ISSN 0132-3474

Номер 1 Январь – Февраль 2024

ЖУРНАЛ РОССИЙСКОЙ АКАДЕМИИ НАУК

ПРОГРАММИРОВАНИЕ

300 лет Российская Академия Наук

НАУКА
1727

СОДЕРЖАНИЕ

Номер 1, 2024

CONTENTS

ПРОГРАММИРОВАНИЕ, *2024*, *№ 1*, *с. 3–14*

ТЕОРИЯ ПРОГРАММИРОВАНИЯ: ФОРМАЛЬНЫЕ МОДЕЛИ И СЕМАНТИКА

УДК 510.22

НЕЧЕТКАЯ МЕРА НА p-АДИЧЕСКИХ ШАРАХ, ЗАДАННЫХ НА ОГРАНИЧЕННОМ ЧИСЛОВОМ МНОЖЕСТВЕ

В. П. Бочарников[a],* (ORCID: 0000-0003-4398-5551),
С. В. Свешников[a] (ORCID: 0000-0001- 8924-4535)

[a]*Консалтинговая группа ИНЭКС-FT, 03011 Киев, ул. Десятинная, д. 13а, Украина*

**E-mail: bocharnikovvp@gmail.com*

Поступила в редакцию 24.04.2023 г.
После доработки 27.05.2023 г.
Принята к публикации 03.07.2023 г.

В статье рассматривается подход к построению нечеткой меры на p-адических шарах, не требующий непосредственного задания плотности меры. Доказаны соотношения, необходимые для определения данной меры произвольного подмножества ограниченного числового множества, представленного как множество p-адических шаров. Рассмотрены равномерные и неоднородные нечеткие меры. Предложен алгоритм определения нечеткой меры на p-адических шарах. Приведены примеры расчета данной меры.

Ключевые слова: нечеткая мера, p-адические шары, p-адические числа, множество, ультраметрика
DOI: 10.31857/S0132347424010011 **EDN:** HVVDTA

1. ВВЕДЕНИЕ

Для практического моделирования сложных энергетических ландшафтов, представляемых скалярными полями, был предложен эффективный подход на основе формализации расположения энергетических бассейнов [1]. Данные бассейны предопределяют иерархическую структуру рассматриваемого пространства X. В скалярном поле каждой точки пространства (как правило, $X=R^n$, где R — множество действительных чисел), ставится в соответствие скалярная величина $v(x)$: $X\to R$, которая может отражать уровень энергии в рассматриваемой точке. В условиях неопределенности значение уровня энергии соответствующего энергетического бассейна может быть описано в виде величины, пропорциональной значению нечеткой меры подмножества пространства X [2]. Исходные данные в этом случае могут быть представлены в виде суждений "S есть P" [3], где S — субъект суждения (понятие, о котором что-либо утверждается), P — предиката суждения (то, что утверждается о субъекте). Например, "Энергия (S) в рассматриваемом энергетическом бассейне возможно (*есть*) высокая (P)". Нечеткая мера $g(\cdot): 2^X \to [0,1]$, как неаддитивная функция множества, является эффективным инструментом формализации таких нечетких исходных данных [2]. Например, значение $g(A)$ может определять величину, пропорциональную относительному значению энергии в бассейне $A \subseteq X$ (как вариант, уверенность в том, что энергия "очень высокая"). В отличие от обычного представления исходных данных в виде "признак–значение", нечеткая мера позволяет учесть модальность суждения, которая относится к связке "*есть*" в суждении. В общем случае модальность (от лат. modus — мера, способ) — способ существования какого-либо объекта или протекания какого-либо процесса, или же способ понимания суждения об объекте, явлении или событии [4]. Для нашего примера могут быть рассмотрены алетические модальности "*необходимость, доверие, вероятность, правдоподобие, возможность*", которые уточняют исходное суждение и существенно повышают адекватность моделирования в условиях неопределенности.

В работе мы будем рассматривать нечеткие меры $g(\cdot)$ Сугэно [2], которые получили наибольшее распространение на практике. Хотя следует отметить, что основные результаты и выводы, полученные при дальнейшем рассмотрении, могут быть использованы и для нечетких мер другого типа (например, нечетких мер Цукомото [5]). Нечеткие меры $g(\cdot)$ Сугэно являются неаддитивными функциями множества, удовлетворяющими свойствам ограниченности $g(\varnothing)=0,\ g(X)=1$, монотонности $\forall A,B\subseteq X, A\subseteq B, g(A)\le g(B)$, непрерывности $\forall F_n \subseteq X,\ g(\lim\limits_{n\to\infty} F_n) = \lim\limits_{n\to\infty} g(F_n)$, где $\{F_n\}$ — монотонная возрастающая или убывающая последовательность подмножеств, а также λ-правилу

$$\forall A,B\subseteq X,\ A\cap B=\varnothing, g\left(A\cup B\right) =$$
$$= \frac{1}{\lambda}\cdot\left(\left(1+g\left(A\right)\cdot\lambda\right)\cdot\left(1+g\left(B\right)\cdot\lambda\right)-1\right) =$$
$$= g\left(A\right)+g\left(B\right)+\lambda\cdot g\left(A\right)\cdot g\left(B\right),$$

где $\lambda \in [-1,+\infty[$. Важнейшим свойством меры $g(\cdot)$ Сугэно является наличие функциональной зависимости алетической модальности меры от параметра нормировки λ нечеткой меры. В частности, если $\lambda=-1$, то мы имеем меры с модальностью возможности, если $\lambda \in [-1,0[$ — то рассматриваются меры правдоподобия, при $\lambda=0$ меры будут иметь модальность "вероятно", если же $\lambda>0$, то рассматриваются меры доверия, а при $\lambda \gg 0$ модальность суждения стремится к необходимости. Таким образом, учет модальности в нечеткой мере $g(\cdot)$ Сугэно позволяет уточнить исходные данные в виде суждений, повысить адекватность моделирования сложных энергетических ландшафтов, представляемых скалярными полями.

Однако проблемой для применения на практике нечетких мер является задание (определение, измерение, оценка ...) их значений. Для непрерывного случая задание нечеткой меры предполагает возможность нахождения функции плотности нечеткой меры $g(x)$: $X \to [0,1]$ в точках $x \in X \subseteq R$ [6] множества вещественных чисел со стандартной метрикой. На практике, как правило, множество X является ограниченным числовым множеством (далее будем полагать $X=I=[0,1]$). То есть предполагается, что данные точки в I можно определить и в них произвести измерение значения меры. Здесь следует отметить, как минимум, два момента. Во-первых, на практике какие-либо реальные измерения могут быть осуществлены только в точках, соответствующих полю рациональных чисел Q [7]. При этом для определения всех $x \in I \subset R$ необходимо выполнить пополнение поля рациональных чисел Q до поля действительных чисел R по евклидовой норме. Во-вторых, даже задав точки $x \in X$, определить функцию плотности меры $g(x)$ сложно, так как концептуально в теории нечеткой меры предполагается, что распределение уверенности по точкам множества I принципиально неизвестно [8] и может быть оценено только на некоторых подмножествах множества I. В этом случае предполагается, что построение нечеткой меры осуществляется по оценкам уверенности на данных подмножествах, которые иногда называют фокальными элементами. Например, может использоваться подход на основе функции меры фокальных элементов [9] $m(\cdot)$: $2^I \to [0,1]$, для которой выполняются условия

$$m(E_j) > 0, \ \sum_{j=1}^{K} m(E_j) = 1,$$

где $\{E_j \subseteq I \mid j=\overline{1,K}\}$, $\bigcup_j E_j = I$. При этом распределение уверенности внутри подмножеств фокальных элементов $E_j \subseteq I$ считается неизвестным. Полученные при этом значения меры и ее модальные свойства будут зависеть от выбранного множества $\{E_j \subseteq I \mid j=\overline{1,K}\}$.

Таким образом, существует противоречие. С одной стороны, надо задать меру на одноточечных подмножествах X, то есть определить плотность меры, а с другой стороны, это противоречит концепции нечеткой меры, которая не может быть локализована в одноточечных подмножествах в реальных условиях измерений, и надо использовать подмножества, где не уточняется распределение меры по точкам. Данное противоречие определяет актуальность приведенных исследований.

В отличие от поля действительных чисел R, поле p-адических чисел Q_p имеет строгую внутреннюю иерархическую структуру, что позволяет конструктивно описывать состояние и динамику скалярного поля в виде совокупности энергетических бассейнов [10], которые представляются образами p-адических шаров. Топологическая структура поля p-адических чисел Q_p обладает рядом свойств, которые позволяют весьма эффективно использовать образы p-адических шаров, для разрешения проблемы задания нечетких мер без необходимости определения функции плотности нечеткой меры. В то же время нахождение распределения нечеткой меры Сугэно на основе использования топологической структуры поля p-адических чисел приводит к необходимости определения нечеткой меры как функций p-адического аргумента, что требует использования алгебраической структуры поля p-адических чисел.

2. ПОСТАНОВКА ЗАДАЧИ

Учитывая результат теоремы Островского [11], существует единственная альтернатива полю действительных чисел R для области определения функции плотности нечеткой меры $g(x)$. Это поле p-адических чисел Q_p, получаемое в результате пополнения поля рациональных чисел Q по неархимедовой p-адической норме [7], удовлетворяющей условию сильного неравенства треугольника, p-адическое число $r \in Q_p$ отличное от нуля, имеет вид [12]:

$$r = \sum_{l=-m}^{+\infty} q_l \cdot p^l, \ q_l = 0,\ldots,p-1, \ q_{-m} \neq 0, \ m \in Z.$$

Бесконечная влево и конечная вправо последовательность целых чисел $q_l=0,\ldots,p-1$ вида $r = (\ldots q_l \ldots q_1 q_0 q_{-1} \ldots q_{-m})_p$ называется канонической формой p-адического числа r. p-адические числа с нормой $|r|_p \le 1$, для которых $l \ge 0$ [13] образуют кольцо

целых p-адических чисел Z_p. Их каноническая запись имеет вид бесконечной влево последовательности $r = (\ldots q_l \ldots q_1 q_0)$ целых чисел q_l. Иногда для записи канонической формы целых p-адических чисел для удобства используют бесконечную вправо последовательность целых чисел q_l в виде $r = (\ldots q_l \ldots q_1 q_0)_p$ [14], которую мы в дальнейшем будем использовать.

В работе [13] было показано, что существует непрерывное отображение вида $\theta(r)$: $Q_p \to R_+$, где R_+ — множество неотрицательных действительных чисел, вида:

$$\theta(r) = \sum_{l=m}^{+\infty} q_l \cdot p^{-l-1}, \; q_l = 0, \ldots, p-1, \; m \in Z.$$

Отображение $\theta(r)$ сюръективно, взаимно однозначно почти всюду, то есть сохраняет меру (переводит p-адическую меру Хаара в меру Лебега на полупрямой), непрерывно и гёльдерово с показателем 1 [15]. При этом непересекающиеся шары отображаются на интервалы, которые не пересекаются или имеют пересечение нулевой меры. Кроме того, образ целого p-адического числа $r \in Z_p$ принадлежит единичному интервалу вещественных чисел при отображении вида [16]:

$$\varphi(r) = p \cdot \theta(r) = \sum_{l=0}^{+\infty} q_l \cdot p^{-l}, \; q_l = 0, \ldots, p-1.$$

В работе [10] было показано, что подмножество $A \subseteq I \subset R$ ограниченного числового множества с точностью до образа p-адического предела центра p-адического шара может быть представлено в виде образа некоторого множества p-адических шаров вида $U_\varepsilon(a) = \{r \in Z_p \,|\, \rho(r,a) \le \varepsilon\}$, где $\rho(r,a)$ — обобщенная метрика Кантора [17], $a \in I$ центр шара, $\varepsilon \in R_+$ его радиус,

$$r = \sum_{l=0}^{+\infty} q_l \cdot p^l, q_l = 0, \ldots, p-1 \; —$$

целое p-адическое число. Все множество p-адических шаров в поле p-адических чисел Q_p образуют топологию со специфическими свойствами [12]. Кроме того, каждая точка шара $U_\varepsilon(a)$ является его центром, и в этом смысле их можно считать равными по важности, что в целом согласуется с концепцией теории нечеткой меры. Указанные топологические свойства Q_p, в частности соотношения между p-адическими шарами, упрощают определение множества фокальных элементов, как образов p-адических шаров для построения нечеткой меры на ограниченном числовом множестве и не требуют необходимости задания плотности нечеткой меры $g(x)$. В свою очередь, использование топологических свойств пространства p-адических чисел влечет за собой необходимость использования алгебраической структуры поля Q_p, так как нечеткие меры будут задаваться как функции p-адического аргумента [9]. В частности, как будет показано ниже, для равномерных аддитивных нечетких мер значение нечеткой меры определяется простейшей функцией сложения p-адических чисел.

Таким образом, в данном материале ставится задача определения нечеткой меры на ограниченном числовом множестве $I \subset R$ с использованием топологической и алгебраической структуры поля p-адических чисел Q_p без необходимости прямого задания плотности нечеткой меры.

3. ОБСУЖДЕНИЕ РЕЗУЛЬТАТОВ

Описание структуры ограниченного числового множества на основе p-адических шаров. В работе [10] были предложены подход и алгоритм представления произвольного подмножества $A \subseteq I$ как образа множества p-адических шаров. При этом множество I представлялось в виде иерархической структуры подмножеств, которые удовлетворяют условиям:

$$\bigcup_{q_{l+1}=0}^{p-1} E_{q_0,\ldots q_{l+1}} = E_{q_0,\ldots q_l},$$

$$E_{q_0} = I, q_l = 0, \ldots, p-1, l = 0, \ldots, +\infty;$$

$$\forall i, j = 0, \ldots, p-1, Card(E_{q_0,\ldots,q_l,i}) = Card(E_{q_0,\ldots,q_l,j});$$

$$\forall i, j = 0, \ldots, p-1, E_{q_0,\ldots q_l,i} \cap E_{q_0,\ldots q_l,j} = \varnothing.$$

На рис. 1 представлен вариант разбиения множества I на подмножества $\{E_{q_0,\ldots q_l} \,|\, q_l = 0, \ldots, p-1\}$ при $p = 3$.

Следует отметить, что в качестве числа разбиения множества I можно взять произвольное составное число s, для которого можно будет сформулировать и доказать приведенные ниже утверждения. Однако в этом случае величина $|\cdot|_s$, построенная по аналогии с p-адической нормой [12], будет являться псевдо-

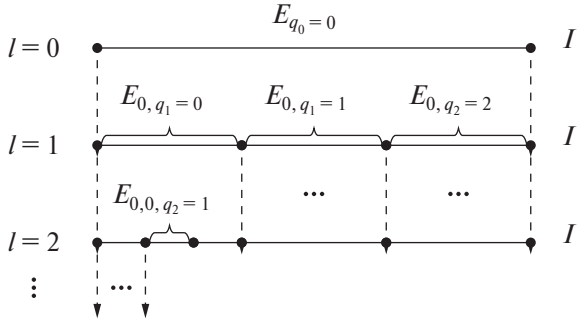


Рис. 1. Разбиение множества I при $p = 3$.

нормой. Тем не менее, как утверждается в работе [12], функция $d(x,y)=|x-y|_s$, будет являться метрикой, и можно рассмотреть пополнение поля Q по данной метрике. Данное пополнение позволяет определить кольцо Q_s, которое для составного числа s не будет являться полем. Однако в соответствии с теоремой Гензеля, доказательство которой представлено также в работе [12], если число s представляется как произведение различных простых чисел, то кольцо Q_s изоморфно прямой сумме p-адических полей $Q_s = Q_{p_1} \oplus \cdots \oplus Q_{p_k}$, где $s = p_1 \cdot \ldots \cdot p_k$. Но в этом случае последовательности, соответствующие p-адическим числам с простыми числами $p_1, ..., p_k$, не обязательно сходятся по псевдонорме $|\cdot|_s$ (если только $s \neq p$) или даже могут быть не ограничены по ней, что осложняет выполнение операций в кольце Q_s. Данный факт существенно осложнит рассмотрение дальнейшего материала. Поэтому далее для большей наглядности определения нечеткой меры на p-адических шарах мы будем рассматривать разбиение множества I для случая простого числа p.

Любое подмножество $E_{q_0,\ldots q_l}$, полученное при разбиении множества I, может быть представлено как образ p-адического шара $U_\varepsilon(r_{q_0,\ldots q_l})$ [10], где $\varepsilon = p^{-l}$ — радиус данного шара, $r_{q_0,\ldots q_l} = (q_0 \ldots q_l (p-1)\ldots)_p$ — целое p-адическое число в канонической форме, определяющее центр шара. Радиус и центр шара полностью определяются последовательностью $Seq_l(r_{q_0,\ldots q_l}) = (q_0, ..., q_l)$ индексов подмножества $E_{q_0,\ldots q_l}$. Далее последовательность $Seq_l(r)$ будем условно называть p-адической координатой шара $U_\varepsilon(r_{q_0,\ldots q_l})$. Множество всех p-адических шаров обладает такими свойствами [12], что если есть U и V два p-адических шара в I, то они либо упорядочены по включению (или $U \subset V$, или $V \subset U$), либо не пересекаются ($U \cap V = \varnothing$); каждый шар в I одновременно открыт и замкнут, а любая точка шара является его центром.

В последующих рассуждениях будем использовать ряд понятий и обозначений, приведенных в работе [10]. Пусть $U_\alpha(r_{q_0,\ldots q_{l-1}})$ — p-адический шар с радиусом $\alpha = p^{1-l}$, а $\mathcal{U}(r_{q_0,\ldots q_{l-1}}) = \{U_\varepsilon(r_{q_0,\ldots q_l}), |q_l = 0, \ldots, p-1, \varepsilon = p^{-l}\}$ — множество шаров мощности $Card(\mathcal{U}(r_{q_0,\ldots q_{l-1}})) = p$. Тогда шар $U_\alpha(r_{q_0,\ldots q_{l-1}})$ является минимальным покрывающим шаром для множества $\mathcal{U}(r_{q_0,\ldots q_{l-1}})$, а также каждого из p-адических шаров данного множества, что обозначается как $\forall U_\varepsilon(r_{q_0,\ldots q_l}), U_\alpha(r_{q_0,\ldots q_{l-1}}) = Cov_\alpha(U_\varepsilon(r_{q_0,\ldots q_l}))$. При этом шары из множества $\mathcal{U}(r_{q_0,\ldots q_{l-1}})$ являются равными по покрытию $U_\alpha(r_{q_0,\ldots q_{l-1}})$, что обозначается как $\forall i, j = 0, \ldots, p-1, U_\varepsilon(r_{q_0,\ldots q_{l-1},i}) \equiv \equiv U_\varepsilon(r_{q_0,\ldots q_{l-1},j})\{Cov|U_\alpha(r_{q_0,\ldots q_{l-1}})\}$. Шары из множества шаров $\mathcal{U}(r_{q_0,\ldots q_{l-1}})$ неотличимы с точки зрения минимального покрытия $U_\alpha(r_{q_0,\ldots q_{l-1}})$. Исходя из этого, можем предположить, что распределение уверенности на множестве равных по покрытию шаров является равномерным.

Равномерная нечеткая мера на p-адических шарах. Равномерной нечеткой мерой называется нечеткая мера, для которой $\forall x \in I$ плотность меры постоянна $g(x) = \text{const} \in [0,1]$. Для построения равномерной нечеткой меры будем учитывать несколько замечаний. Во-первых, точки $x \in I \subset R$ являются образами целых p-адических чисел и в них нет возможности определить плотность нечеткой меры $g(x)$: $I \to [0,1]$. Во-вторых, множество всех возможных p-адических шаров вида $U_\varepsilon(r_{q_0,\ldots q_l})$, заданных на I, определяет иерархическую структуру подмножеств данного множества [10]. В-третьих, будем считать, что известна модальность нечеткой меры $g(\cdot)$: $2^I \to [0,1]$, заданной на множестве I, а соответственно нам известен параметр λ данной нечеткой меры. В-четвертых, учитывая выше сказанное, будем полагать, что нечеткая мера на любом множестве шаров, равных по минимальному покрытию, имеет равномерное распределение плотности.

Для определения нечеткой меры на множестве I мы не можем задать функцию плотности меры в классическом варианте. В то же время на множестве I мы можем определить значение меры для образов всех p-адических шаров $U_\varepsilon(r_{q_0,\ldots q_l})$. Для этого воспользуемся свойством [2] ограниченности нечеткой меры $g(I) = 1$.

Утверждение 1. Нечеткая мера $g(r_{q_0,\ldots,q_l})$ подмножества $E_{q_0,\ldots q_l} \subset I$, являющегося образом p-адического шара $U_{p^{-l}}(r_{q_0,\ldots q_l})$ для равномерной нечеткой меры $g(\cdot)$: $2^I \to [0,1]$ фиксированным параметром λ, если таковая существует, определяется соотношением:

$$g(r_{q_0,\ldots,q_l}) = \frac{1}{\lambda} \cdot \left\{(1+\lambda)^{1/p^l} - 1\right\}, \quad l = 0, \ldots, +\infty.$$

Доказательство. Рассмотрим уровень $l = 1$ разбиения множества I на p подмножеств $E_{q_0,q_1}, q_1 = 0, \ldots, p-1$. В этом случае p-адический шар $U_1(1)$, образом которого является все множество I, является минимальным покрытием для множества $\mathcal{U}(r_{q_0})$ равных по данному покрытию шаров $U_{p^{-1}}(r_{q_0,q_1})$, $Card(\mathcal{U}(r_{q_0})) = p, r_{q_0} = (0,(p-1),\ldots)_p, \varphi(r_{q_0}) = 1$. По условию нормировки $g(I) = 1$ должно выполняться условие:

$$\frac{1}{\lambda} \cdot \left\{ \prod_{q_1=0}^{p-1} \left(1 + \lambda \cdot g(r_{q_0,q_1})\right) - 1 \right\} = 1,$$

где $\forall q_1, g(r_{q_0,q_1}) = \text{const} \in [0,1]$. Тогда имеем:

$$g(r_{q_0,q_1}) = \frac{1}{\lambda} \cdot \left\{ (1+\lambda)^{\frac{1}{p}} - 1 \right\}.$$

Для уровня разбиения $l = 2$ должно выполняться условие:

$$\frac{1}{\lambda} \cdot \left\{ \left(1 + \lambda \cdot g\left(r_{q_0,q_1,q_2}\right)\right)^p - 1 \right\} =$$
$$= g\left(r_{q_0,q_1}\right) = \frac{1}{\lambda} \cdot \left\{ (1+\lambda)^{\frac{1}{p}} - 1 \right\}.$$

Отсюда нечеткая мера $\forall q_2, g(r_{q_0,q_1,q_2}) = \text{const} \in [0,1]$ для образа p-адического шара $U_{p^{-2}}(r_{q_0,q_1,q_2})$ на уровне $l = 2$ будет определяться соотношением:

$$g(r_{q_0,q_1,q_2}) = \frac{1}{\lambda} \cdot \left\{ (1+\lambda)^{1/p^2} - 1 \right\}.$$

Выполняя аналогичную процедуру для произвольного $l = 0, ..., +\infty$ мера для образа p-адического шара $U_{p^{-l}}(r_{q_0,\ldots,q_l})$ будет определяться соотношением:

$$g(r_{q_0,\ldots,q_l}) = \frac{1}{\lambda} \cdot \left\{ (1+\lambda)^{1/p^l} - 1 \right\},$$

где при фиксированных $q_0, \ldots, q_{l-1}$ и $\forall q_l = 0, \ldots, p-1, g(r_{q_0,\ldots,q_l}) = \text{const} \in [0,1]$ ■.

Далее меру $g(r_{q_0,\ldots,q_l})$ будем называть мерой p-адического шара $U_{p^{-l}}(r_{q_0,\ldots q_l})$, понимая, что это мера подмножества $E_{q_0,\ldots q_l} \subseteq I$, являющегося образом данного шара.

Следствие. Мера подмножества множества I, являющегося образом p-адического шара $U_{p^{-l}}(r_{q_0,\ldots q_l})$ для равномерной меры вероятности $Pr(\cdot)$: $2^I \to [0.1]$, определяется радиусом p-адического шара $Pr(r_{q_0,\ldots,q_l}) = p^{-l}$.

Доказательство. $\forall U_{p^{1-l}}(r_{q_0,\ldots,q_{l-1}}), Card(\mathcal{U}(r_{q_0,\ldots q_{l-1}})) = p$. $Pr(I) = Pr(r_{q_0}) = 1$. Тогда для равномерной меры имеем:

$$Pr(r_{q_0,\ldots,q_l}) = \frac{1}{p} \cdot Pr(r_{q_0,\ldots,q_{l-1}}) =$$
$$= \frac{1}{p} \cdot \left\{ \frac{1}{p} \cdot Pr(r_{q_0,\ldots,q_{l-2}}) \right\} = \ldots =$$
$$= \frac{1}{p^l} \cdot Pr(r_{q_0}) = \frac{1}{p^l} \cdot 1 = p^{-l}. \blacksquare$$

Определение равномерной меры для множества p-адических шаров на любом уровне $l = 0, ..., +\infty$ полностью задает нечеткую меру на I. Однако при этом нет необходимости задавать плотность нечеткой меры, которая привязана к точкам множества I. Мера p-адического шара определяется параметром меры λ (ее модальностью) и значением параметра разбиения p. Следует отметить, что наши рассуждения исходили из того, что p является простым числом. Однако приведенные результаты допустимы и для более общего случая, когда $p \in N$, где N — множество натуральных чисел [14]. В дальнейшем для простоты изложения мы будем использовать понятие p-адического числа, понимая, что в общем случае возможно использование произвольного $p \in N$. Заметим, что значение равномерной меры для p-адического шара зависит от p^l. В этом смысле наблюдается сходство построения данной меры с построением p-адических чисел [18].

Рассмотрим порядок расчета равномерной нечеткой меры на p-адических шарах для произвольного подмножества ограниченного числового множества I. Подмножество $A \subseteq I$ с точностью до образа p-адического предела центра p-адического шара является образом множества p-адических шаров $U(A) = \{U_{\varepsilon_i}(r_i) | i = \overline{1, N_A}\}$ [10]. На практике для подмножества $A \subseteq I$ рассматривается конечное множество p-адических шаров $U_{\varepsilon_{apr}}(A) = \{U_{\varepsilon_i}(r_i) | i = \overline{1, N_A}, U_{\varepsilon_i}(r_i) \in U(A),\ \varepsilon_i \geq \varepsilon_{apr} \in [0,1]\} \subseteq U(A)$, которое аппроксимирует множество $A \subseteq I$ с точностью ε_{apr}. В этом случае p-адические шары из $U_{\varepsilon_{apr}}(A)$ минимального радиуса будут лежать на уровне $L = -\log_p \min_{i=\overline{1,N_A}} \varepsilon_i, \varepsilon_i \geq \varepsilon_{apr}$. Пусть $U_l(A) \subseteq U(A)$ включает все шары $U_{\varepsilon_i}(r_i)$, для которых $\varepsilon_i = p^{-l}$, $l = 0, \ldots, +\infty$, и пусть $Card(U_l(A)) = a_l \in N$. Тогда справедливо следующее утверждение.

Утверждение 2. Мера подмножества $A \subseteq I$ для равномерной нечеткой меры на p-адических шарах $g(\cdot)$: $2^I \to [0,1]$ определяется соотношением:

$$g(A) = \frac{1}{\lambda} \cdot \left\{ (1+\lambda)^{r^A} - 1 \right\},$$

где $r^A = \sum_{j=-L}^{+\infty} (q_j \cdot p^j)$ — результирующее p-адическое число, полученное по правилу сложения p-адических чисел r_l^A для всех уровней разложения $l \leq L$, $L \in N$, где

$$r_l^A = \sum_{j=-L}^{+\infty} (q_j \cdot p^j),\ b_j^l = b_n^l,\ j = n - l,$$

а b_n^l — коэффициенты в позиционной p-адической системе счисления для числа $a_l = Card(U_l(A))$.

Доказательство. В соответствии с [10] любое подмножество $A \subseteq I$ однозначно определяется множеством p-адических шаров $U(A)$, имеющих образом

интервалы, которые либо не пересекаются, либо имеют пересечение нулевой меры. В этом случае мера подмножества $A \subseteq I$ может быть определена соотношением:

$$g(A) = \frac{1}{\lambda} \cdot \left\{ \prod_{i=1}^{N_A} (1 + \lambda \cdot g_i) - 1 \right\},$$

где g_i — мера шара $U_{\varepsilon_i}(r_i) \in U(A)$. Для равномерной нечеткой меры справедливо соотношение:

$$g(A) = \frac{1}{\lambda} \cdot \left\{ \prod_{l=0}^{+\infty} (1 + \lambda \cdot g_l)^{a_l} - 1 \right\},$$

где g_l — мера р-адических шаров $U_{\varepsilon_i}(r_i)$ с радиусом $\varepsilon_i = p^{-l}$, $l = 0, \ldots, +\infty$, $a_l = Card(U_l(A))$. Исходя из построения равномерной нечеткой меры на р-адических шарах (Утверждение 1), имеем $(1 + \lambda \cdot g_l)^p = (1 + \lambda \cdot g_{l-1})$. Натуральное число a_l может быть представлено [18] в виде:

$$a_l = \sum_{n=0}^{+\infty} (b_n^l \cdot p^n), \; b_n^l, \; n \in N$$

или в позиционной р-адической системе счисления в виде $a_l = (b_0^l b_1^l b_2^l \ldots b_n^l \ldots)_p$.

Исходя из Утверждения 1, мера g_l определяется соотношением:

$$g_l = \frac{1}{\lambda} \cdot \left\{ (1 + \lambda)^{1/p^l} - 1 \right\}.$$

Тогда, подставив это выражение, получим:

$$(1 + \lambda \cdot g_l)^{\sum_{n=0}^{+\infty} (b_n^l \cdot p^n)} =$$

$$= \left(1 + \lambda \cdot \frac{1}{\lambda} \cdot \left\{ (1 + \lambda)^{\frac{1}{p^l}} - 1 \right\} \right)^{\sum_{n=0}^{+\infty} (b_n^l \cdot p^n)} =$$

$$= (1 + \lambda)^{p^{-l} \sum_{j=-l}^{+\infty} \left(b_n^l \cdot p^j \right)} =$$

$$= (1 + \lambda)^{\sum_{n=0}^{+\infty} (b_n^l \cdot p^{n-l})} = (1 + \lambda)^{\sum_{n=0}^{+\infty} (b_j^l \cdot p^j)},$$

где $b_j^l = b_n^l$, $j = n - l$. Ряд $\sum_{j=-l}^{+\infty} (b_j^l \cdot p^j) = r_l^A$ является р-адическим числом [7]. Тогда можем записать:

$$g(a) = \frac{1}{\lambda} \cdot \left\{ \prod_{l=0}^{+\infty} (1 + \lambda \cdot g_l)^{a_l} - 1 \right\} =$$

$$= \frac{1}{\lambda} \left\{ (1 + \lambda)^{\sum_{l=0}^{+\infty} \left\{ \sum_{j=-l}^{+\infty} \left(b_j^0 \cdot p^j \right) \right\}} - 1 \right\} =$$

$$= \frac{1}{\lambda} \left\{ (1 + \lambda)^{\sum_{j=0}^{+\infty} \left(b_j^0 \cdot p^j \right) + \sum_{j=-1}^{+\infty} \left(b_j^1 \cdot p^j \right) + \sum_{j=-2}^{+\infty} \left(b_j^2 \cdot p^j \right) \cdots} - 1 \right\} =$$

$$= \frac{1}{\lambda} \left\{ (1 + \lambda)^{r_0^A \oplus r_1^A \oplus r_2^A \oplus \cdots} - 1 \right\} = \frac{1}{\lambda} \left\{ (1 + \lambda)^{r^A} - 1 \right\},$$

где $\bigoplus_{l \le L} r_l^A = r^A$ — результирующее р-адическое число, полученное по правилу сложения р-адических чисел r_l^A для всех уровней $l \le L$, $L \in N$. Если для подмножества $A \subseteq I$ рассматривается конечное множество р-адических шаров $U_{\varepsilon_{apr}}(A) \subseteq U(A)$, которое аппроксимирует множество $A \subseteq I$ с точностью ε_{apr}, то результатом р-адического сложения L р-адических чисел r_l^A будет р-адическое число $r^A = \sum_{j=-L}^{+\infty} (q_j \cdot p^j)$, где $L = -\log_p \min_{i=\overline{1,N_A}} \varepsilon_i$, $\varepsilon_i \ge \varepsilon_{apr}$. Исходя из свойства ограниченности нечеткой меры $g(A) \le 1$ следует, что $\sum_{j=-L}^{+\infty} (q_j \cdot p^j) \le 1$, а следовательно, $j \le 0$ и $q_0 \in \{0,1\}$. Тогда можем записать

$$r^A = \begin{cases} \sum_{j=-L}^{-1} (q_j \cdot p^j), \; q_0 = 0; \\ 1, \; q_0 = 1, \end{cases}$$

где $\sum_{j=-L}^{-1} (q_j \cdot p^j)$ является дробным р-адическим числом [13].

Следствие. Мера подмножества $A \subseteq I$ для равномерной нечеткой меры вероятности на р-адических шарах $Pr(\cdot)$: $2^I \to [0,1]$ определяется пределом по р-адической норме последовательности р-адического числа $r^A = \sum_{j=-L}^{+\infty} (q_j \cdot p^j)$, $q_l \in \{0,1,\ldots,p-1\}$, где q_l — коэффициенты канонического разложения результирующего р-адического числа r^A, полученного по правилу сложения р-адических чисел r_l^A для всех уровней разложения $l \le L$, $L \in N$, где $r_l^A = \sum_{j=-l}^{+\infty} (b_j^l \cdot p^j)$, $b_j^l = b_n^l$, $j = n - l$, а b_n^l — коэффициенты в позиционной р-адической системе счисления для числа $a_l = Card(U_l(A))$.

Доказательство. Мера вероятности шара $U_{\varepsilon_i}(r_i) \in U_l(A)$ равна p^{-l}. Шары из множества $U(A)$ не пересекаются, поэтому имеем:

$$Pr(U_l(A)) = \sum_{U_{\varepsilon_i}(r_i) \in U_l(A)} Pr\left(U_{\varepsilon_i}(r_i) \right) = \frac{a_l}{p^l} =$$

$$= p^{-l} \sum_{n=0}^{+\infty} \left(b_n^l \cdot p^n \right) = \sum_{j=-l}^{+\infty} \left(b_j^l \cdot p^j \right) = r_l^A,$$

где $b_j^l = b_n^l$, $j = n - l$. Множества $U_l(A) \subseteq U(A)$ для любых уровней l также между собой не пересекаются. Поэтому можем записать:

$$Pr\big(U(A)\big) = \sum_{l=0}^{+\infty} Pr\big(U_l(A)\big) =$$

$$= r_0^A \oplus r_1^A \oplus r_2^A \oplus \cdots = r^A = \sum_{j=-L}^{+\infty} \big(q_j \cdot p^j\big),$$

где q_l — коэффициенты канонического разложения результирующего p-адического числа r^A, полученного по правилу сложения p-адических чисел r_l^A для всех уровней разложения $l \le L$, $L \in N$. Мера $Pr(U(A))$ равна пределу по p-адической норме последовательности, представляющей p-адическое число r^A. ■

Пример 1. Рассмотрим пример расчета равномерной p-адической меры (при $p = 3$) для подмножества $A \subseteq I$, которое представлено множеством p-адических шаров

$$U(A) = \left\{U_{\frac{1}{3}}(r_{0,1}), U_{\frac{1}{9}}(r_{0,0,1}), U_{\frac{1}{9}}(r_{0,0,2}), U_{\frac{1}{9}}(r_{0,2,0})\right\},$$

где $r_{0,1} = (0122\ldots)_3$, $r_{0,0,1} = (0012\ldots)_3$, $r_{0,0,2} = (0022\ldots)_3$, $r_{0,2,0} = (0202\ldots)_3$. Пусть $\lambda = 0.7$. В соответствии с Утверждением 1 меры p-адических шаров из множества $U(A)$ будут $g(r_{0,1}) \cong 0.2764$, $g(r_{0,0,1}) = g(r_{0,0,2}) = g(r_{0,2,0}) \cong 0.0868$. При этом $g(r_0) = 1$. Множество $U(A)$ по уровням l распределяется в виде:

$$U_0(A) = \varnothing, U_1(A) = \left\{U_{\frac{1}{3}}(r_{0,1})\right\}, U_2(A) =$$

$$= \left\{U_{\frac{1}{9}}(r_{0,0,1}), U_{\frac{1}{9}}(r_{0,0,2}), U_{\frac{1}{9}}(r_{0,2,0})\right\}.$$

Тогда, $a_0 = Card(U_0(A)) = 0$, $a_1 = 1$, $a_3 = 3$. И, следовательно, $r_0^A = 0$, $r_1^A = (0100\ldots)_3$, $r_2^A = (0100\ldots)_3$. Выполнив сложение $r_l^A, l = 0,1,2$ по правилу сложения p-адических чисел, получим $r^A = (0200\ldots)_3$. Пределом последовательности будет $2 \cdot 3^{-1} = 2/3$. Тогда мера подмножества $A \subseteq I$ в соответствии с Утверждением 2 будет

$$g(A) = \frac{1}{0.7} \cdot \left\{(1 + 0.7)^{\frac{2}{3}} - 1\right\} = 0.6063.$$

В данном примере при фиксированном параметре λ равномерная p-адическая мера совпадает с обычной равномерной нечеткой мерой, что полностью подтверждает правомочность Утверждения 1.

Неоднородная нечеткая мера на p-адических шарах. Использование равномерной нечеткой меры на p-адических шарах на практике затруднительно. Прежде всего, это связано с тем, что данная мера не позволяет учитывать различные распределения неопределенности на множестве I. Это приводит к большим ошибкам при моделировании неопределенности в практических задачах. Кроме этого, при увеличении уровня l мера p-адических шаров $g(r_{q_0,\ldots,q_l})$ убывает экспоненциально. Это серьезно затрудняет ее идентификацию на основе существующих методов [19]. Опыт использования нечетких мер при моделировании в условиях неопределенности показывает, что для каждого множества равных по покрытию p-адических шаров $\mathcal{U}(r_{q_0,\ldots q_{l-1}})$ мера должна иметь возможность принимать значения во всем интервале [0,1]. Это позволяет не привязываться к изначально заданному уровню $l = 0$, обеспечить возможность идентификации меры известными методами, а также практичность ее использования.

Ранее мы рассматривали равномерную нечеткую меру на p-адических шарах, для которой $\forall \mathcal{U}(r_{q_0,\ldots q_{l-1}})$, $l = 0,\ldots,+\infty, \forall q_l, g(r_{q_0,\ldots,q_l}) = \text{const}$, а параметр $\forall l, \lambda = \text{const}$. Далее определим неоднородную нечеткую меру на p-адических шарах. Сначала рассмотрим случай, когда параметр λ зависит только от уровня $\forall l, \lambda_l \in [-1,+\infty[$, $\lambda_l = \text{var}$, а мера равных по фиксированному покрытию шаров $U_{p^{-l}}(r_{q_0,\ldots,q_l})$ равномерная.

Утверждение 3. Нечеткая мера подмножества множества I, являющегося образом p-адического шара $U_{p^{-l}}(r_{q_0,\ldots q_l})$, для неравномерной по уровням l нечеткой меры на p-адических шарах с изменяющимся по уровням параметром λ_l определяется соотношением:

$$\begin{cases} g(r_{q_0,\ldots,q_l}) = \dfrac{1}{\lambda_l} \cdot \left\{\left(1 + f_l\right)^{1/p} - 1\right\}; \\ f_l = \dfrac{\lambda_l}{\lambda_{l-1}} \cdot \left((1 + f_{l-1})^{1/p} - 1\right) = \lambda_l \cdot g(r_{q_0,\ldots,q_{l-1}}), \end{cases}$$

где $l = 0,\ldots,+\infty$, $\lambda_l \in [-1,+\infty[$, $g(r_{q_0,\ldots,q_l}) \in [0,1]$, $g(r_{q_0}) = 1$, $f_1 = \lambda_1$.

Доказательство. Сохраним ранее описанную процедуру построения нечеткой меры на p-адических шарах. Тогда на уровне $l = 1$ равномерная нечеткая мера p-адического шара будет:

$$g(r_{q_0,q_1}) = \frac{1}{\lambda_1} \cdot \left\{\left(1 + \lambda_1\right)^{\frac{1}{p}} - 1\right\}.$$

Для согласованности меры на $l = 2$ должно выполняться условие:

$$\frac{1}{\lambda_2} \cdot \left\{(1 + \lambda_2 \cdot g(r_{q_0,q_1,q_2}))^p - 1\right\} =$$

$$= \frac{1}{\lambda_1} \cdot \left\{\left(1 + \lambda_1\right)^{\frac{1}{p}} - 1\right\}.$$

Отсюда значение меры шара $U_{p^{-2}}(r_{q_0,q_1,q_2})$ на уровне $l = 2$ будет:

$$g(r_{q_0,q_1,q_2}) = \frac{1}{\lambda_2}\cdot\left\{\left[\frac{\lambda_2}{\lambda_1}\cdot\left((1+\lambda_1)^{\frac{1}{p}}-1\right)+1\right]^{\frac{1}{p}}-1\right\}.$$

Введем в рассмотрение вспомогательную функцию f_l вида:

$$f_l = \begin{cases}\lambda_1,\ l=1;\\ \dfrac{\lambda_l}{\lambda_{l-1}}\cdot\left((1+f_{l-1})^{1/p}-1\right),\ l>1.\end{cases}$$

Тогда значение меры g_2 будет определяться соотношением:

$$g(r_{q_0,q_1,q_2}) = \frac{1}{\lambda_2}\cdot\left\{(1+f_2)^{\frac{1}{p}}-1\right\}.$$

Повторим процедуру для уровня $l=3$. Тогда:

$$g(r_{q_0,q_1,q_2,q_3}) = \frac{1}{\lambda_3}\times$$

$$\times\left\{\left[\underbrace{\frac{\lambda_3}{\lambda_2}\cdot((1+f_2)^{\frac{1}{p}}-1)+1}_{f_3}\right]^{\frac{1}{p}}-1\right\} =$$

$$= \frac{1}{\lambda_3}\cdot\left\{(1+f_3)^{\frac{1}{p}}-1\right\}.$$

Таким образом, неравномерная по уровням l нечеткая мера на р-адических шарах с изменяющимся параметром λ_l для р-адического шара $U_{p^{-l}}(r_{q_0,\ldots q_l})$ определяется системой рекуррентных уравнений:

$$\begin{cases} g(r_{q_0,\ldots,q_l}) = \dfrac{1}{\lambda_l}\cdot\left\{(1+f_l)^{\frac{1}{p}}-1\right\};\\ f_l = \dfrac{\lambda_l}{\lambda_{l-1}}\cdot\left((1+f_{l-1})^{\frac{1}{p}}-1\right) = \lambda_l\cdot g(r_{q_0,\ldots,q_{l-1}}),\end{cases}$$

где $l=0,\ldots,+\infty$, $\lambda_l\in[-1,+\infty[$, $g(r_{q_0,\ldots,q_l})\in[0,1]$, $g(r_{q_0})=1$, $f_1=\lambda_1$. ■

В общем случае для фиксированного уровня l нечеткая мера шара $U_{p^{-l}}(r_{q_0,\ldots,q_l})$ будет зависеть от последовательности $\lambda_l(r_{q_0,\ldots,q_{l-1}})\in[-1,+\infty[$. В этом случае $\lambda_l(r_{q_0,\ldots,q_{l-1}})$ и вспомогательная функция $f_l(r_{q_0,\ldots,q_l})$ будут функциями р-адической координаты $Seq_{l-1}(r_{q_0,\ldots,q_{l-1}}) = (q_0,\ldots,q_{l-1})$. По мере увеличения уровня l мера $g(r_{q_0,\ldots,q_l})$ шара $U_{p^{-l}}(r_{q_0,\ldots,q_l})$ уменьшается экспоненциально, что определяется ограничением $g(\mathcal{U}(r_{q_0,\ldots q_{l-1}})) = g(r_{q_0,\ldots,q_{l-1}})$. Однако связь между мерами на соседних уровнях может и не удовлетворять данному ограничению. В этом случае для обеспечения согласованности меры для всех шаров $U_{p^{-l}}(r_{q_0,\ldots,q_l})$ введем в рассмотрение функцию проектора (или просто проектор) $Proj(r_{q_0,\ldots,q_{l-1}})$ между уровнями l и $l-1$ для конкретной р-адической координаты $(q_0,\ldots,q_{l-1})$ так, чтобы выполнялось условие:

$$g(\mathcal{U}(r_{q_0,\ldots q_{l-1}})) =$$
$$= Proj(r_{q_0,\ldots,q_{l-1}})\cdot g(r_{q_0,\ldots,q_{l-1}})\in[0,1].$$

Тогда, учитывая доказательство Утверждения 3, неравномерная по уровням l нечеткая мера на р-адических шарах для фиксированного шара $U_{p^{-l}}(r_{q_0,\ldots,q_l})$ будет определяться системой рекуррентных уравнений вида:

$$\begin{cases} g(r_{q_0,\ldots,q_l}) = \dfrac{1}{\lambda_l(r_{q_0,\ldots,q_l})}\cdot\left\{(1+f_l(r_{q_0,\ldots,q_{l-1}}))^{\frac{1}{p}}-1\right\};\\ f_l(r_{q_0,\ldots,q_{l-1}}) = \lambda_l(r_{q_0,\ldots,q_{l-1}})\cdot Proj(r_{q_0,\ldots,q_{l-1}})\cdot g(r_{q_0,\ldots,q_{l-1}}),\end{cases}$$

где $l=0,\ldots,+\infty$, $Proj(r_{q_0,\ldots,q_{l-1}})\cdot g(r_{q_0,\ldots,q_{l-1}})\in[0,1]$,
$g(r_{q_0})=1$, $f_1(r_{q_0})=\lambda_1(r_{q_0})$, $\lambda_l(r_{q_0,\ldots,q_{l-1}})\in[-1,+\infty[$.

Рассмотрим проектор $Proj(r_{q_0,\ldots,q_{l-1}})$. Если $\lambda_l(r_{q_0,\ldots,q_{l-1}})\to-1$, то мы имеем меру правдоподобия (возможности) и в этом случае $g(r_{q_0,\ldots,q_l})\to1$. При этом должно выполняться условие $Proj(r_{q_0,\ldots,q_{l-1}})\cdot g(r_{q_0,\ldots,q_{l-1}})\ge g(r_{q_0,\ldots,q_{l-1}})$ и, следовательно, $Proj(r_{q_0,\ldots,q_{l-1}})\ge1$. Аналогично, если $\lambda_l(r_{q_0,\ldots,q_{l-1}})\to+\infty$, то мы имеем меру доверия (необходимости) и $g(r_{q_0,\ldots,q_l})\to0$. В этом случае $Proj(r_{q_0,\ldots,q_{l-1}})\in[0,1]$. Таким образом, проектор $Proj(r_{q_0,\ldots,q_{l-1}})$ является функцией, зависящей от $g(r_{q_0,\ldots,q_{l-1}})$ и $\lambda_l(r_{q_0,\ldots,q_{l-1}})$, для которой выполняются вышеописанные условия. Вариантом функции проектора может быть функция:

$$Proj(r_{q_0,\ldots,q_{l-1}}) = g(r_{q_0,\ldots,q_{l-1}})^{\lambda_l(r_{q_0,\ldots,q_{l-1}})}.$$

Тогда функция $f_l(r_{q_0,\ldots,q_{l-1}})$ примет вид:

$$f_l(r_{q_0,\ldots,q_{l-1}}) = \lambda_l(r_{q_0,\ldots,q_{l-1}})\cdot g(r_{q_0,\ldots,q_{l-1}})^{1+\lambda_l(r_{q_0,\ldots,q_{l-1}})}.$$

Определим зависимости для расчета неоднородной нечеткой меры на р-адических шарах для подмножества $A\subseteq I$, которое является образом множества р-адических шаров $U(A)=\{U_{\varepsilon_i}(r_i)|i=\overline{1,N_A}\}$ [10]. Пусть $CS(U_{\varepsilon_i}(r_i))$ — множество всех р-адических покрытий $Cov_\alpha(U_{\varepsilon_i}(r_i))$ для шара $U_{\varepsilon_i}(r_i)$. Тогда множество всех покрытий, соответствующих $U(A)$, будет задавать для множества $A\subseteq I$ р-адический ландшафт $LS(U(A))=\{CS(U_{\varepsilon_i}(r_i))|U_{\varepsilon_i}(r_i)\in U(A)\}$. Исходя из этого, для определения меры подмножества $A\subseteq I$ на практике целесообразно знать меры всех покрытий из $LS(U(A))$. Мера для покрытия $U_1(1)$, соответствующего всему множеству I, будет определять нечеткую меру подмножества $A\subseteq I$

в классическом понимании. Следует отметить, что в общем случае мера "полного" шара $U_{p^{1-l}}(r_{q_0,\ldots,q_{l-1}})$ и мера покрытия в виде данного шара $Cov_{p^{1-l}}(U_{p^{-l}}(r_{q_0,\ldots q_l})) = U_{p^{1-l}}(r_{q_0,\ldots,q_{l-1}})$ не совпадают.

Обозначим меру покрытия для шаров $U_{p^{-l}}(r_{q_0,\ldots q_l})$ в виде $g(Cov(r_{q_0,\ldots q_{l-1}}))$. В общем случае выполняется условие $g(r_{q_0,\ldots,q_{l-1}}) \geq g(Cov(r_{q_0,\ldots q_{l-1}}))$. Тогда справедливо следующее утверждение.

Утверждение 4. Значение неравномерной нечеткой меры на p-адических шарах покрытия $Cov_{p^{1-l}}(U_{p^{-l}}(r_{q_0,\ldots q_l})) \in LS(U(A))$ определяется соотношениями:

$$\begin{cases} g(Cov(r_{q_0,\ldots q_{l-1}})) = \dfrac{1}{\lambda_l(r_{q_0,\ldots,q_{l-1}})} \times \\ \times \left\{ F_l(r_{q_0,\ldots,q_{l-1}}) \cdot (1 + f_l(r_{q_0,\ldots,q_{l-1}}))^{\frac{a_l(r_{q_0,\ldots q_{l-1}})}{p}} - 1 \right\}; \\ f_l(r_{q_0,\ldots,q_{l-1}}) = \lambda_l(r_{q_0,\ldots,q_{l-1}}) \cdot g(r_{q_0,\ldots,q_{l-1}})^{1+\lambda_l(r_{q_0,\ldots,q_{l-1}})}; \\ F_l(r_{q_0,\ldots,q_{l-1}}) = \prod_{q_l \notin A_l(r_{q_0,\ldots,q_{l-1}})} (1 + g(Cov(r_{q_0,\ldots q_l})) \times \\ \times \lambda_l(r_{q_0,\ldots,q_{l-1}})), \end{cases}$$

где $r_{q_0,\ldots,q_{l-1}}$ — p-адическое число, соответствующее центру шара покрытия, $A_l(r_{q_0,\ldots,q_{l-1}})$ — множество индексов q_l p-адических чисел $r_{q_0,\ldots,q_l}$ для центров шаров равных по покрытию $U_{p^{1-l}}(r_{q_0,\ldots,q_{l-1}})$ и входящих в $U(A)$, $Card(A_l(r_{q_0,\ldots,q_{l-1}})) = a_l(r_{q_0,\ldots q_{l-1}}) \in \{0,\ldots,p-1\}$, $g(r_{q_0,\ldots,q_{l-1}})$ — мера p-адического шара $U_{p^{1-l}}(r_{q_0,\ldots,q_{l-1}})$, рассчитанная для последовательности $\lambda_l(r_{q_0,\ldots,q_{l-1}})$.

Доказательство. Мера покрытия $Cov_{p^{1-l}}(U_{p^{-l}}(r_{q_0,\ldots q_l}))$ на уровне $l-1$ определяется по формуле:

$$g(Cov(r_{q_0,\ldots q_{l-1}})) = \frac{1}{\lambda_l(r_{q_0,\ldots,q_{l-1}})} \times$$
$$\times \left\{ \prod_{q_l \in \{0,\ldots,p-1\}} \left(1 + g(Cov(r_{q_0,\ldots q_l}))\,\lambda_l(r_{q_0,\ldots q_{l-1}})\right) - 1 \right\}.$$

Все множество индексов q_l на уровне l делится на два подмножества $A_l(r_{q_0,\ldots,q_{l-1}})$, для p-адических шаров $U_{p^{-l}}(r_{q_0,\ldots,q_l}) \in U(A)$, и $\{0,\ldots,p-1\} \setminus A_l(r_{q_0,\ldots,q_{l-1}})$, для $U_{p^{-l}}(r_{q_0,\ldots,q_l}) \notin U(A)$. Тогда на уровне l для $q_l \in A_l(r_{q_0,\ldots,q_{l-1}})$ мера $g(Cov(r_{q_0,\ldots q_l}))$ будет определяться соотношением:

$$\begin{cases} g(Cov(r_{q_0,\ldots q_l})) = g(r_{q_0,\ldots,q_l}) = \dfrac{1}{\lambda_l(r_{q_0,\ldots,q_{l-1}})} \times \\ \times \left\{ (1 + f_l(r_{q_0,\ldots,q_{l-1}}))^{1/p} - 1 \right\}; \\ f_l(r_{q_0,\ldots,q_{l-1}}) = \lambda_l(r_{q_0,\ldots,q_{l-1}}) \cdot g(r_{q_0,\ldots,q_{l-1}})^{1+\lambda_l(r_{q_0,\ldots,q_{l-1}})}. \end{cases}$$

Отсюда:

$$\prod_{q_l \in A_l(r_{q_0,\ldots,q_{l-1}})} \left(1 + g(Cov(r_{q_0,\ldots q_l})) \cdot \lambda_l(r_{q_0,\ldots,q_{l-1}})\right) =$$
$$= \left(1 + g(Cov(r_{q_0,\ldots q_l})) \cdot \lambda_l(r_{q_0,\ldots,q_{l-1}})\right)^{a_l(r_{q_0,\ldots q_{l-1}})} =$$
$$= \left\{ 1 + \left[\frac{1}{\lambda_l(r_{q_0,\ldots,q_{l-1}})} \cdot \left\{ (1 + f_l(r_{q_0,\ldots,q_{l-1}}))^{\frac{1}{p}} - 1 \right\} \right] \times \right.$$
$$\left. \times \lambda_l(r_{q_0,\ldots,q_{l-1}}) \right\}^{a_l(r_{q_0,\ldots q_{l-1}})} = \left(1 + f_l(r_{q_0,\ldots,q_{l-1}})\right)^{\frac{a_l(r_{q_0,\ldots q_{l-1}})}{p}}.$$

Для шаров $U_{p^{-l}}(r_{q_0,\ldots,q_l}) \notin U(A)$, $q_l \notin A_l(r_{q_0,\ldots,q_{l-1}})$ и $g(Cov(r_{q_0,\ldots q_l})) \leq g(r_{q_0,\ldots,q_l})$. Для них определим вспомогательную функцию в виде:

$$F_l(r_{q_0,\ldots,q_{l-1}}) =$$
$$= \prod_{q_l \notin A_l(r_{q_0,\ldots,q_{l-1}})} \left(1 + g(Cov(r_{q_0,\ldots q_l})) \cdot \lambda_l(r_{q_0,\ldots,q_{l-1}})\right).$$

Подставив соответствующие выражения для $g(Cov(r_{q_0,\ldots q_l}))$ и $F_l(r_{q_0,\ldots,q_{l-1}})$ для определения меры $g(Cov(r_{q_0,\ldots q_{l-1}}))$, мы получим соотношение, заданное Утверждением 3. ■

Интересно заметить, что функции $f_l(\cdot)$ и $F_l(\cdot)$ выполняют роли уточняющей и агрегирующей функций соответственно. Уточняющая функция позволяет по заданным параметрам λ определить меры соответствующих p-адических шаров, а агрегирующая функция обеспечивает получение значения меры любого покрытия, входящего в p-адический ландшафт $LS(U(A))$ для множества $A \subseteq I$. Исходя из этого, практический алгоритм расчета неравномерной меры на p-адических шарах для данного подмножества имеет следующий вид.

Алгоритм.

Шаг 1. Определение исходных данных. Для задания меры на p-адических шарах мы должны иметь полную функцию $\lambda_l(r_{q_0,\ldots,q_{l-1}})$ для всех p-адических шаров на множестве I. Подмножество $A \subseteq I$ рассматривается как образ множества p-адических шаров $U(A)$ в соответствии с подходом, представленным в работе [10].

