1007/978-3-030-05807-4_48
5. **Aleeva V. N.** Improving Parallel Computing Efficiency // Proceedings – 2020 Global Smart Industry Conference, GloSIC 2020. IEEE. 2020. P. 113–120. Article number 9267828. DOI: 10.1109/GloSIC50886.2020.9267828
6. **Valentina Aleeva, Rifkhat Aleev.** Investigation and Implementation of Parallelism Resources of Numerical Algorithms // ACM Transactions on Parallel Computing. 2023. Vol. 10. no. 2, Article number 8. P. 1–64. DOI: 10.1145/3583755
7. **Aleeva V. N., Aleev R. Zh.** High-Performance Computing Using Application of Q-determinant of Numerical Algorithms // Proceedings – 2018 Global Smart Industry Conference, GloSIC 2018. IEEE. 2018. 8 p. Article number 8570160. DOI: 10.1109/GloSIC.2018.8570160
8. **Aleeva V., Bogatyreva E., Skleznev A., et al.** Software Q-system for the Research of the Resource of Numerical Algorithms Parallelism // Supercomputing. RuSCDays 2019. Communications in Computer and Information Science. 2019. Vol. 1129. P. 641–652. DOI: 10.1007/978-3-030-36592-9_52
9. **Aleeva V. N., Sharabura I. S., Suleymanov D. E.** Software System for Maximal Parallelization of Algorithms on the Base of the Conception of *Q*-determinant // Parallel Computing Technologies (PaCT 2015). Lecture Notes in Computer Science. 2015. Vol. 9251. P. 3–9. DOI: 10.1007/978-3-319-21909-7_1
10. **Воеводин В. В., Воеводин Вл. В.** Параллельные вычисления. СПб.: БХВ-Петербург, 2002. 608 с.
11. **Antonov A. S., Dongarra J., Voevodin V. V.** AlgoWiki Project as an Extension of the Top500 Methodology // Supercomputing Frontiers and Innovations. 2018. Vol. 5, no. 1. P. 4–10. DOI: 10.14529/jsfi180101
12. **Bosilca G., Bouteiller A., Danalis A., et al.** Flexible Development of Dense Linear Algebra Algorithms on Massively Parallel Architectures with DPLASMA // 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum. 2011. P. 1432–1441. DOI: 10.1109/IPDPS.2011.299
13. **Bosilca G., Bouteiller A., Danalis A., et al.** PaRSEC: Exploiting Heterogeneity to Enhance Scalability // Computing in Science & Engineering. 2013. Vol. 15, no. 6. P. 36–45. DOI: 10.1109/MCSE.2013.98

14. **Balaprakash P., Dongarra J., Gamblin T., et al.** Autotuning in High-Performance Computing Applications // Proceedings of the IEEE. 2018. Vol. 106, no. 11. P. 2068–2083. DOI: 10.1109/JPROC.2018.2841200
15. **Bauer M., Treichler S., Slaughter E., and A. Aiken A.** Legion: Expressing locality and independence with logical regions // SC '12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis. 2012. P. 1–11. DOI: 10.1109/SC.2012.71
16. **Алеева В. Н.** Автоматизированное проектирование и исполнение эффективных программ для численных алгоритмов // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2023. Т. 12, № 3. С. 31–49. DOI: 10.14529/cmse230303
17. **Manatin P., Aleeva V.** Efficient Implementation of Numerical Algorithms Based on a Lexical Analyzer // Parallel Computational Technologies. PCT 2024. Communications in Computer and Information Science. 2024. Vol 2241. P. 107–121. DOI: 10.1007/978-3-031-73372-7_8
18. **Leung J. Y.-T., Zhao H.** Scheduling problems in master-slave model // Annals of Operations Research. 2008. Vol. 159. P. 215–231. DOI: 10.1007/s10479-007-0271-4