Шаг 2. Определение меры p-адических шаров. Меры $g(r_{q_0,\ldots,q_l})$ рассчитываются на основании функции неравномерной меры на p-адических шарах Утверждения 3. Для этого определяются функции $f_l(r_{q_0,\ldots,q_{l-1}})$.

Шаг 3. Нахождение начального условия функции агрегирования. Определяем максимальный уровень l, на котором определены p-адические шары минимального радиуса из множества $U(A)$ и определяем $F_l(r_{q_0,\ldots,q_{l-1}}) = 1$.

Шаг 4. Расчет меры подмножества. Рассчитываем в соответствии с формулой Утверждения 4 меры покрытий $g(Cov(r_{q_0,\ldots q_{l-1}}))$. Принимаем $g(Cov(r_{q_0})) = g(A)$.

Пример 2. Рассмотрим расчет неравномерной нечеткой меры на p-адических шарах для подмножества $A \subseteq I$ из Примера 1. Для расчета меры подмножества $A \subseteq I$ нам необходимо рассчитать меры покрытий $Cov_1(U_{p^{-1}}(r_{0,q_1}))$, $Cov_{p^{-1}}(U_{p^{-2}}(r_{0,0,q_2}))$, $Cov_{p^{-1}}(U_{p^{-2}}(r_{0,2,q_2}))$. При этом $g(Cov(r_0))$ для покрытия $Cov_1(U_{p^{-1}}(r_{0,q_1}))$ будет определять меру $g(A)$ подмножества $A \subseteq I$. Пусть мера на p-адических шарах задана функцией $\lambda_l(r_{q_0,\ldots,q_{l-1}})$, зависящей от p-адической координаты. Пусть значения данной функции для p-адических координат, необходимых нам для расчета покрытий, будут иметь значения $\lambda_1(r_0) = 0.7$, $\lambda_2(r_{0,0}) = -0.8$, $\lambda_2(r_{0,2}) = 1$. Тогда значения уточняющей функции $f_l(r_{q_0,\ldots,q_{l-1}})$ для необходимых покрытий в соответствии с Утверждением 4 будут иметь значения: $f_2(r_{0,0}) = -0.6186$, $f_2(r_{0,2}) = 0.0764$, $f_1(r_0) = 0.7$. Соответственно меры шаров с заданными p-адическими координатами: $g(r_{0,0,1}) = g(r_{0,0,2}) = 0.3435$, $g(r_{0,2,0}) = 0.0248$, $g(r_{0,1}) = 0.2764$.

Для расчета меры покрытий в соответствии с Утверждением 4 рассчитываем агрегирующую функцию $F_l(r_{q_0,\ldots,q_{l-1}})$. Для необходимых покрытий данная функция примет значения $F_2(r_{0,0}) = F_2(r_{0,2}) = 1$, $F_1(r_0) = 1.4394$. Тогда меры рассчитываемых покрытий будут: $g(Cov(r_{0,0})) = 0.5925$, $g(Cov(r_{0,2})) = 0.0248$, $g(Cov(r_0)) = 0.802$. Таким образом, для заданной неоднородной меры на p-адических шарах мера подмножества $A \subseteq I$ будет иметь значение $g(A) = g(Cov(r_0)) = 0.802$. Получение данного результата может быть представлено таблицей 1.

При изменении функции $\lambda_l(r_{q_0,\ldots,q_{l-1}})$ мера примет другое значение. Сама же функция параметра λ зависит от p-адической координаты покрывающего шара. Таким образом, мера на p-адических шарах может учитывать структуру множества I и формировать на нем пространство с нечеткой мерой, имеющей переменную модальность. При этом необходимость непосредственного задания функции плотности нечеткой меры отсутствует.

Одним из вариантов задания функции $\lambda_l(r_{q_0,\ldots,q_{l-1}})$ может быть рассмотрена функция, определяемая p-адической координатой $Seq_{l-1}(r) = (q_0,\ldots q_{l-1})$:

$$\lambda_l\left(r_{q_0,\ldots,q_{l-1}}\right) = \left[\mu\left(\rho\left(Seq_{l-1}(r)\right)\right)\right]^{-2} \times \times \left\{1 - 2 \cdot \mu\left(\rho\left(Seq_{l-1}(r)\right)\right)\right\},$$

где $$\rho(Seq_{l-1}(r)) = \sum_{j=0}^{l-1} \frac{q_j}{p^j},$$

$q_j \in \{0,\ldots,p-1\}$, $\mu(\cdot)\colon [0,1] \to [0,1]$ — функция в единичном квадрате. В простейшем случае $\mu(\cdot)$ является тождественной функцией вида $\mu(x) = x$.

4. ВЫВОДЫ

Предложенная нечеткая мера на p-адических шарах полностью удовлетворяет свойствам нечеткой меры, что вытекает из подстановки исходных данных свойств ограниченности, монотонности и непрерывности в соотношениях Утверждения 4. Данная мера учитывает структуру ограниченного числового множества в виде образов p-адических шаров и не требует задания точек данного множества, а соответственно и плотности нечеткой меры. Нечеткая мера на p-адических шарах полностью определяется лишь функцией параметра нормировки λ (модальностью нечеткой меры), которая однозначно задается p-адической координатой покрывающего шара. Таким образом нечеткая мера, предложенная в работе, является вещественнозначной функцией p-адического аргумента. Использование проектора при построении нечеткой меры снимает проблему определения нулевого уровня, позволяет обеспечить согласованность нечеткой меры на различных уровнях с учетом относительной нормировки в каждом покрывающем p-адическом шаре. Использование неравномерной нечеткой меры на p-адических шарах позволяет моделировать пространства с нестационарной нечеткой мерой, в которой параметр λ

Таблица 1. Расчет неравномерной p-адической меры

l	$U_{p^{-l}}(r_{q_0,\ldots q_l})$	$\lambda_l(r_{q_0,\ldots,q_{l-1}})$	$f_l(r_{q_0,\ldots,q_{l-1}})$	$g(r_{q_0,\ldots,q_l})$	$F_l(r_{q_0,\ldots,q_{l-1}})$	$g(Cov(r_{q_0,\ldots q_{l-1}}))$
1	$U_{p^{-1}}(r_{0,1})$	0.7	0.7	0.2764	1.4394	**0.802**
2	$U_{p^{-2}}(r_{0,0,1})$	–0.8	–0.6186	0.3435	1	0.5925
	$U_{p^{-2}}(r_{0,0,2})$	–0.8	–0.6186	0.3435	1	0.5925
	$U_{p^{-2}}(r_{0,2,0})$	1	0.0764	0.0248	1	0.0248

зависит от p-адической координаты. Применение данной меры открывает возможность моделирования динамики сложных ландшафтов для различных скалярных полей.

СПИСОК ЛИТЕРАТУРЫ

1. *Becker O.M., Karplus M.* The topology of multidimensional protein energy surfaces: theory and application to peptide structure and kinetics // Journal of Chemical Physics. 1997. V. 106. P. 1495–1517.
2. *Grabisch M., Murofushi T., Sugeno M.* Fuzzy Measures and Integrals: Theory and Applications. Berlin, Germany, Physica-Verlag GmbH & Co, 2000, 477 p., ISBN10 3790812587
3. *Веккер Л.М.* Психика и реальность: единая теория психических процессов. — М.: Смысл, 1998. 685 с. Режим доступа: https://vshp.pro/wp-content/uploads/2020/03/Vekker-L.M.-Psihika-i-realnost.-Edinaya-teoriya-psihicheskih-protsessov.pdf
4. Философский энциклопедический словарь. / гл. ред.: Л.Ф. Ильичев, П.Н. Федосеев, С.М. Ковалев, В.Г. Панов. М.: Советская энциклопедия, 1983. 840 с.
5. *Aliev R.A.* Fundamentals of the Fuzzy Logic-Based Generalized Theory of Decisions. Studies in Fuzziness and Soft Computing. Springer, 2013, 324 p. ISBN 3642348955
6. *Keller J.M., Derong L., Fogel D.B.* Fundamentals of Computational Intelligence: Neural Networks, Fuzzy Systems, and Evolutionary Computation. IEEE Press Series on Computational Intelligence. John Wiley & Sons, 2016, 378 p. ISBN 1119214343.
7. *Vladimirov V.S. Volovich I.V., Zelenov E.I.* P-adic Analysis and Mathematical Physics. Series on Soviet and East European Mathematics (Vol 1). World Scientific, 1994, 340 p. ISBN 9814505765.
8. *Yager R.R., Liping L.* Classic Works of the Dempster-Shafer Theory of Belief Functions. Studies in Fuzziness and Soft Computing (Vol 219). Springer, 2008, 806 p. ISBN 354044792X.
9. *Torra V., Narukawa Y., Sugeno M.* Non-Additive Measures: Theory and Applications. Studies in Fuzziness and Soft Computing (Vol 310). Springer, 2013, 201 p. ISBN 3319031554.
10. *Bocharnikov V.P., Sveshnikov S.V.* p-Adic Representation of Subsets of a Bounded Number Set. Programming and Computer Software, 2021, Vol. 47, No. 4, pp. 225–234.
11. *Koblitz N.* p-adic Numbers, p-adic Analysis, and Zeta-Functions. Springer, Science & Business Media, 2012, 153 p. ISBN 1461211123.
12. *Katok S.* p-adic Analysis Compared with Real. Student mathematical library (Vol 37), American Mathematical Society. American Mathematical Soc., 2007, 152 p. ISBN 9780821842201.
13. *Волович И.В., Козырев С.В.* p-Адическая математическая физика: основные конструкции, применения к сложным и наноскопическим системам, Математическая физика и ее приложения. Вводные курсы. Вып. 1, Самарский гос. ун-т, Самара, 2009. http://www.mi.ras.ru/noc/irreversibility/p-adicMF1.pdf
14. *Хренников А.Ю.* Моделирование процессов мышления в p-адических системах координат. — М.: ФИЗМАТЛИТ, 2004. 296 с. ISBN 5-9221-0501-9.
15. *Kozyrev S.V.* Wavelet theory as p-adic spectral analysis. Izv. RAN. Ser. Mat., 2002, Vol. 66, Iss. 2, pp. 149–158.
16. *Кононюк А.Е.* Обобщенная теория моделирования: Книга 2: Числа: количественные оценки параметров модели. Киев: “Освіта України”, 2012. 548 с. ISBN 9789667599508.
17. *Deza M-M, Deza E.* Encyclopedia of distances. Berlin, Springer, 2008. 412 p. (Russ. ed.: Deza M-M, Deza E. Entsiklopedicheskii slovar’ rasstoyanii. Moscow, Nauka Publ., 444 p).
18. *Borevich Z.I., Shafarevich I.R.* Teoriya chisel [Number theory]. Moscow, Science. The main edition of the physical and mathematical literature, 3rd ed., 1985. 504 p.
19. *Bocharnikov V., Bocharnikov I., Sveshnikov S.* Fundamentals of the systemic organization’s management. Theory and Practice. Berlin, LAP LAMBERT Academic Publishing, 2012, 296 p. ISBN 9783659223327

FUZZY MEASURE ON p-ADIC BALLS DEFINED ON A FINITE NUMBER SET

V. P. Bocharnikov[a], S. V. Sveshnikov[a]

[a]*INEKS-FT Consulting Group, ul. Desyatinnaya 13a, Kiev, 03011 Ukraine*

The article explores an approach to constructing a fuzzy measure on p-adic balls that does not require the direct specification of the measure density. The relationships necessary for determining this measure for an arbitrary subset of a bounded numerical set, represented as a set of p-adic balls, are proven. Uniform and non-uniform fuzzy measures are considered. An algorithm for determining the fuzzy measure on p-adic balls is proposed. Examples of calculating this measure are provided.

Keywords: fuzzy measure, p-adic balls, p-adic numbers, set, ultrametric

REFERENCES

1. *Becker O.M., Karplus M.* The topology of multidimensional protein energy surfaces: theory and application to peptide structure and kinetics // Journal of Chemical Physics. 1997. V. 106. P. 1495–1517.
2. *Grabisch M., Murofushi T., Sugeno M.* Fuzzy Measures and Integrals: Theory and Applications. Berlin, Germany, Physica-Verlag GmbH & Co, 2000, 477 p., ISBN10 3790812587
3. *Wekker L.M.* Psyche and reality: a unified theory of mental processes. — M.: Smysl, 1998. — 685 p. — Access mode: https://vshp.pro/wp-content/uploads/2020/03/Vekker-L.M.-Psihika-i-realnost.-Edinaya-teoriya-psihicheskih-protsessov.pdf
4. Philosophical encyclopedic dictionary. / Chief editor: *L. F. Ilyichev, P. N. Fedoseev, S. M. Kovalev, V. G. Panov*. — M.: Soviet Encyclopedia, 1983. — 840 p.
5. *Aliev R. A.* Fundamentals of the Fuzzy Logic-Based Generalized Theory of Decisions. Studies in Fuzziness and Soft Computing. Springer, 2013, 324 p. ISBN 3642348955
6. *Keller J. M., Derong L., Fogel D. B.* Fundamentals of Computational Intelligence: Neural Networks, Fuzzy Systems, and Evolutionary Computation. IEEE Press Series on Computational Intelligence. John Wiley & Sons, 2016, 378 p. ISBN 1119214343
7. *Vladimirov V. S, Volovich I. V, Zelenov E. I.* P-adic Analysis and Mathematical Physics. Series on Soviet and East European Mathematics (Vol 1). World Scientific, 1994, 340 p. ISBN 9814505765
8. *Yager R.R., Liping L.* Classic Works of the Dempster-Shafer Theory of Belief Functions. Studies in Fuzziness and Soft Computing (Vol 219). Springer, 2008, 806 p. ISBN 354044792X
9. *Torra V., Narukawa Y., Sugeno M.* Non-Additive Measures: Theory and Applications. Studies in Fuzziness and Soft Computing (Vol 310). Springer, 2013, 201 p. ISBN 3319031554
10. *Bocharnikov V. P., Sveshnikov S. V.* p-Adic Representation of Subsets of a Bounded Number Set. Programming and Computer Software, 2021, Vol. 47, No. 4, pp. 225–234.
11. *Koblitz N.* p-adic Numbers, p-adic Analysis, and Zeta-Functions. Springer, Science & Business Media, 2012, 153 p. ISBN 1461211123
12. *Katok S.* p-adic Analysis Compared with Real. Student mathematical library (Vol 37), American Mathematical Society. American Mathematical Soc., 2007, 152 p. ISBN 9780821842201
13. *Volovich I. V., Kozyrev S.* V. p-Adic mathematical physics: basic constructions, applications to complex and nanoscopic systems, Mathematical physics and its applications. Introductory courses. Issue 1, Samara State. University, Samara, 2009. http://www.mi.ras.ru/noc/irreversibility/p-adicMF1.pdf
14. *Khrennikov A. Yu.* Modeling of thinking processes in p-adic coordinate systems. — M.: FIZMATLIT, 2004. — 296 p. ISBN 5-9221-0501-9
15. *Kozyrev S. V.* Wavelet theory as p-adic spectral analysis. Izv. RAN. Ser. Mat., 2002, Vol. 66, No. 2, pp. 149–158
16. *Кононюк А. Е.* Обобщённая теория моделирования: Книга 2: Числа: количественные оценки параметров модели. Киев: «Освіта України», 2012. — 548 с. ISBN 9789667599508
17. *Deza M-M, Deza E.* Encyclopedia of distances. Berlin, Springer, 2008. 412 p. (Russ. ed.: Deza M-M, Deza E. Entsiklopedicheskii slovar' rasstoyanii. Moscow, Nauka Publ., 444 p)
18. *Borevich Z.I., Shafarevich I.R.* Teoriya chisel [Number theory]. Moscow, Science. The main edition of the physical and mathematical literature, 3rd ed., 1985. 504 p.
19. *Bocharnikov V., Bocharnikov I., Sveshnikov S.* Fundamentals of the systemic organization's management. Theory and Practice. Berlin, LAP LAMBERT Academic Publishing, 2012, 296 p. ISBN 9783659223327

ПРОГРАММИРОВАНИЕ, *2024*, *№ 1*, *с. 15–29*

УДК 519.6

ПРИМЕНЕНИЕ БИБЛИОТЕКИ ФУНКЦИОНАЛЬНОГО ПРОГРАММИРОВАНИЯ ДЛЯ РАСПАРАЛЛЕЛИВАНИЯ ВЫЧИСЛЕНИЙ НА CUDA

М. М. Краснов[a],[*] (ORCID: 0000-0001-7988-6323),
О. Б. Феодоритова[a],[**] (ORCID: 0000-0002-2792-9376)

[a]*Институт прикладной математики им. М.В. Келдыша Российской академии наук, 125047 Москва, Миусская пл., д. 4*

**E-mail: kmm@kiam.ru*
***E-mail: feodor@kiam.ru*

Поступила в редакцию 15.02.2023
После доработки: 28.02.2023
Принята к публикации: 10.04.2023

Современные графические ускорители (GPU) позволяют существенно ускорить выполнение численных задач. Однако перенос программ на графические ускорители является непростой задачей, иногда требующей практически полного их переписывания. Графические ускорители CUDA, благодаря разработанной компанией NVIDIA технологии, позволяют иметь единый исходный код как для обычных процессоров (CPU), так и для CUDA. Однако распараллеливание на общей памяти все равно делается по-разному и его нужно указывать явно. Применение разработанной авторами библиотеки функционального программирования позволяет скрыть использование того или иного механизма распараллеливания на общей памяти внутри библиотеки и сделать пользовательский исходный код полностью независимым от используемого вычислительного устройства (CPU или CUDA). В настоящей статье показывается, как это можно сделать.

Ключевые слова: C++; функциональное программирование; численные методы; CUDA; OpenMP; OpenCL
DOI: 10.31857/S0132347424010027 **EDN:** HVJUIU

1. ВВЕДЕНИЕ

В последние годы все большее распространение получают графические ускорители (GPU), используемые в качестве вычислительных устройств для численных расчетов. Такие ускорители устанавливаются на многих вычислительных кластерах. Хотя в списке TOP500 самых производительных суперкомпьютеров от июня 2022 г. (JUNE 2022) [1] графические ускорители от компании NVIDIA [2] уступили пальму первенства технологиям от других производителей, в течение многих лет они были в числе первых. Что касается самых высокопроизводительных суперкомпьютеров России, по состоянию на март 2022 г. [3], практически все они оснащены ускорителями от NVIDIA. Скорость численных расчетов на таких ускорителях может быть во много раз выше, чем на CPU (по опыту авторов, ускорение может достигать 10–20 раз), поэтому перенос на графические ускорители программ, реализующих численные методы, является чрезвычайно актуальной задачей.

Однако перенос существующей программы на GPU является непростой задачей. Возможно, идеальный вариант — сразу писать программу так, чтобы она могла считаться на любых вычислителях. В любом случае главным встает вопрос — какую технологию работы на GPU использовать? В настоящий момент существует три основных технологии — это OpenCL (открытый стандарт для гетерогенных систем) [4], OpenACC [5] и CUDA [6] — разработка компании NVIDIA для своих графических ускорителей. Каждая из этих технологий имеет свои преимущества и недостатки. Главное преимущество OpenCL — открытый стандарт. Программа, использующая OpenCL, будет работать на любом вычислительном устройстве, поддерживающем этот стандарт, в том числе на GPU от NVIDIA и от AMD, процессорах Intel Xeon Phi с технологией Intel MIC и даже на обычных CPU. Главным недостатком этой технологии является то, что исходный код программы часто возникает в двух экземплярах: для CPU, который компилируется обычным компилятором и является частью основной программы, и текст для OpenCL в отдельных файлах. При изменениях в алгоритмах правки надо будет вносить в оба места. Преимущества и недостатки технологии CUDA являются зеркальным отражением недостатков и

преимуществ OpenCL. CUDA работает только на GPU от NVIDIA. С другой стороны, в CUDA мы имеем единый исходный текст, который компилируется предварительно и является частью основной программы (в том числе и код, который будет исполняться на GPU). Главный недостаток технологии OpenACC в том, что она пока еще недостаточно распространена. Компилятор, поддерживающий эту технологию, установлен далеко не на всех кластерах с графическими ускорителями.

Мы выбираем технологию CUDA. Наш главный аргумент состоит в том, что в (нашей) реальной жизни мы сталкиваемся исключительно с устройствами от NVIDIA. GPU от AMD и процессоры Intel Xeon Phi достаточно экзотичны и, хотя нам и встречались, реально неактуальны. Поэтому недостаток CUDA недостатком для нас не является, а ее преимущество остается.

Следующая проблема состоит в том, что распараллеливание на общей памяти на CPU и на GPU делается совершенно по-разному. Если мы хотим получить единый текст, который должен компилироваться и для CPU, и для CUDA, то в тех местах, где должно быть распараллеливание, придется писать разный код (например, с помощью конструкции **#ifdef**), что неудобно. И тут возникла идея воспользоваться ранее написанной одним из авторов статьи библиотекой функционального программирования для языка C++ [7]. При использовании этой библиотеки всю специфику вычислительного устройства (CPU или CUDA) можно поместить внутрь библиотеки, и пользовательский исходный код останется платформонезависимым. С другими аналогичными работами авторов можно также ознакомиться в работах [8] и [9].

Настоящая работа состоит из трех основных частей: краткое введение в функциональное программирование (в объеме, необходимом для понимания остального текста), краткое описание библиотеки функционального программирования funcprog и описание применения этой библиотеки для решения численных задач.

2. КРАТКОЕ ВВЕДЕНИЕ В ФУНКЦИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ

В функциональном программировании центральным объектом является (как это и следует из названия) функция. Функции являются полноправными участниками вычислительного процесса, такими же, какими при обычных вычислениях являются числа. Это значит, что функция может быть передана как параметр другой функции и может быть возвращена как результат работы функции (иногда функции, принимающие в качестве параметров другие функции, называют функциями высшего порядка). Функцию можно вычислить так же, как при обычных вычислениях можно вычислить число. Простой пример — композиция двух одноместных функций, которая возвращает новую одноместную функцию, вызывающую последовательно обе функции. В специализированных функциональных языках программирования (таких как Haskell [10]) такие возможности встроены в язык, в то время как реализация композиции функций на языке C++ является нетривиальной задачей, требующей специальных ухищрений. Примеры будут приводиться на языке Haskell, так как этот язык позволяет записывать многие вещи максимально кратко и в то же время понятно.

2.1. Принципы функционального программирования

Функциональное программирование имеет ряд особенностей по сравнению с императивным программированием, которые можно сформулировать в виде нескольких принципов. Некоторые из этих принципов являются обязательными для функционального программирования и поддерживаются всеми языками и библиотеками функционального программирования, а другие — опциональными, то есть в некоторых языках и библиотеках функционального программирования могут отсутствовать. По тому, насколько полно эти принципы реализованы в языке или библиотеке функционального программирования, можно судить о степени ее “функциональности”.

Первый и главный обязательный принцип, уже упоминавшийся выше, — это то, что функция является полноценным участником вычислительного процесса и может быть как передана в качестве параметра, так и возвращена как результат работы некоторой функции. Другой обязательный принцип — наличие лямбда-выражений. Лямбда-выражение — это такое выражение в языке, результатом которого является функция. Собственно, первый принцип без второго практически невозможен. Как правило, функция, возвращающая в качестве результата функцию, фактически возвращает лямбда-выражение. В частности, в современном стандарте языка C++ (начиная с версии C++11) лямбда-выражения имеются. Остальные принципы функционального программирования не столь важны и часто отсутствуют, но их наличие суще-

ственно повышает возможности языка или библиотеки. Опишем основные из них.

“Чистые” функции. Под “чистотой” функции в функциональном программировании подразумевается отсутствие у функции побочных эффектов. Это значит, что результат, возвращаемый функцией, зависит только от переданных аргументов и больше ни от чего.

Следующий принцип функционального программирования — **неизменяемые (immutable) переменные**. Это как если бы в вашей программе на C++ все переменные имели бы модификатор **const** (**int const** i = 5;). Переменные есть, но им что-то присвоить можно только один раз при создании. Именно с такими переменными работают все функциональные языки программирования (Haskell, Lisp). Этот принцип позволяет гарантировать “чистоту” функции, то есть то, что ее результат зависит только от ее параметров, и при одном и том же наборе параметров она всегда возвращает одно и то же. Функция не меняет значение ни одной переменной (потому что не может), значит, она “чистая”.

Каррирование. Названо по фамилии американского математика и логика Хаскелла Карри. Принцип каррирования состоит в том, что при неполном указании параметров функции ошибки не происходит, а вместо этого генерируется функция с меньшим (равным числу недостающих) числом параметров. Реализовано во многих современных функциональных языках программирования (в частности, в языке Haskell). Каррирование позволяет функцию с несколькими аргументами рассматривать как набор функций с одним аргументом.

Ленивые вычисления. Этот принцип в языке Haskell является прямым следствием принципа каррирования. В языке Haskell каррирование “идет до конца”. Это значит, что даже при передаче всех параметров функции фактического вызова не происходит. При применении каждого следующего параметра порождается функция с меньшим на единицу числом параметров, и после применения всех параметров получается функция без параметров. Именно эта функция без параметров передается в качестве аргумента. То есть фактически любые вычисления в языке Haskell — это вычисления функций.

η-редукция (эта редукция, или η-преобразование). Рассмотрим пример. Пусть мы хотим написать функцию с одним параметром — списком чисел, возвращающую список синусов этих чисел. Текст этой функции на языке Haskell очевидный:

```
mapsin lst = map sin lst
```

В этом примере последний параметр функции mapsin передается как последний параметр в правой части. Принцип η-редукции гласит, что в подобных случаях последний параметр в определении функции можно опускать, то есть определение функции mapsin можно записать короче:

```
mapsin = map sin
```

Композиция функций. Композиция функций настолько важна в функциональном программировании, что в языке Haskell эта операция максимально упрощена. Обычно рассматривают композицию одноместных функций (назовем их f и g), в языке Haskell она записывается так:

```
(f . g) x = f (g x)
map (exp . sin) [1,2,3] — пример
```

2.2. Математические основы функционального программирования

В основе функционального программирования (в частности языка Haskell) лежит современная математическая теория — теория категорий. Подробно с этой теорией можно ознакомиться, например, по источникам [11] и [12]. **Категорией** называется совокупность объектов, снабженных стрелками (морфизмами) между ними (некоторыми из них). Между двумя заданными объектами может быть много стрелок. Совокупность стрелок из объекта A в объект B в категории **C** обозначается $Hom_{\mathbf{C}}(A, B)$ или просто $Hom(A, B)$, если конкретная категория подразумевается.

Для стрелок имеются два обязательных условия. Во-первых, из каждого объекта должна существовать стрелка в него самого (“единичная” стрелка, или тождественный морфизм). Тождественный морфизм объекта A обозначается id_A. Таким образом, для любого объекта A $Hom(A, A)$ непусто (всегда имеется тождественный морфизм). Во-вторых, для любых двух стрелок $f \in Hom(A,B)$ и $g \in Hom(B,C)$ должна существовать их композиция $g \circ f \in Hom(A,C)$. Для существования композиции морфизмов важно, чтобы конец первой (правой) стрелки f совпадал с началом второй (левой) стрелки g. Обратите внимание, что первая стрелка указывается справа от оператора композиции, а вторая — слева. Композицию стрелок $g \circ f$ можно читать как “g после f”.

Морфизмы, в которых начало и конец совпадают (морфизмы из некоторого объекта в него самого), называются **эндоморфизмами**. Множество эндоморфизмов объекта A: $End(A) = Hom(A, A)$ является **моноидом** относительно операции композиции с еди-

ничным элементом id_A. Напомним, что моноидом называется множество с определенной на нем бинарной ассоциативной операцией и нейтральным (единичным) элементом относительно этой операции.

Функторами называются отображения категорий, сохраняющие структуру исходной категории. Точнее, функтор $F : \mathbf{C} \to \mathbf{D}$ ставит в соответствие каждому объекту в категории **C** объект в категории **D** и каждому морфизму $F: A \to B$ в категории **C** морфизм $F(f): F(A) \to F(B)$ в категории **D** так, что

- $F(id_A) = id_{F(A)}$

и для двух морфизмов $f: A \to B$ и $g: B \to C$ в категории **C**

- $F(g) \circ F(f) = F(g \circ f)$.

Функторы из категории **C** в категорию **D** также образуют категорию (категорию функторов), морфизмами в которой являются так называемые **естественные преобразования**. Естественное преобразование предоставляет способ перевести один функтор в другой, сохраняя внутреннюю структуру (например, композиции морфизмов).

Пусть F и G — функторы из категории **C** в категорию **D**. Тогда естественное преобразование η: $F \Rightarrow G$ сопоставляет каждому объекту X в категории **C** морфизм η_X: $F(X) \to G(X)$ в категории **D**, называемый компонентой η в X, так, что для любого морфизма $f: X \to Y$ в категории **C** диаграмма (в категории **D**), изображенная на рисунке ниже, коммутативна.

$$\begin{array}{ccc} X & F(X) & \xrightarrow{\eta_X} G(X) \\ {\scriptstyle f}\downarrow & {\scriptstyle F(f)}\downarrow & \downarrow{\scriptstyle G(f)} \\ Y & F(Y) & \xrightarrow[\eta_Y]{} G(Y) \end{array}$$

Коммутативность данной диаграммы означает, что из $F(X)$ в $G(Y)$ можно прийти двумя разными путями, или что выполнено следующее равенство:

$$\eta_Y \circ F(f) = G(f) \circ \eta_X.$$

Можно определить внешнюю (external) бинарную операцию между функторами и естественными преобразованиями (в английской терминологии **whiskering**). Пусть **B**, **C**, **D**, **E** — категории, K: $\mathbf{B} \to \mathbf{C}$, F, G: $\mathbf{C} \to \mathbf{D}$, H: $\mathbf{D} \to \mathbf{E}$ — функторы, η: $F \Rightarrow G$ — естественное преобразование. Тогда мы можем определить естественное преобразование $H\eta$: $H \circ F \Rightarrow H \circ G$, задав

$$(H\eta)_X = H(\eta_X), X \in \mathbf{C}.$$

Здесь задается равенство морфизмов в категории **E**. Аналогично мы можем определить естественное преобразование ηK: $F \circ K \Rightarrow G \circ K$, задав

$$(\eta K)_X = \eta_{K(X)}, X \in \mathbf{B}.$$

Здесь задается равенство морфизмов в категории **D**.

Функторы из категории в нее саму называются **эндофункторами**. Эндофункторы играют важную роль в теории категорий. Между любыми двумя эндофункторами в некоторой категории **C** определена композиция (так как начало и конец совпадают), причем эта композиция ассоциативна, а также существует единичный эндофунктор (обозначим его как $1_{\mathbf{C}}$), оставляющий все объекты и морфизмы в категории **C** без изменения. Эндофункторы в некоторой категории **C** образуют свою категорию с естественными преобразованиями в качестве морфизмов.

Определим теперь теоретико-категорное понятие монады. **Монадой** в теории категорий называется тройка (T, η, μ), где

- T — эндофунктор в некоторой категории **K** (T: $\mathbf{K} \to \mathbf{K}$);
- η: $1_{\mathbf{K}} \to T$ – естественное преобразование;
- μ: $T^2 \to T$ — естественное преобразование;
- следующая диаграмма коммутативна (ассоциативность):

$$\begin{array}{ccc} T^3 & \xrightarrow{T\mu} & T^2 \\ {\scriptstyle \mu T}\downarrow & & \downarrow{\scriptstyle \mu} \\ T^2 & \xrightarrow[\mu]{} & T \end{array}$$

- следующая диаграмма коммутативна (двухсторонняя единица):

$$\begin{array}{ccc} T & \xrightarrow{\eta T} & T^2 \\ {\scriptstyle T\eta}\downarrow & & {\scriptstyle \mu}\downarrow \\ T^2 & \xrightarrow[\mu]{} & T \end{array}$$

Из определения моноида (см. выше) видно, что монада является моноидом в категории эндофункторов **End**(**K**).

2.3. Функторы и монады в программировании

Функторы. Пусть у нас есть некоторый контейнер, хранящий какое-то количество значений, например, список или объект класса Maybe (хранящий одно значение указанного типа или не хранящий

ничего, в стандартной библиотеке C++ этому классу соответствует класс std::optional). Теперь поставим задачу: применить обычную одноместную функцию (например, sin) к значениям в контейнере. Как это сделать в языке Haskell со списками известно — применить функцию map. Но как это сделать с типом Maybe, и в общем случае — как это сделать данными в произвольном контейнере? Универсальный подход состоит в том, чтобы доверить это ответственное дело самому контейнеру. Для этого в языке Haskell определен специальный класс Functor, в котором продекларирована функция fmap:

```
class Functor f where
  fmap :: (a -> b) -> f a -> f b
  <$> = fmap
```

Оператор <$> — синоним функции fmap. Это оператор применения функции к функтору. Он похож на оператор применения функции к обычному значению ($). Прототип функции fmap можно записать в другом эквивалентном виде (это следует из правоассоциативности стрелки вправо):

```
fmap :: (a -> b) -> (f a -> f b)
```

Из последней записи следует, что функцию fmap можно рассматривать как функцию с одним параметрам (“обычной” функцией, то есть функцией, принимающей и возвращающей простые значения), возвращающую функцию, принимающую и возвращающую значения в контейнере.

Любой тип данных можно объявить функтором, реализовав для него экземпляр класса Functor и функцию fmap. Любая реализация функтора должна удовлетворять двум функторным законам:

```
1. fmap id = id
2. fmap (g . f) = fmap g . fmap f
```

Здесь id — функция, возвращающая свой аргумент: id x=x. Первый закон гласит: применение функции id к функтору не должно менять функтор, так же, как применение этой функции к обычному значению его не меняет. Второй закон — это распределительный закон функториой операции относительно композиции функций.

Аппликативы. Если стоит задача применить функцию с двумя аргументами к двум контейнерам (например, просуммировать два списка), то функционала класса Functor будет недостаточно. Для решения этой задачи предназначен другой класс — аппликативный функтор (аппликатив). Вот определение класса Applicative:

```
class Functor f => Applicative f where
  pure :: a -> f a
  (<*>) :: f (a -> b) -> f a -> f b
```

Таким образом, в каждом аппликативе должны быть реализованы две основные операции: функция pure, помещающая обычное значение в “чистый” аппликатив, и оператор (<*>), принимающий в качестве первого параметра функцию, помещенную в аппликатив, и второго параметра — значение, помещенное в тот же аппликатив, и возвращающий результат в том же аппликативе.

Любая реализация аппликатива должна удовлетворять аппликативным законам:

```
1. (Identity)
  pure id <*> v = v
2. (Homomorphism)
  pure f <*> pure x = pure (f x)
3. (Interchange)
  u <*> pure y = pure ($ y) <*> u
4. (Composition)
  pure (.) <*> u <*> v <*> x =
  u <*> (v <*> x)
```

Монады. Напишем “безопасные” функции safe_sqrt и safe_log:

```
safe_sqrt x=if x<0 then Nothing else
  Just(sqrt x)
safe_log x=if x<=0 then Nothing else
  Just(log x)
```

Эти функции возвращают результат типа Maybe, причем если аргумент функции принадлежит области ее определения, то возвращается значение Just value, а иначе — Nothing (признак ошибки). Пусть теперь мы хотим вычислить квадратный корень от логарифма числа. Для обеих операций у нас есть “безопасные” функции, возвращающие результат типа Maybe и проверяющие, что значение аргумента принадлежит области определения функции. Как нам теперь применить функцию safe_sqrt к результату функции safe_log? Функционала функтора и аппликатива для этого недостаточно. Для этой цели служат монады. Монады можно рассматривать как дальнейшее продолжение аппликатива, они предназначены для построения цепочек монадных вычислений.

Определение класса Monad в языке Haskell выглядит так:

```
class Applicative m => Monad m where
  return :: a -> m a
  return = pure
  (>>=) :: m a -> (a -> m b) -> m b
```

Каждая монада имеет две основные функции: **return** и bind (в языке Haskell оператор (>>=). Функция **return** аналогична функции pure для аппликативов (фактически для большинства монад **return** определяется как pure). Операция bind принимает в качестве параметров монаду и функцию, принимающую обычное (не монадное) значение и возвращающую монадное значение (возможно, другого типа, но в той же монаде). Будем называть такие функции монадными. Функции safe_log и safe_sqrt (а также функция **return**) — примеры монадных функций.

Функции **return** и bind должны удовлетворять трем монадным законам. Для того чтобы их сформулировать, введем операцию монадной композиции (оператор (>=>) в языке Haskell). Она определяется следующим образом:

```
(>=>) :: f >=> g = \x -> f x >>= g
```

Оба операнда монадной композиции и ее результат — монадные функции. Следовательно, монадную композицию можно рассматривать как групповую операцию в пространстве таких функций. В терминах этой групповой операции монадные законы формулируются так:

```
1. return >=> f == f
2. f >=> return == f
3. (f >=> g) >=> h == f >=> (g >=> h)
```

Другими словами, функции **return** и bind должны быть определены так, чтобы, во-первых, функция **return** являлась единичным элементом (левым и правым) монадной композиции (первые два закона) и, во-вторых, монадная композиция должна быть ассоциативной (третий закон).

Так как любая монада также является функтором, то для любой монады операцию bind можно определить через операцию fmap следующим образом:

```
x >>= f = join (f <$> x)
```

Функция fmap в данном случае вернет значение “монады в монаде” (например, список списков). Функция join убирает этот один лишний уровень вложенности. Функции fmap и join можно определить через монадные операции:

```
fmap f m = m >>= (return . f)
join n = n >>= id
```

Таким образом, по-умолчанию, bind, join и fmap циклически определяются друг через друга. Чтобы разорвать этот замкнутый круг, нужно какие-то из этих операций определить явно. Как правило, явно определяются fmap и bind.

Если сравнить математическое и “программистское” определения монады, то можно заметить, что естественному преобразованию соответствует функция **return**, а естественному преобразованию μ — функция join.

Возвращаясь к нашим “безопасным” функциям, заметим, что теперь квадратный корень от логарифма можно вычислить так:

```
safe_log 5 >>= safe_sqrt -- Just
  1.2686362411795196
safe_log (-5) >>= safe_sqrt -- Nothing
safe_log 0.5 >>= safe_sqrt -- Nothing
```

Если где-то в цепочке вычислений возникает ошибка, то происходит быстрый выход из всей цепочки вычислений, остальные функции фактически не вычисляются. Это чем-то напоминает исключения (exceptions) в императивных языках (таких, как C++).

3. БИБЛИОТЕКА ФУНКЦИОНАЛЬНОГО ПРОГРАММИРОВАНИЯ

3.1. Общее описание библиотеки

При реализации библиотеки функционального программирования funcprog для языка C++ ставилась задача написать библиотеку, с помощью которой на языке C++ можно было бы писать в стиле, близком к стилю языка Haskell.

Важный вопрос — что такое функция с точки зрения этой библиотеки? В первоначальной версии библиотеки под функцией подразумевался объект класса std::function. Этот вариант нас теперь не устраивает, так как мы хотим, чтобы функция исполнялась на графическом ускорителе, а объект класса std::function может исполняться только на CPU (главным образом потому, что в ее реализации используются виртуальные функции, которые на GPU непереносимы). Это не может быть и обычная функция. Адрес обычной функции передать в CUDA, конечно же, можно, но это не имеет смысла, так как у CPU и у CUDA несовместимые наборы инструкций. Было принято решение в рамках библиотеки funcprog2 создать класс function2 и считать функцией любой объект этого класса. В объект класса function2 можно преобразовать любой функциональный объект (имеющий функциональный оператор ()), в частности, это может быть лямбда-выражение. Напомним, что в современном языке C++ (начиная с версии C++11) лямбда-выражение

начинается с пары квадратных скобок, внутрь которых могут быть помещены переменные, доступные из лямбда-выражения. Для того чтобы объект можно было передавать в CUDA, функциональный оператор должен быть помечен ключевым словом __device__. Чтобы пользовательский исходный код был платформо-независим, в библиотеке funcprog2 имеется макрос __DEVICE, который при компиляции для CUDA расширяется в ключевое слово __device__, а при компиляции для CPU — в пустую строку. Таким образом, функциональный оператор нужно пометить словом __DEVICE. Для преобразования функционального объекта в объект класса function2 в библиотеке имеется специальная функция _ (подчерк). Недостатком класса function2 по сравнению с классом std::function является то, что в метаданные функции попадает не только ее прототип (типы параметров и возвращаемое значение), но и реализация (класс объекта) — такова плата за возможность переноса на CUDA. Вот как реализован класс function2:

```
template<typename FuncType,
  typename FuncImpl> struct function2;
  template<typename Ret, typename... Args,
    typename FuncImpl>
  struct function2<Ret(Args...) ,FuncImpl>{
    function2(FuncImpl const& impl)  :
      impl(impl){}
    Ret operator()(Args... args)
      const {return impl(args...);}
    private: FuncImpl const impl;
};
```

Видно, что, в отличие от класса std::function, в класс function2 передаются два параметра шаблона. Для упрощения работы с такими функциями имеется вспомогательная функция _ (подчерк). Вот как она определена:

```
template<typename FuncImpl,typename Ret,
  typename... Args>
struct function2_type_ {
  using type = function2<Ret(Args...) ,
    FuncImpl>; };

// Common case for functors lambdas
template<class F>
struct function2_type : function2_type
  <decltype( F::operator())>{};

template<class Cls, typename Ret,
  typename... Args>
struct function2_type
  <Ret(Cls::*)(Args...)> :
  function2_type_<Cls, Ret, Args...>{};

template<class Cls, typename Ret,
  typename... Args>
struct function2_type
  <Ret(Cls::*)(Args...) const> :
function2_type_<Cls, Ret, Args...>{};

template<typename F>
using function2_type_t =
  typename function2_type<F>::type;

template<typename F>
function2_type_t<F> _(F f) { return f; }
```

Приведем пример работы с этой библиотекой:

```
double d=(_([](double x){return x*x;})&
_([](double x)return x + 1;}))(5);//36
```

В этом примере мы создали две функции (с помощью функции _), сделали их композицию (с помощью оператора &) и вызвали получившуюся составную функцию с параметром 5. В результате получили число 36.

В библиотеке funcprog2 достаточно полно реализовано каррирование функций. Для этого в библиотеке реализован оператор применения аргумента к функции с помощью оператора сдвига влево <<. При этом создается новая функция, имеющая на один параметр меньше. В частности, если исходная функция имела единственный параметр, то создастся функция без параметров. В библиотеке имеется также функция invoke_f0, которой можно передать любую функцию. Если переданная функция без параметров, то она будет вызвана и вернется результат ее исполнения. Если же переданная функция имеет параметры (один или больше), то будет просто возвращена эта функция. В библиотеке имеется также метафункция remove_f0, принимающая в качестве параметра шаблона тип функции. Если функция без параметров, то в переменной type вернется тип возвращаемого функцией значения, а если параметры есть, то тип самой функции.

3.2. Реализация функторов, аппликативов и монад

Реализация функторов, аппликативов и монад в библиотеке funcprog в чем-то похожа на реализацию этих понятий в языке Haskell. Любой класс может объявить себя функтором, аппликативом или монадой. Для этого достаточно для этого класса реализовать специализацию классов соответственно Functor, Applicative и Monad. В сам класс никаких изменений вносить не требуется.

Любой функтор или монада являются типом с одним параметром. В языке C++ типы с параметром реализуются с помощью шаблонов классов. Рассмотрим определение функтора на примере класса Maybe. Класс Maybe в библиотеке funcprog2 определен так:

```
template<typename A> struct Maybe;
```

Шаблон класса типом не является, и его нельзя передать как параметр шаблона другого класса. Поэтому определяется еще один класс (без шаблона) с именем _Maybe (с подчерком впереди). Это уже настоящий класс, его можно передавать как параметр шаблона:

```
struct _Maybe {};
```

Шаблон класса Maybe наследуется от этого класса:

```
template<typename A>
struct Maybe : std::optional<A>,_Maybe ... {
  ...
};
```

Специализации классов Functor, Applicative и Monad пишутся именно от этого класса _Maybe:

```
template<> struct Functor<_Maybe>{
  ...
};
```

Внутри специализации класса Functor нужно определить статическую функцию fmap. Специализации классов Applicative и Monad определяются аналогично. Для аппликатива методы называются pure и apply, а для монады — return_ и bind. При реализации статических методов этих классов нужно не забывать про выполнение функторных, аппликативных и монадных законов.

Заметим также, что в библиотеке funcprog2 в качестве функторного оператора используется оператор деления, а в качестве аппликативного — умножение.

4. ПРИМЕНЕНИЕ БИБЛИОТЕКИ ДЛЯ ЧИСЛЕННЫХ МЕТОДОВ

Технологически любая задача численного моделирования начинается с построения сетки в области расчета. Причем в областях сложной формы эта сетка, как правило, неструктурная. Интересующие исследователя сеточные функции могут быть заданы как в узлах ячеек, так и в их центрах.

На данном этапе исследования мы рассматриваем только нестационарные задачи и считаем, что для интегрирования по времени используются различные явные схемы, например, в программном комплексе, который мы использовали для перевода на GPU, это явная классическая схема Рунге–Кутты четвертого порядка. Неявные схемы, предполагающие решение линейных систем уравнений, в настоящий момент не рассматривались.

Если метод явный, то вычислять значения сеточных функций в разных точках сетки можно независимо друг от друга, и, следовательно, эти вычисления можно вести параллельно. Таким образом, каждый явный шаг (цикл по индексу сеточной функции) можно распараллеливать на общей памяти. Заметим, что MPI-параллелизация также возможна, но пока не рассматриваются. При расчетах на CPU распараллеливание циклов делается, как правило, с помощью OpenMP. При расчетах на CUDA методы распараллеливания свои, и они сильно отличаются от OpenMP. Чтобы скрыть метод распараллеливания от прикладного программиста-математика, реализующего численный метод, предлагается использовать библиотеку функционального программирования funcprog2 так, как это описывается далее.

4.1. Сеточные выражения и сеточные функции

Введем понятие сеточного выражения. Это объект, определенный для всех индексов сетки, то есть для любого объекта, являющегося сеточным выражением, можно узнать, чему равно его значение для любого индекса сетки. Простейшим частным случаем сеточного выражения является сеточная функция, которая свои значения просто хранит в памяти и их, если нужно, возвращает. Для сеточных выражений определяется шаблон класса grid_expression, от которого должны наследоваться все классы объектов, являющихся сеточными выражениями (в частности, класс grid_function также пронаследован от класса grid_expression). Таким образом, фраза “объект является сеточным выражением” означает, что класс этого объекта пронаследован от класса grid_expression. При таком наследовании используются шаблоны выражений [13] и шаблон проектирования CRTP (Curiously Recurring Template Pattern) [14], при котором в базовый класс в качестве параметра шаблона передается конечный класс. Про этот шаблон и другие методы метапрограммирования можно прочитать в книгах [15], [16], [18].

Любое сеточное выражение можно присвоить сеточной функции. Этот оператор присваивания проходит по всем индексам сеточной функции, которой присваивается сеточное выражение, для каж-

дого индекса запрашивает у сеточного выражения его значение и присваивает это значение сеточной функции по данному индексу. Этот оператор присваивания подразумевает, что значения для разных индексов можно вычислять независимо друг от друга, и, значит, их можно вычислять параллельно. Именно в этом операторе присваивания выполняется тот самый внутренний цикл по элементам сеточной функции. Метод распараллеливания этого цикла выбирается оператором присваивания в зависимости от того, каким компилятором компилируется программа. Если это компилятор для CUDA (определена переменная препроцессора __CUDACC__), то распараллеливание осуществляется с помощью CUDA, иначе — с помощью OpenMP. Таким образом, метод распараллеливания скрыт от прикладного программиста внутри этого оператора присваивания. Покажем, как реализован этот оператор присваивания.

Вначале введем понятие “исполнителя” (executor). Исполнитель — это объект, параллельно выполняющий цикл с указанным телом цикла. Тело цикла задается объектом-замыканием (closure), который все нужное для исполнения, кроме индекса цикла, хранит в своих переменных-членах, а индекс цикла передается ему в функциональном операторе. При этом этот функциональный оператор с разными значениями индекса цикла может быть вызван параллельно. Вот как реализован исполнитель для обычного процессора (CPU):

```
struct cpu_executor {
  template<class Closure>
  void operator()(size_t size,
    Closure const& closure) const {
#pragma omp parallel for
  for(size_t i = 0; i < size; ++i)
    closure(i);
  }
};
using default_executor = cpu_executor;
```

А вот как исполнитель реализован для CUDA (мы используем тот факт, что ядро (kernel) может быть полиморфной функцией):

```
#ifndef BLOCK1_SIZE
#define BLOCK1_SIZE 512
#endif

template<class Closure>
__global__ void cuda_exec(
  size_t size, Closure closure
){
  size_t const i = blockIdx.x *
    blockDim.x + threadIdx.x;
  if(i < size) closure(i);
}

struct cuda_executor {
  template<class Closure>
  void operator()(size_t size,
    Closure const& closure) const
{
  unsigned const
    dimGrid = (size +
      BLOCK1_SIZE - 1) / BLOCK1_SIZE,
    dimBlock = size < BLOCK1_SIZE ?
      size : BLOCK1_SIZE;
    cuda_exec<<<dimGrid, dimBlock>>>
      (size, closure)
  }
};
using default_executor = cuda_executor;
```

Теперь мы можем определить оператор присваивания сеточного выражения сеточной функции:

```
template<typename T>
template<class GE>
void vector_grid_function<T>::operator=
  (grid_expression<GE,
    typename GE::proxy_type> const& gexp)
{
  proxy_type gf_p = get_proxy();
  auto const gexp_p = gexp.get_proxy();
  default_executor()(m_local_size ,
    [gf_p, gexp_p] __DEVICE (size_t i) {
      const_cast<proxy_type&>
        (gf_p)[i] = gexp_p[i];
});
}
```

Говоря про сеточные функции, нужно упомянуть еще один аспект. GPU может работать только со своей памятью, это значит, что при работе на GPU сеточная функция должна память под свои данные запрашивать в памяти CUDA. С этим также проблем нет. Сеточные функции устроены так, что при компиляции на CUDA они запрашивают память в CUDA, иначе — в памяти CPU.

4.2. Объекты-заместители (proxy)

Шаблон класса сеточного выражения grid_expression определен следующим образом:

```
template<class E, class _Proxy = E>
struct grid_expression;
```

Здесь E — конечный класс, а _Proxy — класс заместителя. По умолчанию (если он не указан) класс заместителя совпадает с конечным классом. Он нужен для создания копии объекта. Дело в том, что единственный способ передачи параметров из памяти основного процессора в память CUDA — это передача по значению (то есть делается копия параметра). Передача по адресу и ссылке невозможна. По значению можно передавать только переменные простых типов (числа и указатели) и объекты классов, содержащие только простые типы. Кроме того, эти объекты не должны содержать виртуальных методов, и вызываемые в них методы должны быть доступны из CUDA. Далеко не все объекты удовлетворяют всем этим требованиям. Если же такой объект все-таки нужно передать из CPU в CUDA, то для него можно создать объект-заместитель, удовлетворяющий перечисленным требованиям и хранящий все основные данные из основного объекта. Есть и другая причина того, почему не всегда можно хранить ссылки и указатели на объекты даже на CPU. Дело в том, что в сложных выражениях могут появляться временные безымянные переменные, у которых нет постоянной памяти и ссылки и указатели на которые сохранять нельзя. Такие объекты необходимо копировать. Всегда копировать объекты тоже нельзя, так как бывают “большие” объекты (например, векторы данных), которые при копировании делают копию этих данных. Общее правило следующее. Если объект “маленький” и не имеет виртуальных методов, то его, как правило, можно копировать, и класс-заместитель не нужен, иначе такой класс необходим.

4.3. Сеточные выражения как функторы, аппликативы и монады

Сеточные выражения можно рассматривать как контейнеры (особенно это справедливо для сеточных функций). В библиотеке funcprog контейнеры (например, списки) являются функторами, аппликативами и монадами. Это дает возможность применять к значениям, хранящимся в них, обычные функции (свойство функторов). Сделаем сеточное выражение также функтором, аппликативом и монадой, чтобы и к сеточным выражениям можно было применять функции. Чтобы понять, как это можно сделать, рассмотрим типичный цикл, вычисляющий новое значение сеточной функции по старой:

```
for(size_t i = 0; i < N; ++i)
  f_new[i] = calculate(f_old[i]);
```

здесь calculate — функция, вычисляющая новое значение в ячейке по старому. Ей в качестве параметра передается старое значение в ячейке. В новом подходе мы хотим, чтобы в этом случае можно было написать что-то типа:

```
f_new = _(calculate) / f_old;
```

Если же для вычислений требуются еще несколько сеточных функций (назовем их f2 и f3), то вместо

```
for(size_t i = 0; i < N; ++i)
f_new[i] = calculate(f_old[i], f2[i],
  f3[i]);
```

мы могли бы написать:

```
f_new = _(calculate) / f_old * f2 * f3;
```

то есть для первой сеточной функции мы применили свойство функтора, а для последующих — аппликатива. Если мы хотим в функцию передать еще дополнительно некоторое постоянное значение (не зависящее от индекса цикла), то вместо

```
for(size_t i = 0; i < N; ++i)
f_new[i] = calculate(f_old[i],
  some_value);
```

можно было бы написать

```
f_new = _(calculate) / f_old *
  pure(some_value);
```

Таким образом, результатом применения функции к сеточному выражению (или к нескольким сеточным выражениям в случае аппликатива) должно быть также сеточное выражение, то есть у него можно запросить значение по индексу (должен быть реализован оператор []). Сеточными выражениями, помимо сеточных функций, являются также результаты применения функций к сеточным выражениям как к функторам и монадам. Кроме того, сумма и разность двух сеточных выражений, а также произведение и частное сеточного выражения и числа также являются сеточными выражениями.

Покажем, как реализованы функторы, аппликативы и монады для сеточных выражений и докажем, что эти реализации корректны (удовлетворяют требуемым законам). Доказывать теоремы будем методом эквационального рассуждения (equational reasoning). Этот метод доказательства основан на приведении левой и правой частей доказываемого равенства путем эквивалентных преобразований к одинаковому выражению. Пусть теперь f — применяемая функция, а gexp — сеточное выражение.

Функтор. Функция *fmap* принимает функцию с одним параметром и функтор (в нашем случае сеточное выражение) и возвращает тот же функтор (новое сеточное выражение). Это новое сеточное выражение запоминает параметры функции *fmap* в своих переменных-членах класса (назовем их f и gexp) и реализует оператор [] следующим образом:

```
(fmap f gexp)[i] = f gexp[i]
```

Теорема 1 (Теорема о функторе) *Определенная выше функция fmap удовлетворяет функторным законам.*

Доказательство. Перепишем первый функторный закон полностью (без η-редукции):

```
fmap id gexp = id gexp
```

или (по определению функции id):

```
fmap id gexp = gexp
```

Далее,

```
(fmap id gexp)[i] = -- def of fmap
  id gexp[i] = -- def of id
  gexp[i]
```

то есть, действительно, fmap id gexp = gexp. Первый закон доказан. Перепишем второй функторный закон без η-редукции:

```
fmap(g.f) gexp = (fmap g . fmap f) gexp
```

Тогда

```
(fmap (g . f) gexp)[i] = -- def of fmap
  (g . f) gexp[i] = -- def of
    function composition
  g (f gexp[i])
```

С другой стороны:

```
((fmap g . fmap f) gexp)[i] =        -- def of
    function composition
  (fmap g (fmap f gexp))[i] =        -- fmap
  g (fmap f gexp)[i] =               -- fmap
  g (f gexp[i])
```

Теорема доказана. Определение функтора корректно.

Аппликатив. Функция *pure* принимает некоторое значение и “вносит” его в аппликатив. В нашем случае делает из него сеточное выражение. Определим его оператор [] так, чтобы он для любого индекса возвращал одинаковое значение val:

```
(pure val)[i] = val
```

Функция *apply* (аналог оператора (<*>) в языке Haskell) в нашем случае принимает два сеточных выражения: первый (назовем его gexp_f) возвращает функции, а второй (назовем его gexp) — некоторые значения (параметры этих функций). Определим сеточное выражение функции *apply* следующим образом:

```
(apply gexp_f gexp)[i] =
  gexp_f[i] gexp[i]
```

Теорема 2 (Теорема об аппликативе) *Определенные выше функции pure и apply удовлетворяют аппликативным законам.*

Доказательство. Перепишем аппликативные законы с использованием функции *apply*:

```
1. apply (pure id) gexp = gexp
2. apply (pure f) (pure x) = pure (f x)
3. apply u (pure y) = apply(pure($ y)) u
4. apply (apply (apply (pure (.) ) u) v) x =
   apply u (apply v x)
```

Первый закон (Identity):

```
(apply (pure id) gexp ) | i | =          -- apply
  (pure id) | i | gexp | i | id gexp|i| = -- pure
  id gexp|i| =                           -- id
  gexp [i]
```

т.е. apply (pure id) gexp = gexp. Первый закон доказан.