## References

1. **Aleeva V. N.** Analysis of Parallel Numerical Algorithms. Preprint no. 590. Novosibirsk, Computing Center of the Siberian Branch of the Academy of Sciences of the USSR, 1985. 23 p. (in Russian)
2. **Aleeva V. N., Zotova P. S., Skleznev D. S.** Advancement of Research for the Parallelism Resource of Numerical Algorithms with the Help of Software *Q*-system. *Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering*, 2021, vol. 10, no. 2, pp. 66–81. (in Russian) DOI: 10.14529/cmse210205
3. **Aleeva V. N., Shatov M. B.** Application of the Q-determinant Concept for Efficient Implementation of Numerical Algorithms by the Example of the Conjugate Gradient Method for Solving Systems of Linear Equations. *Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering*, 2021, vol. 10, no. 3, pp. 56–71. (in Russian) DOI: 10.14529/cmse210304
4. **Aleeva V.** Designing a Parallel Programs on the Base of the Conception of Q-Determinant. Supercomputing. RuSCDays 2018. *Communications in Computer and Information Science*, 2019, vol. 965, pp. 565–577. DOI: 10.1007/978-3-030-05807-4_48
5. **Aleeva V. N.** Improving Parallel Computing Efficiency. Proceedings – 2020 Global Smart Industry Conference, GloSIC 2020. IEEE. 2020. P. 113–120. Article number 9267828. DOI: 10.1109/GloSIC50886.2020.9267828
6. **Aleeva V., Aleev R.** Investigation and Implementation of Parallelism Resources of Numerical Algorithms. *ACM Transactions on Parallel Computing*, 2023, vol. 10, no. 2, pp. 1–64. DOI: 10.1145/3583755
7. **Aleeva V. N., Aleev R. Zh.** High-Performance Computing Using Application of Q-determinant of Numerical Algorithms. Proceedings – 2018 Global Smart Industry Conference, GloSIC 2018. IEEE. 2018. 8 p. Article number 8570160. DOI: 10.1109/GloSIC.2018.8570160
8. **Aleeva V., Bogatyreva E., Skleznev A., et al.** Software Q-system for the Research of the Resource of Numerical Algorithms Parallelism. Supercomputing. RuSCDays 2019. *Communications in Computer and Information Science*, 2019, vol. 1129, pp. 641–652. DOI: 10.1007/978-3-030-36592-9_52
9. **Aleeva V. N., Sharabura I. S., Suleymanov D. E.** Software System for Maximal Parallelization of Algorithms on the Base of the Conception of Q-determinant. Parallel Computing Technologies

(PaCT 2015). *Lecture Notes in Computer Science*, 2015, vol. 9251, pp. 3–9. DOI: 10.1007/978-3-319-21909-7_1
10. **Voevodin V. V., Voevodin Vl. V.** Parallel Computing. St.Petersburg, BHV-Petersburg, 2002. 608 p. (in Russian)
11. **Antonov A. S., Dongarra J., Voevodin V. V.** AlgoWiki Project as an Extension of the Top500 Methodology. *Supercomputing Frontiers and Innovations*, 2018, vol. 5, no. 1, pp. 4–10. DOI: 10.14529/jsfi180101
12. **Bosilca G., Bouteiller A., Danalis A., et al.** Flexible Development of Dense Linear Algebra Algorithms on Massively Parallel Architectures with DPLASMA. *2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum*, 2011. P. 14321441. DOI: 10.1109/IPDPS.2011.299
13. **Bosilca G., Bouteiller A., Danalis A., et al.** PaRSEC: Exploiting Heterogeneity to Enhance Scalability. *Computing in Science & Engineering*, 2013, vol. 15, no. 6, pp. 36–45. DOI: 10.1109/MCSE.2013.98
14. **Balaprakash P., Dongarra J., Gamblin T., et al.** Autotuning in High-Performance Computing Applications. *Proceedings of the IEEE*, 2018, vol. 106, no. 11, pp. 2068–2083. DOI: 10.1109/JPROC.2018.2841200
15. **Bauer M., Treichler S., Slaughter E., and A. Aiken A.** Legion: Expressing locality and independence with logical regions. *SC '12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. 2012. P. 1–11. DOI: 10.1109/SC.2012.71
16. **Aleeva V. N.** Computer-aided design and execution of effective programs for numerical algorithms. *Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering*. 2023. Vol. 12, no. 3. P. 31–49. (in Russian) DOI: 10.14529/cmse230303
17. **Manatin P., Aleeva V.** Efficient Implementation of Numerical Algorithms Based on a Lexical Analyzer. Parallel Computational Technologies. PCT 2024. *Communications in Computer and Information Science*, 2024, vol. 2241, pp. 107–121. DOI: 10.1007/978-3-031-73372-7_8
18. **Leung J. Y.-T., Zhao H.** Scheduling problems in master-slave mode. *Annals of Operations Research*, 2008, vol. 159, pp. 215–231. DOI: 10.1007/s10479-007-0271-4

## Информация об авторе

**Алеева Валентина Николаевна,** кандидат физико-математических наук

## Information about the Author

**Valentina N. Aleeva,** Candidate of Science in Physics and Mathematics

*Статья поступила в редакцию 27.12.2024;*
*одобрена после рецензирования 14.01.2025; принята к публикации 14.01.2024*

*The article was submitted 27.12.2024;*
*approved after reviewing 14.01.2025; accepted for publication 14.01.2025*