Второй закон (Homomorphism):

```
(apply (pure f) (pure x))[i] = -- apply
  (pure f)[i] (pure x)[i] =    -- pure
  f x
```

С другой стороны:

```
(pure (f x))[i] = -- def of pure f x
```

Второй закон доказан. Третий закон (Interchange):

```
(apply u (pure y))[i] = -- def of apply
  u[i] (pure y)[i] =    -- def of pure
  u[i] y
```

С другой стороны:

```
(apply (pure ($ y)) u)[i] =  -- apply
  (pure ($ y))[i] u[i] =     -- pure
  ($ y) u[i] =               -- function
    application
  u[i] y
```

Третий закон доказан. Четвертый закон (Composition):

```
(apply (apply (apply (pure (.) ) u) v) x)[i] =
  (apply(apply (pure (.)) u) v[i] x[i] =
  (apply (pure (.)) u) [i] v[i] x[i] =
  (pure(.)) [i] u[i] v[i] x[i] =
  (.) u[i] v[i] x[i] =   -- rewrite
    function composition in infix form
  (u[i] . v[i]) x[i] =   -- function
    composition
  u[i] (v[i] x[i])
```

С другой стороны:

```
(apply u (apply v x))[i] =          -- apply
  u[i] (apply v x)[i] =             -- apply
  u[i] (v[i] x[i])
```

Четвертый закон доказан. Теорема доказана. Определение аппликатива корректно. □

Монада. Монадная функция *return* определена так же, как и аппликативная функция *pure*:

```
(return val)[i] = val
```

Монадная операция *bind* (в языке Haskell и в библиотеке funcprog оператор >>=) принимает монаду (в нашем случае сеточное выражение, обозначим его переменной gexp) и функцию, принимающую обычное (не монадное) значение и возвращающую монаду (сеточное выражение). Определим операцию *bind* следующим образом:

```
(bind gexp f)[i] = (f gexp[i])[i]
```

Теорема 3 (Теорема о монаде) *Определенные выше функции return и bind удовлетворяют монадным законам.*

Доказательство. Перепишем монадные законы в терминах функций return и bind:

```
1. bind (return x) f = f x
2. bind gexp return = gexp
3. bind (bind gexp f) g =
   bind gexp (\x->bind (f x) g)
```

Первый закон:

```
(bind (return x) f)[i] =          -- bind
  (f (return x)[i])[i] =          -- return
  (f x)[i]
```

Первый закон доказан. Второй закон:

```
(bind gexp return)[i] =           -- bind
  (return gexp[i])[i] =           -- return
  gexp[i]
```

Второй закон доказан. Третий закон:

```
(bind (bind gexp f) g)[i] =       -- bind
  (g (bind gexp f)[i])[i] =       -- bind
  (g (f gexp[i])[i])[i]
```

С другой стороны:

```
(bind gexp (\x -> bind (f x) g))[i] =
  ((\x->bind (f x) g) gexp[i])[i] =
    -- beta-reduction, substitute
    -- gexp[i] instead of x
  (bind (f gexp[i]) g)[i] =
  (g (f gexp[i])[i])[i]
```

Результат получился одинаковый. Третий закон доказан. Теорема доказана. □

Итак, сеточные выражения являются функторами, аппликативами и монадами. Это значит, что любую обычную унарную функцию можно “применить” к сеточному выражению (с помощью функции *fmap*). Такое применение вернет новое сеточное выражение. Любую бинарную функцию можно “применить” к двум сеточным выражениям (с помощью функции *apply*), а любую функцию с n аргументами можно “применить” к n сеточным выражениям. Это можно сделать одной строчкой. Например, пусть есть две сеточных функции f и g. Тогда можно написать так:

```
g = _([](double x){return sin(x);} / f;
```

Так как сеточные выражения являются монадами, то из них можно строить цепочки монадных вычислений вида:

```
g = f >>= f1 >>= f2 >>= f3;
```

Здесь f1, f2, f3 — некоторые функции, принимающие обычные (не монадные) значения и возвращающие сеточные выражения.

5. ПРИМЕРЫ ИСПОЛЬЗОВАНИЯ

5.1. Простейший пример

Рассмотрим функцию axpy из библиотеки BLAS. Эта функция принимает константу a и два вектора (x и y) и модифицирует вектор y по формуле $y[i] += a * x[i]$. Ее реализацию для обычного процессора можно записать так:

```
template<typename T>
void axpy(T a, vector<T> const& x,
```

```
  vector &y) {
#pragma omp parallel for
  for(int i = 0; i < y.size(); ++i)
    y[i] += a*x[i];
}
```

Эта функция прекрасно распараллеливается, но способ распараллеливания в данной реализации указан явно и для графических ускорителей не подходит. С использованием функционального программирования эту функцию можно было бы переписать следующим образом:

```
template<typename T>
void axpy(T a,grid_function<T> const& x,
     grid_function<T> &y){
  y = _([](T a,T xi,T &yi,size_t/*i*/){
    yi += a * xi;
  }) / pure(a) * x;
}
```

Лямбда-выражение в теле этой функции принимает в качестве параметров нашу константу a, i-й элемент сеточной функции x, по ссылке i-й элемент сеточной функции y и текущий индекс цикла i (который мы игнорируем). Каждому входному параметру (а их у нас два) соответствует один параметр лямбда-выражения, а сеточной функции, которой делается присваивание выражения, соответствует выходной параметр (передаваемый по ссылке) и индекс цикла. Лямбда-выражение передается функции _ (подчерк), которая преобразует это лямбда-выражение в объект класса function2 (функцию в понятиях библиотеки). Этому объекту передаются входные параметры, первый — как для функтора, а следующие — как для аппликатива. Вот пример обращения к функции axpy:

```
int main(){
  size_t const N = 10;
  grid_function<double> x(N, 2), y(N, 3);
  axpy(5., x, y);
  std::cout << y[0] << std::endl; // 13
  return 0;
}
```

5.2. Более сложный пример

Теперь мы будем вычислять выражение вида z[i]=a*x[i]+b*y[ind[i]] (назовем функцию axpby). Ее особенностью является то, что индекс сеточной функции y содержится в дополнительной сеточной функции ind:

```
template<typename T>
void axpby(T a,grid_function<T>const& x,
  T b, grid_function<T> const& y,
  grid_function<size_t> const& ind,
  grid_function<T> &z)
{
  z = _([](T a, T xi, T b,
    grid_function_proxy<T> const y_p,
    size_t indi, T &zi, size_t /*i*/)
  {
    zi = a * xi + b * y_p[indi];
  }) / pure(a) * x * pure(b)*pr(y)*ind;
}
```

Из существенно нового по сравнению с первым примером здесь то, что сеточную функцию y мы уже не можем передавать через apply. Вместо этого нам нужно для нее создать объект-заместитель (proxy) и передать его через pure. Именно это и делает библиотечная функция pr:

```
template<class F>
auto pr(F const& f) {
  return pure(get_proxy(f));
}
```

Обратим еще внимание на то, что из GPU недоступны глобальные переменные, которые хранятся в памяти CPU (современные CUDA-платформы могут это позволять за счет механизма CUDA Unified Memory, но в любом случае это менее эффективно и, главное, доступно далеко не всегда), поэтому их приходится передавать явно с помощью функции pure. Если нужен буфер для промежуточных вычислений, то его размер должен быть равен числу узлов сетки, так как в случае CUDA мы не знаем абсолютный номер нити исполнения (thread).

Еще одна важная часто возникающая задача — это редукция (например, поиск максимального значения или сумма всех значений). Редукцию также очень важно выполнять параллельно. В исходной версии программы редукция выполнялась средствами OpenMP, но теперь мы этот механизм использовать не можем. К счастью, в современном языке C++ (начиная со стандарта языка C++17) у функций редукции (таких как accumulate и reduce) есть параллельная версия. При работе на CUDA мы используем библиотеку thrust, входящую в состав CUDA Computing Toolkit. В этой библиотеке также имеются параллельные функции редукции, работающие на GPU. Таким образом, при работе на CUDA мы используем параллельные функции редукции из библиотеки thrust, при работе на CPU, если имеются параллельные версии функций редукции (если компилятор их поддерживает), то используются они, иначе редукуция делается последовательно.

6. СРАВНЕНИЕ ЭФФЕКТИВНОСТИ

Мы решали некоторую задачу моделирования течения набегающего потока воздуха в трубе с препятствием с помощью гибридного суперкомпьютера K60, установленного в Центре коллективного пользования ИПМ им. М.В. Келдыша РАН [19], на одном узле в трех режимах: на обычном процессоре (2 x Intel Xeon Gold 6142 v4, 16 ядер) с использованием OpenMP и на GPU (nVidia Volta GV100GL) с использованием OpenCL и с использованием описанного подхода. Ускорение по сравнению с CPU при использовании OpenCL составило около 20 раз, а при использовании нашего подхода — около 12 раз. Ускорение получилось не таким большим, как при использовании OpenCL, но с учетом указанных преимуществ (прежде всего единый исходный код) его можно считать приемлемым.

7. ЗАКЛЮЧЕНИЕ

Декларативные языки программирования, к которым относятся и функциональные языки, позволяют, в отличие от императивных языков, к которым относятся большинство языков программирования, на которых реализуются численные методы, кратко и в то же время достаточно ясно записывать желаемый результат, не вдаваясь в подробности реализации. Конкретная реализация может быть скрыта в языке и зависеть от текущего программно-аппаратного окружения. Язык C++ оказался достаточно мощным, чтобы позволить реализовать на нем библиотеку функционального программирования, позволяющую писать программы в стиле, близком к стилю чисто функциональных языков, таких как Haskell. Такие понятия из мира функционального программирования, как функторы и монады, реализованные в библиотеке функционального программирования, оказались очень удобным средством для переноса численных задач на графические ускорители CUDA. Сеточные выражения были определены как функторы, аппликативы и монады, что позволило применять функции к хранящимся в них значениям. Сами эти функции можно строить, комбинируя сложные функции из простых, что является также сильной стороной функционального программирования. Авторам представляется, что за функциональным программированием будущее технологии программирования вообще и решения численных задач в частности. Мы надеемся, что данная работа будем нашим вкладом в этом направлении.

СПИСОК ЛИТЕРАТУРЫ

1. TOP 500. URL: https://www.top500.org
2. NVIDIA. URL: https://www.nvidia.com
3. TOP 50. URL: http://top50.supercomputers.ru
4. OpenCL. URL: https://www.khronos.org/opencl/
5. OpenACC. URL: https://www.openacc.org
6. CUDA Zone. URL: https://developer.nvidia.com/cuda-zone
7. *Краснов М.М.* Библиотека функционального программирования для языка C++ // Программирование. 2020. № 5. С. 47-59. DOI: 10.31857/S0132347420050040
8. *Краснов М.М.* Операторная библиотека для решения трехмерных сеточных задач математической физики с использованием графических плат с архитектурой CUDA // Математическое моделирование. 2015. Т. 27. № 3. С. 109-120. URL: http://www.mathnet.ru/links/38633e7a627ab2ce1527ae4a092be72f/mm3585.pdf
9. *Краснов М.М.* Канд. дисс. “Сеточно-операторный подход к программированию задач математической физики”. Автореф. URL: http://keldysh.ru/council/1/2017-krasnov/avtoref.pdf
10. Haskell language. URL: https://www.haskell.org/
11. *Маклейн С.* Категории для работающего математика / перевод с англ. под ред. В.А. Артамонова. М.: ФИЗМАТЛИТ, 2004. 352 с. ISBN 5-9221-0400-4.
12. *Bartosz Milewski.* Category Theory for Programmers. URL: https://github.com/hmemcpy/milewski-ctfp-pdf/releases/download/v1.3.0/category-theory-for-programmers.pdf
13. *Veldhuizen T.* Expression Templates. C++ Report. Vol. 7. № 5. June 1995. pp. 26-31.
14. *Coplien J.O.* Curiously recurring template patterns. C++ Report, February 1995, pp. 24–27.
15. *David Abrahams, Aleksey Gurtovoy.* C++ Template Metaprogramming. Addison-Wesley. 2004. 400 с. ISBN 978-0-321-22725-6.
16. *Краснов М.М.* Метапрограммирование шаблонов C++ в задачах математической физики. М.: ИПМ им. М.В. Келдыша, 2017. 84 с. http://dx.doi.org/10.20948/mono-2017-krasnov10.20948/mono-2017-krasnov.
17. *Краснов М.М.* Применение символьного дифференцирования для решения ряда вычислительных задач // Препринты ИПМ им. М.В. Келдыша. 2017. № 4. 24 с. http://dx.doi.org/10.20948/prepr-2017-410.20948/prepr-2017-4.
18. *Краснов М.М.* Применение функционального программирования при решении численных задач // Препринты ИПМ им. М.В. Келдыша. 2019. № 114. 36 с. http://dx.doi.org/10.20948/prepr-2019-11410.20948/prepr-2019-114.
19. Вычислительный комплекс К-60. https://www.kiam.ru/MVS/resourses/k60.html.

THE USE OF FUNCTIONAL PROGRAMMING LIBRARY TO PARALLELIZE ON GRAPHICS ACCELERATORS WITH CUDA TECHNOLOGY

M. M. Krasnov[a], O. B. Feodoritova[a]

[a] *Keldysh Institute of Applied Mathematics of Russian Academy of Sciences, Miusskaya sq., 4 Moscow, 125047, Russia*

Modern graphics accelerators (GPUs) can signi cantly speed up the execution of numerical tasks. However, porting programs to graphics accelerators is not an easy task, sometimes requiring their almost complete rewriting. CUDA graphics accelerators, thanks to technology developed by NVIDIA, allow you to have a single source code for both conventional processors (CPUs) and CUDA. However, in this single source code, you need to somehow tell the compiler which parts of this code to parallelize on shared memory. The use of the functional programming library developed by the authors allows you to hide the use of one or another parallelization mechanism on shared memory within the library and make the user source code completely independent of the computing device used (CPU or CUDA). This article shows how this can be done.

Keywords: C++; functional programming library; CUDA; OpenMP; OpenCL; OpenACC

REFERENCES

1. TOP 500. URL: https://www.top500.org
2. NVIDIA. URL: https://www.nvidia.com
3. TOP 50. URL: http://top50.supercomputers.ru
4. OpenCL. URL: https://www.khronos.org/opencl/
5. OpenACC. URL: https://www.openacc.org
6. CUDA Zone. URL: https://developer.nvidia.com/cuda-zone
7. *Krasnov M.M.* Functional Programming Library for C++ // Programming and Computer Software, 2020, v. 46, no. 5, pp. 330–340. http://dx.doi.org/10.1134/S0361768820050047
8. *Krasnov M. M.* Operator library for solving three-dimensional grid problems of mathematical physics using graphics cards with CUDA architecture // Mathematical Modeling, 2015, v. 27, no. 3, pp. 109-120. URL: http://www.mathnet.ru/links/38633e7a627ab2ce1527ae4a092be72f/mm3585.pdf
9. *Krasnov M. M.* Candidate's thesis "Grid-operator approach to programming problems of mathematical physics". Abstract. URL: http://keldysh.ru/council/1/2017-krasnov/avtoref.pdf
10. Haskell language. URL: https://www.haskell.org/
11. *McLane S.* Categories for the working mathematician / Translation from English edited by V.A. Artamonova. M.: FIZMATLIT, 2004. 352 p. ISBN 5-9221-0400-4.
12. *Milewski B.* Category Theory for Programmers. URL: https://github.com/hmemcpy/milewski-ctfp-pdf/releases/download/v1.3.0/category-theory-for-programmers.pdf
13. *Veldhuizen T.* Expression Templates. C++ Report, Vol. 7. 5, June 1995, pp. 26-31.
14. *Coplien J.O.* Curiously recurring template patterns. C++ Report, February 1995, pp. 24-27.
15. *Abrahams D., Aleksey Gurtovoy.* C++ Template Metaprogramming. Addison-Wesley. — 2004. 400 c. ISBN 978-0-321-22725-6.
16. *Krasnov M. M.* Metaprogramming of C++ templates in problems of mathematical physics. M.: IPM im. M.V. Keldysh, 2017. 84 p. http://dx.doi.org/10.20948/mono-2017-krasnov10.20948/mono-2017-krasnov.
17. *Krasnov M. M.* Application of symbolic differentiation to solve a number of computational problems // Preprints of IAM im. M.V. Keldysh. 2017. No. 4. 24 p. http://dx.doi.org/10.20948/prepr-2017-410.20948/prepr-2017-4.
18. *Krasnov M. M.* Application of functional programming in solving numerical problems // Preprints of IPM im. M.V. Keldysh. 2019. No. 114. 36 p. http://dx.doi.org/10.20948/prepr-2019-11410.20948/prepr-2019-114.
19. Computer complex K-60. URL: https://www.kiam.ru/MVS/resourses/k60.html.

ПРОГРАММИРОВАНИЕ, *2024*, *№ 1*, *с. 30–39*

ТЕОРЕТИЧЕСКИЕ ВОПРОСЫ ПРОГРАММИРОВАНИЯ

УДК 519.713.2

О ЛИНЕЙНЫХ КЛЕТОЧНЫХ АВТОМАТАХ

В. Р. Куликов[a], *, А. А. Кытманов[b], **, А. О. Порошин[a], *, И. В. Тимофеев[c, a], ****, Д. П. Федченко[c, a]**

[a] *Сибирский федеральный университет, 660041 Красноярск, пр. Свободный, 79*

[b] *МИРЭА — Российский технологический университет, 119454 Москва, пр. Вернадского, 78*

[c] *Институт физики им. Л.В. Киренского СО РАН, обособленное подразделение ФИЦ КНЦ СО РАН, 660036 Красноярск, Академгородок, 50*

**E-mail: v.r.kulikov@mail.ru*
***E-mail: aakytm@gmail.com*
****E-mail: poroshin.012332@gmail.com*
*****E-mail: tiv@iph.krasn.ru*
fdp@iph.krasn.ru

Поступила в редакцию 20.01.2023
После доработки 05.03.2023
Принята к публикации 05.04.2023

В работе рассматриваются вольфрамовские клеточные автоматы и демонстрируется их работа на примере задачи моделирования транспортного потока. Для класса одномерных элементарных клеточных автоматов на языке операторов Жегалкина вводится понятие линейности. Приводится алгоритм нахождения линейных операторов Жегалкина с мультипликаторами трех переменных. Алгоритм программно реализован на языке Python.

Ключевые слова: клеточный автомат, код Вольфрама, оператор Жегалкина
DOI: 10.31857/S0132347424010032 **EDN:** HOIZMS

1. ВВЕДЕНИЕ

Конечные автоматы представляют из себя абстрактные математические модели дискретных устройств, имеющих один вход и один выход и в каждый момент времени находящихся в некотором состоянии из множества возможных (см., например, [1], [2]). Они используются в задачах моделирования многих естественных и технологических процессов. Так, в работе [3] был предложен клеточный автомат, развивающийся по очень простым правилам, но моделирующий поведение различных сложных систем. Частным случаем конечных автоматов являются клеточные автоматы, которые используются как инструмент моделирования процессов в различных областях знания, таких как градостроительная наука ([4], [5]), задачи моделирования транспортных потоков [6], физика жидких кристаллов [7] и др. (см., например, [8], [9], [10]).

Клеточный автомат полностью определяется своим начальным состоянием и правилом эволюционирования, которое может быть задано десятичным числом, называемым кодом Вольфрама, впервые введенным в работах [11], [12].

В компьютерных технологиях клеточные автоматы рассматриваются в связи с низкоуровневыми и архитектурными вопросами, такими как аппаратная генерация случайных чисел [13], создание новой вычислительной архитектуры, основанной на квантовых точках [14], вычисления на GPU [15].

Многие вопросы, связанные со свойствами инъективности, сюръективности, обратимости клеточных автоматов, приводят к рассмотрению линейной теории (см., например, [16], [17], [18]).

В данной работе мы вводим понятие линейности для клеточного автомата на одномерной конечной (или бесконечной) решетке со значениями в некотором конечном поле, что позволяет построить матричное представление для таких систем в виде трехдиагональных теплицевых матриц. Для решения данной задачи строится взаимно-однозначное соответствие между кодами Вольфрама и операторами Жегалкина. Следует отметить, что многочлены Жегалкина хорошо известны в дискретной математике и являются удобным средством представления функций булевой логики [19].

2. КЛЕТОЧНЫЕ АВТОМАТЫ И КОДЫ ВОЛЬФРАМА

В работе [11] был введен класс элементарных клеточных автоматов. Рассмотрим бесконечную последовательность из пронумерованных клеток

$$s^0 = (\ldots, s^0_{-1}, s^0_0, s^0_1, \ldots), \quad s^0_i \in \{0,1\},\ i \in \mathbb{Z},$$

которую будем называть начальным состоянием клеточного автомата (здесь верхний индекс <<0>> соответствует начальному состоянию клеточного автомата). Зададим правила преобразования состояния за один шаг (такт времени). Верхний индекс соответствует номеру шага (такта). Окрестностью i-й клетки на k-м шаге назовем множество $U^k_i = (s^k_{i-1}, s^k_i, s^k_{i+1})$, состоящее из самой клетки и двух ее ближайших соседей. Множество всех типов окрестностей имеет мощность 2^3. Правилом клеточного автомата назовем трехместную булеву функцию

$$s^k_i = f\left(s^{k-1}_{i-1}, s^{k-1}_i, s^{k-1}_{i+1}\right),$$

определяющую состояние клетки s_i на k-м шаге в зависимости от состояния ее окрестности на $(k-1)$-м шаге. Очевидно, что для задания такой функции нужно определить ее значения на $2^3 = 8$ наборах аргументов

$$(0,0,0),(0,0,1),\ldots,(1,1,1). \qquad (1)$$

На векторном языке речь идет о задании восьмимерного бинарного вектора

$$\mathbf{r} = (r_0, \ldots, r_7), \quad \mathbf{r}\colon s^{k-1} \mapsto s^k,$$

где аргументы $r_0, \ldots, r_7$ соответствуют наборам (1), а в клетку s^k_i записывается значение r_j, $j = s^{k-1}_{i-1} \times 2^0 + s^{k-1}_i \times 2^1 + s^{k-1}_{i+1} \times 2^2$. Компонента r_j вектора r соответствует двоичному представлению числа j, записанного слева направо (рис. 1).

Определение 1. Элементарным клеточным автоматом назовем пару $\langle s_0, \mathbf{r}\rangle$.

В данной статье мы ограничимся рассмотрением клеточных автоматов, для которых правило преобразования предыдущего состояния одно и то же на каждом такте. Этим свойством не обладают, например, клеточные автоматы, моделирующие поведение фотонных топологических изоляторов [20]. Множество правил для класса элементарных клеточных автоматов состоит из $2^{2^3} = 256$ элементов. Правило $\mathbf{r}$ можно закодировать десятичным числом, равным

$$W = \sum_{j=0}^{7} r_j 2^j,$$

Рис. 1. Правило клеточного автомата с кодом Вольфрама W = 254.

которое мы будем называть кодом Вольфрама. Упорядочим правила клеточного автомата в соответствии с возрастанием кодов Вольфрама.

Правило клеточного автомата $\mathbf{r} = (0,1,1,1,1,1,1,1)$ с кодом Вольфрама W = 254 изображено на рис. 1, где темные клетки — нули, а светлые — единицы. В каждой Т-образной фигуре, соответствующей определенному виду окрестности, верхний ряд клеток — s^{k-1}_{i-1}, s^{k-1}_i, s^{k-1}_{i+1}, нижняя клетка — s^k_i.

Действие клеточного автомата с кодом Вольфрама W = 30 и с начальным состоянием $s^0 = (s^0_0, \ldots, s^0_{102})$, где $s^0_{52} = 1$, $s^0_i = 0$ при $i \neq 52$ изображено на рис. 2.

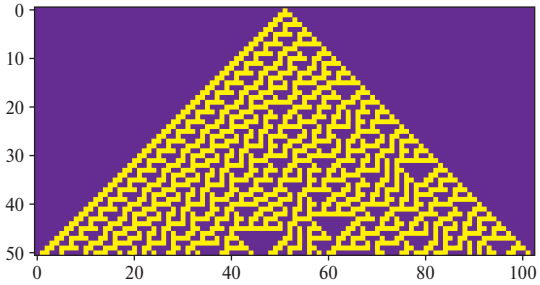


Рис. 2. Первые 50 тактов клеточного автомата с кодом Вольфрама W = 30 с одноточечным начальным состоянием.

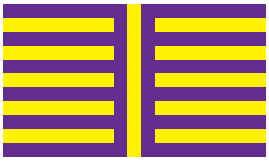
Иногда в случае работы с конечной последовательностью

$$s^0 = (s^0_0, s^0_1, \ldots, s^0_{n-2}, s^0_{n-1})$$

ее замыкают, положив $U^0_0 = (s^0_{n-1}, s^0_0, s^0_1)$ и $U^0_{n-1} = (s^0_{n-2}, s^0_{n-1}, s^0_0)$. Тем самым, избавившись от особенностей на границе, мы можем применять правила преобразования ко всем клеткам последовательности s^0.

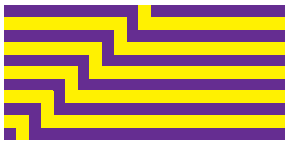
Для иллюстрации построения взаимно-однозначного соответствия между операторами Жегалкина и кодами Вольфрама рассмотрим наиболее простой нетривиальный пример одномерных клеточных автоматов с проколотой окрестностью $U^0_i = (s^0_{i-1}, s^0_{i+1})$, состоящей только из правого и левого соседей клетки. В этом случае правила задаются четырехмерными бинарными векторами, а общее количество правил в этом случае составляет $2^4 = 16$. На рис. 3 представлены все правила клеточного автомата, для которого состояние клетки зависит

$w_0 = [0,0,0,0]$

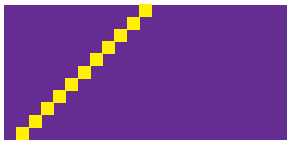


$w_1 = [1,0,0,0]$

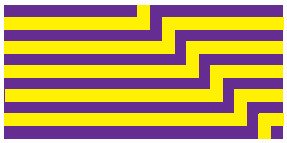
$w_2 = [0,1,0,0]$



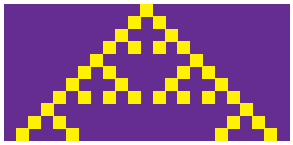
$w_3 = [1,1,0,0]$



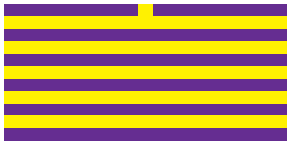
$w_4 = [0,0,1,0]$



$w_5 = [1,0,1,0]$

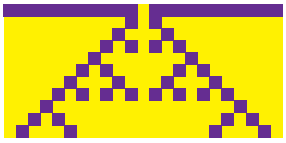


$w_6 = [0,1,1,0]$



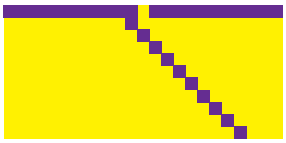
$w_7 = [1,1,1,0]$

$w_8 = [0,0,0,1]$

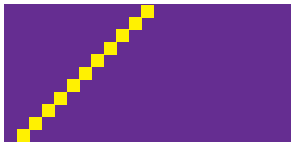


$w_9 = [1,0,0,1]$

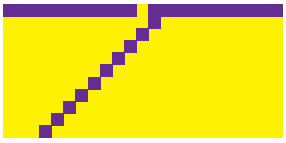
$w_{10} = [0,1,0,1]$



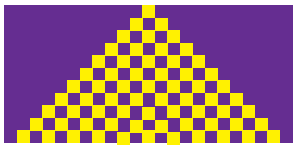
$w_{11} = [1,1,0,1]$



$w_{12} = [0,0,1,1]$



$w_{13} = [1,0,1,1]$



$w_{14} = [0,1,1,1]$

$w_{15} = [1,1,1,1]$

Рис. 3. Все правила клеточного автомата, для которого состояние клетки зависит только от состояний соседних клеток.

от состояний только правого и левого соседей. При этом индекс i в записи правила w_i является кодом Вольфрама соответствующего клеточного автомата.

Код Вольфрама фактически является числовым способом представления правила, управляющего поведением клеточного автомата, и вычисляется с помощью программно реализованного алгоритма. Увеличение размерности пространства или расширение окрестности клеточного автомата приводят к росту числа знаков, необходимых для записи кода Вольфрама определенного правила, так же, как и числа самих правил. Данное обстоятельство влечет необходимость разработки алгоритмов вычисления кодов Вольфрама, проверки правил на их соответствие заданным условиям и т.п.

Примером клеточных автоматов с *большими* кодами Вольфрама являются трехцветные клеточные автоматы, рассмотренные в статье [20]. В данной работе на языке таких автоматов дается описание игры Руднера, являющейся упрощенной моделью топологического изолятора. Количество правил в рассматриваемом классе клеточных автоматов существенно превосходит 10^{100}. Рассмотрим правила 1–4, где

правило 1 = 871896424859609582029110705858
607716968581963006295292858847170257072451849554615145673501346427619604753974631352 21;

правило 2 = 871896424859609582029110705858
607716968656290069375168566464205353070837488470095116884667048071141874921781551246 53;

правило 3 = 871896424859609582029110705858
607716968657588792402221282547761642305110322715914450484937842398638240540827194153 81;

правило 4 = 871896424859609582029110705858
607716968657588794939736287161814820368150788205250465383318162467733254390253505955 33.

Эти правила включаются в определенной последовательности для клеток, в зависимости от четности, которая определяется шахматной раскраской, в соответствии с табл. 1.

Таблица 1. Четырехтактный клеточный автомат, моделирующий игру Руднера

Параменты	Такт 1	Такт 2	Такт 3	Такт 4
Четная клетка	**Пр. 1**	**Пр. 2**	**Пр. 3**	**Пр. 4**
Нечетная клетка	**Пр. 3**	**Пр. 4**	**Пр. 1**	**Пр. 2**

Отметим, что предложенный клеточный автомат может быть реализован с помощью нарушенного полного внутреннего отражения в массиве, составленном из стеклянных призм [21], и описан на языке математических бильярдов (см., например, [22]).

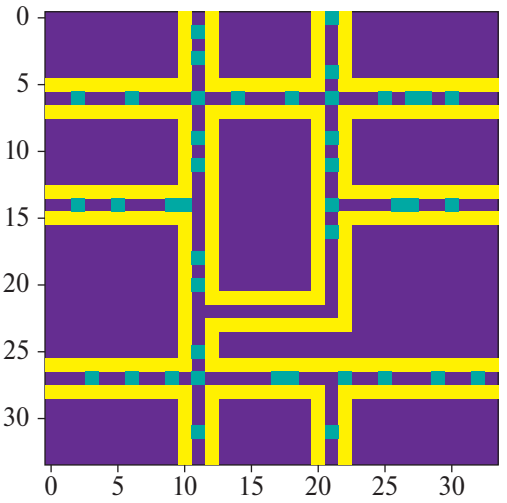


Рис. 4. Элемент транспортной сети, смоделированной с помощью клеточного автомата.

Проиллюстрируем, как предложенная в [20] модель может быть использована при моделировании транспортного потока. Для этого реализуем транспортную сеть в виде трехцветного клеточного автомата, представленного на рис. 4, где светлые клетки изображают край проезжей части, на которой расположены машины. Клеточный автомат работает на основе периодического четырехтактного правила, описанного в табл. 1. Клетки машин перемещаются по проезжей части. Для того чтобы машины не исчезали, элемент транспортной сети превращен в двумерный тор с помощью стандартной процедуры подклеивания противоположных сторон.

На каждом такте наличие машины в клетке приводит к инкрементированию значения, приписанного данной клетке (начальное значение равно 0), а тепловая карта показывает, в каких клетках чаще всего находилась машина. Через 200 тактов работы клеточного автомата получим график распределения машин на элементе транспортной сети, изображенный на рис. 5.

Отметим, что данный трехцветный клеточный автомат на элементе транспортной сети ведет себя согласовано с моделью Нагеля—Шрекенберга (см., например, [6, 2.2.4]).

3. ОПЕРАТОРЫ ЖЕГАЛКИНА

Исследуя подобные системы, мы применяем алгебраические методы в теории клеточных автоматов, переходя при этом на эквивалентный язык операторов Жегалкина.

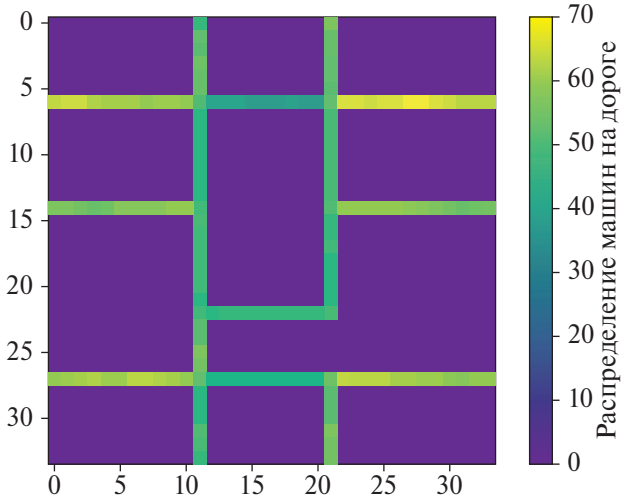


Рис. 5. Результат моделирования загруженности транспортной сети.

Определение 2. Многочленом Жегалкина d от n переменных над полем $\mathbb{Z}/2\mathbb{Z}$ назовем выражение

$$d(x_1,\ldots,x_n) = \bigoplus_I a_I x^I,$$

где $I=(i_1,\ldots,i_n)$, $i_k \in \mathbb{Z}/2\mathbb{Z}$ — мультииндекс длины n, $x^I = x_1^{i_1}\ldots x_n^{i_n}$, $x_k, a_I \in \mathbb{Z}/2\mathbb{Z}$.

Количество мономов в многочлене Жегалкина равно 2^n. Роль произведения здесь играет конъюнкция, а в качестве суммы выступает сложение по модулю 2.

Пример 1. В случае трех переменных многочлен Жегалкина имеет вид

$$d = a_0 \oplus a_1x_1 \oplus a_2x_2 \oplus a_3x_1x_2 \oplus \ldots \oplus a_7x_1x_2x_3. \quad (2)$$

Пример 2. Выпишем все шестнадцать многочленов Жегалкина $\mathrm{d}(x_1,x_2)$ от двух переменных.

$$d_0 = 0,$$
$$d_1 = 1,$$
$$d_2 = x_1,$$
$$d_3 = 1 \oplus x_1,$$
$$d_4 = x_2,$$
$$d_5 = 1 \oplus x_2,$$
$$d_6 = x_1 \oplus x_2,$$
$$d_7 = 1 \oplus x_1 \oplus x_2,$$
$$d_8 = x_1x_2,$$
$$d_9 = 1 \oplus x_1x_2,$$
$$d_{10} = x_1 \oplus x_1x_2,$$
$$d_{11} = 1 \oplus x_1 \oplus x_1x_2,$$
$$d_{12} = x_2 \oplus x_1x_2,$$
$$d_{13} = 1 \oplus x_2 \oplus x_1x_2,$$
$$d_{14} = x_1 \oplus x_2 \oplus x_1x_2,$$
$$d_{15} = 1 \oplus x_1 \oplus x_2 \oplus x_1x_2.$$

Каждый многочлен Жегалкина задает преобразование бинарного вектора по правилам, схожим с теми, что определены для класса элементарных клеточных автоматов.

Пусть $\langle V, \oplus\rangle = (\mathbb{Z}/2\mathbb{Z})^n$ — векторное пространство над полем $\mathbb{Z}/2\mathbb{Z}$.

Определение 3. Оператором Жегалкина с мультипликатором $d(x_1,x_2,x_3)$ назовем отображение $g_d : V \to V$ $(g_d(v)=w)$, действующее по правилу

$$w_1 = d(v_n, v_1, v_2),\ w_n = d(v_{n-1}, v_n, v_1),$$
$$w_{j+1} = d(v_j, v_{j+1}, v_{j+2}),\ j = 1,\ldots,n-2.$$

На рис. 6 проиллюстрированы действия операторов Жегалкина с мультипликаторами двух переменных. Нумерация рисунков совпадает с нумерацией из примера 2.

Определение 4. Оператор g_d назовем линейным, если

$$g_d(v \oplus v') = g_d(v) \oplus g_d(v')$$

для всех $v, v' \in V$.

4. ОПИСАНИЕ АЛГОРИТМА

Для нахождения линейных операторов Жегалкина были разработаны и программно реализованы алгоритмы, описание которых приводится ниже. Программная реализация алгоритмов на языке Python доступна по адресу https://github.com/vrkulikov/cellural_automata.

Приведем краткое описание алгоритмов. Алгоритм 7 преобразует десятичное число в двоичный код. Алгоритм 8 по коду Вольфрама, т.е. десятичному номеру правила клеточного автомата, строит многочлен Жегалкина. Алгоритм 9 вычисляет значение многочлена Жегалкина на векторе x. Алгоритм 10 реализует перебор всех построенных многочленов Жегалкина на трехклеточных шаблонах и выбор тех из них, которые индуцируют линейный оператор Жегалкина согласно определению 4.

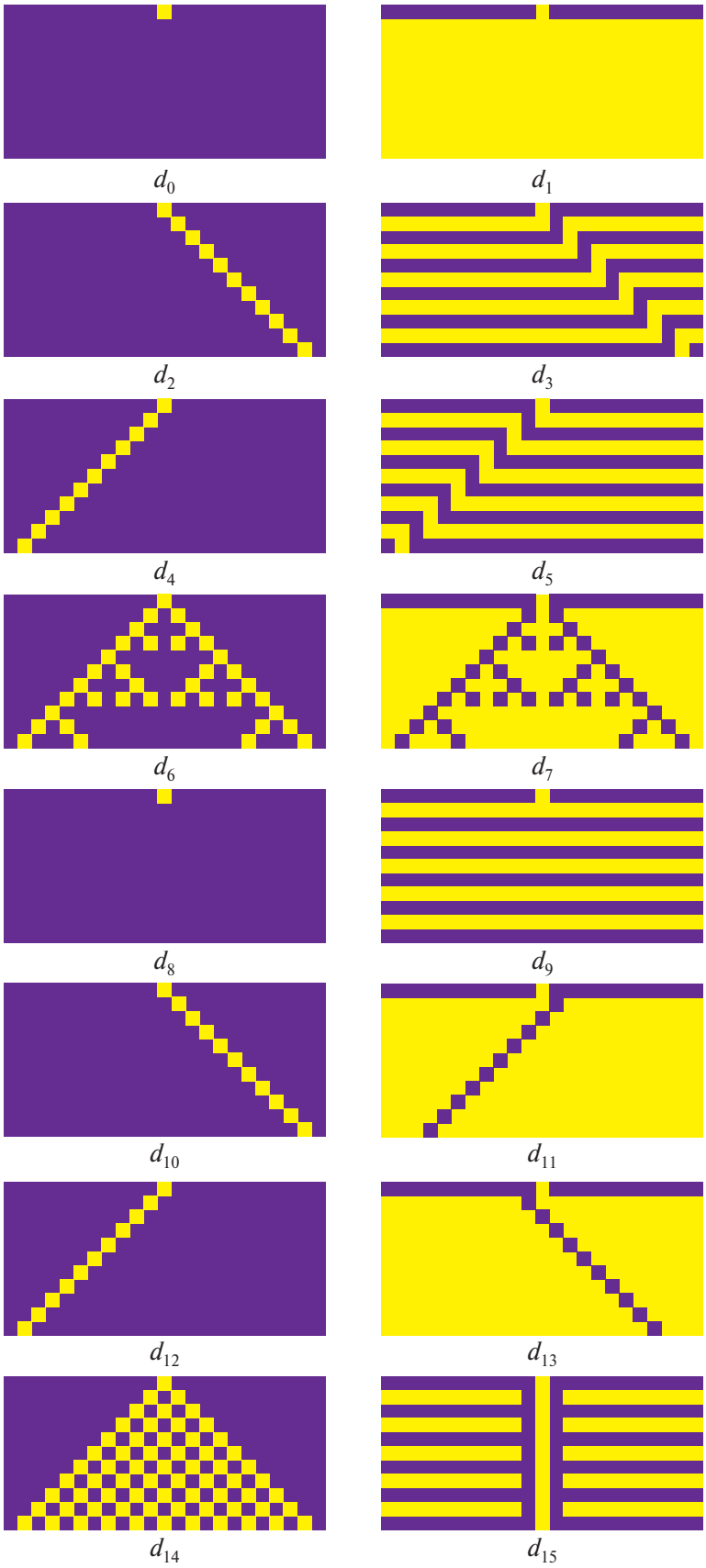


Рис. 6. Действия операторов Жегалкина с мультипликаторами двух переменных.

Алгоритм 1: INT2BIN(N, D):

Data: N, D
Result: Массив битов числа N длины D
out $\leftarrow$ создаем пустой массив длины D
for $i \leftarrow 0$ **to** $D-1$ **do**
  out$[i] \leftarrow N \bmod 2$
  $N \leftarrow N \operatorname{div} 2$
end
Output: out

Алгоритм 2: WOLF2ZHEG(W)

Data: W — целое число от 0 до 255, код Вольфрама
Result: Zh — массив длины 8, коэффициенты многочлена Жегалкина
Zh $\leftarrow$ создаем пустой массив длины 8
$W \leftarrow$ INT2BIN$(W, 8)$
for $i \leftarrow 0$ **to** 7 **do**
  Zh$[i] \leftarrow W[0]$
  for $j \leftarrow 1$ **to** 7 **do**
    $W[j-1] \leftarrow W[j-1] + W[j] \bmod 2$
  end
end
Output: Zh

Алгоритм 3: ZHEGALKIN(a, x)

Data: a — массив коэффициентов многочлена, x — вектор из трех двоичных переменных.
Result: V — значение многочлена
$V \leftarrow a_0 + a_1x_0 + a_2x_1 + a_3x_2 + a_4x_0x_1 + a_5x_0x_2 + a_6x_1x_2 + a_7x_0x_1x_2 \operatorname{div} 2$
Output: V — значение многочлена с коэффициентами a на векторе x.

Алгоритм 4: LINEARRULESSEARCH()

Result: LiearRules — список линейных правил.
LinearRules $\leftarrow$ []
for $i \leftarrow 0$ **to** 255 **do**
  poly = Wolf2Zheg(i)
  counter $\leftarrow 0$ **for** $j \leftarrow 0$ **to** 7 **do**
    for $k \leftarrow 0$ **to** 7 **do**
      $x \leftarrow$ Int2Bin$(j, 3)$
      $y \leftarrow$ Int2Bin$(k, 3)$
      $xy \leftarrow$ Int2Bin$(j \operatorname{xor} k, 3)$
      $f_x =$ Zhegalkin(poly, x)
      $f_y =$ Zhegalkin(poly, y)
      $f_{xy} =$ Zhegalkin(poly, xy)
      if $f_{xy} = f_x \operatorname{xor} f_y$ **then**
        counter $\leftarrow$ counter $+ 1$
      end
    end
    if counter $= 64$ **then**
      LinearRules $=$ LinearRules $\cup$ {poly}
    end
  end
end

Приведенные алгоритмы позволяют получить полный список линейных операторов Жегалкина с мультипликаторами трех переменных, а именно, справедлив следующий результат: *оператор g_d является линейным в точности для следующего набора мультипликаторов*

$$d=0,\ d=x_1,\ d=x_2,\ d=x_3,\ d=x_1\oplus x_2,$$
$$d=x_1\oplus x_3,\ d=x_2\oplus x_3,\ d=x_1\oplus x_2\oplus x_3.$$

Действия линейных операторов Жегалкина проиллюстрированы на рис. 11.

Легко видеть, что действие оператора g_d на вектор $v \in V$ может быть представлено трехдиагональной теплицевой матрицей $T_{g_d} v = w$

$$T_{g_d} = \begin{pmatrix} a_2 & a_4 & 0 & 0 & \vdots & 0 & 0 & a_1 \\ a_1 & a_2 & a_4 & 0 & \vdots & 0 & 0 & 0 \\ 0 & a_1 & a_2 & a_4 & \vdots & 0 & 0 & 0 \\ 0 & 0 & a_1 & a_2 & \ddots & 0 & 0 & 0 \\ 0 & 0 & 0 & a_1 & \ddots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \ddots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \vdots & a_1 & a_2 & a_4 \\ a_4 & 0 & 0 & 0 & \vdots & 0 & a_1 & a_2 \end{pmatrix},$$

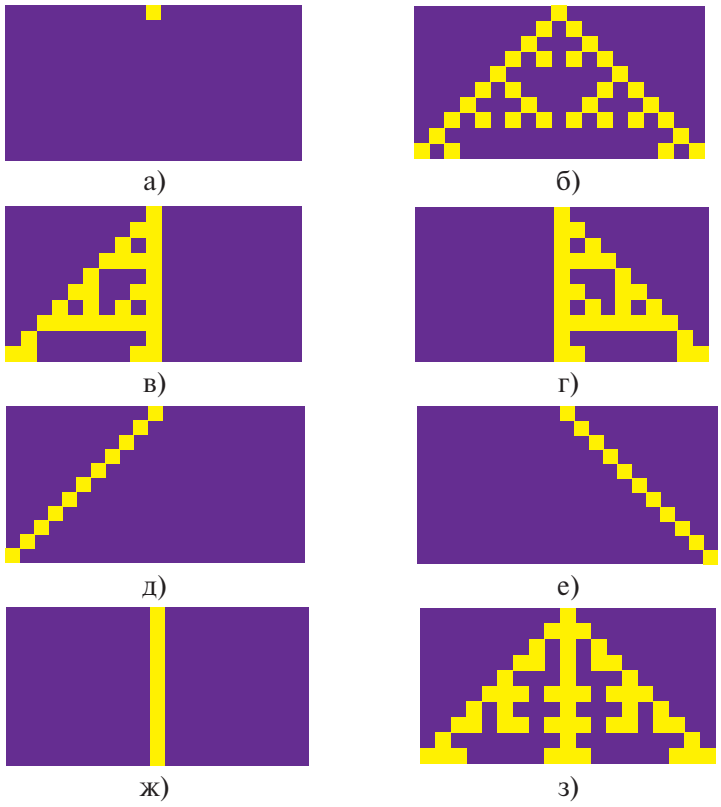


Рис. 7. Линейные операторы Жегалкина с мультипликаторами трех переменных.

где a_1, a_2, a_4 — произвольные коэффициенты многочлена Жегалкина (2), а коэффициенты a_0 и a_3, a_5, a_6, a_7 и равны нулю.

Сопоставим теперь каждому коду Вольфрама многочлен Жегалкина. Пусть $\mathbf{r} = (r_0, r_1, ..., r_7)$ — одно из 256 правил клеточного автомата. Рассмотрим многочлен Жегалкина $d(x_1, x_2, x_3)$ с произвольными коэффициентами $\mathbf{a} = (a_0, a_1, ..., a_7)$. Последовательно подставляя в d вместо переменных (x_1, x_2, x_3) наборы значений (0,0,0),(1,0,0),(0,1,0), ..., (1,1,1), получим систему линейных уравнений

$$A\mathbf{a} = \mathbf{r} \qquad (3)$$

с обратимой нижнетреугольной матрицей коэффициентов

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix},$$

определитель которой равен 1. Решая систему (3) для заданных правых частей (r_j), $j = 0, ..., 7$, найдем коэффициенты многочлена Жегалкина d.

5. ЗАКЛЮЧЕНИЕ

Для одномерных клеточных автоматов на эквивалентном языке операторов Жегалкина найдены линейные правила. Соответствие операторов Жегалкина и кодов Вольфрама может быть установлено в случае клеточных автоматов большей размерности, а также над конечным полем большего порядка. Для этого важно, чтобы сработал метод Гаусса решения системы линейных уравнений (3). Однако вопрос линейности в этих случае требует отдельного изучения.

БЛАГОДАРНОСТИ

Работа поддержана Красноярским математическим центром, финансируемым Минобрнауки РФ (Соглашение 075-02-2023-936).

СПИСОК ЛИТЕРАТУРЫ

1. *Neumann J.* Theory of self-reproducing automata // Edited by Arthur W. Burks. 1966.

2. *Цетлин М.Л.* Некоторые задачи о поведении конечных автоматов //Доклады Академии наук. Российская академия наук, 1961. Т. 139. № 4. С. 830–833.
3. *Conway J. et al.* The game of life //Scientific American. 1970. Т. 223. № 4. С. 4.
4. *Batty M.* Cities as Complex Systems: Scaling, Interaction, Networks, Dynamics and Urban Morphologies. 2009.
5. *Ghosh P. et al.* Application of Cellular automata and Markov-chain model in geospatial environmental modeling-A review //Remote Sensing Applications: Society and Environment. — 2017. Т. 5. С. 64–77.
6. *Гасников А. и др.* (ред.). Введение в математическое моделирование транспортных потоков. Litres, 2022.
7. *Fronczak P. et al.* Cellular automata approach to modeling self-organized periodic patterns in nanoparticle-doped liquid crystals //Physical Review E. 2022. Т. 106. № 4. С. 44705.
8. *Janssens K.G.F.* An introductory review of cellular automata modeling of moving grain boundaries in polycrystalline materials //Mathematics and Computers in Simulation. 2010. Т. 80. № 7. С. 1361–1381.
9. *Lemont B.* Kier and Paul G. Seybold. Cellular Automata Modeling of Complex Biochemical Systems // Encyclopedia of Complexity and Systems Science, 2015.
10. *Kozhoridze G., Dor E.B., Sternberg M.* Assessing the Dynamics of Plant Species Invasion in Eastern-Mediterranean Coastal Dunes Using Cellular Automata Modeling and Satellite Time-Series Analyses //Remote Sensing. 2022 Т. 14. № 4. С. 1014.
11. *Wolfram S.* Statistical mechanics of cellular automata // Reviews of modern physics. – 1983. Т. 55. № 3. С. 601.
12. *Wolfram S. et al.* A new kind of science. Champaign: Wolfram media, 2002. Т. 5. С. 130.
13. *Tomassini M., Sipper M., Perrenoud M.* On the generation of high-quality random numbers by two-dimensional cellular automata // IEEE Transactions on computers. 2000. Т. 49. № 10. С. 1146–1151.
14. *Walus K. et al.* RAM design using quantum-dot cellular automata // NanoTechnology Conference. 2003. Т. 2. С. 160–163.
15. *Cagigas-Muniz D. et al.* Efficient simulation execution of cellular automata on GPU // Simulation Modelling Practice and Theory. 2022. Т. 118. С. 102519.
16. *Sato T.* Decidability for some problems of linear cellular automata over finite commutative rings // Information Processing Letters. 1993. Т. 46. № 3. С. 151–155.
17. *Martin A. et al.* Reversibility of linear cellular automata // Applied Mathematics and Computation. 2011. Т. 217. № 21. С. 8360-8366.
18. *Martin del Rey A., Casado Vara R., Hernández Serrano D.* Reversibility of symmetric linear cellular automata with radius r = 3 //Mathematics. 2019. Т. 7. № 9. С. 816.
19. *Жегалкин И.И.* Арифметизация символической логики //Математический сборник. 1928. Т. 35. № 3–4. С. 311–377.
20. *Федченко Д.П., Новиков В.В., Тимофеев И.В.* Фотонные топологические изоляторы типа Руднера на языке трехцветных клеточных автоматов // Ученые записки физического факультета МГУ. 2021. № 5. С. 2150302.
21. *Fedchenko D.P., Kim P.N., Timofeev I.V.* Photonic Topological Insulator Based on Frustrated Total Internal Reflection in Array of Coupled Prism Resonators // Symmetry. 2022. Т. 14. № 12. С. 2673.
22. *Гальперин Г. А., Земляков А. Н.* Математические бильярды: Бильярдные задачи и смежные вопросы математики и механики. Наука. Гл. ред. физ.-мат. лит., 1990.

ON LINEAR CELLULAR AUTOMATA

V. R. Kulikov[a], A. A. Kytmanov[b], A. O. Poroshin[a], I. V. Timofeev[c, a], D. P. Fedchenko[c, a]

[a] *Siberian State University, Krasnoyarsk, 660041 Russia*
[b] *MIREA – Russian Technological University, Moscow, 119454 Russia*
[c] *Kirensky Institute of Physics, Federal Research Center KSC SB RAS, Krasnoyarsk, 660036 Russia*

Wolfram cellular automata are considered and their operation is demonstrated using an example of traffic flow simulation. For the class of one-dimensional elementary cellular automata, the concept of linearity is introduced in the language of Zhegalkin operators. An algorithm for finding linear Zhegalkin operators with multipliers of three variables is presented. The algorithm is implemented in Python.

Keywords: cellular automaton, Wolfram code, Zhegalkin operators

REFERENCES

1. *von Neumann J.* Theory of Self-Reproducing Automata, Ed. by Burks, A.W., Urbana: Illinois Univ. Press, 1966.
2. *Tsetlin M.L.* Some problems of finite state machine behavior, Dokl. Akad. Nauk SSSR, 1961, vol. 139, no. 4, pp. 830–833.
3. *Conway J. et al.* The game of life, Sci. Amer., vol. 223, no. 4, p. 4.
4. *Batty M.* Cities as Complex systems: Scaling, interaction, networks, dynamics and urban morphologies, in Encyclopedia of Complexity and Systems Science, 2009, pp. 1041–1071.
5. *Ghosh P. et al.* Application of cellular automata and Markov-chain model in geospatial environmental modeling—A review, Remote Sens. Appl.: Soc. Env., 2017, vol. 5, pp. 64–77.
6. Introduction to Mathemetical Modeling of Traffic Flows, Ed. by Gasnikov, A. et al. Litres, 2022 [in Russian].
7. *Fronczak P. et al.* Cellular automata approach to modeling self-organized periodic patterns in nanoparticle-doped liquid crystals, Phys. Rev. E., 2022, vol. 106, no. 4, p. 44705.
8. *Janssens K.G.F.* An introductory review of cellular automata modeling of moving grain boundaries in polycrystalline materials, Math. Comput. in Simul., 2010, vol. 80, no. 7, pp. 1361–1381.
9. *Lemont B.K. and Seybold P.G.* Cellular automata modeling of complex biochemical systems, in Encyclopedia of Complexity and Systems Science, 2015.
10. *Kozhoridze G., Dor E.B., and Sternberg M.* Assessing the dynamics of plant species invasion in Eastern-Mediterranean Coastal Dunes Using Cellular Automata Modeling and Satellite Time-Series Analyses, Remote Sens., 2022, vol. 14, no. 4, p. 1014.
11. *Wolfram S.* Statistical mechanics of cellular automata, Rev. Modern Phys., 1983, vol. 55, no. 3., p. 601.
12. *Wolfram S. et al.* A New Kind of Science, Champaign: Wolfram Media, 2002, vol. 5, p. 130.
13. *Tomassini M., Sipper M., and Perrenoud M.* On the generation of high-quality random numbers by two-dimensional cellular automata, IEEE Trans. Comput., 2000, vol. 49, no. 10, pp. 1146–1151.
14. *Walus K. et al.* RAM design using quantum-dot cellular automata, NanoTechnology Conference, 2003, Vol. 2, pp. 160–163.
15. *Cagigas-Muniz D. et al.* Efficient simulation execution of cellular automata on GPU, Simul. Modell. Pract. Theory, 2022, vol. 118, p. 102519.
16. *Sato T.* Decidability for some problems of linear cellular automata over finite commutative rings, Inf. Proc. Lett., 993, vol 46, no. 3, pp. 151–155.
17. *Martin A. et al.* Reversibility of linear cellular automata, Appl. Math. Comput., 2011, vol. 217, no. 21, pp. 8360–8366.
18. *Martin del Rey A., Casado Vara R., and Hernández S. D.* Reversibility of symmetric linear cellular automata with radius r = 3, Mathematics, 2019, vol. 7, no. 9, p. 816.
19. *Zhegalkin I.I.* Arithmetization of symbolic logic, Mat. Sb., vol. 35, no. 3–4, pp. 311–377.
20. *Fedchenko D.P., Novikov V.V., and Timofeev I.V.* Photonic topological insulators of the Rudner type in terms of of tricolor cellular automata, Uch. Zap. Fiz. Fakul'teta MGU, 2021, No. 5, p. 2150302.
21. *Fedchenko D.P., Kim P.N., and Timofeev I.V.* Photonic topological insulator based on frustrated total internal reflection in array of coupled prism resonators, Symmetry, 2022, vol. 14, no. 12, p. 2673.
22. *Gal'perin G.A. and Zemlyakov A.N.* Mathematical Billiards: Billiard Problems and Related Problems of Mathematics and Mechanics, Moscow: Nauka, 1990 [in Russian].

ПРОГРАММИРОВАНИЕ, *2024*, *№ 1*, *с. 40–52*

АНАЛИЗ ДАННЫХ

УДК 004.932

АЛГОРИТМ И ПРОГРАММНАЯ РЕАЛИЗАЦИЯ АВТОМАТИЧЕСКОГО РАСЧЕТА ДЛИН И ПЛОЩАДЕЙ ТРЕЩИН НА БОРТАХ И ОТВАЛАХ УГОЛЬНЫХ РАЗРЕЗОВ

С. Е. Попов[a], *(ORCID: 0000-0001-9495-6561),
В. П. Потапов[a], **(ORCID: 0000-0002-1530-5902),
Р. Ю. Замараев[a], *(ORCID: 0000-0003-4822-4794)**

[a]*ФГБУН "Федеральный исследовательский центр информационных и вычислительных технологий" (ФИЦ ИВТ), 630090 Новосибирск, пр. Академика Лаврентьева, 6, Россия*

**E-mail: popov@ict.nsc.ru*
***E-mail: vadimptpv@gmail.com*
****E-mail: zamaraev@ict.nsc.ru*

Поступила в редакцию: 21.02.2023
После доработки: 24.05.2023
Принята к публикации: 26.05.2023

В работе представлен алгоритм и описание его программной реализации для обнаружения линеаментов (трещин) на изображениях аэрофотосъемки угольных разрезов. В основе предложенного подхода лежит аппарат сверточных нейронных сетей для семантической классификации бинаризованных изображений объектов, а также теория графов для определения геометрического расположения объектов с последующим вычислением их длин и площадей. В качестве исходных данных использовались трехканальные RGB-изображения аэрофотосъемки высокого разрешения (пиксел 10 × 10 см). Модель программного модуля логически разделена на три уровня: предобработка, детектирование и постобработка. Первый уровень включает в себя предобработку входных данных для формирования обучающей выборки на базе последовательных трансформаций RGB-изображения в бинарное с применением библиотеки OpenCV. Второй уровень информационной модели представлен нейронной сетью типа U-Net, включающей блоки сверточной (Encoder) и развертоной частей (Decoder). На данном уровне реализовано автоматическое детектирование объектов. Третий уровень модели отвечает за расчет площадей и длин. На вход ему передается результат работы сверточной нейронной сети. Площадь трещин вычисляется путем суммирования общего числа точек с умножением на размер пиксела. Длина рассчитывается путем линеаризации площадного объекта в сегментированный объект с узловыми пикселами и последующим расчетом длин между ними с учетом разрешения исходного изображения. Программный модуль может работать с фрагментами исходного изображения путем их объединения. Модуль реализован на языке программирования Python. Код доступен по адресу (https://gitlab.ict.sbras.ru/popov/lineaments/-/tree/master/lineaments-cnn).

Ключевые слова: обнаружение линеаментов, трещин и эрозии грунта; сверточные нейронные сети; семантическая сегментация; обнаружение и накопление; аэрофотосъемка

DOI: 10.31857/S0132347424010044 **EDN:** HLFTJY

1. ВВЕДЕНИЕ

Современный этап развития технологии добычи полезных ископаемых открытым способом характеризуется значительными глубинами извлечения, которые могут составлять несколько сотен метров. В этих условиях возрастают требования к обеспечению устойчивости бортов карьеров. Оползневые деформации уступов приводят к большим финансовым затратам на устранение последствий аварийных ситуаций. Количественная характеристика неоднородностей (промывов, трещин и пр.) имеет основополагающее значение для определения механического поведения не сплошных массивов горных пород [1]. Следовательно, растет интерес к автоматизированному обнаружению неоднородностей на основе компьютерного зрения, чтобы заменить традиционные процедуры проверки человеком.

Методы обнаружения неоднородностей на основе машинного зрения широко применяются на протяжении последнего десятилетия из-за их преимуществ относительно хорошей точности и производительности алгоритмов в реальном времени [2] с учетом дистанционного наблюдения за объектами.

В основе этих алгоритмов лежат методы обработки цифровых данных (значений пикселов) RGB-изображения, включая методы обнаружения краев

[3], преобразование Хафа [4], сегментацию изображения [5], идентификацию и детектирование характерных точек [6, 7], метод корреляции для цифровых изображений (Digital Image Correlation) [8, 9] и фотограмметрии [10] и прочее.

Однако общее ограничение этих подходов заключается в обнаружении линеаментов (трещин) путем поиска по всей области изображения. В результате возникают трудности в дифференциации истинных объектов от подобных им шумов, таких как границы структур (например, стрела экскаватора), крупные наземные коммуникации, провода и трубы темного цвета, тени от объектов линейной формы [11]. В то же время точное обнаружение формы линеамента, ответвлений, начального и конечного пикселов является ключевым фактором для измерения геометрических свойств (длины или площади) и остается сложной, нетривиальной задачей.

В последние годы машинное обучение и, в частности, сверточные нейронные сети продемонстрировали широкие возможности в области семантического обнаружения объектов и их признаковых классов в задачах дистанционного мониторинга состояния различных объектов. Так, в работе [12] рассматривается модель CNN для точной идентификации микросейсмических событий и взрывов. В статье [13] предложен метод, основанный на DCNN, для локализации повреждений строительных конструкций с высокой точностью на необработанных, зашумленных сигналах. Также встречается большое количество работ касательно процессов детектирования трещин на снимках, образованных на различных поверхностях. Авторы в [14] использовали обученную CNN и методы скользящего окна для обнаружения трещин на бетонных поверхностях. В исследовании [15] используется сверточная нейронная сеть типа R-CNN для обнаружения трещин на асфальтированных дорогах. В [16] продемонстрирована модель обнаружения повреждений на базе сквозного метода (end-to-end), основанного на глубоком обучении с использованием широкого набора данных о повреждениях дорог, их местоположении и типе повреждения. Также в работе [17] на основе сквозного метода предложена сверточная сеть для обнаружения трещин мостовых опор. В работе используется свертка с разделением, уменьшающая количество параметров и модуль объединения пространственных пирамид (ASPP). Представленная модель достигает точности обнаружения 96,37%.

Среди работ по обнаружению линеаментов выделяют исследования по практическому применению результатов детектирования, т.е. так называемый пост-процессинг. В частности, в работе [18] рассматривается задача определения длин трещин на шлифах бетонных срезов различных конструкций. Использована комбинация сверточной сети и методов определения вершин трещин на основе морфологических трансформаций графического объекта и функции пороговой сегментации.

В плане коммерческого программного обеспечения в основном на рынке присутствуют разработки для определения физических характеристик (например, определение величины кинетического сдвига и границ дилатации) трещин по их изображениям в динамике на основе алгоритмов машинного зрения. В работе [19] представлена полностью автоматизированная процедура обнаружения трещин и измерения их кинематики в лабораторных экспериментах с использованием цифровой корреляции изображений (DIC), которая позволяет извлекать гораздо более мелкие трещины с их местоположением в образце. Ширина трещины и подвижки измеряются с использованием поля смещения с учетом локальных поворотов образца.

Анализ представленной литературы показал, что подавляющее большинство работ, посвященных в той или иной степени детектированию линеаментов, используют в качестве исходных данных изображения с явно выделяющимися объектами на относительно равномерном по цветовой гамме фоне (асфальтированная дорога, бетонная стена, каменистая порода и т.п.). Авторам не удалось найти обучающих выборок с изображениями реальных бортов и отвалов на карьерах, где, например, на поверхности могут присутствовать камни, результаты рекультивации (молодая поросль), водные отстойники и прочее. Косвенно это подтверждается и отсутствием таких данных на ресурсе Kaggle (https://www.kaggle.com/).

Хотя методы машинного зрения, в том числе нейронные сети глубокого обучения, и обладают наибольшей точностью детектирования линеаментов, однако, остаются вопросы постпроцессинга результатов и имплементации предметного программного обеспечения. Например, расчета геометрических величин с использованием RGB-изображений аэрофотосъемки.

В этом исследовании представлен алгоритм обнаружения трещин и определения их длин и площадей на бортах и отвалах угольных разрезов. Приводится сравнение нейронных сетей для семантической классификации объектов, алгоритм линеаризации пиксельных данных на бинаризованных

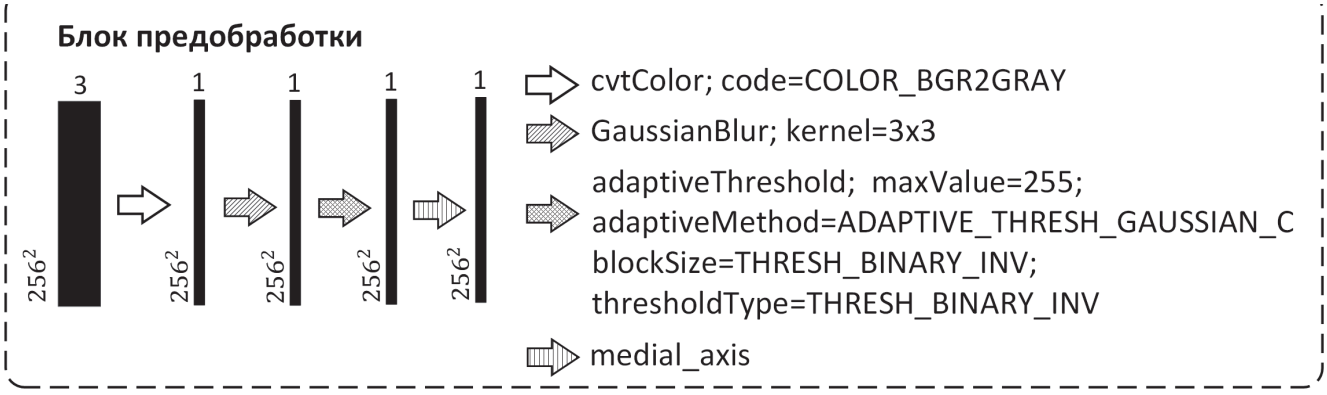


Рис. 1. Схема преобразования RGB-изображения в бинарное изображение. Названия компонентов соответствуют программной библиотеке OpenCV API (https://docs.opencv.org/4.x/) и scikit-image (https://scikit-image.org/). Число по вертикали – размер изображения, по горизонтали – количество каналов.

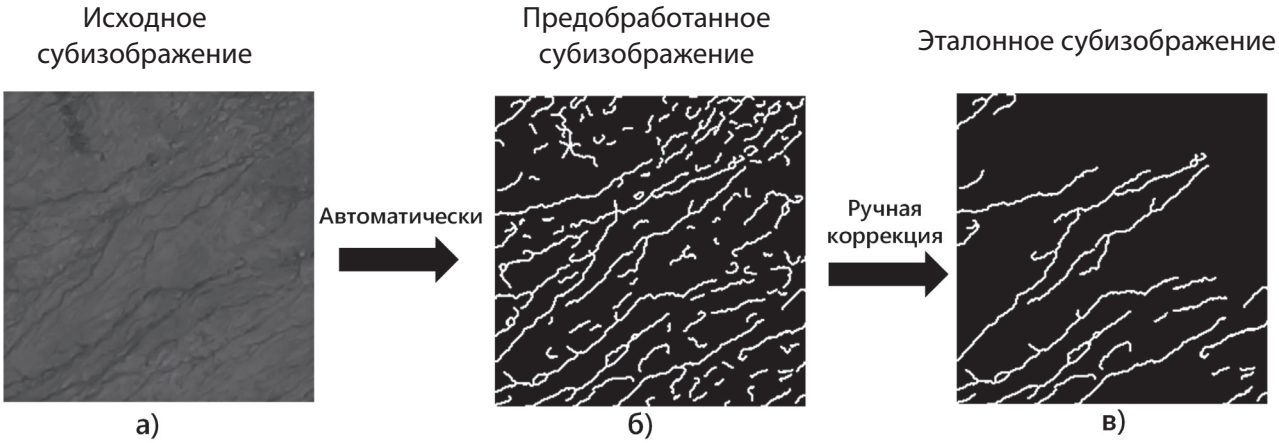


Рис. 2. Схема подготовки обучающей выборки на основе схемы преобразования (рис. 1).

изображениях трещин. Решается задача группировки объектов, подсчета пикселов и расчета на их основе длины и площади с учетом разрешения исходного изображения. Полный алгоритм обработки реализован в виде программного модуля на языке Python (https://gitlab.ict.sbras.ru/popov/lineaments/-/tree/master/lineaments-cnn).

2. ПОДГОТОВКА ДАННЫХ

В качестве исходных данных использовались трехканальные RGB-изображения аэрофотосъемки высокого разрешения (10 × 10 см). Высота съемки 300 м. Снимки получены с беспилотного летательного аппарата с камерой SONY DSC-RX1R. Период съемки 07.2022–08.2022 гг. Полный размер входного изображения 6000 × 4000 пикселов.

Входное изображение разбивалось на базисные субизображения (в текущей версии 256 × 256 пикселов), которые брались за основу для предварительной обработки и процесса ручной разметки данных. Предобработка строилась на базе схемы последовательных преобразований RGB-изображения в бинарное (рис. 1).

В схеме используется преобразование цветного изображения к схеме “градации серого” (cvtColor) с последующим применением процедуры размытия по Гауссу (GaussianBlur) для сглаживания резких переходов (значений) соседних пикселов.

Бинаризация выполняется на основе адаптивного выделения контуров с дилатацией (adaptiveThreshold) из библиотеки OpenCV и выделением медианных пикселов (medial_axis) для линеаризации трещин из библиотеки scikit-image (рис. 3)

На выходе получаются нормализованные бинарные изображения, пригодные для последующей ручной корректировки в рамках формирования обучающей выборки (рис. 2в).

Исходное и эталонное субизображения подвергались вертикальному, горизонтальному, симметричному отражению и случайному вращению (библиотека albumentation, https://albumentations.ai/).

```
...
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
img_gray = cv2.cvtColor (img, cv2.COLOR_BGR2GRAY)
img_blur = cv2.GaussianBlur (img_gray, (3, 3), 0)
img_bw = cv2.adaptiveThreshold (img_blur, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, \
                                cv2 .THRESH_BINARY_INV,
                                template.config['ADAPTIVE_THRESHOLD_BLOCK_SIZE']
                                template.config['ADAPTIVE_THRESHOLD_C'])
img_skel_bool, _ = medial_axis(img_bw, return_distance=True)
img__skel = img_skel_bool.astype (np.uint8)
...
transform = A. Compose ([A.OneOf ([A.HorizontalFlip (p=l), A.VerticalFlip(p=l),
                          A.RandomRotateSO(p=l)], p=0.75)])
image_t, mask_t = list(transform(image=imgs[key], mask=masks[key]).values())
...
```

Рис. 3. Фрагмент кода предобработки субизображений обучающей выборки.

Таким образом была сформирована обучающая выборка из более чем 2000 субизображений (с учетом аугментации), являющихся частями исходных изображений, содержащих как трещины, так и изображения с полным отсутствием таковых (фон). Принималось, что на субизображении трещины обозначены белым цветом (значение 1, класс “трещина”), а остальная часть фона - черным (0, класс “фон”).

Для процесса обучения, валидации и детектирования трещин датасет был поделен на части в пропорции 85% / 10% / 5%. Датасет доступен по ссылке https://www.kaggle.com/datasets/semionpopov/open-pit-cracks.

3. ОБНАРУЖЕНИЕ ТРЕЩИН

3.1 Конфигурация нейронной сети

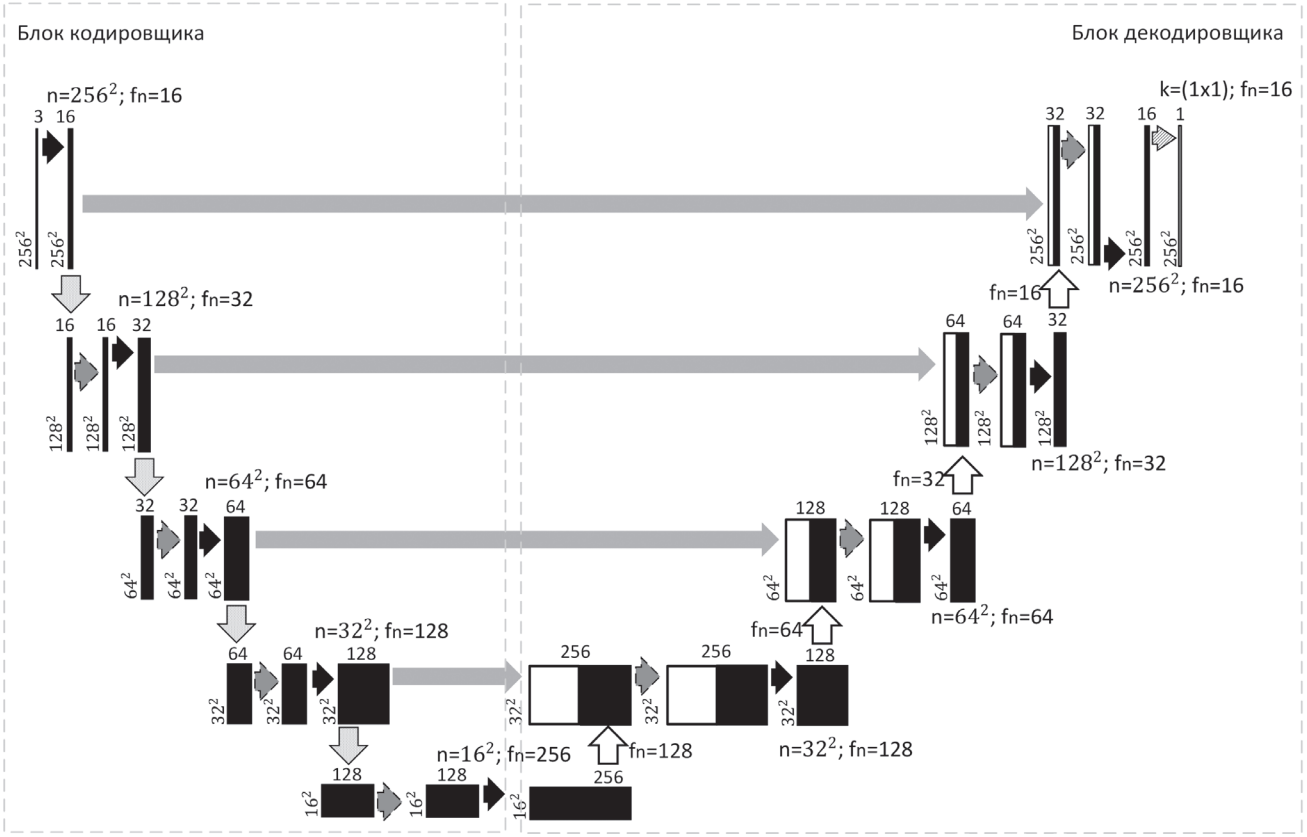
Процесс обнаружения трещин на изображениях строился на базе сверточной нейронной сети типа U-Net. Архитектура сети представляет собой полносвязную сверточную сеть [20], работающую на меньшем количестве примеров (обучающих образов) с более точной сегментацией. Сеть содержит блок кодировщика (Encoder) и блок де-кодировщика (Decoder) (рис. 4а, б). Программная имплементация сети реализована на основе пакета TensorFlow v2.11 с модулем Keras (https://www.tensorflow.org/api_docs/python/tf/keras).

Блок Encoder состоит из модулей свертки, включающих два сверточных слоя (Conv2D) с параметром kernel_size 3×3 (класс tf.keras.layers.Conv2D), слой активации ReLU (класс tf.keras.layers.Activation) и слой нормализации входных данных BatchNorm (класс tf.keras.layers.BatchNormalization). Также используется дополнительный слой пулинга с функцией максимума (параметр pool_size = 2×2) с шагом 2 и слой регуляризации (Dropout) для уменьшения переобучения с 50% исключением (параметр rate). На каждом шаге количество каналов признаков удваивается (параметр filters).

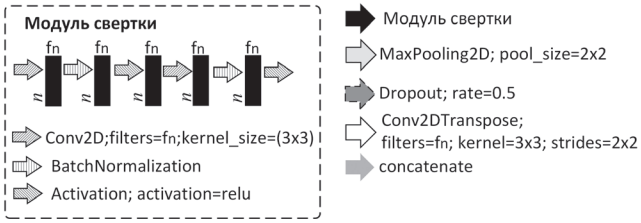
Параметр “фильтр” является ключевым. Фильтры – это массивы заданного размера, которые инициализируются случайными значениями с использованием метода, указанного в аргументе kernel_initializer (класс tf.keras.initializers). Во время обучения сети фильтры обновляются таким образом, чтобы минимизировать потери. В ходе обучения фильтры учатся обнаруживать определенные особенности объектов (например, края и текстуры трещин). Блок Encoder действует как экстрактор признаков и изучает абстрактное представление входного изображения через последовательность фильтров.

Блок Decoder содержит слой обратной свертки (класс tf.keras.layers.Conv2DTranspose), который расширяет карту признаков. После идет конкатенация (слой concatenate) с соответствующим образом обрезанной картой признаков и последовательность слоев свертки, как в части Encoder. Обрезка проводится из-за потери пограничных пикселов в каждой свертке. Завершающий слой (Conv2D) – свертка 1×1 (kernel_size) – используется для приведения каждого 16-компонентного вектора признаков до требуемого количества классов (2 класса: трещина, фон).

Декодирование необходимо для повышения дискретизации нейронной сети за счет объединения карты признаков объектов с более высокими разрешениями из блока кодировщика (апсэмплинг) с целью улучшения априорной оценки более ранних этапов свертки. Это позволяет лучше представить локализацию искомых объектов (трещин). На



а) Блоки кодировщика и де-кодировщика



б) Модуль свертки

Рис. 4. Конфигурация нейронной сети

рис. 5, 6 приведены фрагменты программного кода инициализации блоков Encoder и Decoder, компиляции и старта процесса обучения сети (класс tf.keras.Model).

3.2. Обучение нейронной сети

Сеть обучалась на датасете размером 2015 изображений. В качестве функции потерь использовалась бинарная кросс-энтропия (класс tf.keras.losses.BinaryCrossentropy). Для оценки качества работы моделей использовалась метрика Accuracy (класс tf.keras.metrics.Accuracy), показывающая процент пикселов в изображении, которые были правильно классифицированы. Количество эпох от 30 до 45. На выходе получалась полностью обученная сеть с оптимизированной матрицей весов. Сеть сохранялась в файл формата HDF (.h5) для последующей загрузки на этапе детектирования объектов.

Для оценки релевантности выбора конфигурации представленной сети проведено сравнение результатов обучения с похожей архитектурой нейронной сети DeepLabV3+ [21]. Обучение DeepLabV3+ проводилось на такой же обучающей выборке с аналогичной функцией потерь и метрикой. Ниже представлены сводные таблицы метрик (табл. 1.), матриц ошибок (confusion matrix, табл. 2) и скорости обучения (табл. 3.).

Метрики сети U-Net оказались значительно выше. Это объясняется тем, что сеть DeepLabV3+ использует в слое кодировщика так называемые расширенные свертки. Модель расширенных сверток (Atrous Convolutions) представляет способ ком-

```
def getUnet (inputImage, numFilters=16,
             droupouts=0.1, doBatchNorm=True):

    # Encoder
    c1 = Conv2dBlock (inputImage, numFilters * 1,
                      kernelSize=3, doBatchNorm=doBatchNorm)
    p1 = tf.keras.layers.MaxPooling2D((2, 2))(c1)
    p1 = tf.keras.layers.Dropout (droupouts)(p1)
    ...

    c4 = Conv2dBlock (p3, numFilters * 8,
                      kernelSize=3, doBatchNorm=doBatchNorm)
    p4 = tf.keras.layers.MaxPooling2D((2, 2))(c4)
    p4 = tf.keras.layers.Dropout (droupouts)(p4)
    c5 = Conv2dBlock (p4, numFilters * 16,
                      kernelSize=3, doBatchNorm=doBatchNorm)

    # Decoder
    u6 = tf.keras.layers.Conv2DTranspose (numFilters * 8, (3, 3),
                          strides=(2, 2), padding='same')(c5)
    u6 = tf.keras.layers.concatenate([u6, c4])
    u6 = tf.keras.layers.Dropout(droupouts)(u6)
    c6 = Conv2dBlock (u6, numFilters * 8,
                           kernelSize=3, doBatchNorm=doBatchNorm)
    ...

    u9 = tf.keras.layers.Conv2DTranspose (numFilters * 1, (3, 3),
                          strides=(2, 2), padding='same')(c8)
    u9 = tf.keras.layers.concatenate([u9, c1])
    u9 = tf.keras.layers.Dropout (droupouts)(u9)
    c9 = Conv2dBlock(u9, numFilters * 1,
                          kernelSize=3, doBatchNorm=doBatchNorm)

    output = tf.keras.layers.Conv2D(1, (1, 1), activation='sigmoid')(c9)
    model = tf.keras.Model(inputs=[inputimage], outputs=[output])

    return model
```

Рис. 5. Программная реализация блоков Encoder и Decoder. Переменными “c1-c9” обозначены модули свертки.

бинировать признаки с нескольких масштабов без значительного увеличения количества параметров.

Регулируя показатель расширения, один и тот же фильтр распределяет значение веса дальше в пространстве [21]. Это позволяет изучать более общий контекст. То есть данный метод более подходит для распознавания площадных объектов, за счет увеличения шага пространственных пирамид ASPP на каждом этапе свертки.

Таблица 1. Сравнение нейронных сетей U-Net и DeepLabV3+

Нейронная сеть	Метрика (Accuracy)
U-Net	0,9801
DeepLabV3+	0,9100

Таблица 2. Сравнение матриц ошибок

Реальные значения	Сеть U-Net		Сеть DeepLabV3+	
Трещины	0,9544	0,0456	0,8103	0,1897
Фон	0,0413	0,9587	0,0533	0,9467
	Трещины	Фон	Трещины	Фон
	Предсказанные значения			

Таблица 3. Скорость обучения сети

Нейронная сеть	Скорость обучения на эпоху* (batch_size = 4)
U-Net	52 мс
DeepLabV3+	103 мс

* На GPU Nvidia RTX 3060.

```
...
unet = getUnet (inputs, droupouts=0.07)
unet.compile (optimizer='Adam', loss='binary_crossentropy', metrics=['accuracy'])

fit_result = unet.fit (imgs_array, masks_array, epochs=37, batch_size=1)
unet.save (template.config['MODEL'])
...
```

Рис. 6. Фрагмент кода компиляции и старта процесса обучения сети.

```
def predict(model_file, imgs_array):
    unet_model = tf.keras.models.load_model(model_file)
    predictions = unet_model.predict(imgs_array, batch_size=8)
    predictions = [predictions[i][:, 0] for i in range(len(predictions))]
    return predictions
```

Рис. 7. Фрагмент кода для запуска процедуры обнаружения трещин.

3.3. Тестирование нейронной сети

Для обнаружения трещин на тестовых субизображениях используется метод predict (класс tf.keras. Model) (рис. 7).

На вход метода подаются исходные субизображения (параметр imgs_array) тестовой выборки (рис. 2а). На выходе получается массив значений, каждый элемент которого соответствует положению пиксела на исходном субизображении, а значение – вероятности того, что соответствующий пиксел принадлежит классу “трещина”.

Задается порог значений (например, 0.95), выше которого пиксел классифицируется, как принадлежащий трещине. Таким образом формируется карта вероятностей классификаций пикселов (рис. 8).

Далее всем пикселам со значениями выше порога присваиваются значения, равные 255, а ниже или равными порогу – 0. Формируется бинарное (черно-белое одноканальное изображение) трещин.

После прохождения процедурой predict по всем субизображениям полученные бинарные субизображения объединяются в единый массив данных согласно ширине и длине исходного изображения. Формируется полная бинарная карта. Именно такое изображение используется в постпроцессинге для определения длин и площадей трещин.

4. ПОСТПРОЦЕССИНГ

4.1 Определение длин трещин

Будем рассматривать бинарное субизображение как двухмерный массив (img_bw). Аналогично этапу предобработки применяется метод medial_axis для выделения медианных пикселов. Данный метод уменьшает количество смежных пикселов для текущего медианного до минимального количества (по возможности до 2 пикселов), объекты подвергаются скелетизации (переменная, двухмерного массива img_skel) (рис. 3).

На следующем шаге запускается итерационная процедура формирования наборов точек для каждого объекта-линии. На каждой итерации используется метод label (scipy.ndimage), позволяющий назначить числовую метку (переменная key) пикселам каждой из обособленных трещин, где принадлежность пиксела однозначно определяется его меткой (переменные labeled_array, num_of_labels, рис. 9).

Формируется массив key_points, содержащий массивы координат пикселов (y, x – по вертикали, по горизонтали) для текущей метки объекта-трещины.

Измерение трещины по длине определяется по количеству последовательно-смежных (парных) пикселов между двумя краевыми. Пиксел называется краевым, если у него ровно один смежный пиксел. Если у трещины больше двух краевых пикселов (есть ответвления), то выбирается набор по максимальному количеству.

Таким образом, можно сформулировать задачу нахождения длины трещины, как поиск наибольшего пути в ациклическом графе между двумя заданными вершинами.

Для решения поставленной задачи использовались библиотеки SciPy (spatial.KDTree, https://docs.scipy.org/doc/scipy/) и NetworkX (объект Graph, https://networkx.org/documentation/stable/ reference/index.html). Поиск смежных пикселов осуществлялся при помощи метода query_ball_point с параметрами x, y – координаты текущего пиксела и длина радиуса (переменная r), в пределах которого ищутся все пикселы. Если длина получившегося массива (переменная neighbor_counter) равна 2, то текущий пиксел считается краевым (переменная edge_points).

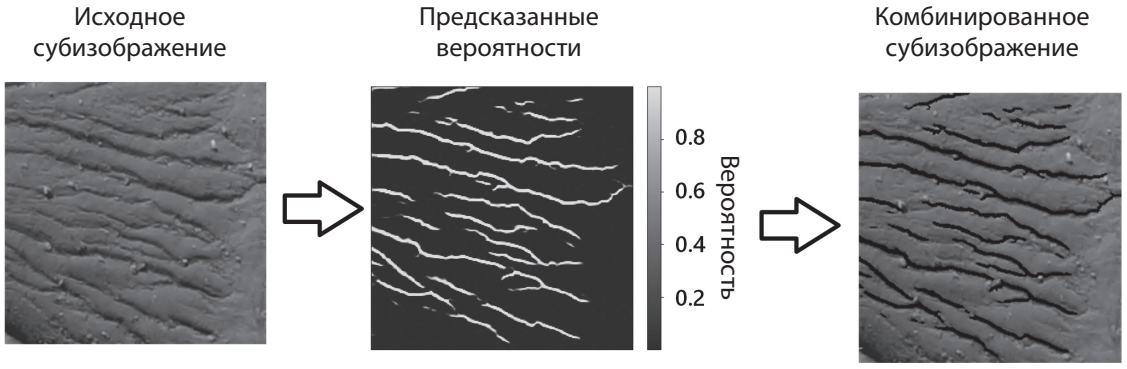


Рис. 8. Схема процесса детектирования трещин для тестового субизображения.

```
labeled_array, num_of_labels = label(img_skel, structure=s)
for key in range(num_of_labels):
    key_points = np.where(labeled_array == key + 1)
    key_points = np.column_stack((key_points[0], key_points[1]))

    labeled_points_tree = KDTree(key_points)
    ...
    for point in key_points:
        neighbor counter = labeled points tree
            .query_ball_point ([point[0], point[1]], r=np.sqrt(2))

        if neighbor_counter == 2: edge_points.append(point)

    labeled_point_pairs_index = np.array(labeled_points_tree
        .query_pairs(r=np.sqrt(2), output_type='ndarray'))

    G = nx.Graph()
    G.add_edges_from(labeled_point_pairs_index, weight=1)
    ...
    longest_path = get_longest_path(edge_points, labeled_points_tree, G)
    ...
    if len(longest_path) > SEGMENT_MIN_LENGTH:
        point_sets.append(key_points[longest_path])
        ...


def get_longest_path(edge_points, labeled_points_tree, G):
    longest_path = []
    for i in range(len(edge_points)):
        for j in range(i + 1, len(edge_points)):
            source = labeled_points_tree.query(edge_points[i])[1]
            target = labeled_points_tree.query(edge_points[j])[1]
                path = nx.shortest_path(G, source=source, target=target)
            if len(path) > len(longest_path):
                longest_path = path
    return longest_path
```

Рис. 9. Фрагмент программного кода получения набора координат пикселов для каждого объекта-трещины.

Для поиска всех возможных пар пикселов для текущей метки применялся метод query_pairs с параметрами r и типом выходного массива output_type, равном “ndarray”. Метод возвращает массив пар (переменная labeled_point_pairs_index), где в паре указаны два индекса из массива key_points, являющихся смежными (рис. 9).

Далее для составления объекта-графа (рис. 10) используется конструктор объекта Graph() (переменная G) и G.add_edges_from, на вход которому передаются переменные labeled_point_pairs_index и weight равная 1. Здесь длина ребра берется равной единице. Вне зависимости от того, как расположены точки в паре по вертикали, горизонтали или по диа-

гонали, важен факт количества точек в пути от одной краевой вершины графа до другой.

Для всех возможных попарных комбинаций пикселов из edge_points в цикле формируются пути прохождения графа между ними, используя метод shortest_path. На вход ему подаются переменная G и переменные source (начальный краевой пиксел) и target (конечный краевой пиксел), на выходе получаем массив индексов элементов (переменная longest_path, рис. 9) из key_points. При этом в цикле ищется максимально возможный по длине массив path, а также удовлетворяющий проверке на пороговое значение длины (переменная SEGMENT_MIN_LENGTH = 5).

Пикселы, соответствующие массиву longest_path, сохраняются в переменной (point_sets), а в массиве img_skel им присваиваются значения 0. Итерация переходит к следующему значению переменной key.

Данный алгоритм также учитывает случай, когда некоторые пикселы образуют три и более пар (трещины имеют ответвления). Тогда после завершения итераций по всем значениям переменной key алгоритм строит новую серию меток и повторяет серию вышеуказанных процедур снова, затем останавливается пока на какой-то итерации, длина массива longest_path будет всегда меньше, чем пороговое значение SEGMENT_MIN_LENGTH.

В переменной point_sets сохраняются координаты пикселов для каждой трещины на бинаризованном изображении. Общая длина трещины складывается из элементарных длин между двумя попарно смежными пикселами, то есть между вертикально- и горизонтально-смежными расстояние равно 20 см, а по диагонали ~28 см (при разрешении 10×10 см).

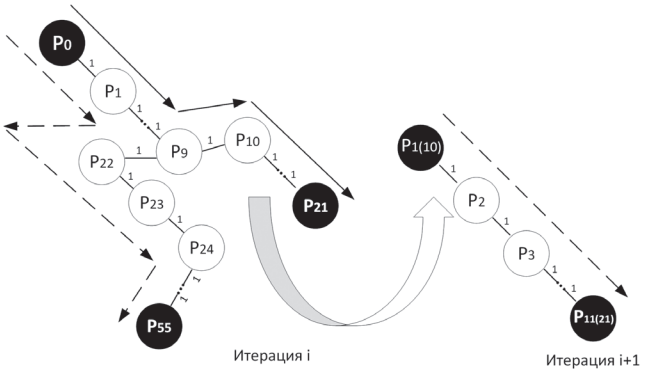


Рис. 10. Пример объекта-графа. Черным обозначены вершины, соответствующие краевым пикселам, стрелками – пути обхода графа между двумя краевыми точками. Штрихованные стрелки – наибольший путь по количеству пикселов на каждой итерации. Ребра графа имеют веса, равные 1.

4.2 Определение площадей трещин

Аналогично для вычисления площади трещины используется метод label. Далее для каждого ключа key подсчитывается общее количество пикселов в текущем объекте-трещине и умножается на 100 (при разрешении изображения 10×10 см).

На рис. 11а, б представлен результат расчета длин и площадей трещин на произвольном изображении аэрофотосъемки (на примере угольного карьера Восточный, Кемеровская область, Россия).

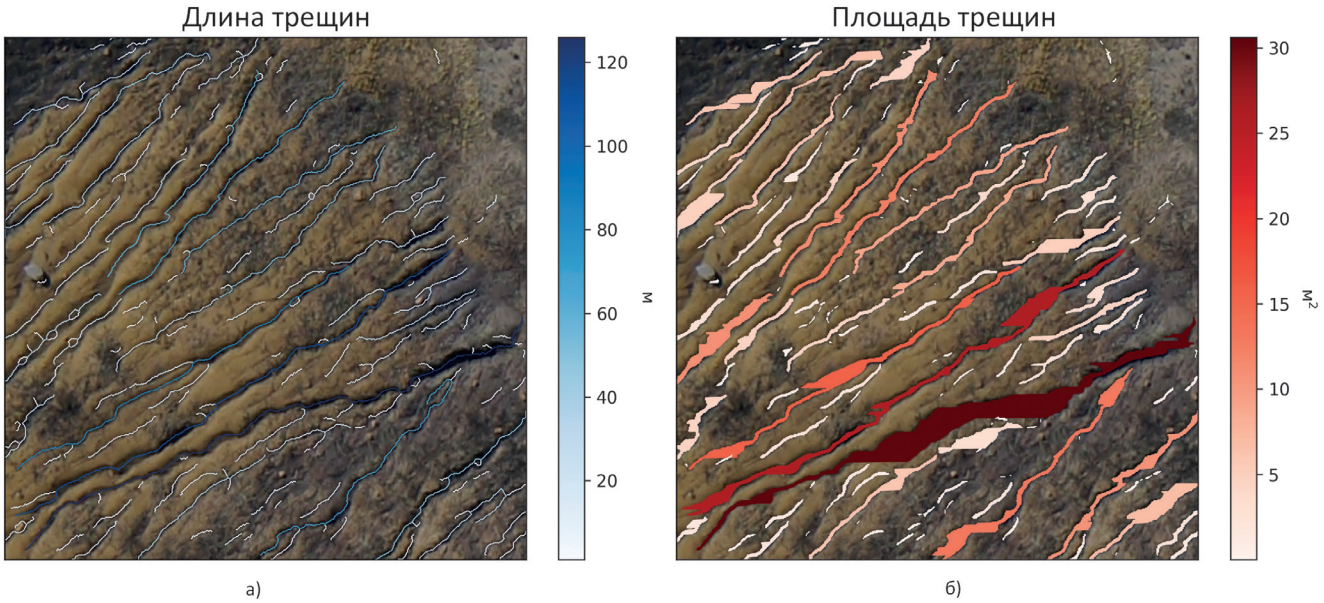


Рис. 11. Результаты расчета длин (слева) и площадей (справа) трещин.

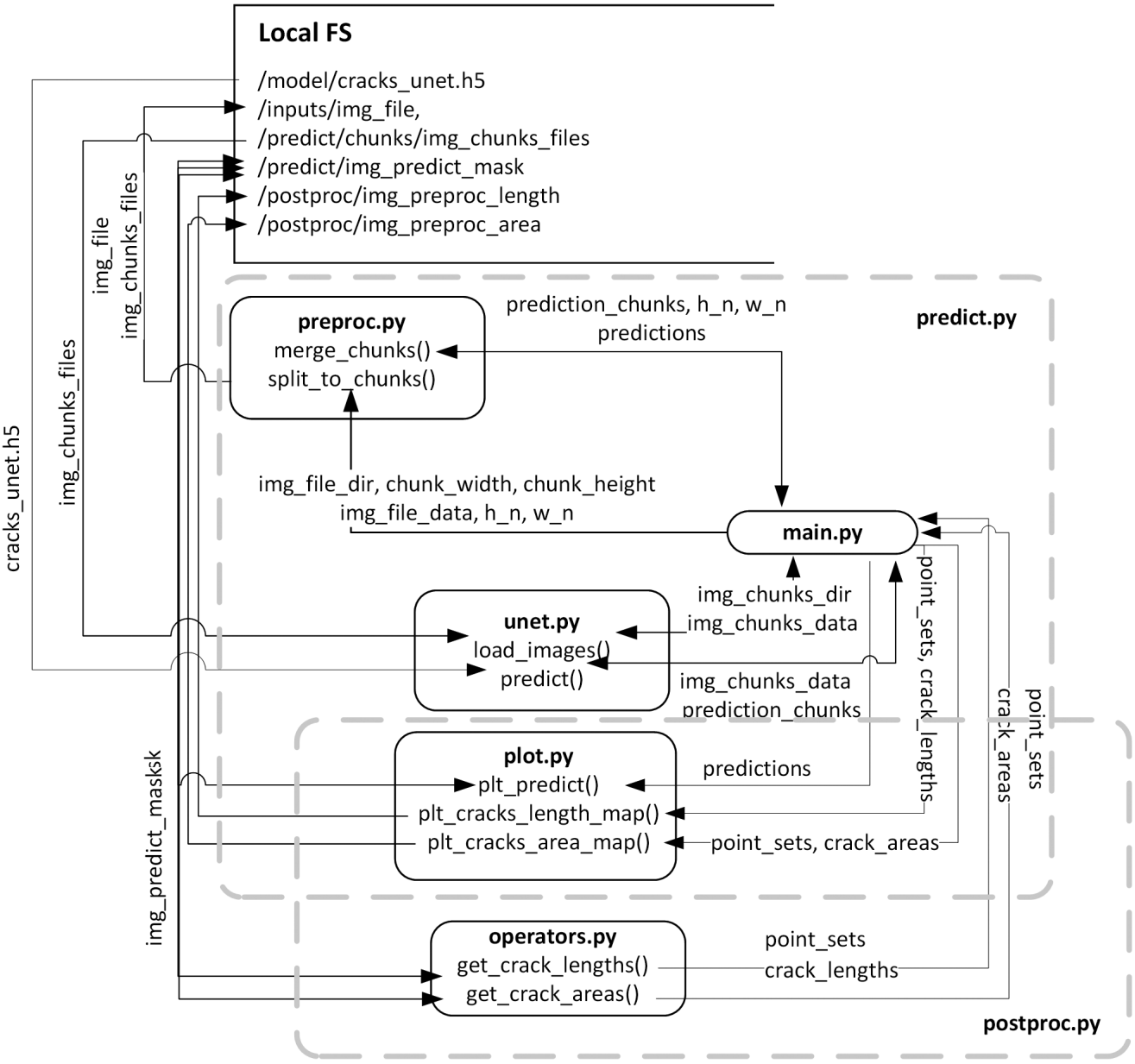


Рис. 12. DFS-диаграмма потоков данных процесса обнаружения трещин и расчета их длин и площадей.

Диаграмма потоков данных для полного алгоритма расчета длин и площадей трещин представлена на рис. 12.

5. ЗАКЛЮЧЕНИЕ

Разработаны алгоритм и его программная реализация, позволяющие автоматизировать процесс детектирования трещин на изображениях, полученных с летательных аппаратов с фото/видео- оборудованием мониторинга геодинамического состояния объектов угледобычи.

Предложенный подход для распознавания трещин на базе аппарата сверточных нейронных сетей позволяет использовать в качестве входных датасетов различные изображения с соответствующей бинаризованной разметкой объектов семантической сегментации. Нейронная сеть поддерживает процесс дообучения. Гибкие параметрические настройки алгоритма дают возможность пропорционально (с шагом разбивки) обрабатывать большие входные изображения нейронной сетью. Сравнение с другими аналогичными нейронными сетями показали относительное превосходство предложенного выбора сети по скорости и точности метрик.

Постпроцессинг, расчет длин и площадей, строится на количественной оценки или подсчете пикселов объектов с пороговой вероятностью больше или равной предсказанной, использует только

величину разрешения изображения и не зависит от RGB-значений. Это позволяет использовать алгоритм на других монохромных (черно-белых) изображениях.

Предложенный подход позволяет в процессе постобработки результатов вычислять количественные характеристики поверхностных неоднородностей (трещин), что имеет практическое значение для определения механического поведения не сплошных массивов горных пород.

Новизна предложенного подхода заключается в применении аппарата нейронных сверточных сетей для автоматизированного поиска характерных риск-объектов на открытых участках (отвалах) угольных разрезов, используя изображения аэрофотосъемки высокого разрешения. Результаты исследования могут быть применены в других предметных задачах многоуровневой адаптивной семантической классификации.

СПИСОК ЛИТЕРАТУРЫ

1. *Potapov V.P., Oparin V.N., Mikov L.S., Popov S.E.* Information Technologies in Problems of Nonlinear Geomechanics. Part I: Earth Remote Sensing Data and Lineament Analysis of Deformation Wave Processes. Journal of Mining Science. 2022. T. 58. P. 486–50.
2. *Hao X., Du W., Zhao Y., Sun Z., Zhang Q., Wang S., Qiao H.* Dynamic tensile behaviour and crack propagation of coal under coupled static-dynamic loading. Int. J. Min. Sci. Technol. 2020. T. 30. P. 659–668.
3. *Krull B., Patrick J., Har, K., White S., Sottos N.* Automatic optical crack tracking for double cantilever beam specimens. Exp. Tech. 2016. T. 40. P. 937–945.
4. *Sun H., Liu Q., Fang L.* Research on fatigue crack growth detection of M (T) specimen based on image processing technology. J. Fail. Anal. Prev. 2018. T. 18. P. 1010–1016.
5. *Zhang W., Zhang Z., Qi D., Liu Y.* Automatic crack detection and classification method for subway tunnel safety monitoring. Sensors. 2014. T. 14. P. 19307–19328.
6. *Kong X., Li J.* Vision-based fatigue crack detection of steel structures using video feature tracking. Comput.-Aided Civ. Inf. 2018. T. 33. P. 783–799.
7. *Kong X., Li J.* Non-contact fatigue crack detection in civil infrastructure through image overlapping and crack breathing sensing. Automat. Constr. 2019. T. 99. P. 125–139.
8. *Li D., Huang P., Chen Z., Yao G., Guo X., Zheng X., Yang Y.* Experimental study on fracture and fatigue crack propagation processes in concrete based on DIC technology. Eng. Fract. Mech. 2020. T. 235. P. 107–166.
9. *Vanlanduit S., Vanherzeele, J., Longo, R. Guillaume P.* A digital image correlation method for fatigue test experiments. Opt. Laser. Eng. 2009. T. 47. P. 371–378.
10. *Valença J., Dias-da-Costa D., Júlio E., Araújo H., Costa H.* Automatic crack monitoring using photogrammetry and image processing. Measurement. 2013. T. 46. P. 433–441.
11. *Yeum C.M., Dyke S.J.* Vision-based automated crack detection for bridge inspection. Comput.-Aided Civ. Inf. 2015. T. 30. P. 759–770.
12. *Dong L., Tang Z., Li X., Chen Y., Xue J.* Discrimination of mining microseismic events and blasts using convolutional neural networks and original waveform. J. Cent. S. Univ. 2020. T. 27. P. 3078–3089.
13. *Yu Y., Wang C., Gu X., Li J.* A novel deep learning-based method for damage identification of smart building structures. Struct. Health Monit. 2019. T. 18. P. 143–163.
14. *Su C., Wang W.* Concrete Cracks Detection Using Convolutional Neural Network Based on Transfer Learning. Mathematical Problems in Engineering. 2020. Article ID 7240129. 10 p. DOI: 10.1155/2020/7240129
15. *Pauly L., Hogg D., Fuentes R., Peel H.* Deeper networks for pavement crack detection. In Proc. 34th ISARC. 2017. P. 479–485.
16. *Maeda H., Sekimoto Y., Seto T., Kashiyama T., Omata Y.* Road damage detection using deep neural networks with images captured through a smartphone. arXiv preprint. 2018. P. 1-14. URL: https://arxiv.org/abs/1801.09454
17. *Xu H., Su X., Wang Y., Cai H., Cui K., Chen X.* Automatic bridge crack detection using a convolutional neural network. Applied Sciences. 2019. V. 9(14). P. 2867. DOI: 10.3390/app9142867
18. *Yuan Y., Ge Z., Su X., Guo X., Suo T., Liu Y., Yu Q.* Crack Length Measurement Using Convolutional Neural Networks and Image Processing. Sensors. 2021. T. 21. P. 5894.
19. *Gehri N., Mata-Falcón J., Kaufmann W.* Automated crack detection and measurement based on digital image correlation. Construction and Building Materials. 2020. T. 256. P. 119383. DOI: 10.1016/j.conbuildmat.2020.119383
20. *Shelhamer E., Long J., Darrell T.* Fully Convolutional Networks for Semantic Segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2017. T. 39. № 4. P. 640-651. DOI: 10.1109/TPAMI.2016.2572683
21. *Chen L., Zhu Y., Papandreou G., Schroff F., Adam H.* Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. Proc. European conference on computer vision (ECCV). 2018. P. 801–818. URL: https://arxiv.org/abs/1802.02611

SOFTWARE IMPLEMENTATION OF THE ALGORITHM FOR AUTOMATIC DETECTION OF LINEAMENTS AND THEIR PROPERTIES ON OPEN-PIT DUMPS

S. E. Popov[a], V. P. Potapov[a], R. Y. Zamaraev[a]

[a] *Federal Research Center for Information and Computational Technologies, 6, Academician M.A. Lavrentiev av., Novosibirsk, 630090, Russia*

The paper presents an algorithm and a description of its software implementation for detecting lineaments (ground erosions or cracks) in aerial photography images of open-pits. The proposed approach is based on the apparatus of convolutional neural networks based on the semantic classification of binarized images of objects (lineaments), as well as graph theory for determining the geometric location of linearized objects, followed by determining their lengths and areas. Three-channel RGB images of high-resolution aerial photography (pixel 10x10 cm) were used as initial data. The software unit of the model is logically divided into three layers: pre-processing, detection and post-processing. The first level includes preprocessing of input data to form a training sample based on successive transformations of an RGB image into a binary one using the OpenCV library. A neural network of the U-Net type, which includes blocks of the convolutional (Encoder) and scanning parts (Decoder), represents the second level of the information model. At this level, automatic lineament detection (washouts) is implemented. The third level of the model is responsible for calculating the areas and lengths of lineaments. The result of the work of the convolutional neural network is transferred to the input. Lineament area is calculated by summing the total number of points multiplied by the pixel size. The length of the lineaments is computed by linearizing a plane object into a line segmental object with nodal points and then calculating the lengths between them, also relying on the resolution of the original image. The software module can work with input images, with their subsequent resulting merging to the size of the original image.

Keywords: detection of lineaments, cracks and ground erosion, convolutional neural networks, semantic segmentation, detection and accumulation, aerial photography

REFERENCES

1. *Potapov V.P., Oparin V.N., Mikov L.S., Popov S.E.* Information Technologies in Problems of Nonlinear Geomechanics. Part I: Earth Remote Sensing Data and Lineament Analysis of Deformation Wave Processes. Journal of Mining Science, 2022, vol. 58, pp. 486–50.
2. *Hao X., Du W., Zhao Y., Sun Z., Zhang Q., Wang S., Qiao H.* Dynamic tensile behaviour and crack propagation of coal under coupled static-dynamic loading. Int. J. Min. Sci. Technol, 2020, vol. 30, pp. 659–668.
3. *Krull B., Patrick J., Har, K., White S., Sottos N.* Automatic optical crack tracking for double cantilever beam specimens. Exp. Tech., 2016, vol. 40, pp. 937–945.
4. *Sun H., Liu Q., Fang L.* Research on fatigue crack growth detection of M (T) specimen based on image processing technology. J. Fail. Anal. Prev., 2018, vol. 18, pp. 1010–1016.
5. *Zhang W., Zhang Z., Qi D., Liu Y.* Automatic crack detection and classification method for subway tunnel safety monitoring. Sensors, 2014, vol. 14, pp. 19307–19328.
6. *Kong X., Li J.* Vision-based fatigue crack detection of steel structures using video feature tracking. Comput.-Aided Civ. Inf., 2018, vol. 33, pp. 783–799.
7. *Kong X., Li J.* Non-contact fatigue crack detection in civil infrastructure through image overlapping and crack breathing sensing. Automat. Constr., 2019, vol. 99, pp. 125–139.
8. *Li D., Huang P., Chen Z., Yao G., Guo X., Zheng X., Yang Y.* Experimental study on fracture and fatigue crack propagation processes in concrete based on DIC technology. Eng. Fract. Mech., 2020, vol. 235, pp. 107–166.
9. *Vanlanduit S., Vanherzeele, J., Longo, R. Guillaume P.* A digital image correlation method for fatigue test experiments. Opt. Laser. Eng., 2009, vol. 47, pp. 371–378.
10. *Valença J., Dias-da-Costa D., Júlio E., Araújo H., Costa H.* Automatic crack monitoring using photogrammetry and image processing. Measurement, 2013, vol. 46, pp. 433–441.
11. *Yeum C.M., Dyke S.J.* Vision-based automated crack detection for bridge inspection. Comput.-Aided Civ. Inf., 2015, vol. 30, pp. 759–770.
12. *Dong L., Tang Z., Li X., Chen Y., Xue J.* Discrimination of mining microseismic events and blasts using convolutional neural networks and original waveform. J. Cent. S. Univ., 2020, vol. 27, pp. 3078–3089.
13. *Yu Y., Wang C., Gu X., Li J.* A novel deep learning-based method for damage identification of smart building structures. Struct. Health Monit., 2019, vol. 18, pp. 143–163.
14. *Su C., Wang W.* Concrete Cracks Detection Using Convolutional Neural Network Based on Transfer Learning. Mathematical Problems in Engineering, 2020, Article ID 7240129, 10 p. DOI: 10.1155/2020/7240129
15. *Pauly L., Hogg D., Fuentes R., Peel H.* Deeper networks for pavement crack detection. In Proc. 34th ISARC, 2017, pp. 479–485.

16. *Maeda H., Sekimoto Y., Seto T., Kashiyama T., Omata Y.* Road damage detection using deep neural networks with images captured through a smartphone. arXiv preprint, 2018, pp. 1–14. URL: https://arxiv.org/abs/1801.09454
17. *Xu H., Su X., Wang Y., Cai H., Cui K., Chen X.* Automatic bridge crack detection using a convolutional neural network. Applied Sciences, 2019, vol. 9, no. 14, p. 2867. DOI: 10.3390/app9142867
18. *Yuan Y., Ge Z., Su X., Guo X., Suo T., Liu Y., Yu Q.* Crack Length Measurement Using Convolutional Neural Networks and Image Processing. Sensors, 2021, vol. 21, p. 5894.
19. *Gehri N., Mata-Falcón J., Kaufmann W.* Automated crack detection and measurement based on digital image correlation. Construction and Building Materials, 2020, vol. 256. p. 119383. DOI: 10.1016/j.conbuildmat.2020.119383
20. *Shelhamer E., Long J., Darrell T.* Fully Convolutional Networks for Semantic Segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017, vol. 39, no. 4, pp. 640–651. DOI: 10.1109/TPAMI.2016.2572683
21. *Chen L., Zhu Y., Papandreou G., Schroff F., Adam H.* Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. Proc. European conference on computer vision (ECCV). 2018. pp. 801-818. URL: https://arxiv.org/abs/1802.02611

ПРОГРАММИРОВАНИЕ, *2024*, *№ 1*, *с. 53–65*

ПРОГРАММНАЯ ИНЖЕНЕРИЯ, ТЕСТИРОВАНИЕ И ВЕРИФИКАЦИЯ ПРОГРАММ

УДК: 004.428

БИБЛИОТЕКА KIAM ASTRODYNAMICS TOOLBOX ДЛЯ ПРОЕКТИРОВАНИЯ ОРБИТАЛЬНОГО ДВИЖЕНИЯ КОСМИЧЕСКИХ АППАРАТОВ

М. Г. Широбоков[a, *] (ORCID: 0000-0002-1747-6430),
С. П. Трофимов[a, **] (ORCID: 0000-0002-2850-5292)

[a]*Институт прикладной математики им. М.В. Келдыша РАН,*
125047 Москва, Миусская пл., д. 4, Россия

**E-mail: shirobokov@keldysh.ru*
***E-mail: trofimov@keldysh.ru*

Поступила в редакцию 16.02.2023
После доработки: 15.04.2023
Принята к публикации: 05.05.2023

Представлен обзор новой программной библиотеки для проектирования и моделирования орбитального движения космического аппарата KIAM Astrodynamics Toolbox. Библиотека создана в Институте прикладной математики им. М.В. Келдыша РАН на языках Fortran и Python и, таким образом, сочетает в себе скорость и гибкость. Библиотека будет полезна специалистам в области механики космического полета, а также обучающимся по соответствующим программам подготовки.

Ключевые слова: библиотека программного обеспечения, траектория, космический аппарат, межпланетный полет, Python, Fortran

DOI: 10.31857/S0132347424010058 **EDN:** HKPXDM

1. ВВЕДЕНИЕ

Проектирование и оптимизация космических траекторий – краеугольный камень в процессе разработки космических миссий. Соответствующие задачи, решаемые специалистами по механике космического полета (астродинамике), разнообразны: поиск оптимальных по затратам топлива или времени полета траекторий с учетом ограничений, исследование траекторий на чувствительность к начальным условиям и параметрам аппарата, исследование динамики вблизи небесных тел и условий радиовидимости аппарата с наземных станций наблюдения, расчет накопленной дозы ионизирующего излучения и продолжительности движения в тени небесных тел. Решение этих задач выполняется, как правило, численными методами и требует быстрых и удобных способов моделирования управляемого и неуправляемого движения космических аппаратов.

Существует множество программных комплексов, помогающих решать задачи астродинамики. Среди тех, что пользуются большим интересом, можно назвать продукт Systems Tool Kit (STK) компании Analytical Graphics [23]. Эта программа первоначально была создана для анализа спутниковых орбит и называлась Satellite Tool Kit; сейчас эта программа предоставляет интегрированную среду для анализа сложных космических, воздушных и морских систем для аэрокосмических и оборонных приложений. STK имеет графический интерфейс и позволяет производить анализ и выбор орбит, оптимизировать траектории перелета, моделировать стыковку аппаратов и учитывать внешние возмущения. Сходными возможностями для решения астродинамических задач обладает комплекс FreeFlyer компании a.i. solutions [24]. Кроме стандартных возможностей этот комплекс позволяет рассчитывать оптимальные маневры перехода между орбитами, причем как для импульсной, так и для непрерывной модели управления. Еще один популярный программный комплекс, General Mission Analysis Tool (GMAT), разрабатываемый специалистами космического агентства NASA, служит для проектирования реальных космических миссий в околоземном пространстве и дальнем космосе [25]. Это свободное программное обеспечение (ПО) с открытым исходным кодом представляет собой отдельную программу с интерфейсами для популярных языков программирования, таких как Python и MATLAB. Программа позволяет учесть многочисленные технические особенности реализации миссий, например, моделировать навигационные системы и связь. Одной из проблем при использовании данной программы является отсутствие возможностей автоматизации моделирования (кодирование происходит на собственном ограниченном языке сценариев). Другие примеры включают High-Precision Orbit

Propagator (HPOP) компании Microcosm, SatTrack Suite компании Bester, Aero/Astro Vehicle Control компании Princeton Satellite Systems, но у них меньше возможностей, чем у вышеназванных крупных программных комплексов. В Институте прикладной математики им. М.В. Келдыша РАН тоже был создан программный комплекс для моделирования вращательного и орбитального движения космических аппаратов [26]. Комплекс позволяет решать большое разнообразие задач, содержит модели внешней среды, модели датчиков и актюаторов, модель работы бортового компьютера и разнообразные алгоритмы управления движением. Комплекс заточен на задачи вращательного и околоземного орбитального движения и, как и продукты выше, представляет собой отдельное ПО со своим интерфейсом.

К известным библиотекам методов проектирования движения космического относится библиотека Fortran Astrodynamics Toolkit [27], разработанная Джейкобом Уильямсом (Jacob Williams) на современной версии языка Fortran. Библиотека содержит средства для решения задачи Ламберта (двухточечной краевой задачи в рамках динамики двух тел), методы интегрирования уравнений движения, функции для моделирования гравитационного поля Земли, обращения к эфемеридам небесных тел, преобразования орбитальных элементов и построения гало-орбит (периодических орбит вокруг точек либрации в модели круговой ограниченной задачи трех тел). Библиотека является открытой и свободной для использования. Отметим, что библиотека имеет существенно ограниченные функциональные возможности для того, чтобы можно было ее применять на стадиях предварительного анализа миссии. Так, она не содержит систем координат, связанных с Землей и Луной, функций для расчета сложного поля Луны и некоторых часто встречающихся в анализе миссий возмущений (например, давления солнечного излучения, силы сопротивления атмосферы). К сожалению, проект сейчас не развивается.

Библиотека pykep – это открытая программная библиотека, разработанная Европейским космическим агентством для исследований в области астродинамики и решения в первую очередь задач орбитального маневрирования [28]. Основа библиотеки написана на C++ и доступна для Python. В основе библиотеки лежат реализация эффективного решателя многооборотной задачи Ламберта, инструменты для решения задач оптимизации с малой тягой, эффективные кеплеровские пропагаторы, интеграторы Тейлора и многое другое. Выпуски новых версий pykep происходят один или два раза в год. Из недостатков можно отметить отсутствие преобразований между полезными для прикладных исследований систем координат, отсутствие возможности интегрирования траекторий с возмущениями, а также технические проблемы – необходимость сборки пакета на своей машине, отсутствие документированной установки сторонних пакетов и самостоятельную настройку Python–C++ интерфейса. Распространены и нерешенные проблемы с установкой пакетов из репозитория conda-forge [29].

Poliastro – еще одна открытая библиотека методов на языке Python для моделирования движения в орбитальной механике [30]. Библиотека сфокусирована на быстром и удобном рисовании орбит и позволяет аналитически и численно интегрировать дифференциальные уравнения движения космических аппаратов, выполнять преобразования между переменными, системами координат, рассчитывать маневры между орбитами, работать с эфемеридами. Этот проект может быть полезен для обучающихся астродинамике, содержит интересные объектно-ориентированные инструменты для моделирования орбит и решения задач маневрирования, однако библиотека недостаточно развита для ее использования в реальных приложениях: весь код написан на высокоуровневом языке Python, что делает его исполнение медленным в высоконагруженных расчетах, а интегрирование орбит даже с часто встречающимися возмущениями подразумевает их ручную реализацию.

SPICE Toolkit – крупная программная библиотека астродинамических методов, созданная в NASA для помощи в планировании и интерпретации научных наблюдений с использованием космических приборов, а также для моделирования, планирования и выполнения мероприятий в рамках миссий по исследованию планет [31, 32]. Методы в рамках этой библиотеки позволяют рассчитывать положения, скорости, размеры, форму и ориентацию планет, спутников, комет и астероидов, а также ориентацию аппарата и его составных частей. Библиотека также позволяет осуществлять временную привязку различных событий, например, определять интервалы времени пребывания спутника в тени планеты или на заданных высотах над небесным телом. Методы SPICE в своей работе обращаются к заранее подготовленным осведомленными источниками первичным наборам данных (“kernels”), которые содержат информацию о небесных телах, системах координат, научных приборах, космических аппаратах, их физические и геометрические параметры. Библиотека была изначально создана на языке

FORTRAN 77, но теперь доступна также на языках C, MATLAB, Java, Python, Julia. К недостаткам SPICE Toolkit можно отнести отсутствие моделей углового и орбитального движения аппарата, средств интегрирования уравнений как пассивного, так и управляемого движения. Библиотека представляет собой набор низкоуровневых подпрограмм; их использование в реальных проектах так или иначе подразумевает создание пользователем вспомогательных высокоуровневых интерфейсов.

Для многих исследователей и специалистов по астродинамике было бы удобно иметь единую поддерживаемую программную библиотеку методов моделирования и проектирования движения аппарата, которая, с одной стороны, была бы написана на распространенном высокоуровневом языке программирования, а с другой стороны, работала быстро. Такая библиотека не только должна иметь астрофизические и геометрические постоянные небесных тел, позволять выполнять преобразования между системами координат, переменными и времени, но также содержать разнообразные настраиваемые модели движения аппарата, возможности проектирования и оптимизации траекторий в выбранных моделях. Указанные выше программные библиотеки полностью таким требованиям не удовлетворяют.

Сейчас стали актуальны и процессы импортозамещения ПО отечественными разработками. Особенную важность появление таких свободно распространяемых библиотек имеет и в процессе обучения новых специалистов. Студенты, обучающиеся по соответствующим программам подготовки, зачастую получают от своих научных руководителей научно-исследовательские задачи, решение которых предполагает пользование средствами моделирования движения космического аппарата. В результате они вынуждены затрачивать время на написание вспомогательных подпрограмм (преобразование переменных, переход между системами координат, решение уравнений в вариациях и пр.) вместо того, чтобы сосредотачиваться на исходной астродинамической задаче. Студенты могут сравнивать собственные реализации вспомогательных программ с теми, что есть в библиотеке.

Данная статья посвящена описанию разработанной в Институте прикладной математики им. М.В. Келдыша РАН программной библиотеки численных методов KIAM Astrodynamics Toolbox для моделирования орбитального движения космического аппарата. Библиотека изначально была написана в среде MATLAB [33, 34, 35] и на данный момент полностью переписана на языках Fortran и Python. На Fortran реализованы часто используемые в механике космического полета преобразования и численные методы (преобразования систем координат, функции правых частей дифференциальных уравнений, численные методы интегрирования и др.). На языке Python реализованы интерфейсы для запуска скомпилированных на Fortran функций и высокоуровневый класс Trajectory для работы с объектами, моделирующими траектории космических аппаратов.

Предлагаемая библиотека методов отличается от существующих аналогов тем, что содержит как низкоуровневые часто используемые операции перевода между системами координат, переменных и времени, астрофизические и геометрические параметры небесных тел, так и широко применяемые модели движения, средства распространения траекторий как пассивных, так и с управлением, уравнения в вариациях, матрицы Якоби преобразований, а также высокоуровневые средства проектирования траекторий с автоматическим преобразованием между системами координат, переменными и системами единиц. Библиотека не является надстройкой над сторонними библиотеками, часть из которых не поддерживается и/или имеет собственные интерфейсы, а является единой системой, содержащей наиболее часто используемые функции для проведения исследований орбитальной динамики. Таким образом, библиотека собирает в себе преимущества других отдельно взятых библиотек. Уникальным в библиотеке является автоматический расчет последовательности преобразований систем координат, переменных и систем единиц на основе графов, а также возможность задавать произвольную функцию управления на языке Python, в то время как траектория будет проинтегрирована на Fortran.

В главе 2 приводится общее видение проекта разработки библиотеки и его особенности. В главе 3 приводится Fortran/Python-структура библиотеки, ее основные компоненты и способы компиляции для работы. В главе 4 перечислены основные функциональные возможности библиотеки. Глава 5 резюмирует все сказанное.

2. ВИДЕНИЕ И ОСОБЕННОСТИ ПРОЕКТА

Предполагаемыми пользователями библиотеки являются специалисты в области механики космического полета, проводящие исследования динамики и схем перелетов, решающие свои астродинамические задачи, а также студенты, обучающиеся астродинамике. Во время решения астродинамических задач библиотека может играть вспомогательную

роль, например, через реализацию множества регулярно используемых действий с массивами фазовых состояний космического аппарата и помощь в ответах на часто встречающиеся вопросы о динамике аппарата. Таким образом, пользователи программируют решения своих астродинамических задач, но для удобства могут использовать готовые реализованные средства библиотеки.

На наш взгляд, такое ПО должно быть открытым, так как открытое ПО обладает полезными для разрабатываемой библиотеки преимуществами:

1. Проверяемость. Открытое ПО, исходный код которого открыт, может быть проверено любым человеком, в том числе специалистами из сторонних заинтересованных организаций со своими методами исследования и решения задач, вследствие чего ошибки могут быть обнаружены на ранней стадии проектирования.

2. Привлечение разработчиков для развития проекта. Участие в открытых проектах проще для всех желающих. Современные веб-сервисы помогают создавать ветви разработки и предоставляют удобные инструменты для контроля версий проекта. Это особенно важно для развития предлагаемой библиотеки, так как множество используемых специалистами методов велико и требуются разработчики, в том числе из сторонних организаций, которые способны сделать свой вклад в проект. Кроме того, в разработке ПО сопровождение зачастую является длительным процессом и занимает намного больше времени, чем программирование перед первым выпуском, и у авторов, как ученых в своей предметной области, может не хватать времени на сопровождение. Вместе с тем у них появляются ученики, решающие задачи в своих направлениях, которым было бы полезно добавлять в код новые функциональные возможности и совершенствовать старые.

3. Распространение. Открытое ПО значительно проще распространять, что удобно, так как эта библиотека может быть использована при исполнении контрактов с индустриальными партнерами. Обратная связь благоприятно сказывается на развитии проекта.

4. Обучение. При обучении механике космического полета студенты иногда сталкиваются с проблемами входа в специальность и начала участия в крупных проектах, подразумевающих проектирование траекторий. Наибольшие трудности при этом приходятся на технические (смена систем координат, преобразование переменных, расчет матрицы монодромии и т.п.), а не содержательные аспекты работы. Существование библиотеки численных методов проектирования траекторий облегчает освоение специальности.

Отметим и некоторые недостатки открытого ПО и возможности им противостоять:

1. Опасность для бизнеса. Размещение в открытом доступе исходных кодов проекта может повлечь утечку идей в сторонние организации, имеющие конфликт интересов с организацией, в которой разрабатывается проект. Для того чтобы этого избежать, исходные коды для решения конкретных астродинамических задач не размещаются в открытом доступе и не добавляются в библиотеку. Некоторые исходные коды для решения общих задач и общематематических функций могут содержаться в библиотеке, но быть при этом закрытыми. Наконец, некоторые коды могут быть временно закрыты и открыты спустя некоторое время, когда острая актуальность в них пропадет.

2. Разветвление плохих проектов. На основе изначально размещенного проекта могут быть созданы ветви, которые станут популярными, но которые обладают неочевидными недостатками и ненадежностью. Эту проблему можно частично решить размещением в открытом доступе информации о тех ветвях проекта, которые, с точки зрения авторов начального проекта, являются надежными.

3. Нерегулярность выпусков. Версии открытого ПО, как правило, выходят нерегулярно, но эту проблему можно контролировать, во всяком случае для главной ветви проекта.

В связи с этим было принято решение сделать проект открытым и для удобства контроля версий и ветвей изменений разместить его на веб-сервисе GitHub [36]. На указанном сайте можно найти короткую справку о проекте, скачать библиотеку или присоединиться к ее разработке. Библиотека распространяется в рамках лицензии MIT License [37].

Еще одной особенностью проекта является сочетание скорости и гибкости, что достигается за счет использования компилируемого (Fortran) и популярного сейчас интерпретируемого (Python) языков программирования. На Fortran реализован почти весь функционал, связанный с проведением расчетов: интегрирование уравнений движения, расчет возмущающих ускорений, преобразование массивов переменных, систем координат. На Python реализованы интерфейсы для обращения к скомпилированным на Fortran функциям, а также высокоуровневые классы, например класс Trajectory для проектирования траекторий. Запуск расчетов на компилируемом языке делает их гораздо более быст-

рыми, чем если бы они были реализованы на интерпретируемом языке. Вместе с тем код на Python является гибким, немногословным и простым для освоения.

3. СТРУКТУРА БИБЛИОТЕКИ

Библиотека состоит из модулей на языке Fortran с реализованными астродинамическими функциями и модулей на языке Python, которые представляют собой интерфейсы для обращения к скомпилированным на Fortran функциям и высокоуровневые классы для проектирования траекторий.

Перечислим модули на языке Fortran:

1. **ConstantsAndUnits.f90** содержит астрономические постоянные согласно стандарту Международного астрономического союза (International Astronomical Union, IAU) 2009/2012. Сюда входят гравитационные параметры и размеры небесных тел Солнечной системы и подпрограммы, которые по названию небесного тела (или пары небесных тел, если речь идет о задаче трех тел) выдают системы согласованных единиц расстояния, скорости, времени и ускорения. Перейти к таким единицам бывает полезно, так как тогда уравнения движения записываются в более простом безразмерном виде.

2. **LinearAlgebraInterfaces.f90** содержит интерфейсы к подпрограммам LAPACK, осуществляющим операции линейной алгебры (решение системы линейных уравнений, матричные и векторные умножения). Подпрограммы LAPACK в виде кодов на языке Fortran находятся в открытом доступе [38].

3. **BaseMeansToolbox.f90** содержит набор подпрограмм для удобного создания матриц и векторов (матриц из нулей, единичной матрицы), расчета нормы вектора, скалярного и векторного произведения векторов.

4. **Ephemeris.f90** содержит интерфейсы к программам NASA›s Jet Propulsion Laboratory для расчета положений и скоростей небесных тел. Эти подпрограммы на языке Fortran также находятся в открытом доступе [39]. К этому модулю поставляется файл JPLEPH с эфемеридами модели DE430, который можно скачать на указанном выше сайте. Вместо файла JPLEPH можно использовать любой другой файл с эфемеридами той же структуры, например, эфемериды лаборатории эфемеридной астрономии Института прикладной астрономии РАН [40].

5. **EquationsModule.f90** содержит подпрограммы для расчета правых частей уравнений движения космического аппарата с возмущениями (сложное гравитационное поле Луны, давление солнечного излучения, возмущение за счет гармоники J_2 геопотенциала, атмосферное сопротивление). Реализованы как сами уравнения движения, так и уравнения в вариациях, которые используются для расчета матриц чувствительности фазовых векторов к начальным условиям. Во вспомогательном модуле GravityMoonCoefficients50.f90 содержатся коэффициенты сложного поля Луны до степени и порядка 50 включительно.

6. **Transformations.f90** содержит подпрограммы для преобразования переменных, описывающих движение (например, из декартовых координат в орбитальные элементы и обратно), и систем координат (например, между селеноцентрической небесной системой координат и системой Mean Earth/Mean-Rotation, связанной с селенографическими координатами на поверхности Луны).

7. **OdeToolbox.f90** содержит реализации численных методов интегрирования систем обыкновенных дифференциальных уравнений. Среди имеющихся на данный момент методов – классический метод Рунге–Кутты 4-го порядка, метод Рунге–Кутты 8-го порядка с постоянным шагом, метод Дормана–Принса 5(4), а также многошаговый метод Адамса переменного порядка (до 12-го включительно) с адаптивным шагом.

8. **PropagationModule.f90** содержит подпрограммы для интегрирования уравнений движения с какими-либо из упомянутых выше возмущений указанными методами интегрирования и указанными возмущениями.

9. **VisibilityModule.f90** содержит подпрограммы для расчета радиовидимости космического аппарата из точек на поверхности небесного тела.

Эти модули были скомпилированы в файл **FKIAMToolbox.cp39-win_amd64.pyd** с использованием программы Fortran to Python interface generator (F2PY) [41], поставляемой в пакете numpy с использованием компилятора Intel(r) Visual Fortran Compiler, который свободно распространяется в рамках системы oneAPI [42] компании Intel. В результате в программах на Python можно импортировать скомпилированную библиотеку как **import FKIAMToolbox** и обращаться к функциям внутри через точку. Например, команда

```
FKIAMToolbox.ephemeris.juliandate(year,
month, day, hour, minute, second)
```

обратится к подпрограмме juliandate модуля ephemeris и возвратит юлианскую дату для заданного года, месяца, дня, часа, минуты и секунды. Если

на вход какой-то Fortran-функции нужно передать числовой массив, он должен иметь тип numpy.ndarray в Python.

Из-за особенностей языка программирования Fortran и создаваемого программой F2PY Fortran-Python интерфейса, обращение в прикладных программах к скомпилированному модулю FKIAMToolbox неудобно и даже небезопасно (глобальные параметры внутри FKIAMToolbox легко изменить извне), поэтому для удобства пользователей был написан Python-модуль kiam.py, содержащий интерфейсы для обращения к FKIAMToolbox. Именно эти интерфейсы проверяют формат и правильность данных, которые затем будут переданы в процедуры FKIAMToolbox: эти процедуры направлены на максимизацию скорости вычислений и проверку входных данных, кроме типа данных, не осуществляют.

Кроме модуля kiam.py в библиотеку также входит модуль Trajectory.py, реализующий класс Trajectory для проектирования траекторий космических аппаратов. Траектории могут быть произвольными, они определяются начальными условиями и длиной интервала времени. Класс Trajectory позволяет создавать объекты, содержащие поля states (массивы фазовых состояний) и times (моменты времени), а также другие вспомогательные атрибуты. Класс содержит метод change_system(target_system), переводящий одновременно все фазовые состояния в целевую систему координат, метод change_vars(target_vars), преобразующий одновременно все фазовые состояния из states к новым переменным, и другие методы.

Библиотека включает модуль engine.py, содержащий классы двигательных установок, которые отличаются силой тяги, удельным импульсом, массой, размерами и прочими параметрами. Например, реализованы классы SPT50, SPT50M, SPT70 двигателей малой тяги [43].

Подробную справку о модулях, функциях и классах можно получить по ссылкам на GitHub-странице проекта [44].

Опишем теперь более подробно возможности класса Trajectory для проектирования траекторий.

4. КЛАСС TRAJECTORY ДЛЯ ПРОЕКТИРОВАНИЯ ТРАЕКТОРИЙ

Класс Trajectory определен в модуле Trajectory.py. Его назначение – быстрая и простая генерация траекторий в выбранной модели с заданными начальными условиями и удобное преобразование фазовых векторов траектории между различными системами координат, переменными и единицами измерения.

Метод __init__ принимает на вход начальный вектор состояния, начальный момент времени, начальную юлианскую дату, название используемых для задания состояния переменных, название системы координат и системы единиц измерения. Например, команда

```
tr = Trajectory(numpy.array([1, 0, 0, 0, 1, 0]), 0.0,
2459905.5, 'rv', 'gcrs', 'earth')
```

создает объект tr класса Trajectory с начальным состоянием numpy.array([1,0,0,0,1,0] (радиус-вектор [1,0,0], скорость [0,1,0]), начальным моментом времени 0.0, начальной юлианской датой 2459905.5 (отвечает дате 21.11.2022), в переменных 'rv' (положение–скорость), в системе координат GCRS (геоцентрическая небесная система координат), в системе единиц 'earth' (единица расстояния – средний радиус Земли, единица скорости – первая космическая скорость).

Возможно создание объекта в классических или равноденственных орбитальных элементах, других инерциальных или неинерциальных системах координат, связанных с Солнцем, Луной и планетами Солнечной системы, в других безразмерных или размерных системах единиц. Подробную информацию о всех возможностях можно найти на GitHub-странице проекта [44].

Метод set_model устанавливает модель движения. Например, строка

```
tr.set_model('rv', 'nbp', 'earth', ['moon', 'sun'])
```

задаст модель задачи n-тел ('nbp') в переменных положение–скорость ('rv'), с центральным телом Земля ('earth') и возмущениями от гравитационного притяжения Луной и Солнцем (['moon', 'sun']). Модель движения в случае задачи n-тел может быть задана и в других переменных, например, равноденственных элементах ('ee' вместо 'rv'). При этом не важно, в каких переменных была инициализирована траектория 'tr'. Каждая определенная в классе Trajectory модель имеет собственные переменные, систему координат и систему единиц, которые могут не совпадать с заданными пользователем переменными, системами координат и системами единиц во время создания объекта траектории.

Некоторые данные, используемые для последующего интегрирования траектории, необходимо установить вручную. Например,

```
tr.model['data']['jd_zero'] = jd0
```

установит юлианскую дату jd0, соответствующую $t = 0$ в уравнениях движения,

```
tr.model['data']['mass'] = 100.0
```

задаст массу космического аппарата в кг,

```
tr.model['data']['area'] = 2.0
```

задаст площадь миделева сечения в м^2,

```
tr.model['data']['order'] = 5
```

задаст степень и порядок сложного гравитационного поля Луны.

Распространение по времени (интегрирование) траекторий осуществляется с помощью метода propagate. Например,

```
tr.propagate(100, 1000)
```

запустит интегрирование траектории в рамках выбранной модели на 100 единиц времени вперед и сохранит 1000 равномерно отстоящих по времени узлов, при этом в tr.times запишет узлы времени, а в tr.states запишет фазовые векторы в этих узлах.

Особенностью класса Trajectory является автоматическое преобразование переменных интегрирования, систем координат и систем единиц из тех, что задал пользователь, в те, что используются моделью движения для ее интегрирования. Опишем подробно механизм этого автоматического преобразования.

Класс Trajectory содержит графы преобразований между переменными, между системами координат и между системами единиц. Граф преобразований переменных показан на рис. 1. Здесь rv, oe, ee обозначают соответственно переменные положение–скорость, классические орбитальные элементы и равноденственные орбитальные элементы. Суффикс _stm означает те же векторы переменных, расширенные переходной матрицей (state transition matrix, STM). Суффикс m означает те же векторы переменных, расширенные добавлением массы аппарата. Стрелка означает преобразование, которое реализовано явно в классе Trajectory. Надпись над стрелкой указывает на ограничения, при которых это преобразование может быть выполнено. В данном случае некоторые преобразования ограничены используемыми системами единиц: либо 'earth', либо 'moon'.

Некоторые преобразования действуют только в одну сторону. Например, легко может быть преобразован вектор, содержащий положение, скорость и массу аппарата, в вектор только из положения

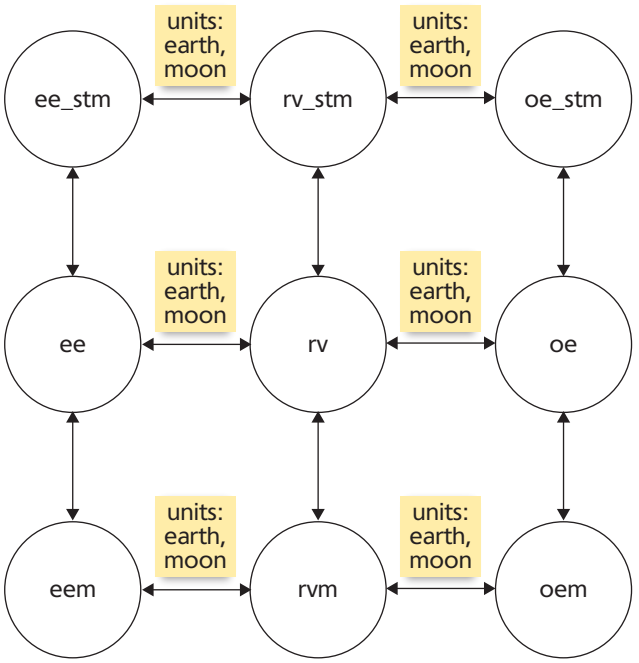


Рис. 1. Граф преобразований переменных в классе Trajectory для моделей движения, использующих эфемериды небесных тел.

и скорости, а обратного преобразования не существует.

Метод change_vars позволяет переключаться между переменными. Например, пусть траектория описывается переменными oe и записана команда tr.change_vars('ee').

В этом случае алгоритм найдет кратчайший пусть от вершины 'oe' до вершины 'ee' и выполнит цепочку преобразований (по порядку обратится к функциям, осуществляющим соответствующие преобразования). Метод change_vars автоматически запускается в методе propagate перед интегрированием уравнений движения, осуществляя переход к переменным выбранной пользователем модели.

Часть графа преобразований систем координат для моделей движения, использующих эфемериды небесных тел, показана на рис. 2. Здесь есть несколько групп систем координат. Системы координат sors, scrs, mer, ssrf_em связаны с Луной, системы itrs, gcrs, gsrf_se, gsrf_em связаны с Землей, а системы hcrs, hsrf_se связаны с Солнцем. Описания аббревиатур можно найти в справке на GitHub [44]. Аналогично в библиотеке реализованы преобразования, отвечающие стрелкам. Метод change_system, реализующий преобразования между любыми системами координат, находит кратчайший путь между вершинами и выполняет всю цепочку необходимых преобразований. Реализованы также преобразования между системами координат в других моделях

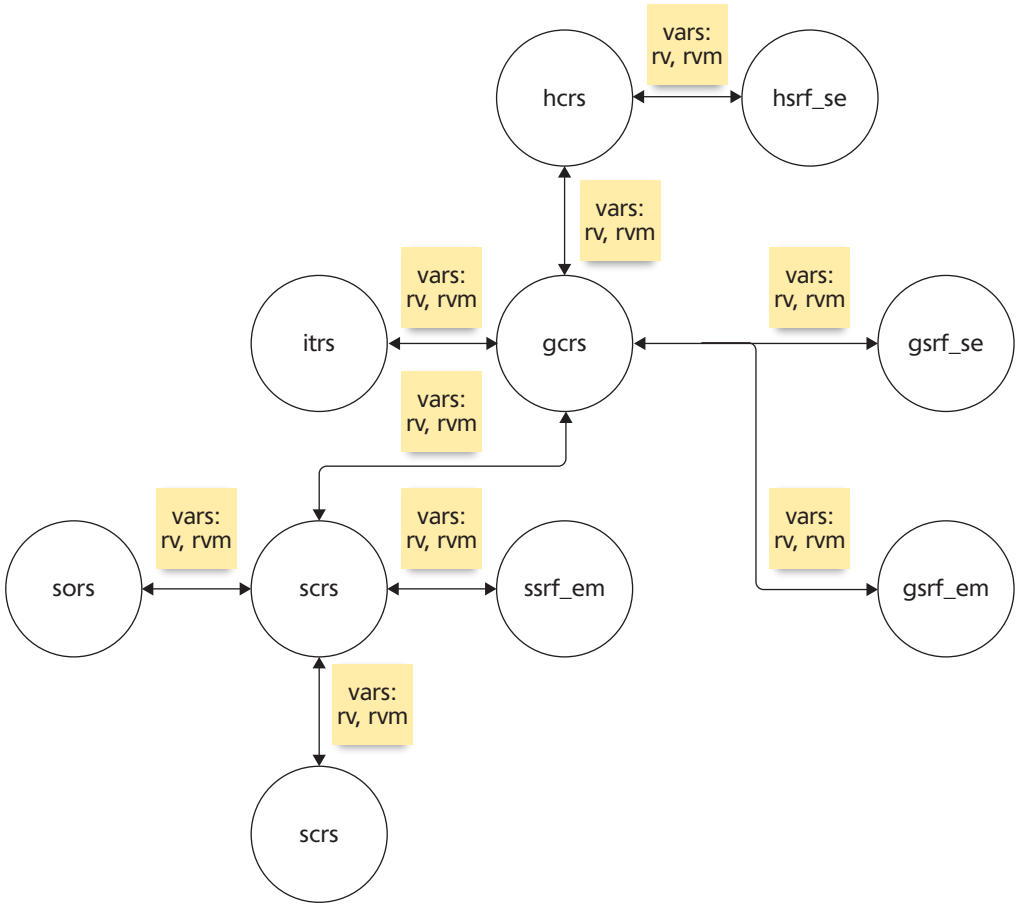


Рис. 2. Граф преобразований систем координат, связанных с Землей, Луной и Солнцем в классе Trajectory.

движения – круговой ограниченной задаче трех тел и бикруговой ограниченной задаче четырех тел, а также в задаче Хилла.

Похожий граф преобразований есть и для преобразования систем единиц; метод, отвечающий за это преобразование, называется change_units.

Итак, в любой момент пользователь может вызвать методы change_vars, change_system, change_units, и тогда преобразования затронут все фазовые состояния в массиве фазовых состояний tr.states. Преобразования переменных и систем координат реализованы в модуле Translations.f90.

Рассмотрим несколько конкретных примеров использования модуля Trajectory. Начнем с простейшего примера. В модели задачи двух тел круговую орбиту можно получить следующим образом (с точностью до настроек осей графика и шрифтов):

```
t0 = 0.0
s0 = numpy.array([1, 0, 0, 0, 1, 0])
jd0 = kiam.juliandate(2022,4,30,0,0,0)
tr = Trajectory.Trajectory(s0, t0, jd0, 'rv', 'gcrs',
'earth')
tr.set_model('rv', 'r2bp', 'earth', [])
tr.propagate(2*numpy.pi, 10000)
fig = tr.show('xy', language='rus')
```

В результате будет рассчитана траектория с начальным временем $t_0 = 0$, начальным состоянием (положение и скорость) x_0=[4.0,0,0,0,0.5,0] в системе отсчета GCRS (геоцентрическая небесная система координат) в безразмерной системе единиц 'earth' (гравитационный параметр равен единице, единица расстояния – радиус Земли, единица скорости – первая космическая скорость) на одном витке (период орбиты равен одной безразмерной единице времени). Последняя строчка запускает команду отображения орбиты в проекции на плоскость земного экватора на рисунке (рис. 3).

Приведем пример интегрирования траектории с управлением (с точностью до настроек осей графика и шрифтов):

```
units = kiam.units('sun')
AU = units['AccUnit']
TU = units['TimeUnit']
engine = SPT50()
```

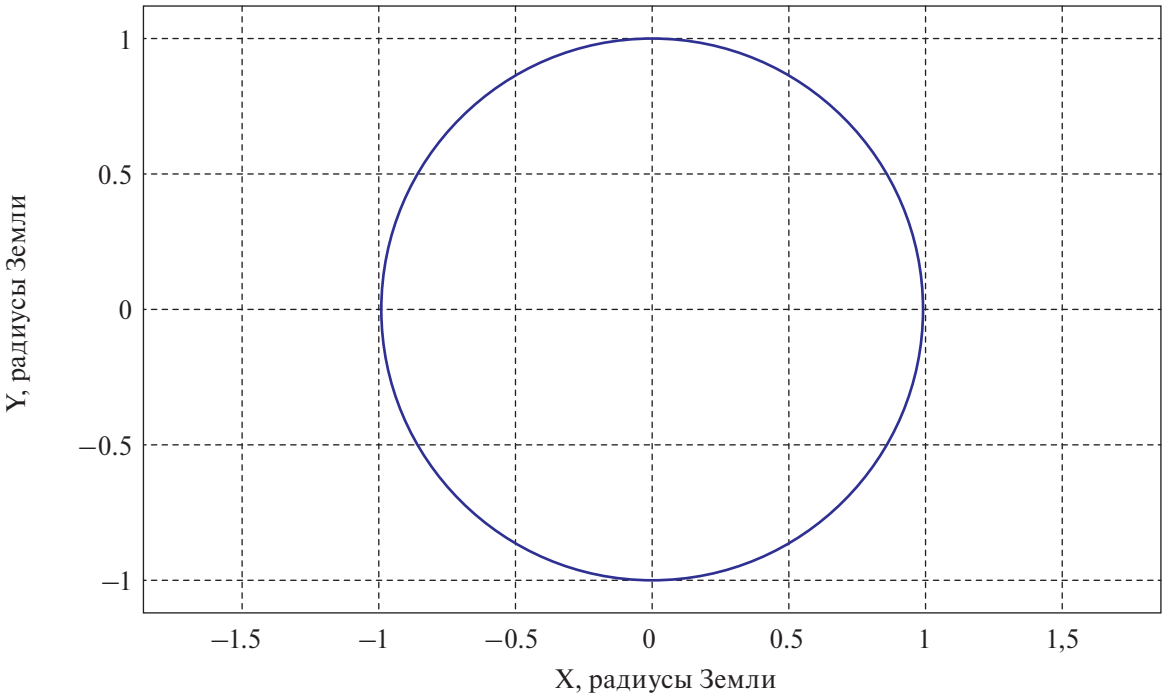


Рис. 3. Круговая экваториальная орбита, проинтегрированная на одном витке в модели задачи двух тел.

```
engine.force /= AU
engine.specific_impulse /= TU*24*3600

def control(t, x):
force_vector = x[3:6]/norm(x[3:6]) * engine.force
specific_impulse = engine.specific_impulse
return force_vector, specific_impulse

t0 = 0.0
jd0 = kiam.juliandate(2023,1,1,0,0,0)
s0 = numpy.array([1,0,0,0,1,0,100])
tr = Trajectory(s0, t0, jd0, 'rvm', 'hcrs', 'sun')
tr.set_model('rvm', 'nbp', 'sun', ['jupiter'])
tr.model['data']['jd_zero'] = jd0
tr.model['data']['area'] = 2.0
tr.model['data']['mass'] = s0[6]
tr.model['control'] = control
tr.propagate(500/TU, 100)
tr.show('xy')
```

В первых трех строках загружается система единиц, связанная с Солнцем: единицей расстояния является 1 а.е., единицей скорости – круговая скорость вокруг Солнца на расстоянии 1 а.е., гравитационный параметр Солнца равен единице. Далее создается экземпляр класса SPT50, содержащий информацию о двигателе малой тяги СПД-50; его сила и удельный импульс приводятся к безразмерной системе единиц. Далее определяется функция control, которая сопоставляет моменту времени и фазовому вектору аппарата силу тяги и удельный импульс. Затем задаются начальные условия, объект траектории, модель. Траектория распространяется на интервале 500 дней. В процессе интегрирования используются уравнения движения и интегратор, реализованные на Fortran, но при каждом обращении к функции правой части производится вызов функции control, написанной на Python. Функция control кроме вектора силы тяги выдает также удельный импульс: это может быть полезно в тех случаях, когда величина удельного импульса может быть переменной. В результате выводится график траектории на плоскости международной небесной системы координат (рис. 4).

Рассмотрим пример с более сложной динамикой: движение вокруг Луны с учетом гравитационного притяжения Земли и Солнца, а также сложного поля Луны до 10-й степени и порядка включительно (с точностью до настроек осей графика и шрифтов):

```
units = kiam.units('moon')
t0 = 0.0
s0 = numpy.array([3, 0.01,
kiam.deg2rad(80), 0, 0, 0])
jd0 = kiam.juliandate(2022,4,30,0,0,0)
tr = Trajectory.Trajectory(s0, t0, jd0, 'oe', 'mer',
'moon')
tr.set_model('rv', 'nbp', 'moon', ['earth', 'sun',
'cmplxmoon'])
tr.model['data']['jd_zero'] = jd0
```

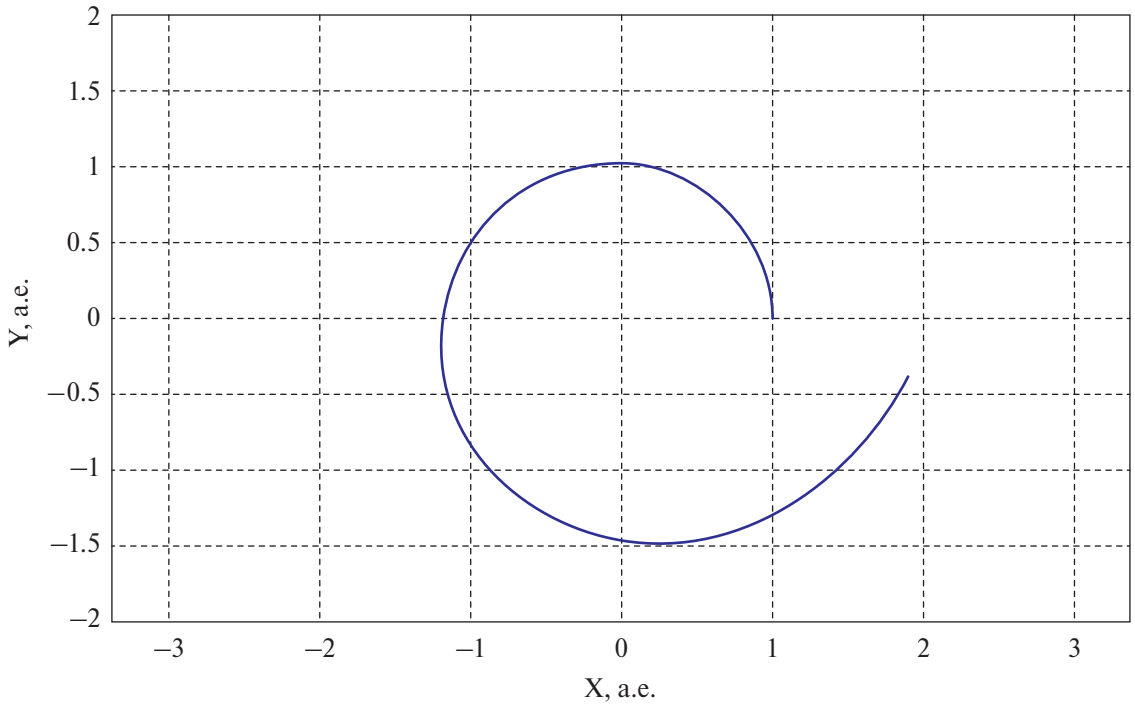


Рис. 4. Пример плоской траектории с малой тягой вокруг Солнца.

```
tr.model['data']['mass'] = 100.0
tr.model['data']['area'] = 2.0
tr.model['data']['order'] = 10
tr.propagate(10/units['TimeUnit'], 5000)
tr.change_system('mer')
tr.change_vars('oe')
tr.change_units('dim')
tr.show('e', language='rus')
```

В результате будет получена траектория с начальным временем $t_0 = 0$, начальными классическими орбитальными элементами $a_0 = 3$ (большая полуось, безразмерные единицы), $e_0 = 0.01$ (эксцентриситет орбиты), $i_0 = 80°$ (наклонение орбиты), $\Omega_0 = 0°$ (долгота восходящего узла), $\omega_0 = 0°$ (аргумент перицентра), $\vartheta_0 = 0°$ (истинная аномалия) на интервале времени 10 дней. Орбитальные элементы даны в связанной с Луной системе координат MER.

При исполнении команды tr.propagate переменные будут трансформированы в декартовы положение и скорость, а система координат – в систему SCRS, так как именно в этих терминах определяется модель и правые части уравнений, соответствующие параметрам 'rv' и 'nbp' в строке с tr.set_model. После интегрирования уравнений движения переменные останутся 'rv' (положение и скорость), а система координат останется SCRS. Так как орбитальными элементами обычно интересуются в связанной с небесным телом системе координат, в примере выше после интегрирования идут команды, которые возвращают все фазовые состояния рассчитанной траектории в систему MER, классическим орбитальным элементам и размерной системе единиц. Последняя строка отображает график эволюции эксцентриситета с течением времени (рис. 5).

Для анализа видимости поверхности небесного тела полезно визуализировать подспутниковую трассу орбиты – точки поверхности тела, в которых ее пересекает радиус-вектор аппарата в ходе орбитального движения. На следующем примере показано, как отобразить подспутниковые точки на низкой окололунной орбите. Сначала рассчитывается окололунная орбита:

```
    units = kiam.units('moon')
    t0 = 0.0
    s0 = numpy.array([1.2, 0.01,
kiam.deg2rad(80), 0, 0, 0])
    jd0 = kiam.juliandate(2022,4,30,0,0,0)
    tr = Trajectory.Trajectory(s0, t0, jd0, 'oe', 'mer',
'moon')
    tr.set_model('rv', 'nbp', 'moon', ['earth', 'sun',
'cmplxmoon'])
    tr.model['data']['jd_zero'] = jd0
    tr.model['data']['mass'] = 100.0
    tr.model['data']['area'] = 2.0
    tr.model['data']['order'] = 40
    tr.propagate(1.0/units['TimeUnit'], 10000)
    tr.change_system('mer')
```

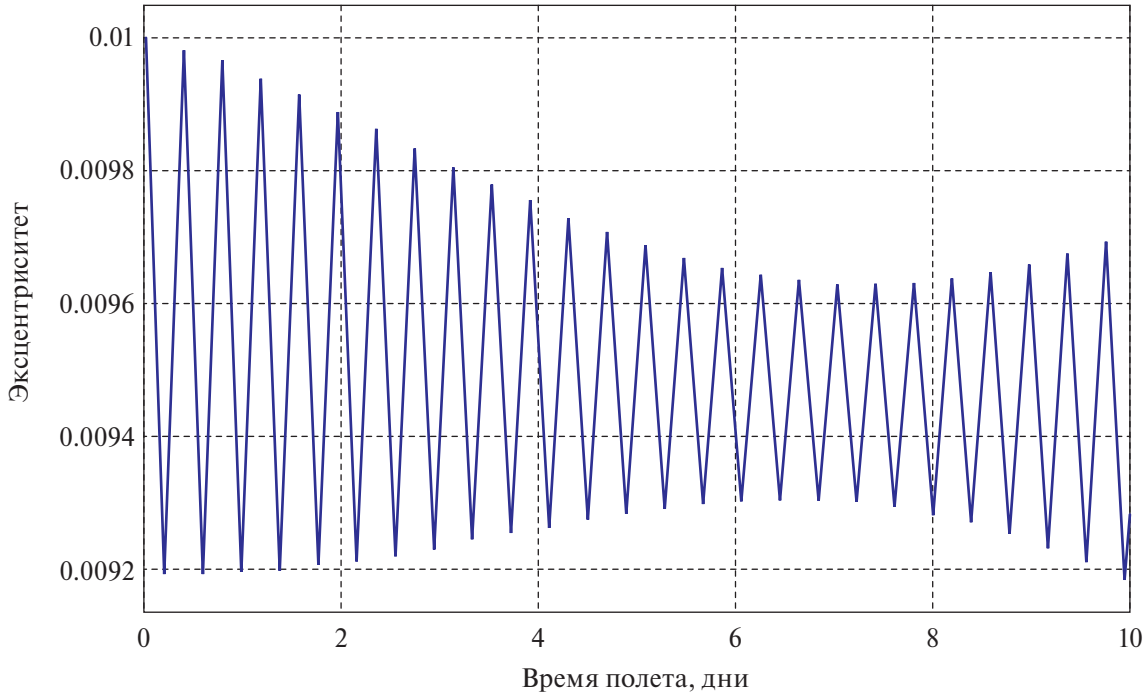


Рис. 5. Эволюция эксцентриситета окололунной орбиты в сложной модели движения.

Далее средствами рисования графиков (например, пакета plotly) формируется графический объект траектории

```
traj_go = go.Scatter3d(
x=tr.states[0, :],
y=tr.states[1, :],
z=tr.states[2, :],
mode='lines',
line='color': 'rgb(178,34,34)', 'width': 3),
```

рассчитываются расстояния до притягивающего центра

```
r = numpy.sqrt(tr.states[0, :] ** 2 +
tr.states[1, :] ** 2 +
tr.states[2, :] ** 2)
```

координаты подспутниковых точек (радиус Луны принят равным единице)

```
traj_surf_go = go.Scatter3d(
x=tr.states[0, :]/r,
y=tr.states[1, :]/r,
z=tr.states[2, :]/r,
mode='lines',
line='color': 'rgb(178,34,34)', 'width': 3)
```

В модуле kiam.py есть возможность отображения поверхности Луны:

```
fig_moon = kiam.body_surface('moon')
moon_go = fig_moon['data'][0]
```

Наконец, на экран выводятся все графические объекты:

```
fig_traj = go.Figure()
fig_traj.add_trace(traj_go)
fig_traj.add_trace(moon_go)
fig_traj.add_trace(traj_surf_go)
```

Результат приводится на рис. 6.

5. ЗАКЛЮЧЕНИЕ

Разработана новая программная библиотека для проектирования и моделирования орбитального движения космического аппарата KIAM Astrodynamics Toolbox. Библиотека написана на языках Fortran (основные вычислительные модули) и Python (интерфейсы к ним, а также высокоуровневые классы). Библиотека позволяет выполнять преобразования переменных, систем координат и систем единиц, получать эфемеридную информацию о движении небесных тел, интегрировать и визуализировать траектории космических аппаратов в выбранной модели с заданными начальными условиями. Библиотека находится в открытом доступе, свободна для использования и рассчитана на специалистов в области механики космического полета и обучающихся по соответствующим программам подготовки. Она является альтернативой существующим проприетарным решениям с пользовательским графическим интерфейсом, а также аналогичным прог-

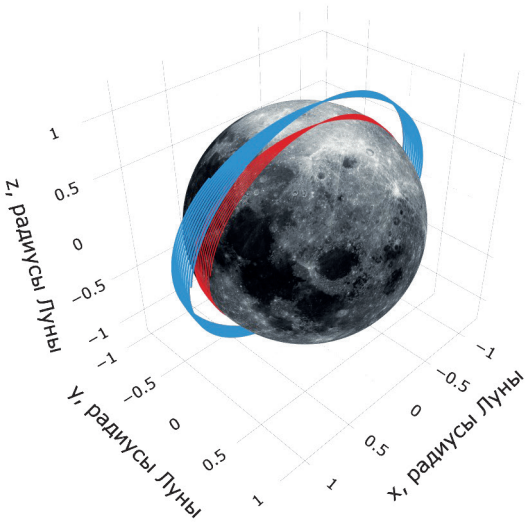


Рис. 6. Низкая окололунная орбита с подспутниковой трассой.

раммным библиотекам, заметно ограниченным по функционалу.

СПИСОК ЛИТЕРАТУРЫ

1. AGI Products // Сайт компании «AGI» (https://www.agi.com/products). Просмотрено: 14.04.2023.
2. FreeFlyer // Сайт компании «a.i. solutions» (https://ai-solutions.com/freeflyer/). Просмотрено: 14.04.2023.
3. GMAT // Хостинг проектов «SourceForge.net» (https://sourceforge.net/projects/gmat/files/GMAT/GMAT-R2020a/). Просмотрено: 14.04.2023.
4. *Иванов Д.С., Овчинников М.Ю., Ролдугин Д.С., Ткачев С.С., Трофимов С.П., Шестаков С.А., Широбоков М.Г.* Программный комплекс для моделирования орбитального и углового движения спутников // Математическое моделирование. 2019. Т. 31. № 12. С. 44–56.
5. Fortran Astrodynamics Toolkit // Страница на веб-сервисе GitHub (https://github.com/jacobwilliams/Fortran-Astrodynamics-Toolkit). Просмотрено: 14.04.2023.
6. Pykep // Страница на веб-сервисе GitHub (https://esa.github.io/pykep/). Просмотрено: 14.04.2023.
7. Can't install anything using conda, it hangs in solving environment // Страница на веб-сервисе GitHub (https://github.com/conda/conda/issues/8051). Просмотрено: 14.04.2023.
8. Poliastro // Страница команды разработчиков (https://docs.poliastro.space/en/stable/index.html). Просмотрено: 14.04.2023.
9. SPICE, An Observation Geometry System for Space Science Missions // Главная страница сайта про SPICE Toolkit (https://naif.jpl.nasa.gov/naif/index.html). Просмотрено: 14.04.2023.
10. An Overview of SPICE // Презентация библиотеки SPICE Toolkit (https://naif.jpl.nasa.gov/pub/naif/toolkit_docs/Tutorials/pdf/individual_docs/03_spice_overview.pdf). Просмотрено: 14.04.2023.
11. *Широбоков М.Г., Трофимов С.П.* Высокоуровневые средства моделирования межпланетного орбитального движения // XLV Академические чтения по космонавтике, посвященные памяти академика С.П. Королёва и других выдающихся отечественных ученых – пионеров освоения космического пространства. Сб. тез. в 4 т. 2021. Т. 1. С. 415–418.
12. *Широбоков М.Г.* KIAM Astrodynamics Toolbox 2.0 для проектирования космических миссий // Тр. 64-й Всерос. науч. конф. МФТИ. 29 ноября – 03 декабря 2021 г. Прикладная математика и информатика. 2021.
13. *Широбоков М.Г.* Высокоуровневое моделирование управляемого орбитального движения в KIAM Astrodynamics Toolbox // Тр. 63-й Всерос. науч. конф. МФТИ. 23–29 ноября 2020 года. Прикладная математика и информатика. 2020. С. 52–54.
14. KIAM Astrodynamics Toolbox // Страница на веб-сервисе GitHub (https://github.com/shmaxg/KIAMToolbox). Просмотрено: 14.04.2023.
15. MIT License // Страница на веб-сервисе GitHub (https://choosealicense.com/licenses/mit/). Просмотрено: 14.04.2023.
16. Netlib Repository at UTK and ORNL // Сайт репозитория программного обеспечения Netlib (www.netlib.org). Просмотрено: 14.04.2023.
17. Solar System Dynamics // Сайт лаборатории реактивного движения НАСА (https://ssd.jpl.nasa.gov/). Просмотрено: 14.04.2023.
18. Эфемериды EPM // Сайт лаборатории эфемеридной астрономии Института прикладной астрономии Российской академии наук (https://iaaras.ru/dept/ephemeris/epm/). Просмотрено: 14.04.2023.
19. F2PY user guide and reference manual // Сайт программы F2PY пакета numpy (https://numpy.org/doc/stable/f2py/). Просмотрено: 14.04.2023.
20. Intel oneAPI Toolkits // Сайт системы Intel oneAPI (https://www.intel.com/content/www/us/en/developer/tools/oneapi/toolkits.html). Просмотрено: 14.04.2023.
21. Продукция ОКБ Факел // Страница сайта ОКБ Факел (https://fakel-russia.com/produkciya). Просмотрено: 14.04.2023.
22. KIAM Astrodynamics Toolbox Wiki // Страница на веб-сервисе GitHub (https://github.com/shmaxg/KIAMToolbox/wiki). Просмотрено: 14.04.2023.

KIAM ASTRODYNAMICS TOOLBOX FOR SPACECRAFT ORBITAL MOTION DESIGN

M. G. Shirobokov[a], S. P. Trifimov[a]

[a] *Keldysh Institute of Applied Mathematics of Russian Academy of Sciences, Miusskaya sq. 4, Moscow, 125047, Russia*

The KIAM Astrodynamics Toolbox, a new software library for designing spacecraft orbital motion, is introduced. The toolbox is developed at the Keldysh Institute of Applied Mathematics of the Russian Academy of Sciences using Fortran and Python languages, thus combining the computational speed and the flexibility of program design. The library can be useful for space flight mechanics specialists, as well as for students in relevant educational programs.

Keywords: Software library, trajectory, spacecraft, interplanetary flight, Python, Fortran

REFERENCES

1. AGI Products // AGI company website (https://www.agi.com/products). Viewed: 14.04.2023.
2. FreeFlyer // Сайт компании «a.i. solutions» (https://ai-solutions.com/freeflyer/). Viewed: 14.04.2023.
3. GMAT // Хостинг проектов «SourceForge.net» (https://sourceforge.net/projects/gmat/files/GMAT/GMAT-R2020a/). Viewed: 14.04.2023.
4. *Ivanov D.S., Ovchinnikov M.Yu., Roldugin D.S., Tkachev S.S., Trofimov S.P., Shestakov S.A., Shirobokov M.G.* Software package for modeling orbital and angular motion satellites // Mathematical modeling. 2019. Vol. 31. No. 12. P. 44–56.
5. Fortran Astrodynamics Toolkit // Page on the GitHub web service (https://github.com/jacobwilliams/Fortran-Astrodynamics-Toolkit). Просмотрено: 14.04.2023.
6. Pykep // Page on the GitHub web service (https://esa.github.io/pykep/). Viewed: 14.04.2023.
7. Can›t install anything using conda, it hangs in solving environment // Page on the GitHub web service (https://github.com/conda/conda/issues/8051). Viewed: 14.04.2023.
8. Poliastro // Development team page (https://docs.poliastro.space/en/stable/index.html). Viewed: 14.04.2023.
9. SPICE, An Observation Geometry System for Space Science Missions // Home page of the site about SPICE Toolkit (https://naif.jpl.nasa.gov/naif/index.html). Viewed: 14.04.2023.
10. An Overview of SPICE // Presentation of the SPICE Toolkit library (https://naif.jpl.nasa.gov/pub/naif/toolkit_docs/Tutorials/pdf/individual_docs/03_spice_overview.pdf). Viewed: 14.04.2023.
11. *Shirobokov M.G., Trofimov S.P.* High-level tools for modeling interplanetary orbital motion // XLV Academic readings on astronautics, dedicated to the memory of Academician S.P. Korolev and other outstanding domestic scientists – pioneers of space exploration: collection of abstracts: in 4 volumes, 2021. Vol. 1. P. 415–418.
12. *Shirobokov M.G.* KIAM Astrodynamics Toolbox 2.0 for designing space missions // Proceedings of the 64th All-Russian Scientific Conference of MIPT. November 29 – December 03, 2021 Applied mathematics and computer science. 2021.
13. *Shirobokov M.G.* High-level modeling of controlled orbital motion in KIAM Astrodynamics Toolbox // Proceedings of the 63rd All-Russian Scientific Conference of MIPT. November 23–29, 2020. Applied mathematics and computer science. 2020. P. 52–54.
14. KIAM Astrodynamics Toolbox // Page on the GitHub web service (https://github.com/shmaxg/KIAM-Toolbox) Viewed: 14.04.2023.
15. MIT License // Страница на веб-сервисе GitHub (https://choosealicense.com/licenses/mit/) Viewed: 14.04.2023.
16. Netlib Repository at UTK and ORNL // Netlib software repository website (www.netlib.org) Viewed: 14.04.2023.
17. Solar System Dynamics // NASA Jet Propulsion Laboratory website (https://ssd.jpl.nasa.gov/) Viewed: 14.04.2023.
18. Эфемериды EPM // Website of the Laboratory of Ephemeris Astronomy of the Institute of Applied Astronomy of the Russian Academy of Sciences (https://iaaras.ru/dept/ephemeris/epm/) Viewed: 14.04.2023.
19. F2PY user guide and reference manual // Website of the F2PY program of the numpy package (https://numpy.org/doc/stable/f2py/) Viewed: 14.04.2023.
20. Intel oneAPI Toolkits // Intel oneAPI system website (https://www.intel.com/content/www/us/en/developer/tools/oneapi/toolkits.html) Viewed: 14.04.2023.
21. Продукция ОКБ Факел // OKB Fakel website page (https://fakel-russia.com/produkciya) Viewed: 14.04.2023.
22. KIAM Astrodynamics Toolbox Wiki // Page on the GitHub web service (https://github.com/shmaxg/KIAMToolbox/wiki) Viewed: 14.04.2023.

ПРОГРАММИРОВАНИЕ, *2024*, *№ 1*, *с. 66–87*

ПРОГРАММНАЯ ИНЖЕНЕРИЯ, ТЕСТИРОВАНИЕ И ВЕРИФИКАЦИЯ ПРОГРАММ

УДК 004.4

МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ С ПОМОЩЬЮ ПРОГРАММНЫХ КОМПЛЕКСОВ NUT3D, BIC3D, ЭГАК И МИМОЗА ТУРБУЛЕНТНОГО ПЕРЕМЕШИВАНИЯ В ГАЗОВЫХ СИСТЕМАХ С КОНТАКТНОЙ ГРАНИЦЕЙ В ВИДЕ ШЕВРОНА

М. Д. Брагин[a], Н. В. Змитренко[a], В. В. Змушко[b], П. А. Кучугов[a, *], Е. В. Левкина[b], К. В. Анисифоров[b], Н. В. Невмержицкий[b], А. Н. Разин[b], Е. Д. Сеньковский[b], В. П. Стаценко[b], В. Ф. Тишкин[a], Ю. В. Третьяченко[b], Ю. В. Янилкин[b]

[a]*Федеральное государственное учреждение "Федеральный исследовательский центр Институт прикладной математики им. М.В. Келдыша Российской академии наук",*
125047 Москва, Миусская пл., д. 4, Россия
[b]*Российский федеральный ядерный центр – Всероссийский научно-исследовательский институт экспериментальной физики,*
607188 Нижегородская обл., Саров, пр. Мира, 37, Россия
**E-mail: pkuchugov@gmail.com*

Поступила в редакцию: 18.07.2023
После доработки: 25.07.2023
Принята к публикации: 30.07.2023

В работе представлены расчетные и экспериментальные исследования эволюции турбулентного перемешивания в трехслойных газовых системах при развитии гидродинамических неустойчивостей Рихтмайера-Мешкова и Кельвина-Гельмгольца под действием ударных волн. Одна из контактных границ газов была плоской, вторая – с изломом в виде шеврона. Численные расчеты выполнены как без начальных возмущений контактных границ веществ, так и в присутствии возмущений. Показано, что шероховатость контактной границы существенно влияет на ширину зоны перемешивания.

Ключевые слова: турбулентное перемешивание, математическое моделирование, гидродинамические неустойчивости, ударная труба
DOI: 10.31857/S0132347424010061 **EDN:** НКМРЕМ

1. ВВЕДЕНИЕ

Гидродинамические неустойчивости Рихтмайера-Мешкова (НРМ) [1, 2], Кельвина–Гельмгольца (НКГ) [3], Рэлея–Тейлора (НРТ) [4] и вызванное ими турбулентное перемешивание веществ (ТП) могут приводить к диссипативным потерям и к нарушению симметрии схождения мишеней управляемого термоядерного синтеза (ИТС), что в итоге не позволяет получить зажигание горючего [5]. Поэтому исследования этих сложнейших течений в настоящее время интенсивно проводятся как экспериментально, так и численно во многих научных центрах. Разрабатываемые математические коды всегда требуют экспериментальной проверки.

На начальном этапе имеющиеся на контактной границе веществ малые возмущения под действием неустойчивостей начинают расти и довольно быстро выходят на насыщение и нелинейную стадию, на которой происходит взаимодействие и разрушение различных мод и формирование зоны турбулентного перемешивания. Законы развития этой зоны перемешивания, а также варианты ее взаимодействия с классическими сильными и слабыми разрывами представляют существенный интерес [6–8]. На протяжении десятилетий одним из основных вопросов остается вопрос о влиянии начальных (случайных или детерминированных) возмущений на переход к развитому перемешиванию.

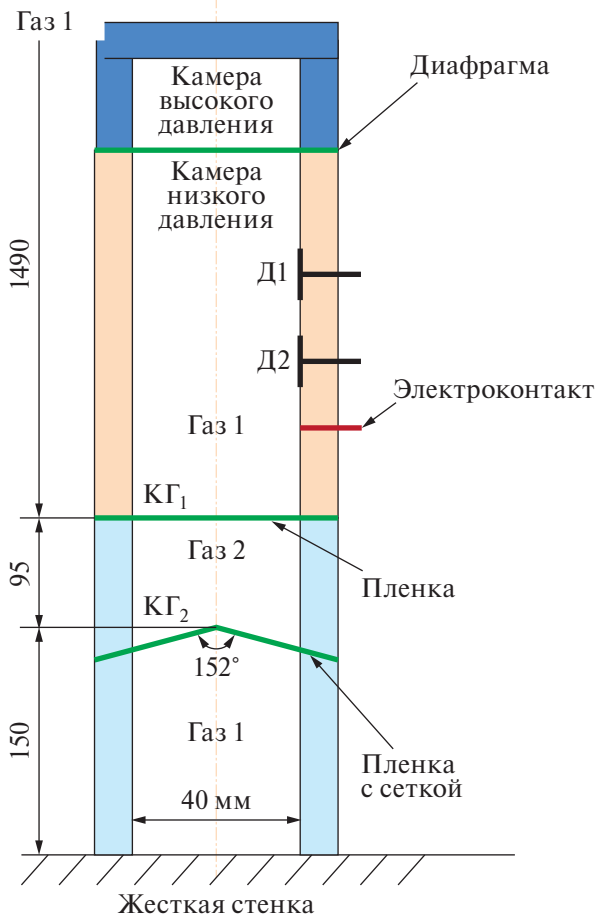
Ориентируясь на задачи ИТС, значительный интерес представляют многослойные системы, в которых возможно не только развитие НРМ, НКГ, но также взаимодействие вторичных волн с зонами перемешивания, развитие вторичных неустойчивостей и т.д. Такой комплексной задачей являются, например, трехслойные газовые системы в ударной трубе. Контактные границы между газами могут иметь различную форму, газы могут иметь различное соотношение плотностей (воздух-SF_6, воздух-He, воздух-Xe), конец трубы может быть закрыт или открыт (прошедшая ударная волна отражается от стенки и повторно взаимодействует с зонами перемешивания). Все эти вариации приводят к суще-

ственно различающимся волновым картинам и реализуют различные режимы роста начальных возмущений. Исследованию этих задач посвящено большое число работ [9–19], где экспериментально и численно изучаются различные аспекты реализующегося течения.

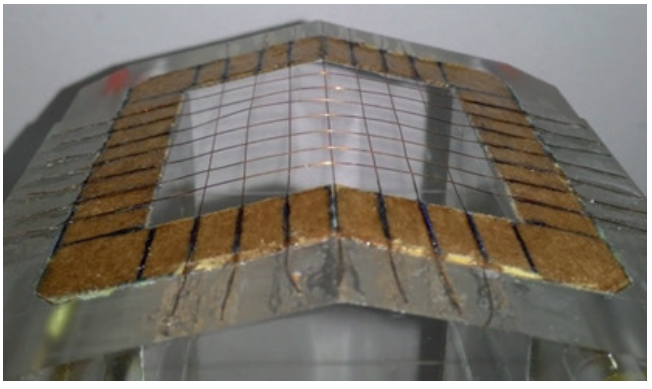
В данной работе представлены задачи этого типа, которые можно рассматривать как хорошую возможность взаимной проверки диагностических данных, получаемых в натурных экспериментах и численных кодах, использующихся для моделирования гидродинамических неустойчивостей.

2. ЭКСПЕРИМЕНТАЛЬНЫЕ РЕЗУЛЬТАТЫ

Эксперименты проводились на ударной трубе в РФЯЦ-ВНИИЭФ (рис. 1). Техника эксперимента и диагностические методики подробно описаны в предыдущих сходных работах [9, 15, 16]. Ударная труба имеет сечение внутреннего канала 40×40 мм, состоит из камеры высокого давления (драйвера), камеры низкого давления и измерительной секции, состоящей из двух оптически прозрачных отсеков, между которыми в ряде опытов устанавливалась 3D- или 2D-сетка из медной проволочки или из полихлорвиниловых нитей. Контактная граница, на которую накладывалась сетка ($КГ_2$), имела угол излома 152°. На сетку накладывалась тонкая (34 мкм) полимерная пленка. Размер ячейки 3D-сетки составлял 5 × 5 мм, толщина медных проволочек — 0,1 мм, толщина нитей ПВХ — 0,06 мм. В 2D-сетке поперечный ряд проволочек отсутствовал, расстояние между проволочками составляло 5 мм.



а)



б)

Рис. 1. Схема постановки опытов.
а) схема ударной трубы; б) фотография измерительной секции с 3D-сеткой 5×5 мм; Д1, Д2 – отметчики времени; $КГ_1$, $КГ_2$ – контактные границы

Верхний отсек измерительной секции ($КГ_1$) отделялся от камеры низкого давления трубы тонкой (3–4 мкм) плоской полимерной пленкой. Камеры высокого и низкого давления трубы разделялись между собой диафрагмой из лавсана толщиной $\sigma \approx 0{,}1$ мм. Камера высокого давления заполнялась сжатым гелием или сжатым воздухом до определенного избыточного давления. В камере низкого давления и в нижнем отсеке измерительной секции находился воздух при атмосферном давлении. Верхний отсек измерительной секции заполнялся либо гелием, либо SF_6 при атмосферных условиях. Нижний торец измерительной секции в ряде опытов был закрыт жесткой горизонтальной стенкой из оргстекла толщиной 20 мм. В вертикальных стенках ударной трубы располагались отметчики времени (пьезокерамические датчики давления Д1, Д2) для определения скорости падающей ударной волны.

Ударная труба работала следующим образом. Саморазрыв диафрагмы, отделяющей камеру высокого от камеры низкого давления, происходил, когда при заполнении драйвера сжатым газом давление превышало критическое. По камере низкого давления (по воздуху) распространялась ударная волна, которая при выходе на $КГ_1$ инициировала развитие

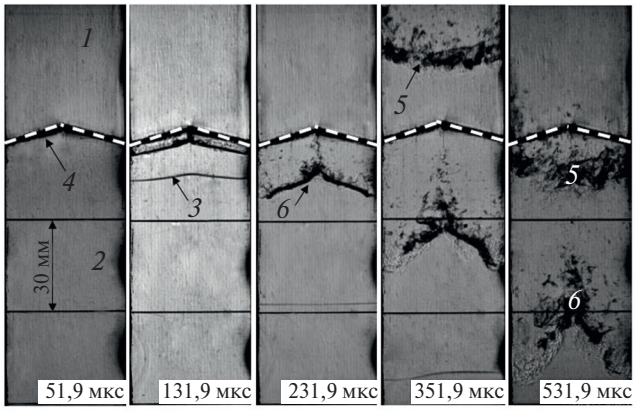
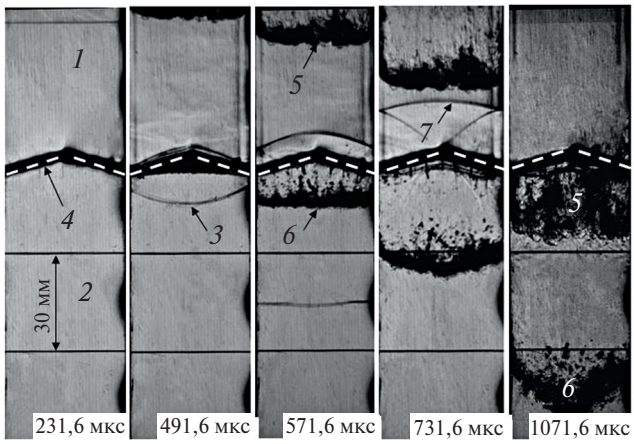
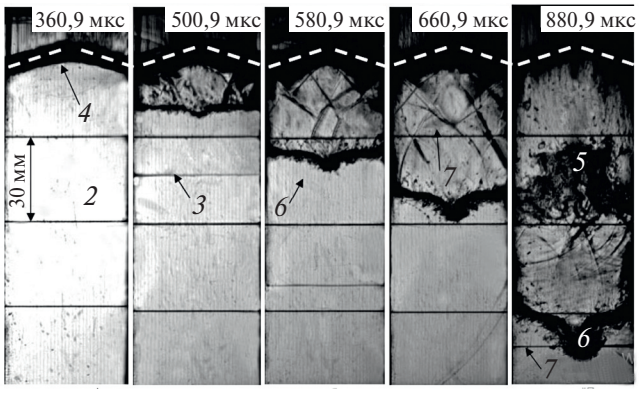


Рис. 2. Кинограмма течения в опыте № 1 со слойкой воздух-гелий-воздух (M = 1,25).

1 — гелий; *2* — воздух; *3* – ударная волна; *4* – $КГ_2$; *5* — $ЗТП_1$ на $КГ_1$; *6* — $ЗТП_2$ на $КГ_2$; время отсчитывается от прихода падающей ударной волны на $КГ_1$



а)



б)

Рис. 3. Кинограмма течения в опытах со слойкой воздух-SF_6-воздух.

а) опыт № 6 с 3D-сеткой, M = 1,2; б) опыт № 25 без сетки, M = 1,4.

1 – SF_6; *2* – воздух; *3* – ударная волна; *4* – $КГ_2$; *5* – $ЗТП_1$ на $КГ_1$; *6* – $ЗТП_2$ на $КГ_2$; *7* – отраженная от жесткой стенки волна; время отсчитывается от прихода падающей ударной волны на $КГ_1$

НРМ. Со временем ударная волна достигала $КГ_2$ и инициировала совместное развитие НРМ и НКГ (из-за излома $КГ_2$), приводящих к турбулентному перемешиванию газов. Регистрация течения проводилась скоростной видеокамерой шлирен-методом, которая запускалась от электроконтакта.

На рис. 2 и 3 представлены кинограммы некоторых экспериментов. На кинограммах M – число Маха падающей на $КГ_1$ ударной волны.

По кинограммам видно, что при проходе УВ из гелия в воздух возмущение от излома $КГ_2$ не переворачивается (рис. 2), при проходе УВ из SF_6 в воздух – переворачивается (рис. 3). Кроме этого, при отсутствии сетки на $КГ_2$ из углового возмущения растет ярко выраженный пузырь (локальное возмущение), при наличии сетки пузырь менее выражен. Это говорит о том, что интенсивно развивающаяся из-за сетки зона ТП частично поглощает локальное возмущение. Ширина зоны ТП при наличии сетки больше, чем без сетки, т.е. увеличение шероховатости КГ приводит к увеличению ширины зоны турбулентного перемешивания (ЗТП).

3. РЕЗУЛЬТАТЫ ЧИСЛЕННОГО МОДЕЛИРОВАНИЯ ПО ПРОГРАММАМ NUT3D И BIC3D

Численное моделирование опыта № 1 в ИПМ РАН было выполнено с использованием двух программных комплексов NUT3D [20-22] и BIC3D [23–26], реализующих две различные численные методики. Методики включают в себя численное решение уравнений невязкой газодинамики в форме Эйлера.

Моделирование проводится без привлечения каких-либо дополнительных моделей турбулентности для описания пульсационных составляющих характеристик течения – так называемый неявный метод крупных вихрей. Данный подход подразумевает, что используемая численная схема действует подобно явной подсеточной модели в классическом методе крупных вихрей, определяя масштаб, на котором происходит диссипация турбулентных пульсаций и пренебрегая вкладом от более мелких масштабов.

3.1. Постановка задачи

Общая длина ударной трубы составляла 1730 мм, что при масштабе амплитуды начальных возмущений порядка 0,1 мм и использовании однородной сетки привело бы к числу сеточных элементов порядка 10^9 и значительным вычислительным ресурсам. По этой причине для трехмерного моделиро-

вания был выбран только участок ударной трубы от нижнего торца до координаты по вертикали чуть выше $КГ_1$, а именно, $z = 360$ мм. В части трубы $z > 260$ мм задавались параметры течения, соответствовавшие зафиксированной в экспериментах скорости ударной волны. Кроме того, базовый размер ячеек в поперечном направлении (x, y) составлял $\Delta_1 = 0{,}4$ мм, а в вертикальном направлении (z) – $\Delta_2 = 0{,}2$ мм. В вертикальной области от 120 мм до 240 мм сетка имела уплотнение с размером ячеек $\Delta_3 = 0{,}1$ мм. Таким образом, максимальное число ячеек составляло $2{,}4 \cdot 10^7$, что позволило существенно ослабить требования к вычислительным ресурсам и выполнить большее число расчетов с различными вариантами начальных возмущений.

Параметры газов соответствовали давлению 1 атм и температуре 20 °C. Скорость ударной волны, инициированной драйвером, составляла 429 м/с.

При прохождении падающей ударной волны через пленку, разделяющую изначально газы, происходит ее разрушение на фрагменты различной величины, которые отвечают за мелкомасштабные начальные возмущения. Разрушение пленки определяется ее внутренним состоянием и в гидродинамическом расчете не моделируется, поэтому возникает необходимость задания реалистичных мелкомасштабных возмущений, имитирующих разрушение пленки. Как известно, от вида начальных возмущений достаточно сильно зависит переход к перемешиванию [27], поэтому желательно задать их наиболее приближенными к эксперименту. Из опытов приблизительно известны диапазон длин волн и амплитуда, определяемые по размерам фрагментов пленки. Следуя работам [13, 28], для имитации разрушения пленки будем задавать случайное возмущение контактных поверхностей в виде:

$$\zeta(x,y) = S \sum_{m,n=1}^{N} \begin{pmatrix} a_{mn}\cos(k_{xm}x)\cos(k_{yn}y) + \\ + b_{mn}\cos(k_{xm}x)\sin(k_{yn}y) + \\ + c_{mn}\sin(k_{xm}x)\cos(k_{yn}y) + \\ + d_{mn}\sin(k_{xm}x)\sin(k_{yn}y) \end{pmatrix}, \quad (1)$$

где Фурье-коэффициенты $a_{mn},\ldots,d_{mn}$, выбираются случайным образом на основе нормального распределения, а затем нормируются для получения желаемого среднеквадратичного отклонения амплитуды, S — нормировочный множитель. Волновые векторы определяются выражениями $k_{xm} = 2\pi m/L_x$ и $k_{yn} = 2\pi n/L_y$, L_x и L_y — поперечные размеры ударной трубы. Суммирование в (1) проводится по допустимому диапазону длин волн, которым будут соответствовать определенные пары волновых чисел m и n. Посредством варьирования спектра начальных возмущений возможно добиться лучшего согласия между численными и экспериментальными данными. Численные расчеты для выбранного опыта были выполнены как в присутствии начальных возмущений на контактных границах, так и без них.

3.2. Физико-математическая модель

В качестве основной физико-математической модели выступала модель многокомпонентной невязкой нетеплопроводной газовой динамики. Соответствующая система уравнений выглядит следующим образом:

$$\begin{cases} \dfrac{\partial \rho}{\partial t} + \dfrac{\partial}{\partial x_i}(\rho v_i) = 0 \\ \dfrac{\partial}{\partial t}(\rho C_\alpha) + \dfrac{\partial}{\partial x_i}(\rho C_\alpha v_i) = 0 \\ \dfrac{\partial}{\partial t}(\rho v_i) + \dfrac{\partial}{\partial x_i}(\rho v_i v_j + p\delta_{ij}) = 0 \\ \dfrac{\partial}{\partial t}(\rho E) + \dfrac{\partial}{\partial x_i}(\rho E v_i + p v_i) = 0, \end{cases} \quad (2)$$

где ρ — массовая плотность, v_i – компоненты скорости, C_α — массовая концентрация, p – давление, E — удельная полная энергия: $E = \varepsilon + v_k v_k/2$. По повторяющимся индексам подразумевается суммирование. Замыкающие уравнения состояния записываются в модели идеального газа: $p_\alpha = (\gamma_\alpha - 1)\rho_\alpha \varepsilon_\alpha$, где ρ_a – фазовые или парциальные плотности веществ, γ_a – показатели адиабаты, ε_a – удельная внутренняя энергия компоненты a. В предположении термодинамического равновесия различных компонент для идеальных газов можно аналитически получить выражение для связи термодинамических параметров смеси, а именно, $p = (\gamma - 1)\rho\varepsilon$, где

$$\gamma = 1 + \frac{1}{C_V}\sum_\alpha C_\alpha C_{V\alpha}(\gamma_\alpha - 1),\ C_V = \sum_\alpha C_\alpha C_{V\alpha},$$

C_V – теплоемкость вещества при постоянном объеме. Использование данной модели физически оправдано и не требует каких-либо дополнительных приближений, так как в качестве исследуемых веществ используются легкие газы. Систему (2) удобно записать в векторном виде, удобном для дальнейшего использования при описании используемых численных схем:

$$\partial_t \mathbf{U} + \partial_i \mathbf{F}^i(\mathbf{U}) = 0, \quad (3)$$

где $\mathbf{U}$ — вектор консервативных переменных, $\mathbf{F}^i(\mathbf{U})$ — соответствующие потоковые функции вдоль координатных направлений.

3.3. Численный метод

NUT3D – это трехмерный параллельный программный комплекс, ориентированный на решение задач физики высоких плотностей энергии. Помимо модуля расчета газодинамики (2), NUT3D содержит модули расчета переноса тепла в диффузионном приближении, а также работает с любыми полными термодинамически согласованными уравнениями состояния. Моделирование может вестись в одной из ортогональных систем координат: декартовой, цилиндрической или сферической.

Для получения дискретных аналогов исходных дифференциальных уравнений (2) используется метод конечных разностей. Для этого область моделирования представляется в виде численной ортогональной сетки. Таким образом, дискретная система уравнений может быть записана в следующем виде:

$$\frac{(U^{n+1}_{ijk} - U^{n}_{ijk})V_{ijk}}{\Delta t_n} + \left(F^{1}_{i+1/2jk}S^{1}_{i+1/2jk} - F^{1}_{i-1/2jk}S^{1}_{i-1/2jk}\right) + \ldots + \left(F^{3}_{ijk+\frac{1}{2}}S^{3}_{ijk+\frac{1}{2}} - F^{3}_{ijk-\frac{1}{2}}S^{3}_{ijk-\frac{1}{2}}\right) = R_{ijk}V_{ijk}, \quad (4)$$

где U – вектор консервативных переменных, F – потоки через грани ячеек, S – соответствующие площади граней ячейки, V – объем ячейки. Вычисление потоков через грани ячеек происходит как значений функций, зависящих от решения задачи о распаде произвольного разрыва. В соответствии с работой [29] для определения левых и правых значений вектора переменных на грани используется линейная интерполяция с применением ограничителей антидиффузионных потоков [30]. Решение задачи о распаде разрыва может происходить как точно, так и приближенно. Точное решение задачи о распаде разрыва возможно при использовании УРС идеального газа, а также задействует итерационные методы нахождения корня нелинейного уравнения для давления, вырабатывающегося в результате распада разрыва. По этой причине в расчетах с другими УРС, а также достаточно ресурсоемких, предпочтительнее использовать приближенные методы решения задачи о распаде разрыва. В NUT3D основным методом решения задачи о распаде разрыва является HLLC [31].

В соответствии с методом расщепления по физическим процессам на следующем этапе при необходимости численно решается уравнение теплопроводности:

$$\frac{\partial \rho\varepsilon(T)}{\partial t} + \nabla q_T = 0, \quad (5)$$

где $q_T = -\kappa(T)\nabla T$. Численное решение уравнения (5) происходит в соответствии с работой [32], в которой предложен итерационный процесс по нелинейности в коэффициенте теплопроводности. Соответствующая численная схема выглядит следующим образом:

$$\frac{\left(\varepsilon(T^{s}_{ijk}) + \frac{\partial\varepsilon}{\partial T}(T^{s}_{ijk})(T^{s+1}_{ijk} - T^{s}_{ijk}) - \varepsilon(T^{n}_{ijk})\right)\rho_{ijk}V_{ijk}}{\Delta t_n} + \left(q^{1}_{i+1/2jk}S^{1}_{i+1/2jk} - q^{1}_{i-1/2jk}S^{1}_{i-1/2jk}\right) + \ldots + \left(q^{3}_{ijk+\frac{1}{2}}S^{3}_{ijk+\frac{1}{2}} - q^{3}_{ijk-\frac{1}{2}}S^{3}_{ijk-\frac{1}{2}}\right) = 0, \quad (6)$$

где $q^{1}_{i+1/2jk} = -\kappa\left(T^{*}\right)\frac{T^{s+1}_{i+1jk} - T^{s+1}_{ijk}}{\Delta x^{1}_{ijk}}$, T^{*} на грани в соответствии с работой [33] дается выражением

$$T^{*} = \frac{\kappa\left(T^{s}_{i+1jk}\right)\Delta x^{1}_{i-1jk}T^{s}_{i+1jk} + \kappa\left(T^{s}_{i-1jk}\right)\Delta x^{1}_{i+1jk}T^{s}_{i-1jk}}{\kappa\left(T^{s}_{i+1jk}\right)\Delta x^{1}_{i-1jk} + \kappa\left(T^{s}_{i-1jk}\right)\Delta x^{1}_{i+1jk}}.$$

Остальные потоки тепла аппроксимируются аналогичным образом. Таким образом, получается система линейных алгебраических уравнений (СЛАУ) относительно вектора переменных T^{s+1}.

При расщеплении по координатным направлениям возможно решение исходной СЛАУ (6) последовательным обращением соответствующего количества трехдиагональных матриц методом прогонки. Данный подход может способствовать выделению какого-либо приоритетного направления, поэтому необходимо чередовать последовательность обработки координатных направлений. Также возможно решение исходной СЛАУ (6) каким-либо итерационным методом, например методом сопряженных градиентов с предобуславливателем.

Для вычисления термодинамических параметров в смешанных ячейках используется pT-приближение, предполагающее, что за время гидродинамического шага давление и температура различных компонент смеси успевают выровняться и достигнуть термодинамического равновесия. Также предполагается, что движение всех компонент описывается одной средней скоростью смеси. Кроме того, предполагается, что между компонентами в смешанной ячейке имеется контактная граница. В этом случае массовые и объемные концентрации вводятся выражениями: $C_\alpha = m_\alpha/m$ и $f_\alpha = V_\alpha/V$. Плотности

компонент по определению $\rho_\alpha = m_\alpha / V_\alpha = C_\alpha / f_\alpha \rho$. Таким образом, возникает следующая система уравнений для нахождения равновесного давления, температуры и плотностей компонент (или объемных концентраций) по известным внутренней энергии и плотности смеси после решения задачи о распаде разрыва:

$$\begin{cases} \varepsilon = \sum C_\alpha \varepsilon_\alpha(\rho_\alpha, T); \\ \dfrac{1}{\rho} = \sum \dfrac{C_\alpha}{\rho_\alpha}; \\ p = p_\alpha(\rho_\alpha, T). \end{cases} \quad (7)$$

Данная система содержит $N_\alpha + 2$ уравнений, где N_α — число компонент в ячейке, при необходимости число уравнений может быть уменьшено до N_α.

Для решения системы (7) используется комбинация метода деления отрезка пополам и метода Ньютона. В связи с тем, что решение данной системы зачастую лежит близко к асимптотам, то использование метода деления отрезка пополам является обязательным условием для поиска начального приближения для метода Ньютона. В результате в большинстве случаев выполняется два и более (в зависимости от числа компонент) циклов итераций, что при большом числе смешанных ячеек негативно сказывается на общей производительности программы.

Отметим, что если в качестве независимых термодинамических переменных выбрать p и T вместо ρ и T, тогда система, аналогичная (7), содержала бы всего 2 уравнения, что является более приемлемым вариантом с вычислительной точки зрения.

Возможно использование другого подхода к описанию состояния в смешанных ячейках, при котором каждая компонента смеси занимает весь объем ячейки (кинетический подход). В этом случае система уравнений для определения равновесного давления и температуры выглядит следующим образом:

$$\begin{cases} \varepsilon = \sum C_\alpha \varepsilon_\alpha(\rho_\alpha, T); \\ \rho_\alpha = C_\alpha \rho; \\ p = \sum p_\alpha(\rho_\alpha, T). \end{cases} \quad (8)$$

При известных ρ, ε и C_α данная система требует решения только уравнения для нахождения равновесной температуры.

BIC3D – это параллельный программный комплекс для численного решения трехмерных эволюционных гиперболических систем законов сохранения по бикомпактным схемам из [23–26].

Предполагается, что расчетная область имеет форму параллелепипеда. Пространственная бикомпактная аппроксимация четвертого порядка строится на декартовой сетке, дополненной вспомогательными узлами с полуцелыми индексами:

$$\Omega = \Omega_1^x \times \Omega_1^y \times \Omega_1^z = \{(x_j, y_k, z_l)\};$$

$$\Omega_1^d = \left\{ d_0 < d_{1/2} < d_1 < d_{3/2} < d_2 < \ldots < d_{N_d} \right\};$$

$$h_{d,i} = d_{i+1} - d_i \text{ — шаг по } d,\ i = 0, 1, \ldots, N_d - 1. \quad (9)$$

На временной оси вводятся слои t_n, $n = 0, 1, \ldots$; $t = \tau_n = t_{n+1} - t_n$ — шаг по t. Численное решение представляется в виде сеточной функции $\mathbf{U}^n_{i,j,k}$, значения которой аппроксимируют значения точного решения $\mathbf{U}(x,y,z,t)$ в узлах сетки Ω в моменты времени $t = t_n$.

В бикомпактных схемах из [23, 24] используется локально-одномерное расщепление по пространству. Помимо него, осуществляется потоковое расщепление Лакса–Фридрихса–Русанова (ЛФР). Совместное применение этих расщеплений означает, что при каждом переходе со слоя на слой трехмерная система (2) разбивается на шесть одномерных подсистем вида

$$\partial_t \mathbf{U} + \partial_d \mathbf{F}^{d,\pm}(\mathbf{U}) = 0, \quad (10)$$

где $\mathbf{F}^{d,\pm}(\mathbf{U}) = 0.5\mathbf{F}^d(\mathbf{U}) \pm C_2^d \mathbf{U}$ (формула для C_2^d будет приведена позднее). Одномерные подсистемы (2) решаются при помощи одномерных бикомпактных схем вдоль линий сетки – прямых, проходящих через узлы сетки Ω параллельно координатным осям. Обозначим оператор послойного перехода всякой одномерной бикомпактной схемы для (2) как $S_B^{d,\pm}(\tau)$. Введем операторы $S_B^d(\tau) = S_B^{d,-}(\tau) S_B^{d,+}(\tau)$. Результирующая локально-одномерная бикомпактная схема для исходной системы (2) имеет оператор послойного перехода

$$S_B(\tau) = S_B^x\left(\frac{\tau}{2}\right) S_B^y\left(\frac{\tau}{2}\right) S_B^z(\tau) S_B^y\left(\frac{\tau}{2}\right) S_B^x\left(\frac{\tau}{2}\right). \quad (11)$$

Заметим, что точность пространственного расщепления повышена по методу Марчука–Стрэнга.

Аппроксимация по времени в бикомпактных схемах строится при помощи чисто неявных и неявно-явных методов Рунге–Кутты (РК). В BIC3D реализованы методы РК обоих типов от первого до четвертого порядков включительно. Временные дискретизации четных порядков лучше подходят для счета гладких решений, нечетных – для счета разрывных решений. Неявно-явные бикомпактные схемы экономичнее, так как в них исключены ньютоновские итерации; чисто неявные бикомпактные схемы точнее и генерируют меньшие осцилляции у фронтов сильных разрывов.

В расчетах, результаты которых представлены в следующем разделе, операторы $S_B^{d,\pm}(\tau)$ отвечали неявно-явной схеме BiC4-IMEX3 из [24]. Эта схема имеет четвертый и третий порядки аппроксимации по пространству и времени соответственно.

Для подавления эффекта Гиббса в бикомпактных схемах повышенного порядка аппроксимации по времени применяется метод консервативной монотонизации [25]. Он имеет один настраиваемый параметр $C_1 \geqslant 0$. При $C_1 = 0$ монотонизация отсутствует, при $C_1 \to \infty$ она максимальна (от численного решения в каждой ячейке остается лишь интегральное среднее). Метод [25] основан на сравнении решения высокоточной схемы (B, в данном случае бикомпактной) с решением монотонной схемы-партнера (A). В BIC3D в качестве схемы A выбран и реализован локально-одномерный "явный уголок" с расщеплением ЛФР. Оператор послойного перехода этой схемы имеет вид:

$$S_A(\tau) = S_A^z(\tau) S_A^y(\tau) S_A^x(\tau). \qquad (12)$$

Здесь $S_A^d(\tau) = S_A^{d,-}(\tau) S_A^{d,+}(\tau)$, где $S_A^{d,\pm}(\tau)$ – операторы послойного перехода одномерного "явного уголка" для (2).

Коэффициенты C_2^d и шаг τ задаются перед каждым переходом со слоя на слой по формулам

$$C_2^d = (0.5 + \delta) V_{\max}^d;$$
$$\tau = \frac{\kappa}{1+\delta} \min_d \frac{\min_i h_{d,i}}{V_{\max}^d}; \qquad (13)$$
$$V_{\max}^d = \max_{\Omega} V^d(\mathbf{U}_{j,k,l}^n),$$

где $V^d(\mathbf{U})$ – мажоранты для модулей собственных значений матриц $\mathbf{A}^d(\mathbf{U}) = \partial_{\mathbf{U}} \mathbf{F}^d(\mathbf{U})$, $\delta > 0$ – коэффициент запаса, $\kappa > 0$ – число Куранта.

3.4. Программная реализация

BIC3D реализован на языке C++ в рамках технологии MPI и предназначен для использования на многопроцессорных вычислительных системах. В соответствии со структурой численного подхода (см. предыдущий раздел), а именно, использование локально одномерного расщепления по координатным направлениям, в качестве идеологии распараллеливания используется геометрический параллелизм. В этом случае в пределах одной стадии такого расщепления вычисления на параллельных линиях сетки не связаны между собой, и потому могут осуществляться одновременно. Метод консервативной монотонизации из [25] также легко распараллеливается: для построения монотонизированного решения в каждом узле нужны данные лишь от ячеек, которым он принадлежит.

Пусть число процессов равно n_p. Разобьем всю совокупность ячеек сетки на n_p горизонтальных слоев равной толщины $M_z = N_z/n_p$ (в ячейках) и n_p вертикальных слоев равной толщины $M_x = N_x/n_p$. Ради простоты записи мы полагаем, что N_z, N_x делятся нацело на n_p. Процесс ранга $r = \overline{0, n_p - 1}$ имеет дело не со всей сеткой Ω, а лишь с ее подмножествами

$$\omega_r = \Omega_1^x \times \Omega_1^y \times \{z_l : 0 \leq l - rM_z \leq M_z\};$$
$$\omega_r' = \{x_j : 0 \leq j - rM_x \leq M_x\} \times \Omega_1^y \times \Omega_1^z. \qquad (14)$$

Соответственно, процесс ранга r оперирует двумя сеточными функциями (массивами) решения, определенными на сетках ω_r, ω_r'.

Опишем алгоритм перехода со слоя t_n на слой t_{n+1} с точки зрения процесса ранга r.

1. Найти $\max_{\omega_r} V^d(\mathbf{U}_{j,k,l}^n)$. Если $r \neq 0$, то отправить эти максимумы процессу ранга 0 и в ответ получить величины $V_{\max}^d$. Если $r = 0$, то собрать данные от остальных процессов, вычислить величины $V_{\max}^d$ и разослать их остальным процессам. Вычислить C_2^d и τ по формулам (4).

2. Рассчитать на ω_r промежуточное решение

$$\mathbf{Y}^B = S_B^y\left(\frac{\tau}{2}\right) S_B^x\left(\frac{\tau}{2}\right) \mathbf{U}^n.$$

Действие, например, $S_B^{x,+}(\tau/2)$ на $\mathbf{U}^n$ реализуется так: с помощью одномерной бикомпактной схемы для всех $0 \leq k \leq N_y$, $0 \leq l - rM_z \leq M_z$ решить подсистему (2) с индексами (x, +) вдоль прямой $y = y_k$, $z = z_l$ при половинном временном шаге, используя в качестве начальных данных $\mathbf{U}^n$.

3. Разбить массив решения $\mathbf{Y}^B$ на ω^r сообразно разбиению множества ячеек на вертикальные слои; полученные части этого массива разослать по соответствующим процессам (рис. 1); собрать данные, присланные другими процессами. Как результат, у данного процесса в распоряжении окажется все то же решение $\mathbf{Y}^B$, но теперь уже на ω_r'.

4. Рассчитать на ω_r' промежуточное решение $\tilde{\mathbf{Y}}^B = S_B^z(\tau)\mathbf{Y}^B$.

5. Выполнить операцию, обратную (симметричную по x, z) операции из шага 3. У данного процесса получится решение $\tilde{\mathbf{Y}}^B$ на ω^r.

6. Рассчитать на ω^r финальное решение схемы B на слое t_{n+1}: $\mathbf{U}^B = S_B^x\left(\frac{\tau}{2}\right) S_B^y\left(\frac{\tau}{2}\right) \tilde{\mathbf{Y}}^B$.

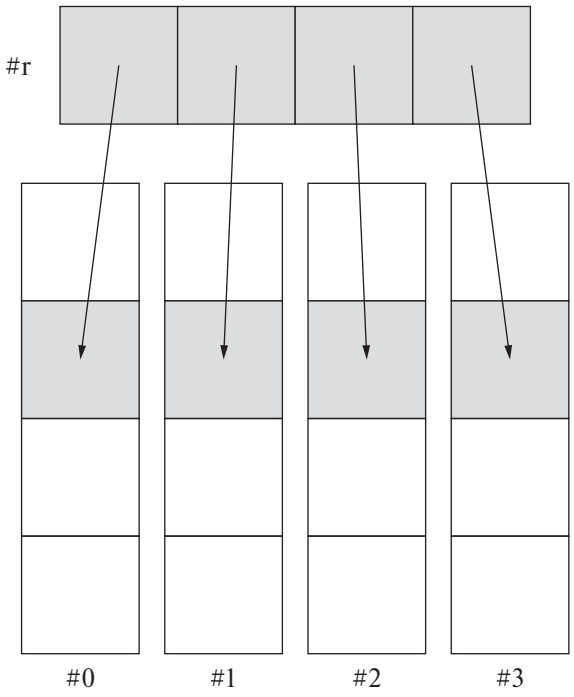


Рис. 4. Рассылка решения с горизонтального слоя ($n_p=4$, $r=2$).

7. Получить на ω^r вспомогательное решение $\mathbf{U}^A$, повторяя шаги 2–5 для схемы A, учитывая при этом формулу (3) для ее оператора послойного перехода.

8. Провести монотонизацию решения $\mathbf{U}^B$ при помощи решения $\mathbf{U}^A$ по методу из [22]. При исполнении этой операции необходимо обменяться с процессами $r \pm 1$ данными с плоскостей $l=rM_z$, $l=(r+1)M_z$.

9. Искомое решение $\mathbf{U}^{n+1}$ на ω^r найдено, переход на слой t_{n+1} завершен.

Барьерные синхронизации процессов выполняются в начале каждого перехода со слоя на слой и перед операциями обменов. Ясно, что при $C_1 = 0$ из алгоритма исключаются шаги 7, 8.

В работе [26] было проведено исследование BIC3D на эффективность распараллеливания: она составляет от 70% при условии $n_p \le (\min_d N_d)/2$.

В завершение дадим практические рекомендации по работе с BIC3D. При использовании комплекса пользователь задает:

- Физическую модель – функции $\mathbf{F}^d(\mathbf{U})$ и $V^d(\mathbf{U})$. Решение задачи Римана (Riemann solver) специфицировать не нужно из-за глобального расщепления ЛФР.
- Пространственную сетку Ω через задание сеток Ω_1^d (как массивов узлов).
- Начальные условия в виде функции для вычисления начальных значений либо уже готового файла данных, содержащего эти значения (очевидно, размеры файла должны соответствовать сетке Ω).
- Типы условий на всех границах расчетной области. Поддерживаются: условия первого рода (для них необходимо дополнительно задать функции для вычисления граничных значений), условия отражения, условия свободного выхода и периодические условия.
- Параметры δ, κ, C_1. Рекомендуемые значения: $C_1=0$ и любое κ при счете гладких решений, $C_1=10$, $\kappa \geqslant 0.6$ при счете разрывных решений (диссипация в бикомпактных схемах нечетного порядка растет с увеличением κ), $\delta=0.5$. При $\kappa>1$ шаг по времени в схеме A автоматически дробится так, чтобы она сохраняла устойчивость.

Численный код NUT3D в последовательном варианте реализован на языке C. Параллельная версия, использовавшаяся для моделирования в данной работе, была получена с помощью набора инструментальных средств, созданных в ИПМ им. М.В. Келдыша РАН и упрощающих процесс распараллеливания. Применение этих инструментов состоит из нескольких этапов. На первом этапе программа на языке Fortran или C подается на вход системе автоматизированного распараллеливания (далее системе SAPFOR [34, 35]). Блок анализа системы переводит программу во внутреннее представление. Программа анализируется с помощью различных алгоритмов статического анализа, в результате которого выявляются особенности программы, влияющие на ее распараллеливание. Затем на втором этапе происходит подбор подходящих схем распараллеливания программы. Отбор подходящих схем распараллеливания осуществляется на основе прогнозируемых характеристик эффективности выполнения параллельной программы. Система может предлагать пользователю на выбор разные схемы распараллеливания, сопровождая их прогнозируемыми характеристиками эффективности и причинами отказа от распараллеливания фрагментов программы. Пользователь определяет значимые фрагменты программы и исследует причины отказа от их распараллеливания. Возможно уточнить результаты автоматического анализа (например, используя дополнительные спецификации) и повторить второй шаг этап, и/или преобразовать текст последовательной программы и начать с первого. При успешном выполнении этого этапа система по отобранным пользователем схемам распараллеливания и внутреннему представлению программы генерирует текст

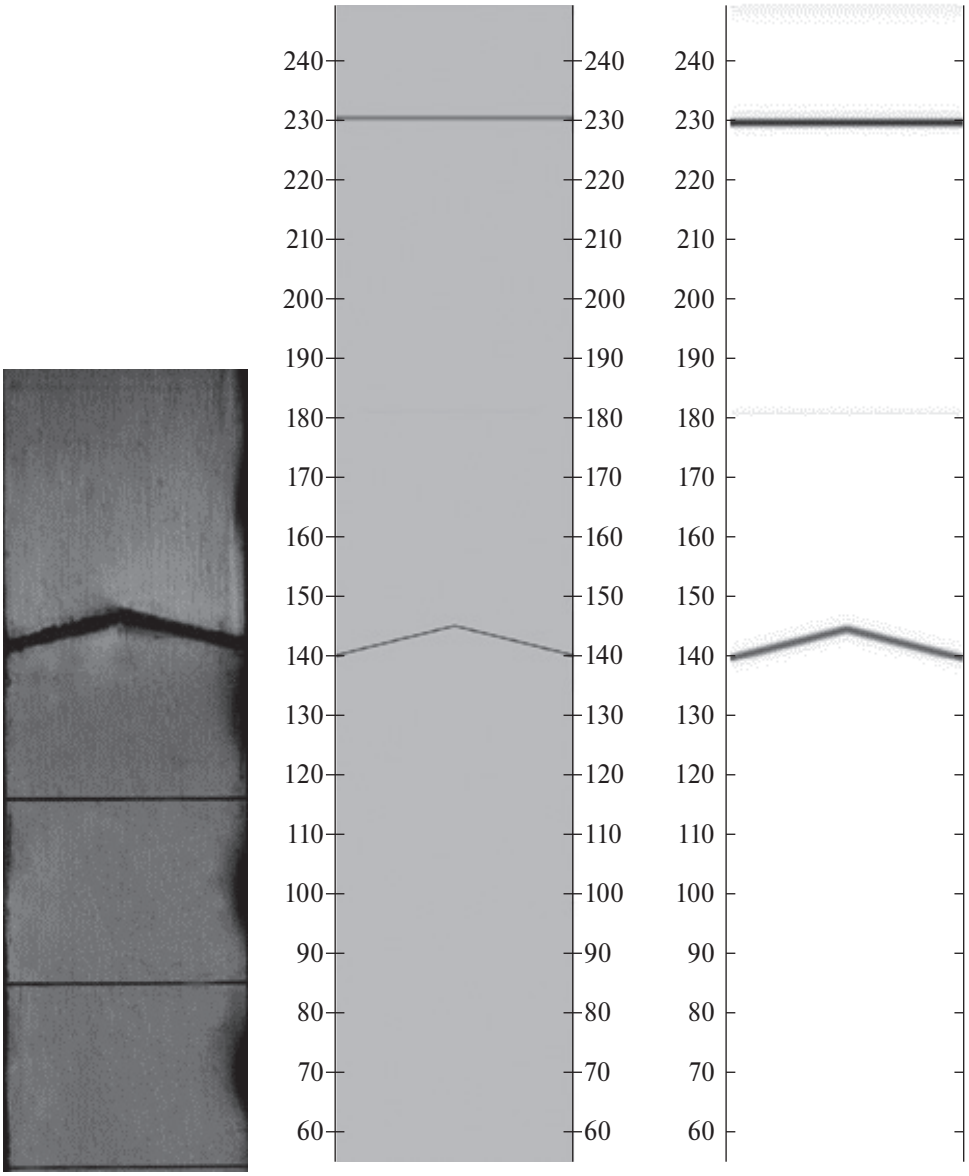


Рис. 5 (начало).

параллельной программы в модели DVMH на языке Fortran-DVMH или C-DVMH.

Модель программирования DVMH [36] позволяет разрабатывать параллельные программы для кластеров, в узлах которых помимо универсальных многоядерных процессоров установлены ускорители и сопроцессоры.

Модель параллелизма базируется на специальной форме параллелизма по данным: одна программа-множество потоков данных. В этой модели одна и та же программа выполняется на всех процессорах, но каждый процессор выполняет свое подмножество операторов в соответствии с распределением данных.

Для отображения DVMH-программы на параллельную ЭВМ используются специальные компиляторы, которые входят в состав DVM-системы.

DVMH-компилятор преобразует исходную программу в параллельную программу на языке Fortran или C с вызовами функций системы поддержки параллельного выполнения DVMH-программ (библиотеки Lib-DVMH). Система поддержки реализована на основе стандартных технологий параллельного программирования MPI (для взаимодействия между узлами кластера), OpenMP (для распределения вычислений между ядрами узла), CUDA (для распределения вычислений и отображения данных на графические ускорители).

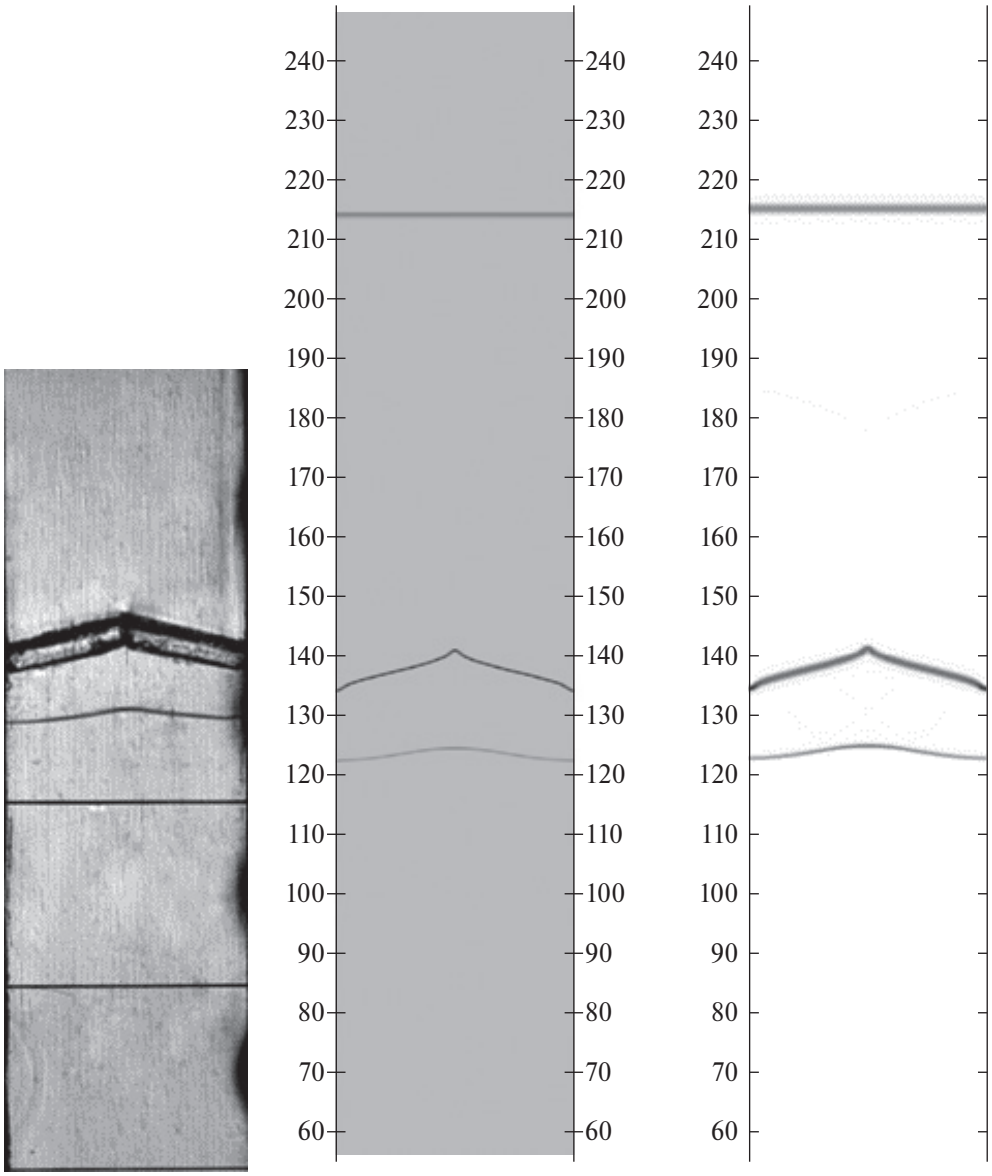


Рис. 5 (продолжение).

3.5. Результаты расчетов и сравнение с экспериментом

На первом этапе были выполнены расчеты без начальных возмущений контактных границ. Целью этих расчетов было сопоставление динамики различных фронтов и контактных поверхностей. Ниже на рис. 5 приведены численные шлирен-изображения течения в различные моменты времени. В данном случае под шлирен-изображением понимается распределение модуля градиента плотности без изменения масштаба вариации данной величины.

В приведенной на рис. 5 картине течения ($t = 0{,}052$ мс) на экспериментальной фотографии в кадре нет верхней (плоской) контактной поверхности. При этом видна ударная волна, прошедшая в гелий, положение фронта которой совпадет и в эксперименте, и в численных расчетах. В следующий приведенный момент времени ($t = 0{,}132$ мс) падающая ударная волна уже прошла вторую контактную границу и вышла из гелия, изменив форму фронта. Здесь уже наблюдаются различия в координате фронта этой ударной волны – в численных расчетах она распространяется быстрее. Далее ($t = 0{,}232$ мс) данная тенденция сохраняется. На численных шлирен-изображениях наблюдается ударная волна, отраженная от $КГ_2$ и прошедшая через $КГ_1$ ($z = 224$ мм). В $t = 0{,}352$ мс в кадре появляется

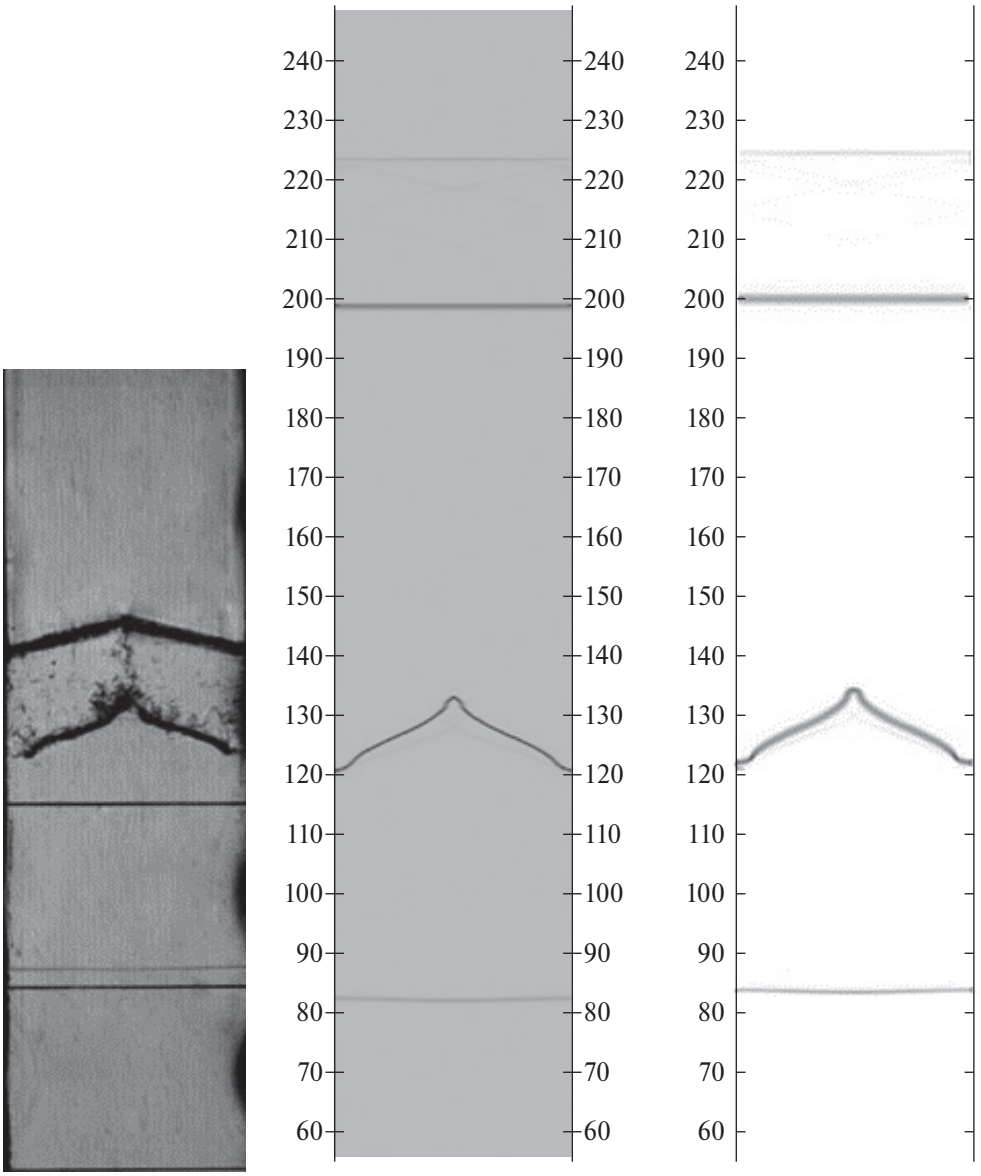


Рис. 5 (продолжение).

возмущенная $КГ_1$, однако расчетное ее положение существенно отличается от экспериментального, отставая от него. Также отставание начинает наблюдаться и для $КГ_2$, движение которой в более ранние моменты времени совпадало с экспериментальным. Финальный момент времени $t = 0{,}532$ мс лишь подтверждает наметившуюся тенденцию. На основании рис. 5 можно отметить следующие обстоятельства. Во-первых, расчеты, выполненные с использованием двух различных методик (NUT3D и BIC3D), хорошо согласуются между собой. Во-вторых, в эксперименте скорость движения контактных границ выше, чем в расчетах. Это может быть связано с неточными данными по положению пленки, разделяющей воздух и гелий в измерительной секции трубы, и скорости падающей ударной волны. Кроме того, в расчетах не учитываются особенности взаимодействия падающей ударной волны с пленкой, которые могут приводить к ее выгибанию и смещению перед разрушением.

В следующей серии расчетов на обеих контактных границах были заданы случайные начальные возмущения. Длина волны возмущений задавалась в интервале от $\lambda_{min} = 1$ до $\lambda_{max} = 3$ мм, среднеквадратичное отклонение составляло $0{,}1\lambda_{min}$. На рис. 6 приведены шлирен-изображения, полученные в расчетах с использованием NUT3D и BIC3D.

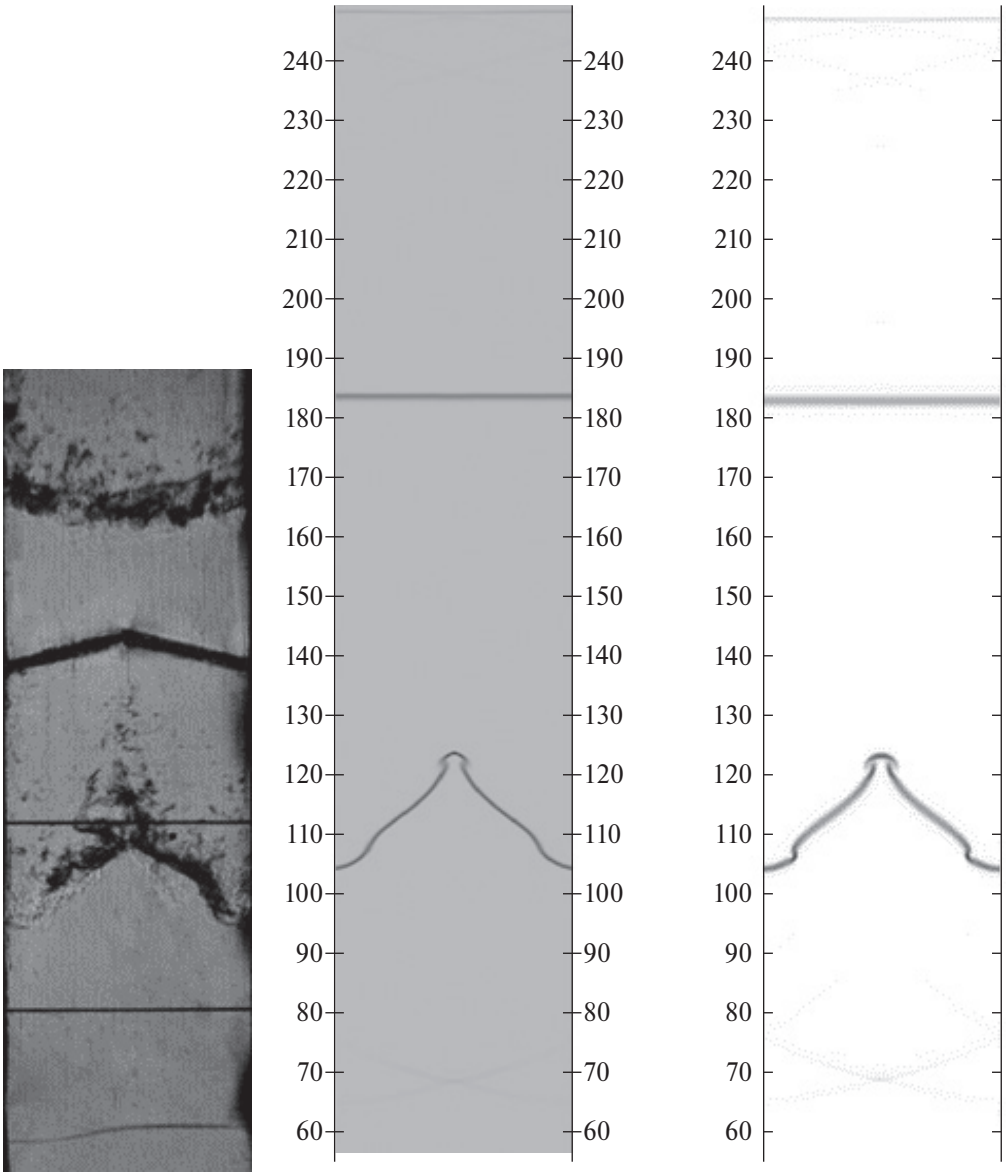


Рис. 5 (продолжение).

На основании рис. 6 можно отметить следующее. Наличие возмущений на контактных границах незначительно влияет на скорость их движения – положения в различные моменты времени идентичны расчетам без возмущений (рис. 5). Характер роста возмущений на $КГ_1$ и $КГ_2$ отличается, во-первых, в связи с дополнительным развитием НКГ на $КГ_2$, а во-вторых, вследствие инверсии фаз начальных возмущений на $КГ_1$ при прохождении исходной падающей ударной волны. Сформировавшиеся в результате развития начальных возмущений зоны перемешивания не являются идентичными. На $КГ_2$ можно отчетливо рассмотреть вихреобразные структуры, а именно дорожку вихрей, характерную для развития неустойчивости Кельвина-Гельмгольца. Также можно отметить, что развитие возмущений на этой контактной границе происходит медленнее, чем на $КГ_1$.

4. ВЕРИФИКАЦИЯ ЧИСЛЕННЫХ КОДОВ ЭГАК И k-ε МОДЕЛИ

В данном разделе представлены результаты проверки численных кодов ЭГАК и k-ε модели на эксперименте № 6.

Методика ЭГАК является лагранжево-эйлеровой, в которой аппроксимация уравнений многокомпонентной газодинамики производится в два этапа

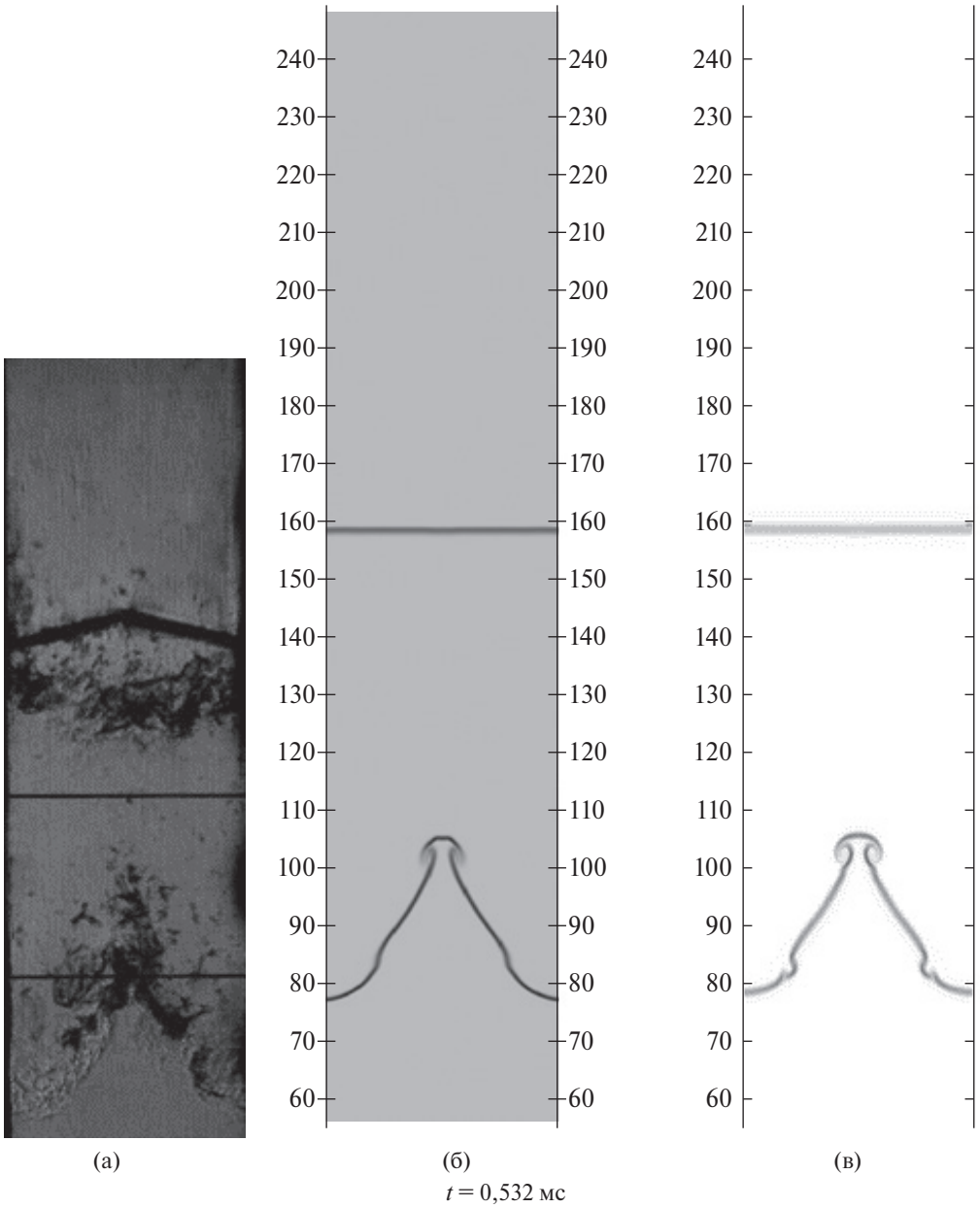


Рис. 5 (окончание). Экспериментальные фотографии (а) и численные шлирен-изображения (б) NUT3D, (в) BIC3D, в различные моменты времени.

на неподвижной прямоугольной, как правило, квадратной счетной сетке [37].

На лагранжевом этапе вычислений используется схема типа “крест”. Все компоненты среды выделяются своими термодинамическими параметрами и объемной долей. Для замыкания уравнений газодинамики в смешанных ячейках, содержащих два и более компонента, используются несколько методов, основанные на разных предположениях относительно термодинамического состояния компонентов [38].

На эйлеровом этапе в чистых (однокомпонентных) ячейках для аппроксимации уравнения адвекции используется метод PPM [39] третьего порядка аппроксимации. Для определения потоков из смешанных ячеек используется метод концентраций (VoF) [40].

В методике разработан специальный (неподвижный несжимаемый) компонент, граница которого проходит по линиям сетки, и на которой реализовано условие идеального скольжения.

Для моделирования турбулентного перемешивания в методике используется как k-ε модель, так и прямое моделирование на подробной сетке без использования каких-либо моделей турбулентности.

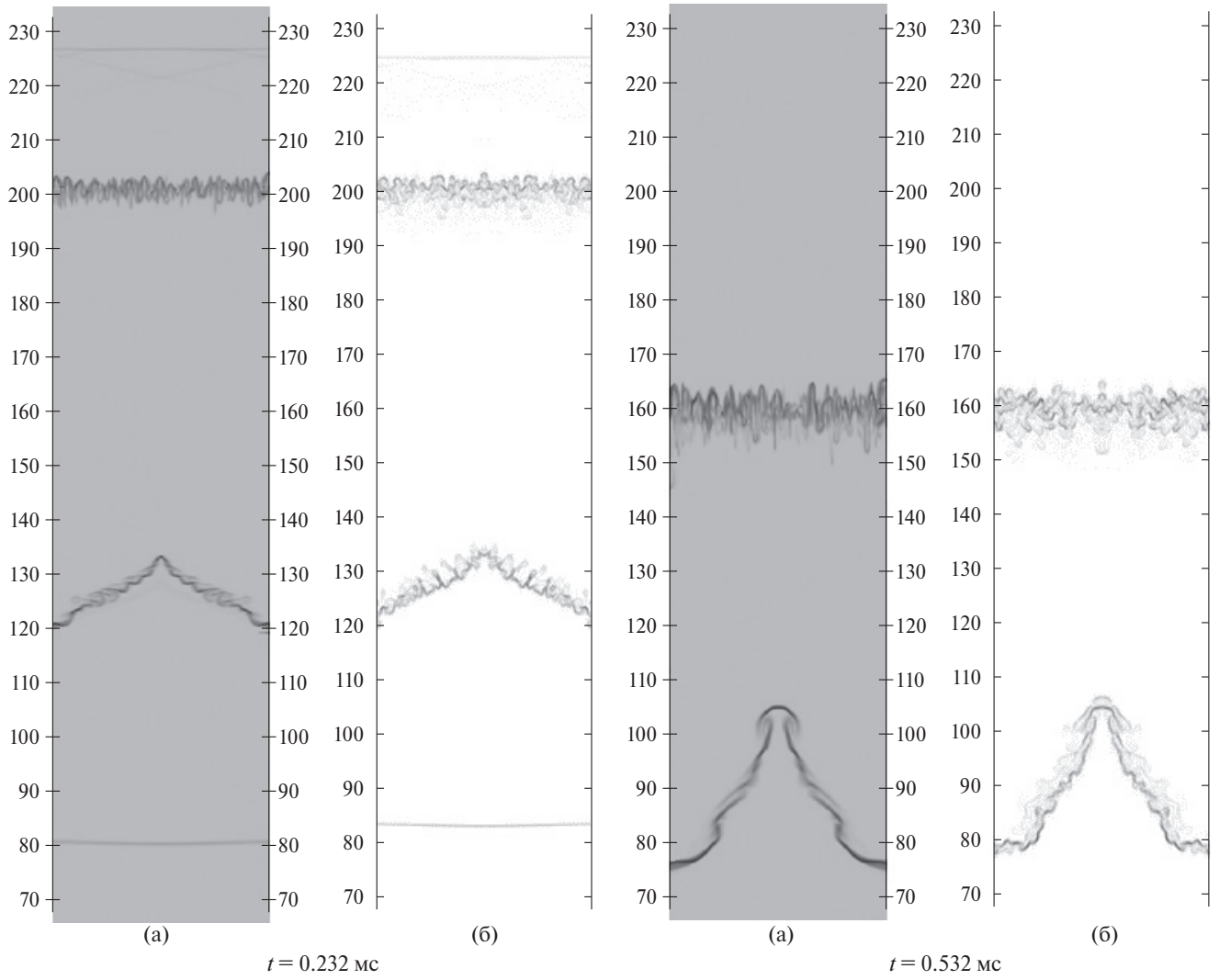


Рис. 6. Численные шлирен-изображения в различные моменты времени: (а) NUT3D, (б) BIC3D.

В настоящей работе использовался подход, основанный на решении уравнений невязкой газодинамики в форме Эйлера.

4.1. Математическая постановка задачи

Начальная геометрия эксперимента взята из рис. 1. На $КГ_2$ в эксперименте была задана двухмерная или трехмерная сетка из медной или лавсановой проволоки с $\lambda = 0{,}5$ см. В расчетах эта сетка была неподвижной и моделировалась с использованием несжимаемого компонента. При этом в расчетах диаметр проволоки задавался равным $d_{\text{расчет}} = d + 2s$, где d — реальный диаметр проволоки, s — толщина натянутой на сетку пленки. Это сделано исходя из предположения, что пленка после разрыва в процессе течения прилипает к проволоке и ее фактическая толщина увеличивается. В двух расчетах сама пленка не задавалась, в третьем расчете она задавалась в виде разорванных кусков толщиной 0,005 см и шириной 0,03 см, расстояние между кусочками составляло 0,01 см.

В расчетах используется следующая система единиц: г, см, мс $= 10^{-3}$ с, T, К, давление в атм. На $КГ_1$ задавалось условие свободного протекания. На боковых стенках задавалось условие “жесткая стенка”. $КГ_1$ была возмущена в виде синусоиды с длиной волны $\lambda = 0{,}03$ см и с амплитудой возмущений $a = 0{,}01$ см.

Уравнение состояния газов:

$$\varepsilon = c_V \cdot T = \frac{p}{(\gamma - 1)\rho},$$

здесь γ — постоянная адиабаты. В расчетах с пленкой для пленки использовался УРС Ми—Грюнайзена с $\rho_0 = 0{,}92$, $c_0 = 2{,}65$, $n = 4{,}5$, $\gamma = 2{,}35$, $c_V = 1$, $p_0 = 1$.

Начальные данные задачи зависели от использованных в опытах газов. Отметим, что в области

“газ 1” (воздух) задавался соответствующий газ с параметрами за фронтом ударной волны, которые определялись следующим образом.

Начальное давление в области “газ 1”

$$p_1 = \frac{2 \cdot \rho_0 \cdot u_{\text{ув}}^2 - (\gamma_1 - 1) \cdot p_0}{(\gamma_1 + 1)}. \qquad (15)$$

Здесь скорость ударной волны $u_{\text{ув}}$ берется из эксперимента.

Далее находим начальную плотность в “газ 1”

$$\rho_1 = \rho_0 \cdot \frac{(\gamma_1 + 1) \cdot p_1 + (\gamma_1 - 1) \cdot p_0}{(\gamma_1 + 1) \cdot p_0 + (\gamma_1 - 1) \cdot p_1}. \qquad (16)$$

И наконец, находим начальную скорость в “газ 1”

$$u_z = -\sqrt{(p_1 - p_0) \cdot \left(\frac{1}{\rho_0} - \frac{1}{\rho_1}\right)}. \qquad (17)$$

Начальные данные опыта № 6: в области “газ 1”: $\rho_{10} = 0{,}0017$, $u_{z0} = 21{,}62$, $p_{10} = 1{,}5844$, $\gamma_1 = 1{,}4$; в области “газ 2”: $\rho_{SF6\,0} = 6{,}5 \cdot 10^{-3}$, $u_0 = 0$, $p_{20} = 1$, $\gamma_2 = 1{,}0945$; в области ниже $КГ_2$ $\rho_0 = 1{,}25 \cdot 10^{-3}$, $u_0 = 0$, $p_0 = 1$, $\gamma_1 = 1{,}4$.

Проведены три расчета на счетной сетке с $h = 0{,}005$: расчет 1 – без пленки и k-ε модели; расчет 2 – без k-ε модели с кусками пленки, заданными как указано выше; расчет 3 – без пленки с k-ε моделью. В расчете с k-ε моделью на $КГ_2$ задавались начальные затравочные значения турбулентной энергии и скорости ее диссипации: $k = 10^{-4}$, $\varepsilon = 10^{-5}$.

Численное моделирование задачи производилось в 2D-постановке по методике ЭГАК [37].

4.2. Результаты расчетов

Ниже приводится сравнение с экспериментами (с двумерной 2D- и с трехмерной 3D-сеткой) указанных вариантов двумерных расчетов. Эксперименты представлены шлирен-фотографиями, расчеты – распределением градиента плотности, по сути, это расчетная шлирен-картина (рис. 7–10).

Отметим некоторые различия, имеющиеся в экспериментальных данных. Имеется некоторое опережение движения волн в эксперименте с 2D-сеткой. На рис. 7 видно, что ударная волна пришла на сетку на более раннее время по сравнению с 3D-сеткой. Такое небольшое отличие может быть следствием изменения сетки или же это вследствие экспериментальной погрешности в определении скорости ударной волны, движущейся по ударной трубе. Как бы там ни было, в этой ситуации от расчетов правомерно ожидать качественного согласия с экспериментальными данными, не претендуя на большую точность по времени. Кроме того, имеются отличия в интенсивности перемешивания и скорости распространения ЗТП вдоль трубы, что хорошо видно на рис. 10.

Отметим, что на всех расчетных рисунках приводятся распределения градиентов плотности в диапазоне от 0 до 0,01, соответственно этому цвет меняется от синего до красного. Таким образом, красные оттенки в расчетах дают представление о процессах перемешивания, в частности о зоне турбулентного перемешивания.

Из сравнения расчетных и экспериментальных распределений можно сделать следующие выводы. В целом результаты расчетов согласуются между собой и с экспериментальными данными, при этом расчеты лучше согласуются с экспериментом с 2D-сеткой, что не удивительно, так как они моделируют именно этот опыт (расчеты двумерные). Наличие проволочной сетки в расчете 1 чувствуется до конца моделирования, в расчетах хорошо видны струйки с большими градиентами, совпадающими с положением сетки. Менее ярко этот эффект проявляется в расчете 2, наличие фрагментов пленки хаотизирует

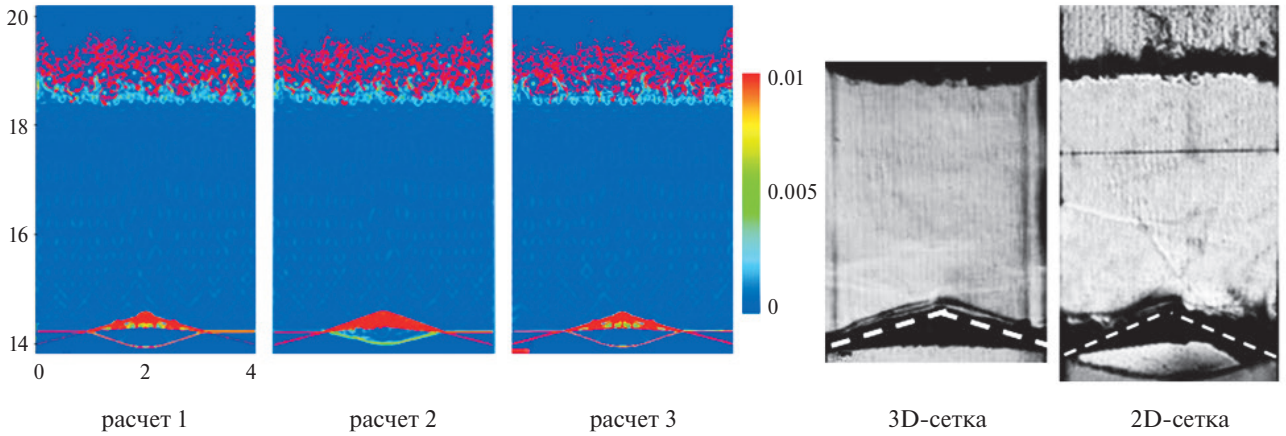
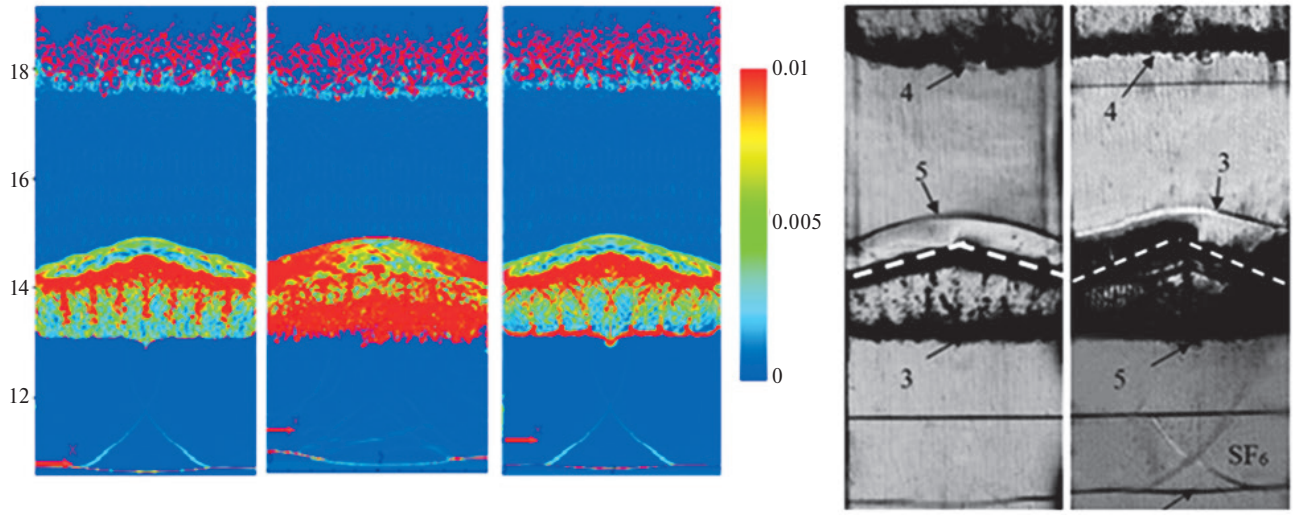
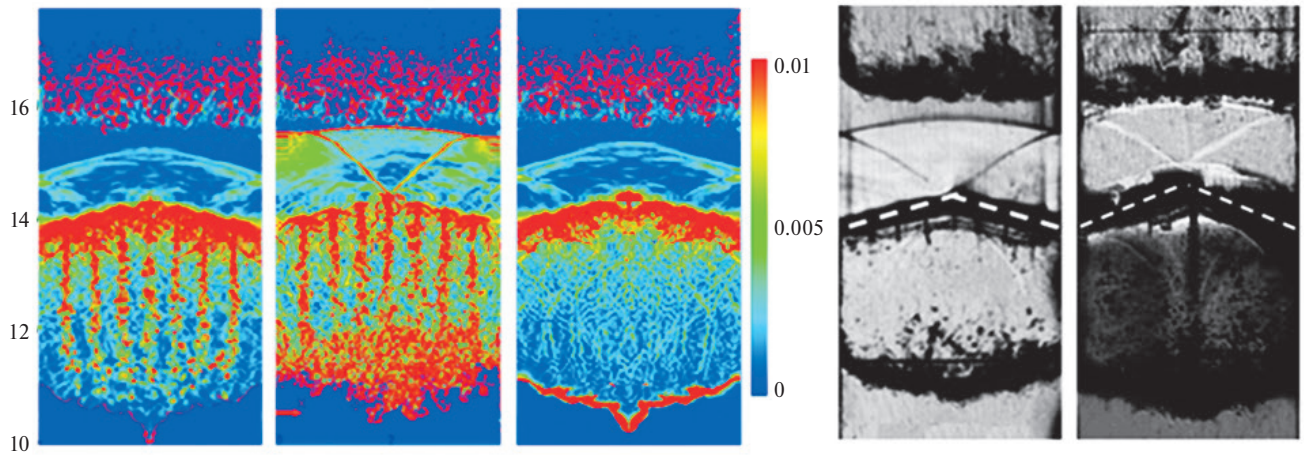


Рис. 7. Картины течения: расчеты $t = 491$, эксперимент с 3D-сеткой $t = 491{,}6$, эксперимент с 2D сеткой $t = 481$.



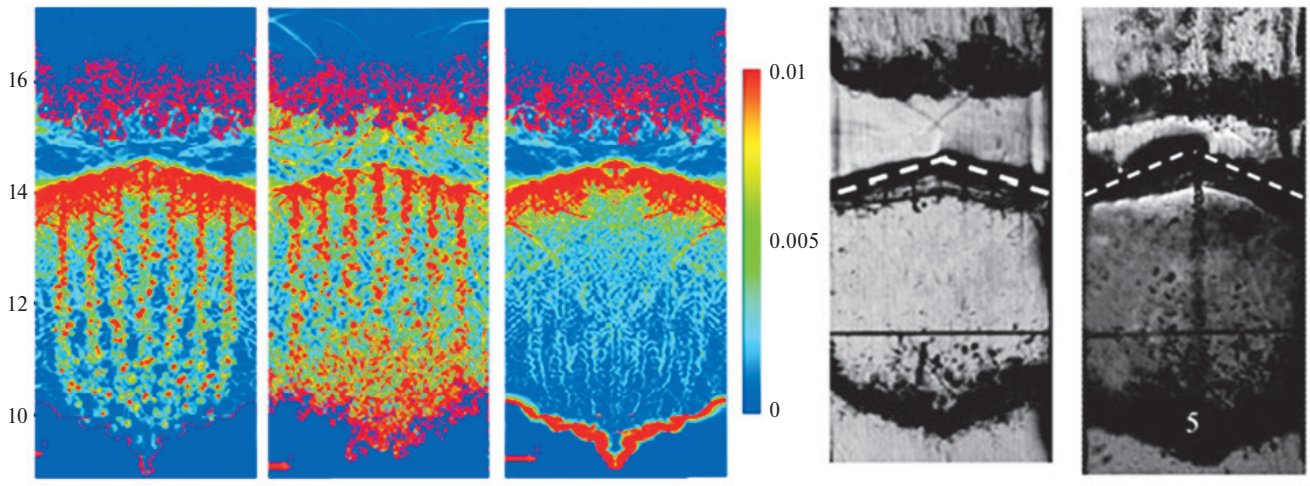
расчет 1 расчет 2 расчет 3 3D-сетка 2D-сетка

Рис. 8. Картины течения: расчеты $t = 571$, эксперимент с 3D-сеткой $t = 571{,}6$, эксперимент с 2D-сеткой $t = 565{,}1$.



расчет 1 расчет 2 расчет 3 3D-сетка 2D-сетка

Рис. 9. Картины течения: расчеты t = 732, эксперимент с 3D-сеткой t = 731,6, эксперимент с 2D-сеткой t = 732,1.



расчет 1 расчет 2 расчет 3 3D-сетка 2D-сетка

Рис. 10. Картины течения: расчеты $t = 812$, эксперимент с 3D-сеткой $t = 811{,}6$, эксперимент с 2D-сеткой $t = 815{,}6$.

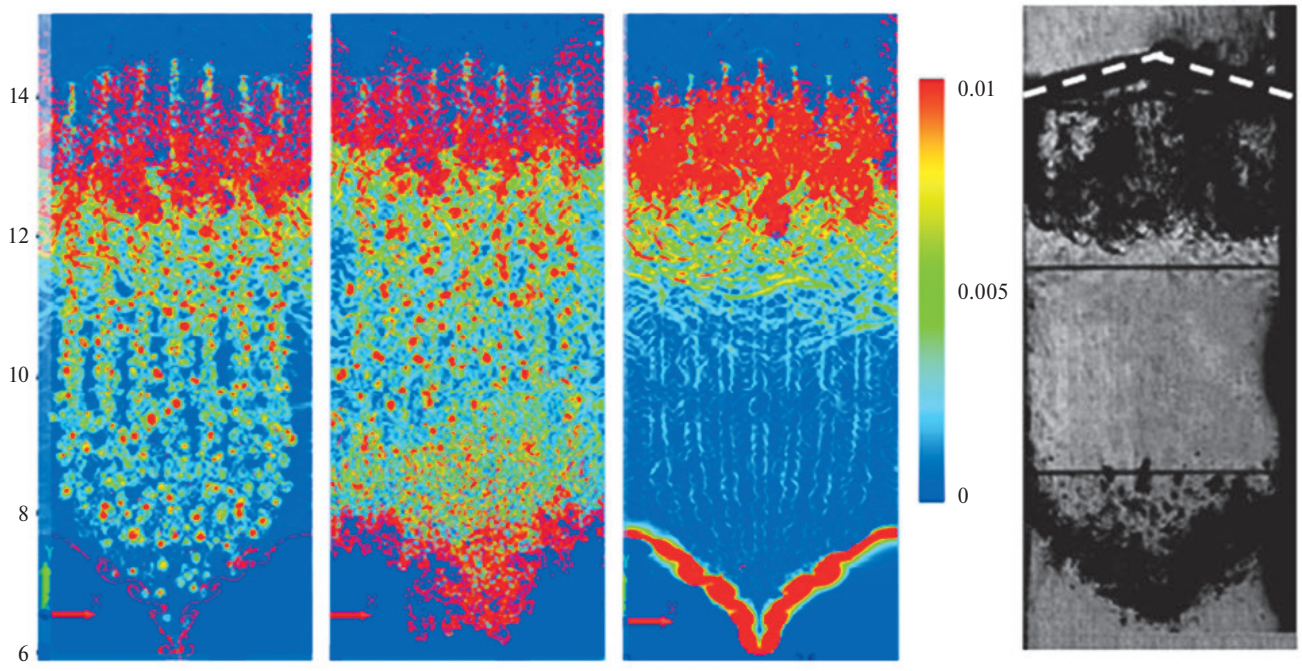


Рис. 11. Картины течения: расчеты $t = 1000$, эксперимент с 3D сеткой $t = 1071{,}6$ (для эксперимента с 2D-сеткой данных нет).

течение и эти струйки со временем размываются. Еще более выделяется расчет с k-ε моделью, в котором наличие диффузии сглаживает этот эффект с самого начала течения, но он все равно заметен (рис. 10).

Однако имеются и более существенные отличия расчетов от экспериментов. Во всех расчетах есть небольшое отставание ударной волны, приходящей на место расположения проволочной сетки (рис. 7), по сравнению с экспериментальными данными, на рис. 9 видно, что расчетная ударная волна опережает экспериментальную. Наиболее согласующиеся с экспериментом данные получены в расчете 2 с учетом фрагментов пленки. В частности, на рис. 9 в расчете 2 отраженная от пленки волна (которая распространяется вверх по трубе) заметно ближе к экспериментальному положению. Еще более заметно, что в расчете 2 ЗТП на границе воздух–SF_6 согласуется с экспериментальными данными, в то время как в расчете 1 она практически отсутствует, а имеются лишь завитушки вследствие проявления сдвиговой неустойчивости, а в расчете 3 ЗТП имеется, однако она значительно тоньше. Таким образом, пленка играет важную роль в данном течении и для корректного численного моделирования необходимо ее учитывать.

Отметим, что для 3D-сетки надо бы провести 3D-моделирование, однако подобный расчет требует неоправданно больших ресурсов компьютера. На качественном уровне результаты вполне адекватны и при 2D-моделировании.

5. ВЕРИФИКАЦИЯ ЧИСЛЕННОЙ МЕТОДИКИ МИМОЗА

Моделирование опыта № 1 выполнялось в постановке, приведенной на рис. 1. В трехслойной газовой системе центральный слой (между контактными границами $КГ_1$ и $КГ_2$) заполнялся легким газом (He). Вне центрального слоя располагался воздух. Все газы находились при атмосферном давлении.

УВ формировалась на левой границе и двигалась направо с числом Маха $M = 1{,}3$. Граничные условия: на верхней и нижней границах ставились периодические условия, за фронтом УВ газодинамические параметры соответствовали величинам за фронтом скачка.

Выполнено два расчета. В первом расчете на обеих КГ задавались двухмодовые ($a_1, a_2 \neq 0, a_3 = 0$) начальные возмущения, во втором – трехмодовые ($a_1, a_2, a_3 \neq 0$) возмущения по соотношению

$$D(y) = a_1 \sin(k_1 y) + a_2 \sin(k_2 y) + a_3 \sin(k_3 y).$$

Амплитуды и длины волн составляли $a_1 = 0{,}01$ см, $\lambda_1 = 3$ см, $a_2 = 0{,}01$ см, $\lambda_2 = 0{,}4$ см, $a_3 = 0{,}002$ см, $\lambda_3 = 0{,}01$ см.

5.1. Методика расчета

Математическое моделирование опыта № 1 выполнялось в 2D-постановке по методике МИМОЗА с использованием уравнений Эйлера без привлечения каких-либо моделей учета ТП (ILES моделирование). Расчетная методика основана на лагранжево-эйлеровой стратегии и выделении веществ концентрациями (подробнее см. [41–44]). Такой подход является эффективным при моделировании задач механики сплошной среды с большими деформациями.

Расчет счетного шага состоит из двух этапов: на первом этапе выполняется интегрирование уравнений Эйлера, записанных в лагранжевых координатах, на втором этапе производится пересчет полученных сеточных значений на первоначальную квадратную сетку [42]. Пересчет величин осуществляется при помощи алгоритма, основанного на расщеплении по координатным направлениям и использовании одномерного алгоритма повышенного порядка точности.

На лагранжевом этапе расчета границы ячеек сетки перемещаются со скоростью вещества, массы ячеек не изменяются. Интегрирование системы уравнений выполняется на разнесенной разностной сетке. Термодинамические параметры задачи относятся к центру счетной ячейки, координаты и компоненты скорости – к узлам. Используется полностью консервативная разностная схема “предиктор–корректор”, аналогичная [44]. С целью предотвращения размытия КГ между веществами применяется алгоритм их выделения с помощью концентраций. Для подавления паразитических осцилляций численного решения в окрестности больших градиентов газодинамических величин вводится искусственная вязкость, являющаяся суммой квадратичной и линейной вязкостей. В смешанных ячейках (ячейках, содержащих несколько веществ) давление вычисляется покомпонентно, а затем усредняется с учетом объемных концентраций веществ.

Расчет многокомпонентной сплошной среды осуществляется посредством двух методов отслеживания КГ материалов: метода концентраций – VolumeofFluid (VoF) [43] и метода моментов концентраций – MomentofFluid (MoF) [45]. Особенностью метода VoF является зависимость алгоритма от объемных концентраций материала в соседних счетных ячейках. Другая особенность VoF-методов заключается в том, что при наличии в смешанной ячейке более двух компонентов результат реконструкции КГ зависит от порядка обработки веществ в алгоритме.

В методе MoF, зная положение центра массы вещества и занимаемую им долю объема в ячейке, путем решения задачи минимизации определяется КГ вещества в виде прямой линии без привлечения данных из соседних счетных ячеек. Метод MoF имеет второй порядок точности, что позволяет точно реконструировать линейные КГ. Метод MoF позволяет точнее рассчитывать потоки объемов компонент на этапе адвекции и позволяет дольше удерживать целостность КГ материалов в процессе счета.

5.2. Результаты расчетов

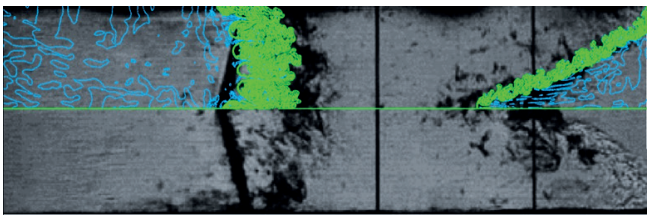
Результаты моделирования динамики границ и зон перемешивания, полученные с использованием методов VoF и MoF, дали близкие результаты. На рис. 12 сопоставляются результаты экспериментов и расчетов, полученных методом VoF. Рассчитанные по методике МИМОЗА зоны перемешивания соприкасающихся газов на $КГ_1$ и $КГ_2$ изображены на рисунке зеленым цветом.

Анализ полученных результатов показал, что:

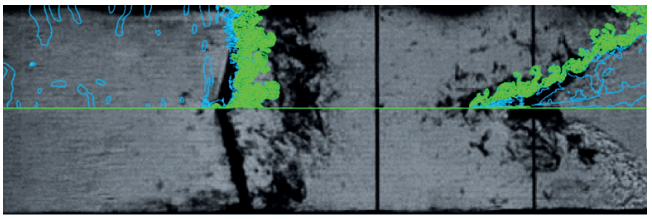
- ширина области перемешивания при трехмодовых начальных возмущениях заметно меньше, чем при двухмодовых (наиболее ярко это наблюдается на $КГ_1$);
- расчетная скорость $КГ_1$ несколько меньше, чем в эксперименте.

6. ЗАКЛЮЧЕНИЕ

Проведены численные расчеты динамики роста малых возмущений на контактных границах газов в трехслойных системах воздух–He–воздух и воздух–SF_6–воздух вследствие развития НРМ и НКГ. Эти



Двухмодовые возмущения, $t \approx 530$ мкс



Трехмодовые возмущения, $t \approx 530$ мкс

Рис. 12. Сравнение расчетной и экспериментальной динамики границ и зон перемешивания, слойка воздух-гелий-воздух, $t \approx 530$ мкс.

данные сопоставлены с данными натурных экспериментов, выполненных в ударной трубе. Получено удовлетворительное согласие расчетных и экспериментальных результатов по динамике зоны турбулентного перемешивания. Некоторые временные отличия могут быть связаны с экспериментальными погрешностями измерения скорости падающей на контактную границу ударной волны.

Эксперименты и расчеты показывают, что с увеличением размера начальных возмущений на контактной границе газов ширина зоны перемешивания увеличивается, поэтому для корректного численного моделирования турбулентного перемешивания веществ требуется строго знать спектр начальных возмущений.

Расчеты ИПМ им. М.В. Келдыша РАН выполнены с использованием вычислительных ресурсов Суперкомпьютерного центра коллективного пользования.

П.А. Кучугов выражает благодарность разработчикам системы SAPFOR и DVMH за реализацию параллельной версии программы NUT3D.

Работа выполнена в рамках научной программы Национального центра физики и математики по Государственному контракту № Н.4ц.241.4Д.23.1085.

СПИСОК ЛИТЕРАТУРЫ

1. *Richtmyer R.D.* Taylor instability in shock acceleration of compressed fluids, Commun. Pure Appl. Math. 13, 297, 1960.
2. *Мешков Е.Е.* Неустойчивость границы раздела двух газов, ускоряемой ударной волной. Изв. АН СССР. Механика жидкости и газа. 5. С. 151–158. 1969.
3. *Helmholtz H.L.F.* Uber discontinuilisch Flussigkeits-Bewegungen. Monatsberichte Konigl. Preus. Akad. Wiss. Berlin. 1868. P. 215.
4. *Taylor G.I.* The instability of liquid surfaces when accelerated in a direction perpendicular to their planes. I. Proc.Roy.Soc. 1950. V. A201. P.192.
5. *Bel'kov S.A., Bondarenko S.V., Demchenko N.N. et al.* Compression and burning of a direct-driven thermonuclear target under the conditions of inhomogeneous heating by multi-beam megajoule laser, PPCF, 61, 025011, 2019.
6. *Невмержицкий Н.В.* Гидродинамические неустойчивости и турбулентное перемешивание веществ. Лабораторное моделирование. Монография / под ред. доктора техн. наук А.Л. Михайлова. Саров: ФГУП "РФЯЦ-ВНИИЭФ", 2018. С. 246.
7. *Luo X., Guan B., Si T. et al.* Richtmyer-Meshkov instability of a three-dimensional SF6-air interface with a minimum-surface feature, Phys. Rev E, 93, 013101, 2016.
8. *Brouillette M.* The Richtmyer-Meshkov Instability, Annu. Rev. Fluid Mech., 34, 445, 2002.
9. *Невмержицкий Н.В., Разин А.Н., Трутнев Ю.А. и др.* Исследование развития турбулентного перемешивания в трехслойных газовых системах с наклонной контактной границей. ВАНТ. Сер. Теоретическая и прикладная физика, 2, 12–17, 2008.
10. *Козлов В.И., Разин А.И., Шапоренко Е.В. и др.* Результаты моделирования по методике КОРОНА газодинамических опытов по турбулентному перемешиванию в двумерных течениях. ВАНТ. Сер. Теоретическая и прикладная физика, 1, 31–38, 2009.
11. *Разин А.Н.* Взаимодействие ударной волны с наклонной контактной границей. ВАНТ. Сер. Теоретическая и прикладная физика, 2, 3–11, 2008.
12. *Henderson L.F.* On the refraction of shock waves, J. Fluid Mech., 198, 365–386, 1989.
13. *Hahn M., Drikakis D., Youngs D.L. et al.* Richtmyer-Meshkov turbulent mixing arising from an inclined material interface with realistic surface perturbations and reshoked flow, Phys. Fluids, 23, 046101, 2011.
14. *Разин А.Н.* Моделирование неустойчивости и турбулентного перемешивания в слоистых системах. Саров: ФГУП РФЯЦ ВНИИЭФ. 414. 2010.
15. *Змушко В.В., Разин А.Н., Синельникова А.А.* Влияние начальной шероховатости контактных границ на развитие неустойчивости после прохождения ударной волны, Прикладная механика и техническая физика. 63. 3. 34–42. 2022.
16. *Бодров Е.В., Змушко В.В., Невмержицкий Н.В. и др.* Расчетно-экспериментальное исследование развития турбулентного перемешивания в газовой слойке при прохождении ударной волны. Известия РАН. Механика жидкости и газа. 3. 54–62. 2018.
17. *Smith A.V., Holder D.A., Barton C.J. et al.* Shock tube experiments on Richtmyer-Meshkov instability across a chevron profiled interface, Proceedings of the 8th IWPCTM, 2001.
18. *Holder D.A., Barton C.J.* Shock tube Richtmyer-Meshkov experiments: Inverse chevron and half height, Proceeding of the 9th IWPCTM, 2004.
19. *Holder D.A., Smith A.V., Barton C.J. et al.* Shock-tube experiments on Richtmyer-Meshkov instability growth using an enlarged double-bump perturbation, Laser and Particle Beams, 23, 411, 2003.
20. *Ладонкина М.Е.* Численное моделирование турбулентного перемешивания с использованием высокопроизводительных систем. Институт математического моделирования РАН. 2005.
21. *Кучугов П.А.* Динамика процессов турбулентного перемешивания в лазерных мишенях. Институт прикладной математики им. М.В. Келдыша РАН. 2014.
22. *Кучугов П.А.* Моделирование имплозии термоядерной мишени на гибридных вычислительных системах, Сб. тр. междунар. науч. конф. "Параллельные

вычислительные технологии 2017", г. Казань, Республика Татарстан, 3–7 апреля 2017 г., 399-409, 2017.
23. *Брагин М.Д., Рогов Б.В.* О точном пространственном расщеплении многомерного скалярного квазилинейного гиперболического закона сохранения. Докл. АН. 2016. Т. 469. № 2. С. 143–147.
24. *Брагин М.Д.* Неявно-явные бикомпактные схемы для гиперболических систем законов сохранения. // Матем. моделирование. 2022. Т. 34. № 6. С. 3–21.
25. *Bragin M.D., Rogov B.V.* Conservative limiting method for high-order bicompact schemes as applied to systems of hyperbolic equations. Appl. Numer. Math. 2020. V. 151. P. 229–245.
26. *Брагин М.Д.* Влияние монотонизации на спектральное разрешение бикомпактных схем в задаче о невязком вихре Тейлора-Грина // Ж. вычисл. матем. и матем. физ. 2022. Т. 62. № 4. С. 625–641.
27. *Groom M., Thornber B.* The influence of the initial perturbation power spectra on the growth of a turbulent mixing layer induced by Richtmyer-Meshkov instability, Phys. D, 407, 132463, 2020.
28. *Youngs D.L.* Three-dimensional numerical simulation of turbulent mixing by Rayleigh-Taylor instability, Phys. Fluids A, 3, 1312, 1991.
29. *Тишкин В.Ф., Никишин В.В., Попов И.В., Фаворский А.П.* Разностные схемы трехмерной газовой динамики для задач о развитии неустойчивости Рихтмайера-Мешкова // Мат. модел. 7. 5. 15–25. 1995.
30. *Вязников К.В., Тишкин В.Ф., Фаворский А.П.* Построение монотонных разностных схем повышенного порядка аппроксимации для систем линейных дифференциальных уравнений с постоянными коэффициентами гиперболического типа // Мат. модел. 1. 5. 95–120. 1989.
31. *Toro E.F., Spruce M., Speares W.* Restoration of the Contact Surface in the HLL-Riemann Solver, Shock Waves, 4, 25-34, 1994.
32. *Самарский А.А., Соболь И.М.* Примеры численного расчета температурных волн. ЖВМиМФ. 3. 4. 702–719. 1963.
33. *Авдошина Е.В., Бондаренко Ю.А., Горбунов А.А., Дмитриева Ю.С., Наумов А.О., Проневич С.Н., Рудько Н.М., Тихомиров Б.П.* Исследование точности различных методов усреднения коэффициента теплопроводности на стороне ячейки интегрирования при численном решении уравнения теплопроводности // ВАНТ. Сер. Математическое моделирование физических процессов. 3. 32. 2014.
34. *Колганов А.С.* Автоматизация распараллеливания Фортран-программ для гетерогенных кластеров: автореф. дисс. к.ф.-м.н. М.: 2020. 22 с.
35. *Kataev N.* Application of the LLVM Compiler Infrastructure to the Program Analysis in SAPFOR. In: Voevodin V., Sobolev S. (eds) Supercomputing. RuSCDays 2018. Communications in Computer and Information Science, vol 965. Springer, Cham. DOI: https://doi.org/10.1007/978-3-030-05807-4_41
36. *Bakhtin V.A., Krukov V.A.* DVM-Approach to the Automation of the Development of Parallel Programs for Clusters // Programming and Computer Software. 2019. V. 45. № 3. P. 121-132.
37. *Янилкин Ю.В., Беляев С.П., Бондаренко Ю.А., Гаврилова Е.С., Гончаров Е.А., Горбенко А.Д., Городничев А.В., Губков Е.В., Гужова А.Р., Дегтяренко Л.И., Жарова Г.В., Колобянин В.Ю., Софронов В.Н., Стадник А.Л., Ховрин Н.А., Чернышова О.Н., Чистякова И.Н., Шемяков В.Н.* Эйлеровы численные методики ЭГАК и ТРЭК для моделирования многомерных течений многокомпонентной среды. Тр. РФЯЦ-ВНИИЭФ. Науч.-иссл. изд. Саров: РФЯЦ-ВНИИЭФ. Вып. 12. 54-65. 2008.
38. *Янилкин Ю.В.* Модели замыкания уравнений лагранжевой газодинамики и упругопластики в многокомпонентных ячейках. Часть 1. Изотропные модели // ВАНТ. Сер. ММФП. Вып. 3. 3–21. 2017.
39. *Янилкин Ю.В., Колобянин В.Ю., Чистякова И.Н., Егужова М.Ю.* Применение метода PPM в расчетах по методикам ЭГАК и ТРЭК // ВАНТ. Сер. ММФП. Вып. 4. 69–79. 2005.
40. *Бахрах С.М., Глаголева Ю.П., Самигулин М.С., Фролов В.Д., Яненко Н.Н., Янилкин Ю.В.* Расчет газодинамических течений на основе метода концентраций // ДАН СССР. Т. 257. № 3. 566–569. 1981.
41. *Змушко В.В., Плетенёв Ф.А., Сараев В.А., Софронов И.Д.* Методика решения трехмерных уравнений газовой динамики в смешанных лагранжево-эйлеровых координатах // ВАНТ. Сер. Методики и программы численного решения задач математической физики. 1988. Вып. 1. С. 22–27.
42. *Софронов И.Д., Афанасьева Е.А., Винокуров О.А. и др.* Комплекс программ МИМОЗА для решения многомерных задач механики сплошной среды на ЭВМ "Эльбрус-2" // Вопр. атом. науки и техники. Сер. Математическое моделирование физических процессов. 1990. Вып. 2. С. 3–9.
43. *Zmushko V.V.* Computation of convective flows and their realization in MIMOZA code // International Workshop "New Models of Numerical Codes for Shock Wave Processes in Condensed Media" / Oxford / September 15–19. 1997.
44. *Ладагин В.К., Пастушенко А.М.* Об одной схеме расчета газодинамических течений // Численные методы механики сплошной среды. 1977. Т. 8. № 2. С. 66–72.
45. *Benson D.J.* Volume of fluid interface reconstruction methods for multi-material problems. // Applied Mechanics Review 55(2). 2002. C. 151-165.
46. *Dyadechko V., Shashkov M.* Multi-material interface reconstruction from the moment data. Technical report LA-UR-07-0656, LANL, 2006.

MATHEMATICAL MODELING OF TURBULENT MIXING IN GAS SYSTEMS WITH A CHEVRON CONTACT BOUNDARY USING NUT3D, BIC3D, EGAK, AND MIMOSA NUMERICAL CODES

M. D. Bragin[a], N. V. Zmitrenko[a], V. V. Zmushko[b], P. A. Kuchugov[a], E. V. Levkina[b], K. V. Anisiforov[b], N. V. Nevmerzhitskiy[b], A. N. Razin[b], E. D. Sen'kovskiy[b], V. P. Statsenko[b], N. V. Tishkin[a], Yu. V. Tret'yachenko[b], Yu. V. Yanilkin[b]

[a] *Keldysh Institute of Applied Mathematics, Russian Academy of Sciences, Miusskaya pl. 4, Moscow, 125047 Russia*
[b] *Russian Federal Nuclear Center – All-Russian Research Institute of Experimental Physics, pr. Mira 37, Sarov, Nizhni Novgorod oblast, 607188 Russia*

The paper presents computational and experimental studies of the evolution of turbulent mixing in three-layer gas systems with the development of hydrodynamic instabilities, in particular Richtmyer-Meshkov and Kelvin-Helmholtz, under the action of shock waves. One of the contact boundaries of the gases was flat, the other with a break in the form of a chevron. Numerical calculations are performed both without initial perturbations of the contact boundaries, and in the presence of perturbations. It is shown that the roughness of the contact boundary significantly affects the width of the mixing zone.

Keywords: turbulent mixing, mathematical modeling, hydrodynamic instabilities, shock tube

REFERENCES

1. *Richtmyer R.D.* Taylor instability in shock acceleration of compressed fluids, Commun. Pure Appl. Math. 13, 297, 1960.
2. *Meshkov E.E.* Instability of the interface between two gases, accelerated by a shock wave, Izv. Academy of Sciences of the USSR, Mechanics of Liquid and Gas, 5, 151–158, 1969.
3. *Helmholtz H.L.F.* Uber discontinuilisch Flussigkeits-Bewegungen. Monatsberichte Konigl. Preus. Akad. Wiss. Berlin. 1868. P. 215.
4. *Taylor G.I.* The instability of liquid surfaces when accelerated in a direction perpendicular to their planes. I. Proc.Roy.Soc., 1950. V.A201. P.192.
5. *Bel'kov S.A., Bondarenko S.V., Demchenko N.N. et al.* Compression and burning of a direct-driven thermonuclear target under the conditions of inhomogeneous heating by multi-beam megajoule laser, PPCF, 61, 025011, 2019.
6. *Nevmerzhitsky N.V.* Hydrodynamic instabilities and turbulent mixing of substances. Laboratory modeling. Monograph / Ed. Doctor of Technical Sciences A.L. Mikhailova. – Sarov: FSUE "RFNC-VNIIEF". 2018. P. 246.
7. *Luo X., Guan B., Si T. et al.* Richtmyer-Meshkov instability of a three-dimensional SF6-air interface with a minimum-surface feature, Phys. Rev E, 93, 013101, 2016.
8. *Brouillette M.* The Richtmyer-Meshkov Instability, Annu. Rev. Fluid Mech., 34, 445, 2002.
9. *Nevmerzhitsky N.V., Razin A.N., Trutnev Yu.A. and others.* Study of the development of turbulent mixing in three-layer gas systems with an inclined contact boundary, VANT. Series: theoretical and applied physics, 2, 12–17, 2008.
10. *Kozlov V.I., Razin A.I., Shaporenko E.V. and others*. Results of modeling gas-dynamic experiments on turbulent mixing in two-dimensional flows using the CORONA method, VANT. Series: theoretical and applied physics, 1, 31–38, 2009.
11. *Razin A.N.* Interaction of a shock wave with an inclined contact boundary, VANT. Series: theoretical and applied physics, 2, 3–11, 2008.
12. *Henderson L.F.* On the refraction of shock waves, J. Fluid Mech., 198, 365-386, 1989.
13. *Hahn M., Drikakis D., Youngs D.L. et al.* Richtmyer-Meshkov turbulent mixing arising from an inclined material interface with realistic surface perturbations and reshoked flow, Phys. Fluids, 23, 046101, 2011.
14. *Razin A.N.* Modeling of instability and turbulent mixing in layered systems, Sarov, FSUE RFNC VNIIEF, 414, 2010.
15. *Zmushko V.V., Razin A.N., Sinelnikova A.A.* The influence of the initial roughness of contact boundaries on the development of instability after the passage of a shock wave, Applied Mechanics and Technical Physics, 63, 3, 34–42, 2022.
16. *Bodrov E.V., Zmushko V.V., Nevmerzhitsky N.V. and others.* Computational and experimental study of the development of turbulent mixing in a gas layer during the passage of a shock wave, Izvestiya RAS. Mechanics of Liquid and Gas, 3, 54–62, 2018.
17. *Smith A.V., Holder D.A., Barton C.J. et al.* Shock tube experiments on Richtmyer-Meshkov instability across a chevron profiled interface, Proceedings of the 8th IWPCTM, 2001.
18. *Holder D.A., Barton C.J.* Shock tube Richtmyer-Meshkov experiments: Inverse chevron and half height, Proceeding of the 9th IWPCTM, 2004.
19. *Holder D.A., Smith A.V., Barton C.J. et al.* Shock-tube experiments on Richtmyer-Meshkov instability growth using an enlarged double-bump perturbation, Laser and Particle Beams, 23, 411, 2003.
20. *Ladonkina M.E.* Numerical modeling of turbulent mixing using high-performance systems, 2005, Institute of Mathematical Modeling of the Russian Academy of Sciences.

21. *Kuchugov P.A.* Dynamics of turbulent mixing processes in laser targets, 2014, Institute of Applied Mathematics named after. M.V. Keldysh RAS.
22. *Kuchugov P.A.* Modeling the implosion of a thermonuclear target on hybrid computing systems, Collection of proceedings of the international scientific conference «Parallel Computing Technologies 2017», Kazan, Republic of Tatarstan, April 3-7, 2017, 399-409, 2017.
23. *Bragin M.D., Rogov B.V.* On the exact spatial splitting of a multidimensional scalar quasilinear hyperbolic conservation law. Dokl. AN. 2016. T. 469. No. 2. P. 143–147.
24. *Bragin M.D.* Implicit-explicit compact schemes for hyperbolic systems of conservation laws. Math. modeling. 2022. T. 34. No. 6. P. 3-21.
25. *Bragin M.D., Rogov B.V.* Conservative limiting method for high-order bicompact schemes as applied to systems of hyperbolic equations. Appl. Numer. Math. 2020. V. 151. P. 229–245.
26. *Bragin M.D.* The influence of monotonization on the spectral resolution of bicompact schemes in the Taylor-Green inviscid vortex problem. J. Comput. math. and math. physical 2022. T. 62. No. 4. P. 625–641.
27. *Groom M., Thornber B.* The influence of the initial perturbation power spectra on the growth of a turbulent mixing layer induced by Richtmyer-Meshkov instability, Phys. D, 407, 132463, 2020.
28. *Youngs D.L.* Three-dimensional numerical simulation of turbulent mixing by Rayleigh-Taylor instability, Phys. Fluids A, 3, 1312, 1991.
29. *Tishkin V.F., Nikishin V.V., Popov I.V., Favorsky A.P.* Difference schemes of three-dimensional gas dynamics for problems on the development of Richtmyer-Meshkov instability, Mat. model., 7, 5, 15–25, 1995.
30. *Vyaznikov K.V., Tishkin V.F., Favorsky A.P.* Construction of monotonic difference schemes of higher order of approximation for systems of linear differential equations with constant coefficients of hyperbolic type, Mat. model., 1, 5, 95–120, 1989.
31. *Toro E.F., Spruce M., Speares W.* Restoration of the Contact Surface in the HLL-Riemann Solver, Shock Waves, 4, 25–34, 1994.
32. *Samarsky A.A., Sobol I.M.* Examples of numerical calculation of temperature waves, ZhVMiMF, 3, 4, 702–719, 1963.
33. *Avdoshina E.V., Bondarenko Yu.A., Gorbunov A.A., Dmitrieva Yu.S., Naumov A.O., Pronevich S.N., Rudko N.M., Tikhomirov B.P.* Study of the accuracy of various methods for averaging the thermal conductivity coefficient on the side of the integration cell in the numerical solution of the heat equation, VANT, ser. Mathematical modeling of physical processes, 3, 32, 2014.
34. *Kolganov A.S.* Automation of parallelization of Fortran programs for heterogeneous clusters: Abstract of the dissertation of Ph.D., M.: 2020. 22 p.
35. *Kataev N.* Application of the LLVM Compiler Infrastructure to the Program Analysis in SAPFOR. In: Voevodin V., Sobolev S. (eds) Supercomputing. RuSCDays 2018. Communications in Computer and Information Science, vol 965. Springer, Cham. DOI: https://doi.org/10.1007/978-3-030-05807-4_41
36. *Bakhtin V.A., Krukov V.A.* DVM-Approach to the Automation of the Development of Parallel Programs for Clusters // Programming and Computer Software. 2019. Vol. 45. № 3. P. 121–132.
37. *Yanilkin Yu.V., Belyaev S.P., Bondarenko Yu.A., Gavrilova E.S., Goncharov E.A., Gorbenko A.D., Gorodnichev A.V., Gubkov E.V., Guzhova A.R., Degtyarenko L.I., Zharova G.V., Kolobyanin V.Yu., Sofronov V.N., Stadnik A.L., Khovrin N.A., Chernyshova O.N., Chistyakova I .N., Shemyakov V.N.* Euler numerical methods EGAC and TREC for modeling multidimensional flows of a multicomponent medium. Proceedings of RFNC-VNIIEF. Scientific research publication, Sarov: RFNC-VNIIEF, 200, Vol. 12, P. 54–65.
38. *Yanilkin Yu.V.* Models for closing the equations of Lagrangian gas dynamics and elastic-plasticity in multicomponent cells. Part 1. Isotropic models // VANT, ser. MMFP, Vol. 3, 2017, P. 3–21.
39. *Yanilkin Yu.V., Kolobyanin V.Yu., Chistyakova I.N., Eguzhova M.Yu.* Application of the PPM method in calculations using the EGAC and TREC methods // VANT, ser. MMFP, 2005, Vol. 4, P. 69–79.
40. *Bakhrakh S.M., Glagoleva Yu.P., Samigulin M.S., Frolov V.D., Yanenko N.N., Yanilkin Yu.V.* Calculation of gas-dynamic flows based on the concentration method // DAN USSR, 1981, Vol. 257, No. 3, P. 566–569.
41. *Zmushko V.V., Pletenev F.A., Saraev V.A., Sofronov I.D.* Methodology for solving three-dimensional gas dynamics equations in mixed Lagrangian-Eulerian coordinates // VANT. Ser. Methods and programs for numerical solution of problems of mathematical physics. 1988. Issue 1. P. 22–27.
42. *Sofronov I.D., Afanasyeva E.A., Vinokurov O.A. and others.* Complex of MIMOZA programs for solving multidimensional problems of continuum mechanics on the Elbrus-2 computer // Vopr. atom. science and technology. Ser. Mathematical modeling of physical processes. 1990. Issue 2. P. 3–9.
43. *Zmushko V.V.* Computation of convective flows and their realization in MIMOZA code // International Workshop "New Models of Numerical Codes for Shock Wave Processes in Condensed Media" / Oxford / September 15–19. 1997.
44. Ladagin V.K., Pastushenko A.M. On one scheme for calculating gas-dynamic flows // Numerical methods of continuum mechanics. 1977. T.8. No. 2. P.66-72.
45. *Benson D.J.* Volume of fluid interface reconstruction methods for multi-material problems. // Applied Mechanics Review 55(2). 2002. C. 151–165.
46. *Dyadechko V., Shashkov M.* Multi-material interface reconstruction from the moment data. Technical report LA-UR-07-0656, LANL, 2006.

ПРОГРАММИРОВАНИЕ, *2024*, *№ 1*, *с. 88–99*

ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ

УДК: 004.43

ОПИСАНИЕ СЕМАНТИКИ ЯЗЫКА PARALOCKS В TLA+

© 2024 г. А. А. Тимаков[a], * (ORCID: 0000-0003-4306-789X)

[a]*МИРЭА – Российский технологический университет, 119454 Москва, пр. Вернадского, д. 78*

**E-mail: timakov@mirea.ru*

Поступила в редакцию 15.03.2023
После доработки: 07.04.2023
Принята к публикации: 03.07.2023

Одним из основных аспектов реализации контроля информационных потоков в программном обеспечении является язык описания политик безопасности или меток. Важно, чтобы такой язык позволял формировать политики безопасности элементов среды вычислений на основе правил управления доступом системного уровня. Соответственно язык должен быть достаточно гибким, поскольку на системном уровне могут использоваться разные механизмы: ролевое, мандатное управление доступом и др. Кроме того, в приложении могут действовать некоторые дополнительные ограничения. Наконец, желательно, чтобы язык позволял естественным образом учитывать возможность деклассификации данных (контролируемого раскрытия) в процессе вычислений. Одним из таких языков является *Paralocks*. Работа посвящена реализации семантики несколько упрощенной версии этого языка с использованием *TLA+*. *Paralocks* является языковой частью разработанной с участием автора платформы анализа информационных потоков в хранимых программных блоках баз данных *PLIF*. Приводятся доказательства свойств заданного отношения частичного порядка и решетки, построенной на основе множества всех возможных политик безопасности.

Ключевые слова: контроль потока информации, язык политики безопасности, логическая семантика, формальная верификация, проверка модели, программные модули

DOI: 10.31857/S0132347424010073 **EDN:** HJYWFT

1. ВВЕДЕНИЕ

Статья будет полезна читателям, хорошо знакомым с формальными моделями безопасности компьютерных систем, основанными на управлении доступом и контроле информационных потоков (КИП), обладающим достаточными знаниями в области формальной верификации и навыками работы с соответствующими инструментальными средствами. Необходимую предварительную информацию можно получить, обратившись к источникам: [1]—[5].

Подавляющее число реализованных механизмов контроля информационных потоков [4], [5] при описании множеств политик безопасности и абстрактной семантики информационных потоков опираются на понятие алгебраической решетки. При этом доказательства свойств предлагаемых решеток и корректности операторов вычисления минимальной верхней (максимальной нижней) грани часто не приводятся. Несмотря на кажущуюся очевидность предположений, отсутствие таких доказательств несколько подрывает доверие к реализованным механизмам.

В работе представлена семантика упрощенной версии языка описания политик безопасности *Paralocks* [4], лежащего в основе механизма КИП платформы *PLIF* [6], а также приводятся исчерпывающие пояснения к доказательствам свойств решетки, заданной на множестве всех возможных политик.

Для проверки обозначенных свойств применялся следующий подход. На предварительном этапе осуществлялся прогон модели на небольшом наборе исходных данных с использованием инструмента *TLC*. Отсутствие выявленных на этом этапе противоречий сделало оправданным переход ко второму этапу – доказательству свойств с использованием формальной логической системы *TLAPS*. Указанная система позволяет выстраивать доказательства в иерархическим стиле, который упрощает применение метода естественной дедукции. *TLAPS* опирается на инструменты автоматизированного доказательства теорем, такие как Isabelle [7], Zennon [8]. Описанный подход предоставляет дополнительные гарантии, что достаточно важно в области формальных моделей. По мнению некоторых авторов [9] до трети доказательств, предлагаемых в различных исследованиях, являются некорректными.

Отметим также, что определенной инновацией явилось использование подсистемы проверки типов *Apalache* [10] для доказательства инвариантов типов некоторых структур, используемых для описания

предложений политик безопасности и самих политик в спецификациях *TLA+*. Это, в свою очередь, позволило обоснованно использовать в доказательствах некоторые неочевидные предположения.

2. СЕМАНТИКА PARALOCKS В TLA+

Язык описания политик безопасности в программной среде *Paralocks* [4] является основой перспективной платформы контроля информационных потоков в программных блоках баз данных *PLIF*. Преимущества языка, предопределившие его выбор, описаны в нашей предыдущей статье, опубликованной в этом журнале.

Политика безопасности P_k определяется формулой логики предикатов, представимой в виде конъюнкции определенных дизъюнктов Хорна: $P_k \triangleq C_1 \wedge C_2 \wedge ...$, здесь C_n – предложение политики вида:

$$\forall x_1, ..., x_m \, l_1(\cdot) \wedge l_2(\cdot) \, l_3(\cdot) \ldots Flow(u),$$

где u – связанная переменная или константа, обозначающая пользователя, $Flow(u)$ – предикат, обозначающий легальность потока данных к u, $l_1 \ldots l_n$ — условия (блокировки), выполнение которых требуется для того, чтобы $Flow(u)$ принял значение *TRUE*. Предикаты $l_1 \ldots l_n$ могут быть параметрическими или непараметрическими.

Например, выражение политики: “Поток к произвольному пользователю *x* возможен, если а) открыта блокировка *guest* — для *x* задана роль *guest* – и открыта блокировка *t_expire* – истек заданный интервал времени **или** б) открыта блокировка *manager* — для *x* задана роль *manager*” – имеет формальный вид:

$$\forall x. \, (t\_expire \wedge guest(x) \Rightarrow Flow(x)) \wedge \\ \wedge \, (manager(x) \Rightarrow Flow(x)).$$

Для описания политик в спецификации *Paralocks.tla* [6] вводятся следующие константы: *UU* — множество допустимых имен связанных переменных, *U* — множество актеров (конкретных пользователей системы), E_0 — множество имен непараметрических блокировок, E_1 — множество имен параметрических (с одним параметром) блокировок. Известные варианты использования *Paralocks* для кодирования политик элементов программной среды (переменных, параметров функций, исключений и т.д.), преемственных к наиболее распространенным механизмам управления доступом (мандатному, ролевому), позволяют ограничиться лишь непараметрическими и однопараметрическими блокировками, а также внести иные допущения, упрощающие описание соответствующих сущностей в спецификациях *TLA+*. Положим, что множество имен связанных переменных *UU* включает один элемент, также будем считать, что, если связанная переменная появляется в одной из блокировок некоторого предложения политики, то эта же переменная является аргументом и в литерале Flow(·) того же предложения, и если связанная переменная является аргументом Flow(·) некоторого предложения политики, то все параметрические блокировки того же предложения также будут принимать эту переменную в качестве аргумента.

Множество возможных предложений политик безопасности и множество самих политик определяются как:

$$\begin{array}{l} ClausesSet \;\triangleq \\ \quad \{\langle u, \langle e0, e1\rangle\rangle : u \;\in (U \cup UU), \\ \quad\; e0 \in [E0 \to \text{SUBSET } \{NONE\}], \\ \quad\; e1 \in [E1 \to ((\text{SUBSET } (U) \cup \text{SUBSET } (UU)) \\ \qquad \setminus \{\{\}\}) \cup \{\{NONE\}\}]\} \\ \\ PoliciesSet \;\triangleq \\ \quad \{p \in \text{SUBSET } ClausesSet : \\ \quad\; \forall\, c1 \in p : \neg\exists\, c2 \in p : \\ \qquad \wedge\, c1 \neq c2 \\ \qquad \wedge\; \vee\, (compareClause(c1, c2) \\ \qquad\quad \vee\, compareClause(c2, c1))\} \\ \quad\; \cup \{\{\}\} \end{array}$$

Политика описывается множеством двумерных кортежей. Каждый кортеж – отдельное предложение политики, его первым элементом является связанная переменная или константа, обозначающие пользователя, к которому разрешен информационный поток; второй элемент – это двумерный кортеж, с его помощью описывается множество блокировок, которые должны быть открыты. Элементами этого кортежа являются записи, их поля соответствуют именам блокировок, определяемых константами модели, значения полей задают множества фактических параметров блокировок. В одной политике не может быть двух разных упорядоченных предложений.

Описанный ранее пример политики в спецификациях *PLIF* имеет вид[1]:

[1] Для интерпретации трасс, приводящих к ошибкам, PLIF используется упрощенная нотация, в которой пример выглядит как:

```
x: manager(x)
x: t_expire
```

$$\{\langle x, \langle [t\_expire \mapsto \{\}], [guest \mapsto \{x\},$$
$$reviewer \mapsto \{NONE\}, manager \mapsto \{NONE\},$$
$$organizer \mapsto \{NONE\}]\rangle\}\rangle],$$
$$\langle x, \langle [t\_expire \mapsto \{NONE\}], [guest \mapsto \{NONE\},$$
$$reviewer \mapsto \{NONE\}, manager \mapsto \{x\},$$
$$organizer \mapsto \{NONE\}]\rangle\rangle\}$$

Здесь: $x \in UU$, $t\_expire \in E_0$, {*guest, reviewer, manager, organizer*} $\subseteq E_1$, $NONE$ — специальное значение, которое обозначает, что открытие блокировки не требуется (в *TLA+* записи являются функциями, при этом частично определенные функции не поддерживаются).

Также с учетом заданных ограничений справедливы следующие предположения:

$$\text{ASSUME } U\_ASSM \triangleq U \neq \{\} \wedge NONE \notin U$$

$$\text{ASSUME } UU\_ASSM \triangleq UU = \{\text{“x”}\}$$

$$\text{ASSUME } Clause\_ASSM2 \triangleq$$
$$\forall c \in ClausesSet :$$
$$(\vee \wedge c[1] \in UU$$
$$\wedge \forall e1 \in E1 : c[2][2][e1] = UU$$
$$\vee \wedge c[1] \notin UU$$
$$\wedge \forall e1 \in E1 : c[2][2][e1] \cap UU = \{\})$$

Пояснения к некоторым иным предположениям, заданным в *Paralocks.tla* и используемым в формальных доказательствах, приводятся несколько позже.

На множестве предложений политик безопасности задан оператор сравнения:

$$compareClause(c1, c2) \triangleq$$
$$\wedge substMap3Equality(c1, c2)$$
$$\wedge \forall k \in E0 : \vee c1[2][1][k] = c2[2][1][k]$$
$$\vee c2[2][1][k] = \{\}$$
$$\wedge \forall e \in E1 : \vee c1[2][2][e] = \{NONE\}$$
$$\vee matchLocks(c1, c2, e)$$
$$\subseteq c2[2][2][e]$$

Оператор $compareClause(c1, c2)$ возвращает значение *TRUE* – предложение $c2$ является как минимум таким же строгим, как и предложение $c1$ — если выполняются два условия: а) существует подстановка, приводящая предикат *Flow*(·) предложения $c1$ к виду, идентичному предикату *Flow*(·) предложения $c2$; б) после применения соответствующей подстановки к блокировкам предложения $c1$ множество блокировок предложения $c1$ будет представлять подмножество блокировок предложения $c2$. Логическая интерпретация заданного таким образом отношения частичного порядка приводится в [4].

Параметром предиката *Flow*(·) может выступать связанная переменная (элемент множества *UU*) или конкретный пользователь (элемент множества *U*), поэтому истинность первого условия определяется результатом вычисления функции:

$$substMap3Equality(c1, c2) \triangleq$$
$$c1[1] \in UU \vee c1[1] = c2[1]$$

В принятой нотации вторым элементом предложения политики является двумерный кортеж записей, определяющих необходимые наборы непараметрических и параметрических блокировок. Если открытие некоторой непараметрической блокировки не требуется, соответствующее поле записи принимает значение {*NONE*}, в противном случае – {}. Поэтому вхождение множества непараметрических блокировок предложения $c1$ во множество непараметрических блокировок предложения $c2$ фактически означает:

$$\forall k \in E0 : \vee c1[2][1][k] = c2[2][1][k]$$
$$\vee c2[2][1][k] = \{\}$$

Для однопараметрических блокировок аналогично – значение {*NONE*} в поле соответствующей блокировки означает, что открытие блокировки не требуется. Если поле блокировки содержит не пустое множество элементов, отличных от {*NONE*}, значит, легитимность информационного потока к пользователю, заданному предикатом *Flow*(·), требует открытия некоторых экземпляров этой блокировки (определяются указанным множеством элементов). Функция *matchLocks*(·) применяет при необходимости соответствующую подстановку к выражениям однопараметрических блокировок предложения $c1$:

$$matchLocks(c1, c2, e) \triangleq$$
$$\text{LET } c \triangleq c1[2][2][e]$$
$$\text{IN}$$
$$\text{IF } c \cap UU \neq \{\}$$
$$\text{THEN } (c \setminus UU) \cup \{c2[1]\}$$
$$\text{ELSE } c$$

Для сравнения политик безопасности используется функция *comparePol*(·), ее определение тривиально:

$$comparePol(p1, p2) \triangleq \forall\, c2 \in p2 : (\exists\, c1 \in p1 : compareClause(c1, c2))$$

Далее в пояснениях к выражениям *TLAPS* в некоторых случаях вместо *compareClause*(·) и *comparePol*(·) будем использовать оператор.

Наиболее важными правилами семантиками языка *Paralocks* являются правила вычисления наименьшей верхней (НВГ) и наибольшей нижней граней (ННГ) двух заданных политик безопасности. ННГ двух произвольных политик безопасности $p1$ и $p2$ является некоторая максимально строгая политика p_{GLB}, такая, что $p_{\text{GLB}} \subseteq p1$ и $p_{\text{GLB}} \subseteq p2$. Поскольку каждое предложение политики представляет собой отдельную возможность возникновения разрешенного информационного потока, очевидно, справедливо:

$$GLB(p1, p2) \triangleq p1 \cup p2$$

НВГ двух произвольных политик безопасности $p1$ и $p2$ является некоторая минимально строгая политика p_{LUB}, такая, что $p1 \sqsubseteq p_{LUB}$ и $p2 \sqsubseteq p_{LUB}$. Для ее вычисления необходимо для всех возможных пар $\langle c1, c2\rangle$, таких, что $c1 \in p1$, $c2 \in p2$, и существует подстановка, приводящая предикат *Flow*(·) предложения $c1$ к виду, идентичному предикату *Flow*(·) предложения $c2$ или, наоборот, предикат *Flow*(·) предложения $c2$ — к виду, идентичному предикату *Flow*(·) предложения $c1$, сгенерировать новое предложение политики, применив соответствующую подстановку к блокировкам предложения $c1$ или $c2$ и объединив после этого полученные множества блокировок исходных предложений:

$$LUB(p1, p2) \triangleq \{x \in \{unionCl(c1, c2) : c1 \in p1, c2 \in p2\} : x \neq \langle\rangle\}$$

Здесь функция *unionCl*($c1, c2$) вычисляет новое предложение политики p_{LUB} на основе предложений $c1$ и $c2$, если существует соответствующая подстановка, или возвращает пустое предложение $\langle\rangle$:

```
unionCl(c1, c2) ≜
  LET capMap ≜ [e0 ∈ E0 ↦
      c1[2][1][e0] ∩ c2[2][1][e0]]
  IN
    IF substMap3Equality(c1, c2)
      THEN ⟨c2[1], ⟨capMap,
             [e1 ∈ E1 ↦
             IF  ∧ NONE ∈ c1[2][2][e1]
                 ∧ NONE ∈ c2[2][2][e1]

                 THEN {NONE}
                 ELSE (matchLocks(c1, c2, e1)
                   ∪ c2[2][2][e1]) \ {NONE}]⟩⟩
      ELSE
    IF substMap3Equality(c2, c1)
      THEN ⟨c1[1], ⟨capMap,
             [e1 ∈ E1 ↦
             IF  ∧ NONE ∈ c1[2][2][e1]
                 ∧ NONE ∈ c2[2][2][e1]
                 THEN {NONE}
                 ELSE (matchLocks(c2, c1, e1)
                   ∪ c1[2][2][e1]) \ {NONE}]⟩⟩
      ELSE ⟨⟩
```

Локальная функция *capMap* объединяет непараметрические блокировки предложений $c1$ и $c2$ (поле, соответствующее непараметрической блокировке, принимает значение {*NONE*}, если блокировка не требуется, в противном случае — {}).

Для доказательства некоторых свойств с использованием *TLAPS*, как уже отмечалось, удобно использовать допущения – предположения (assumptions). Доказательство их справедливости иногда является нетривиальной задачей, а порой невозможной, поскольку *TLAPS* имеет ряд ограничений. Например, в системе отсутствует полноценная поддержка множеств, заданных с использованием кортежей. Проверка таких свойств с использованием механизма проигрывания моделей на неполных наборах данных не дает формальных гарантий. Для инвариантов типов проблема, как показывает исследование, может быть частично решена с использованием инструмента вывода типов *Snowcat*, который является частью платформы *Apalache* [10]. Еще одним предположением, используемым в доказательствах, является *LUB_PS*. Его обоснованность подтверждается проверкой с использованием *Snowcat*. Стоит отметить, однако, что использование указанного инструмента как правило требует специальной разметки спецификации:

@*typeAlias*: $CLAUSE = \langle U, \langle E0 \to Set(U), E1 \to Set(U)\rangle\rangle$
@*typeAlias*: $POLICY = Set(CLAUSE)$;

@type: $Set(CLAUSE)$;
$ClausesSet \triangleq \{\ldots\}$

@type: $Set(POLICY)$;
$PoliciesSet \triangleq \{\ldots\}$

@type: $(POLICY, POLICY) \implies POLICY$;
$LUB(p1, p2) \triangleq \{x \in \{unionCl(c1, c2) : c1 \in p1, c2 \in p2\} : x \neq EC\}$

ASSUME $LUB\_PS \triangleq \forall p1, p2 \in PoliciesSet : LUB(p1, p2) \in PoliciesSet$

Ряд иных тривиальных предположений, описанных в *Paralocks.tla*, в данной работе не разбирается.

3. ПРОВЕРКА СВОЙСТВ РЕШЕТКИ, ЗАДАННОЙ НА МНОЖЕСТВЕ ПОЛИТИК БЕЗОПАСНОСТИ

Процесс вывода формальных доказательств необходимых свойств алгебраических структур, описанных в спецификации *Paralocks.tla*, с использованием логической системы *TLAPS* является весьма трудоемким, поэтому на начальном этапе указанные свойства проверялись для ограниченного набора данных с использованием инструмента оценки выражений над константами среды *TLA+Toolbox*, а в некоторых случаях и с использованием инструмента проигрывания моделей *TLC*. Результаты предварительного этапа в данной работе не приводятся, однако их можно обнаружить в файлах проекта [6]. Представим общий вид доказательства того, что функция *comparePol*(·) является отношением частичного порядка на множестве политик *PoliciesSet*.

Рефлексивность

Проверку удобно разбить на два случая: когда множество предложений произвольной политики является пустым и когда данное множество не является пустым.

THEOREM $Reflexitivity \triangleq \forall p \in PoliciesSet : comparePol(p, p)$
⟨1⟩ USE DEF *comparePol*, *compareClause*, *substMap3Equality*, *matchLocks*
⟨1⟩1 TAKE $pol \in PoliciesSet$
⟨1⟩2 CASE pol = { }
...
⟨1⟩3 CASE pol ≠ { }
...
⟨1⟩4. QED
BY ⟨1⟩2, ⟨1⟩3

В первом случае очевидно:

$\forall c2 \in pol : (\exists c1 \in pol : \text{compareClause}(c1,c2))$,

доказательство следует из определения *comparePol*(·).

Во втором случае требуется для произвольного $c1 \in pol$ доказать истинность *compareClause*(*c*1,*c*1) или с учетом определения *compareClause*(·):

...
⟨1⟩3.CASE $pol \neq \{\}$
⟨2⟩1 TAKE $c1 \in pol$
...
⟨2⟩3 $substMap3Equality(c1, c1)$
...
⟨2⟩4 $\forall k \in E0 : \lor c1[2][1][k] = c1[2][1][k] \lor c1[2][1][k] = \{\}$
...
⟨2⟩5 $\forall e \in E1 : \lor c1[2][2][e] = \{NONE\} \lor matchLocks(c1, c1, e) \subseteq c1[2][2][e]$
...
⟨2⟩6 QED
BY ⟨2⟩3, ⟨2⟩4, ⟨2⟩5
...

Доказательство шага ⟨2⟩ 3 очевидно, оно следует из определения *substMap3Equality*(·) и истинности $c1[1] = c1[1]$. Шаг ⟨2⟩ 4 также не требует доказательства. Для доказательства шага ⟨2⟩ 3 интересен случай, когда $c1[2][2][e] \neq NONE$.

...
⟨2⟩5 $\forall e \in E1 : \lor c1[2][2][e] = \{NONE\} \lor matchLocks(c1, c1, e) \subseteq c1[2][2][e]$
⟨3⟩1 TAKE $e \in E1$
⟨3⟩2 CASE $c1[2][2][e] = \{NONE\}$
⟨4⟩1 $c1[2][2][e] = \{NONE\}$
PROOF BY ⟨3⟩2
⟨4⟩2 QED BY ⟨4⟩1
⟨3⟩3 CASE c1 [2] [2] [e] ≠ { NONE }
...
⟨3⟩4 QED BY ⟨3⟩2, ⟨3⟩1, ⟨3⟩3

```
...
```

В этом случае требуется доказательство истинности *matchLocks*(*c*1, *c*1, *e*) ⊆ *c*1[2][2][*e*]. Если *c*1[2][2][*e*] ∩ *UU* = {}, то доказательство следует из определения *matchLocks* (·) и истинности c1[2][2][*e*] ⊆ ⊆ *c*1[2][2][*e*]:

```
...
⟨4⟩1 matchLocks(c1, c1, e) ⊆ c1[2][2][e]
    ⟨5⟩1 CASE c1[2][2][e] ∩ UU = {}
        ⟨6⟩1 c1[2][2][e] ⊆ c1[2][2][e]
            PROOF OBVIOUS
        ⟨6⟩2 QED BY ⟨5⟩1, ⟨6⟩1 DEF matchLocks
...
```

Для случая c1[2][2][e] ∩ UU ≠ {} доказательство легко может быть получено следующим образом:

```
...
⟨4⟩1 matchLocks(c1, c1, e) ⊆ c1[2][2][e]
    ⟨5⟩1 CASE c1[2][2][e] ∩ UU = {}
        ...
    ⟨5⟩2 CASE c1[2][2][e] ∩ UU ≠ {}
        ⟨6⟩1 c1 ∈ ClausesSet
            PROOF BY ONLY DEF PoliciesSet
        ⟨6⟩2 {c1[1]} = UU
            PROOF BY ⟨5⟩2, ⟨6⟩1,
                     Clause_ASSM2, UU_ASSM
        ⟨6⟩3 QED BY ⟨5⟩2, ⟨6⟩2
...
```

Транзитивность

В виде теоремы в *TLAPS* свойство формулируется следующим образом:

```
THEOREM Transitivity ≜
          ∀ p1, p2, p3 ∈ PoliciesSet :
            ∧ comparePol(p1, p2)
            ∧ comparePol(p2, p3)
                    ⟹ comparePol(p1, p3)
...
```

Выбрав *p*1, *p*2, *p*3 из *PoliciesSet* и положив: *comparePol*(*p*1,*p*2) ∧ *comparePol* (*p*2,*p*3), необходимо доказать: ∀*c*3 ∈ *p*3: (∃*c*1 ∈ *p*1 : *compareClause*(*c*1,*c*3))[2]:

[2]Доказательство опирается на определения функций: *ComparePol*(·), *CompareClause*(·), *substMap*3*Equality*(·) и *MatchLocks*(·).

```
...
⟨1⟩ USE DEF comparePol, compareClause,
            substMap3Equality, matchLocks
⟨1⟩1 TAKE p1 ∈ PoliciesSet, p2 ∈ PoliciesSet,
          p3 ∈ PoliciesSet
⟨1⟩3 HAVE ∧ comparePol(p1, p2)
          ∧ comparePol(p2, p3)
⟨1⟩4 ∀ c3 ∈ p3 : (∃ c1 ∈ p1 :
                  compareClause(c1, c3))
    ...
⟨1⟩5 QED BY ⟨1⟩4
```

Очевидно, для любого *c*3 ∈ *p*3 можно найти такие *c*2 ∈ *p*2 и *c*1 ∈ *p*1, что справедливым будет: *c*2⊑ *c*3 ∧ *c*1⊑ c2. Тогда для доказательства теоремы достаточно выбрать соответствующие *c*1, *c*2, *c*3 и доказать справедливость *c*1⊑ *c*3 или с учетом определения *substMap*3*Equality*(·):

```
...
⟨2⟩1 TAKE c3 ∈ p3
    ⟨2⟩2 PICK c2 ∈ p2 : compareClause(c2, c3)
        PROOF BY ⟨1⟩3
    ⟨2⟩3 PICK c1 ∈ p1 : compareClause(c1, c2)
        PROOF BY ⟨1⟩3
    ⟨2⟩4 c1 [ 1 ] ∈ UU ∨ c1 [ 1 ] = c3 [ 1 ]
        PROOF BY ⟨2⟩3 , ⟨2⟩2
    ⟨2⟩5 ∀ k ∈ E0 : ∨ c1[2][1][k] = c3[2][1][k]
                    ∨ c3[2][1][k] = {}
        PROOF BY ⟨2⟩3, ⟨2⟩2
    ⟨2⟩6 ∀ e ∈ E1 : ∨ c1[2][2][e] = {NONE}
                    ∨ matchLocks(c1, c3, e)
                        ⊆ c3[2][2][e]
        ...
    ⟨2⟩7 QED BY ⟨2⟩4, ⟨2⟩5, ⟨2⟩6
...
```

Доказательства шагов ⟨2⟩ 4 и ⟨2⟩ 5 следуют из истинности ⟨2⟩ 2, ⟨2⟩ 3 и определений: *compareClause*(·), *substMap*3*Equality*(·). Для доказательства шага ⟨2⟩ 6 интересен случай, когда *c*1[2][2][*e*] ≠ NONE:

```
...
⟨2⟩6 ∀ e ∈ E1 : ∨ c1[2][2][e] = {NONE}
                ∨ matchLocks(c1, c3, e)
                    ⊆ c3[2][2][e]
```

⟨3⟩1 TAKE $e \in E1$
⟨3⟩2 CASE $c1[2][2][e] = \{NONE\}$
PROOF BY ⟨3⟩2
⟨3⟩3 CASE $c1[2][2][e] \neq \{NONE\}$
...
⟨3⟩4 QED BY ⟨3⟩1, ⟨3⟩2, ⟨3⟩3
...

В этом случае требуется доказательство истинности $matchLocks(c1,c3,e) \subseteq c3[2][2][e]$. Если $c1[2][2][e] \cap UU \neq \{\}$, доказательство может быть получено следующим образом:

...
⟨4⟩2 CASE $c1[2][2][e] \cap UU \neq \{\}$
⟨5⟩1 $((c1[2][2][e] \setminus UU) \cup \{c2[1]\}) \subseteq c2[2][2][e]$
PROOF BY ⟨4⟩2, ⟨3⟩3, ⟨2⟩3
⟨5⟩2 $((c1[2][2][e] \setminus UU) \cup \{c3[1]\})$
$\subseteq ((c2[2][2][e] \setminus UU) \cup \{c3[1]\})$
PROOF BY ⟨5⟩1
⟨5⟩3 $c2 \in ClausesSet$
PROOF BY ONLY DEF $PoliciesSet$
⟨5⟩4 $c2[1] \neq NONE$
PROOF BY ⟨5⟩3, U_ASSM, UU_ASSM, $Clause\_ASSM3$
⟨5⟩5 $c2[2][2][e] \neq \{NONE\}$
PROOF BY ⟨5⟩4, ⟨5⟩1
⟨5⟩6 QED BY ⟨5⟩5, ⟨5⟩2, ⟨2⟩2, ⟨2⟩3
⟨4⟩3 CASE $c1[2][2][e] \cap UU = \{\}$
...
⟨4⟩4 QED BY ⟨4⟩3, ⟨4⟩2
...

Для случая $c1[2][2][e] \cap UU = \{\}$ доказательство выводится аналогично.

Антисимметричность

Свойство антисимметричности отношения частичного порядка на множестве политик безопасности *PoliciesSet* можно выразить следующим образом:

THEOREM $Antisymmetry \triangleq$
$\forall p1, p2 \in PoliciesSet :$
$\wedge\ comparePol(p1, p2)$
$\wedge\ comparePol(p2, p1) \implies p1 = p2$

Общая схема доказательства не требует пояснений:

...
⟨1⟩ USE DEF $comparePol$, $compareClause$, $substMap3Equality$, $matchLocks$
⟨1⟩1 TAKE $p1 \in PoliciesSet, p2 \in PoliciesSet$
⟨1⟩2 HAVE $\wedge\ comparePol(p1, p2) \wedge comparePol(p2, p1)$
⟨1⟩4 $p1 = p2$
⟨2⟩1 $\forall c2 \in p2 : \exists c1 \in p1 : c2 = c1$
⟨2⟩2 $\forall c1 \in p1 : \exists c2 \in p2 : c1 = c2$
...
⟨2⟩5 QED BY ⟨2⟩1, ⟨2⟩2
⟨1⟩5 QED BY ⟨1⟩4

Доказательства шагов ⟨2⟩ 1 и ⟨2⟩ 2, очевидно, идентичны.

Рассмотрим далее структуру доказательства шага ⟨2⟩ 1. Предложение политики безопасности c, как уже отмечалось, состоит из трех компонент: предиката $Flow(\cdot) - c[1]$ (представлен параметром предиката), множества непараметрических блокировок и множества параметрических блокировок — $c[2][1]$ и $c[2][2]$ (имеют форму записей, в которых поля соответствуют наименованиям блокировок, значения — параметрам). Поэтому доказательство сводится к проверке эквивалентности соответствующих элементов:

...
⟨2⟩1 $\forall c2 \in p2 : \exists c1 \in p1 : c2 = c1$
...
⟨3⟩7 $c1[1] = c2[1]$
...
⟨3⟩8 $c1[2][1] = c2[2][1]$
...
⟨3⟩9 $c1[2][2] = c2[2][2]$
...
⟨3⟩10 QED BY ⟨3⟩7, ⟨3⟩8, ⟨3⟩9
DEF $PoliciesSet$, $ClausesSet$
...

На начальном этапе возьмем произвольное $c2$ из $p2$ и выберем такие $c1$ из $p1$ и $c3$ из $p2$, что $c1 \sqsubseteq c2$ и $c3 \sqsubseteq c1$. Докажем, что $c3 = c2$:

...
⟨3⟩1 TAKE $c2 \in p2$
⟨3⟩3 PICK $c1 \in p1 : compareClause(c1, c2)$
PROOF BY ⟨1⟩2
⟨3⟩4 PICK $c3 \in p2 : \wedge\ compareClause(c3, c1)$
PROOF BY ⟨1⟩2

```
    ...
⟨3⟩6 c3 = c2
...
```

Поскольку выражение политики безопасности не может содержать двух различных предложений $c1$ и $c2$, таких, что $c1 \sqsubseteq c2$ или $c2 \sqsubseteq c1$, для доказательства шага $\langle 3 \rangle 6$ достаточно доказать, что $c3 \sqsubseteq c2$:

```
...
  ⟨3⟩5 compareClause(c3, c2)
      ...
  ⟨3⟩6 c3 = c2
      PROOF BY ⟨3⟩5  DEF PoliciesSet
...
```

Истинность первых двух конъюнктов выражения *compareClause*($c3$,$c2$) непосредственно следует из истинности шагов: $\langle 3 \rangle$ 3, $\langle 3 \rangle$ 4 ($c1 \sqsubseteq c2 \wedge c3 \sqsubseteq c1$), определения *compareClause*($\cdot$) и предположения о том, что для любого c, принадлежащего *ClausesSet*, справедливо: $c[2][1] \in [E0 \to \{\{NONE\},\{\}\}]$:

```
...
⟨3⟩5 compareClause(c3, c2)
    ⟨4⟩1 c3[1] ∈ UU ∨ c3[1] = c2[1]
        PROOF BY ⟨3⟩3, ⟨3⟩4
    ⟨4⟩2 ∀ e ∈ E0 :
          ∧ ∨ c2[2][1][e] = c3[2][1][e]
            ∨ c2[2][1][e] = {}
          PROOF BY ⟨3⟩3, ⟨3⟩4, Clause_ASSM4
    ⟨4⟩3 ∀ e ∈ E1 : ∨ c3[2][2][e] = {NONE}
                    ∨ matchLocks(c3, c2, e)
                        ⊆ c2[2][2][e]
    ...
    ⟨4⟩4 QED BY ⟨4⟩1, ⟨4⟩2, ⟨4⟩3
...
```

Для доказательства истинности последнего условия выражения *compareClause*($c3$,$c2$) необходимо рассмотреть три случая:

```
...
⟨4⟩3 ∀ e ∈ E1 : ∨ c3[2][2][e] = {NONE}
                ∨ matchLocks(c3, c2, e)
                    ⊆ c2[2][2][e]
    ⟨5⟩1 TAKE e ∈ E1
    ⟨5⟩2 CASE c3[2][2][e] = {NONE}
        PROOF BY ⟨5⟩2
    ⟨5⟩3 CASE c3[2][2][e] ∩ UU = {}
              ∧ c3[2][2][e] ≠ {NONE}
    ...
    ⟨5⟩4 CASE c3[2][2][e] ∩ UU ≠ {}
              ∧ c3[2][2][e] ≠ {NONE}
    ...
    ⟨5⟩5 QED BY ⟨5⟩2, ⟨5⟩3, ⟨5⟩4
                DEF matchLocks
...
```

Доказательство шага $\langle 5 \rangle$ 2 тривиально. Доказательства шагов: $\langle 5 \rangle$ 3 и $\langle 5 \rangle$ 4 в полном объеме для краткости не приводятся. Отметим лишь, они основаны на том факте, что $c1 \sqsubseteq c2 \wedge c3 \sqsubseteq c1$, а также на предположениях: *U_ASSM* и *UU_ASSM*.

Опираясь на определения *compareClause*($\cdot$) и *substMap3Equality*($\cdot$), предположение о том, что множество допустимых связанных переменных состоит из одного элемента (*UU_ASSM*), а также на истинность шагов: $\langle 3 \rangle$ 3, $\langle 3 \rangle$ 4 и $\langle 3 \rangle$ 6 имеем: ($c3[1] = x \vee$ $\vee c3[1] = c1[1]) \wedge (c1[1] = x \vee c1[1] = c2[1]) \wedge c2[1] =$ $= c3[1]$. Отсюда следует истинность шага $\langle 3 \rangle$ 7 — $c1[1] = c2[1]$. Доказательство шага $\langle 3 \rangle$ 8 — $c1[2][1] =$ $= c2[2][1]$ — является также тривиальным.

```
...
⟨3⟩7 c1[1] = c2[1]
    PROOF BY ⟨3⟩3, ⟨3⟩4, ⟨3⟩6, UU_ASSM
⟨3⟩8 c1[2][1] = c2[2][1]
    ⟨4⟩1 ∀ e0 ∈ E0 : c1[2][1][e0] = c2[2][1][e0]
        PROOF BY ⟨3⟩3, ⟨3⟩4, ⟨3⟩6
    ⟨4⟩2 QED BY ⟨4⟩1, Clause_ASSM4
            DEF PoliciesSet
...
```

Остается доказать истинность шага $\langle 3 \rangle$ 9 — $c1[2][2] = c2[2][2]$. Для этого достаточно выбрать произвольное значение $e1$ из $E1$ и доказать: $c1[2][2]$ $[e1] = c2[2][2][e1]$. Общая структура доказательства этого шага может быть представлена как:

```
...
⟨3⟩9 c1[2][2] = c2[2][2]
    ⟨4⟩1 SUFFICES ASSUME NEW e1 ∈ E1
        PROVE c1[2][2][e1] = c2[2][2][e1]
        PROOF BY Clause_ASSM1
            DEF PoliciesSet
    ⟨4⟩2 CASE c2[2][2][e1] = {NONE}
        ...
```

```
⟨4⟩3 CASE  ∧ c2[2][2][e1] ∩ UU = {}
           ∧ c2[2][2][e1] ≠ {NONE}
           ...
⟨4⟩4 CASE  ∧ c2[2][2][e1] ∩ UU ≠ {}
           ∧ c2[2][2][e1] ≠ {NONE}
           ...
⟨4⟩5 QED BY ⟨4⟩2, ⟨4⟩3, ⟨4⟩4
...
```

Если $c2[2][2][e1] = NONE$, то из $c1 \sqsubseteq c2$, определений *compareClause*(·) и *matchLocks*(·), предположений о представлении параметрических блокировок в предложениях политик безопасности, допустимых множествах связанных переменных и актеров следует:

```
...
⟨4⟩2 CASE c2[2][2][e1] = {NONE}
    ⟨5⟩1 ∨ c1[2][2][e1] = {NONE}
         ∨ (c1[2][2][e1] \ UU)
           ∪ {c2[1]}  ⊆ {NONE}
         ∨ c1[2][2][e1] ⊆ {NONE}
         PROOF BY ⟨3⟩3, ⟨4⟩2, Clause_ASSM1,
                  U_ASSM, UU_ASSM
                  DEF matchLocks
    ⟨5⟩2 c1 ∈ ClausesSet ∧ c2 ∈ ClausesSet
         PROOF BY  DEF PoliciesSet
    ⟨5⟩3 QED BY ⟨4⟩2, ⟨5⟩1, ⟨5⟩2,
                Clause_ASSM3, Clause_ASSM1,
                U_ASSM, UU_ASSM
...
```

Вывод доказательства для случая ⟨4⟩ 2 далее тривиален. Для шагов ⟨4⟩ 3 и ⟨4⟩ 4 с целью сохранить краткость изложения далее приводится лишь общая структура вывода, для которого ключевыми являются факты, определенные шагами: ⟨3⟩ 3, ⟨3⟩ 4 и ⟨3⟩ 6.

```
...
⟨4⟩3 CASE  ∧ c2[2][2][e1] ∩ UU = {}
           ∧ c2[2][2][e1] ≠ {NONE}
    ⟨5⟩1 ∨ (c1[2][2][e1] \ UU) ∪ {c2[1]}
           ⊆ c2[2][2][e1]
         ∨ c1[2][2][e1] ⊆ c2[2][2][e1]
         ...
    ⟨5⟩2 CASE c1[2][2][e1] ⊆ c2[2][2][e1]
         ...
    ⟨5⟩3 CASE (c1[2][2][e1] \ UU)
              ∪ {c2[1]} ⊆ c2[2][2][e1]
         ...
    ⟨5⟩4 QED BY ⟨5⟩1, ⟨5⟩2, ⟨5⟩3
⟨4⟩4 CASE  ∧ c2[2][2][e1] ∩ UU ≠ {}
           ∧ c2[2][2][e1] ≠ {NONE}
         ...
    ⟨5⟩9  c2[2][2][e1] ⊆ c1[2][2][e1]
         ...
    ⟨5⟩10 c1[2][2][e1] ⊆ c2[2][2][e1]
         ...
    ⟨5⟩11 QED BY ⟨5⟩9, ⟨5⟩10, UU_ASSM
...
```

Полное доказательство транзитивности, заданного на множестве *PoliciesSet* отношения *comparePol*(·), можно найти в [6].

Решетка

Предположим, что заданная ранее функция $LUB(\cdot)$ действительно является функцией нахождения наименьшей верхней грани двух политик безопасности, и сформулируем в виде теоремы утверждение о том, что множество *PoliciesSet* с заданным на нем отношением частичного порядка *comparePol*(·) представляет собой алгебраическую решетку:

```
THEOREM ParalocksLattice ≜
        ∀ p1, p2 ∈ PoliciesSet :
          ∧ comparePol(p1, LUB(p1, p2))
          ∧ comparePol(p2, LUB(p1, p2))
          ∧ ∀ y ∈ PoliciesSet :
            ∧ comparePol(p1, y)
            ∧ comparePol(p2, y)
              ⟹ comparePol(LUB(p1, p2), y)
```

Отметим, даже на малых наборах данных (две параметрических блокировки, одна непараметрическая блокировка, один конкретный пользователь во множестве *U*, одна связанная переменная во множестве *UU*) проверить справедливость теоремы с использованием инструмента оценки выражений над константами среды *TLA + Toolbox* не удалось. На предварительном этапе пришлось воспользоваться инструментом проигрывания моделей *TLC*. В качестве переменных были выбраны: *Policies2Set* — множество всех возможных пар политик из множества *PoliciesSet*, *curSet* — текущая пара политик. *LUB*(*p*1, *p*2) – возвращает для пары политик *p*1 и *p*2 минимальную верхнюю грань, если она существует.

Ввиду того, что иерархическое доказательство является достаточно объемным, ограничимся лишь

его общей структурой. Этапы доказательства очевидны:

```
⟨1⟩1  TAKE p1, p2 ∈ PoliciesSet
⟨1⟩2  LUB(p1, p2) ∈ PoliciesSet
      PROOF BY LUB_PS
⟨1⟩3  PICK l ∈ PoliciesSet : l = LUB(p1, p2)
      PROOF BY ⟨1⟩2
⟨1⟩4  ∀ l1 ∈ l :
        ∧ ∃ c1 ∈ p1 : compareClause(c1, l1)
        ∧ ∃ c2 ∈ p2 : compareClause(c2, l1)
⟨1⟩5  ∀ y ∈ PoliciesSet :
        comparePol(p1, y) ∧ comparePol(p2, y)
                          ⟹ comparePol(l, y)
⟨1⟩6  QED BY ⟨1⟩3, ⟨1⟩4, ⟨1⟩5  DEF comparePol
```

Доказательство шага ⟨1⟩ 4 строится на том факте, что любое предложение политики *l* является результатом применения некоторой подстановки к блокировкам предложения *c*1 или *c*2 с последующим объединением блокировок этих предложений — *unionCl*(*c*1,*c*2), где *c*1 ∈ *p*1 и *c*2 ∈ *p*2. Для доказательства шага ⟨1⟩ 5 в общем виде необходимо выбрать произвольную политику *y* ∈ *PoliciesSet*, произвольное предложение *y*1 ∈ *y* и доказать: ∃*c*1, *c*2, *l*1 : *c*1 ∈ *p*1 ∧ *c*2 ∈ *p*2 ∧ *l*1 ∈ *l* (*c*1 ⊑ *y*1 ∧ *c*2 ⊑ *y*1 => *l*1 ⊑ *y*1):

```
...
⟨1⟩5 ∀ y ∈ PoliciesSet :
     comparePol(p1, y) ∧ comparePol(p2, y)
                       ⟹ comparePol(l, y)
   ⟨2⟩1 TAKE y ∈ PoliciesSet
   ⟨2⟩2 SUFFICES ∀ y1 ∈ y :
          ( ∧ ∃ c1 ∈ p1 : compareClause(c1, y1)
            ∧ ∃ c2 ∈ p2 : compareClause(c2, y1)) ⟹
                ∃ l1 ∈ l : compareClause(l1, y1)
          PROOF BY ⟨1⟩3  DEF comparePol
   ⟨2⟩3 TAKE y1 ∈ y
   ⟨2⟩4 HAVE ∧ ∃ c1 ∈ p1 : compareClause(c1, y1)
             ∧ ∃ c2 ∈ p2 : compareClause(c2, y1)
   ⟨2⟩5 PICK c1 ∈ p1 : compareClause(c1, y1)
        PROOF BY ⟨2⟩4
   ⟨2⟩6 PICK c2 ∈ p2 : compareClause(c2, y1)
        PROOF BY ⟨2⟩4
...
```

Далее необходимо сгенерировать некоторое предложение *u* = *unionCl*(*c*1, *c*2) и доказать: а) *u* принадлежит политике *l* = *LUB*(*c*1, *c*2) и б) *u* ⊑ *y*1. Условие б) требует декомпозиции на три подусловия в соответствии с определением *compareClause*(·):

```
...
⟨2⟩9 PICK u : u = unionCl ( c1 , c2 )
     PROOF OBVIOUS
...
⟨2⟩13 u ∈ l
      PROOF BY ...
...
⟨2⟩16 u [ 1 ] ∈ UU ∨ u [ 1 ] = y1 [ 1 ]
      PROOF BY ...
...
⟨2⟩24 ∀ k ∈ E0 : ∨ u[2][1][k] = y1[2][1][k]
                 ∨ y1 [ 2 ] [ 1 ] [ k ] = { }
      PROOF BY ...
⟨2⟩25 ∀ e ∈ E1 :
      ∨ u[2][2][e] = {NONE}
      ∨ matchLocks(u, y1, e) ⊆ y1[2][2][e]
      ⟨3⟩1 TAKE e ∈ E1
      ⟨3⟩2 CASE u[2][2][e] = {NONE}
           PROOF BY ⟨3⟩2
      ⟨3⟩3 CASE ∧ u[2][2][e] ≠ {NONE}
                ∧ u[2][2][e] ∩ UU = {}
           ...
      ⟨3⟩4 CASE ∧ u[2][2][e] ≠ {NONE}
                ∧ u[2][2][e] ∩ UU ≠ {}
           ...
      ⟨3⟩5 QED BY ⟨3⟩2, ⟨3⟩3, ⟨3⟩4
⟨2⟩26 QED BY ⟨2⟩16, ⟨2⟩13, ⟨2⟩24, ⟨2⟩25
        DEF compareClause, substMap3Equality
...
```

Здесь доказательство шага ⟨2⟩ 25 является наиболее трудоемким и требует рассмотрения трех случаев. В целом логический вывод доказательств для каждого из них осуществляется в таком же стиле, что и выводы доказательств для аналогичных этапов ранее рассмотренных утверждений – последовательная проверка выражений функции *compareClause*(·), отвечающих за сравнение параметрических блокировок предложений политики безопасности, с опорой на определенный набор обоснованных предположений.

4. ЗАКЛЮЧЕНИЕ

Контроль информационных потоков стоит в одном ряду с некоторыми иными [12], динамично развивающимися направлениями теории языков программирования. Работа является продолжением серии публикаций, посвященных разработанной с участием автора технологии контроля информационных потоков в программном обеспечении автоматизированных информационных систем уровня

предприятия. В отличие от известных подобных технологий, основанных на методах статического и динамического анализа программного обеспечения и требующих от программистов решения дополнительных нетривиальных задач, связанных с разметкой исходного кода, а также интерпретацией результатов анализа, разрабатываемая технология позволяет строго разделить функции написания кода и контроля корректности его логики с учетом специфики предметной области и с привязкой к принятой в системе политике безопасности.

Представленное исследование обладает самостоятельной ценностью, поскольку дает вариант описания политик *Paralocks*, имеющих внедрение в *Java* (проект *Paragon* [13], [11]), в формальном языке *TLA+*, который часто применяется для моделирования программ и проверки их свойств. Таким образом, в работе делается шаг в сторону создания применимой на практике платформы КИП, основанной на методах формальной верификации и проигрывания моделей.

Сама идея проверки безопасности информационных потоков на основе методов формальной верификации ранее выдвигалась и получила определенное признание [14]. Однако в указанной работе проводились только теоретические исследования, не привязанные к конкретному языку программирования, алфавит политик безопасности в ней представлен простой двухуровневой решеткой $\langle\{\text{Hi,Low}\}, \subseteq\rangle$.

Полученные в рамках исследования доказательства формальных свойств решетки выражений, составляющих алфавит политик безопасности, являются важным аспектом разработки механизма контроля информационных потоков.

Практическое значение может иметь предложенный вариант использования формальной логической системы *TLAPS*. Как уже отмечалось, многие объемные доказательства, публикуемые в современных статьях, содержат некорректные утверждения. Человеческий фактор является причиной возникновения простейших ошибок. Например, часто на основе установленного (заданного) факта вида: “$\forall x \in X$ выполняется $P(x)$” делается шаг: “возьмем $x \in X$, для которого выполняется $P(x)$”. Шаг является некорректным, так как требует проверки дополнительного условия “X не является пустым”. Другим ошибочным примером является использование для доказательства истинности некоторого утверждения фактов вида: $x1 =$ if $e1$ then $x2$ else $x3$ и $x1 = x2$. Недостающим условием в данном случае является: $x2 \neq x3$, его проверка может оказаться нетривиальной.

Использование таких средств, как *TLAPS*, безусловно влечет дополнительные трудозатраты, вызванные необходимостью трансляции доказательств в формулы *TLA+*, но в то же время предоставляет дополнительные гарантии корректности получаемых результатов.

5. БЛАГОДАРНОСТИ

Автор выражает признательность Джуру Куковецу (Институт разработки информационных систем, Вена, https://informatics.tuwien.ac.at/orgs/e194), одному из разработчиков инструмента *Apalache* [10], предназначенного для проверки спецификаций *TLA+* на основе символьного выполнения, за важные рекомендации по оптимизации отдельных функций и операторов, образующих *TLA+* семантику используемой версии языка *Paralocks*. Несмотря на то, что на данном этапе полностью адаптировать спецификации *PLIF* для применения *Apalache* не удалось, дальнейшая работа в этом направлении, по мнению автора, имеет перспективу и будет продолжена.

СПИСОК ЛИТЕРАТУРЫ

1. *Девянин П.Н. и др.* Интеграция мандатного и ролевого управления доступом и мандатного контроля целостности в верифицированной иерархической модели безопасности операционной системы // Тр. Института системного программирования РАН. 2020. Т. 32. № 1. С. 7–26.
2. *Lamport L.* Specifying systems: the TLA+ language and tools for hardware and software engineers. 2002.
3. *Denning, Dorothy E.* A lattice model of secure information flow // Communications of the ACM. 1976, № 5. P. 236–243.
4. *Broberg, Niklas, Sands, David.* Paralocks: Role-based information flow control and beyond // Conference Record of the Annual ACM Symposium on Principles of Programming Languages, 2010. P. 431–444.
5. *Myers, C. Andrew, Liskov, Barbara.* A decentralized model for information flow control // ACM SIGOPS Operating Systems Review. 1997. № 5. P. 129–142.
6. *Тимаков А.А.* PLIF. 2021. GitHub. https://github.com/timimin/plif.
7. *Blanchette J. C., Bulwahn L., Nipkow T.* Automatic proof and disproof in Isabelle/HOL //Frontiers of Combining Systems: 8th International Symposium, FroCoS 2011, Saarbrucken, Germany, October 5–7, 2011. Proceedings 8. Springer Berlin Heidelberg, 2011. P. 12–27.
8. *Bonichon R., Delahaye D., Doligez D. Zenon.* An extensible automated theorem prover producing checkable proofs //LPAR. 2007. Т. 4790. P. 151–165.
9. *Lamport L.* How to write a proof //The American mathematical monthly. 1995. Т. 102. № 7. P. 600–608.
10. *Kukovec J., Konnov I.* Type Inference for TLA in Apalache.

11. *Broberg N.* Thesis for the Degree of Doctor of Engineering Practical, Flexible Programming with Information Flow Control. 2011.
12. *Кораблин Ю.П.* Эквивалентность схем программ на основе алгебраического подхода к заданию семантики языков программирования // Russian Technological Journal. 2022. № 10(1). С. 18–27.
13. *Broberg N., van Delft B., Sands D.* Paragon for practical programming with information-flow control //Programming Languages and Systems: 11th Asian Symposium, APLAS 2013, Melbourne, VIC, Australia, December 9–11, 2013. Proceedings 11. Springer International Publishing, 2013. С. 217–232.
14. *Clarkson M. R. et al.* Temporal logics for hyperproperties // Principles of Security and Trust: Third International Conference, POST 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5–13, 2014, Proceedings 3. – Springer Berlin Heidelberg, 2014. С. 265–284.

DESCRIPTION OF PARALOCKS LANGUAGE SEMANTICS IN TLA+

A. A.Timakov[a]

[a]*MIREA – Russian Technological University, Pr. Vernadskogo 78, Moscow, 119454 Russia*

One of the basic aspects of information flow control in applications is security policy language. Such language should allow to define security policies for evaluation environment elements in coherence with higher level access control rules. So the language is expected to be flexible because there may be different access control paradigms implemented on the system level: mandatory, role-based etc. An application may also have its own specific restrictions. Finally it is also desirable that the language support declassification (controlled release of information) during the computation. One of such languages is Paralocks. The research is devoted to the logical semantics of modified version of Paralocks realized in TLA+. Paralocks represents a language basis for the perspective information flow control platform PLIF which is being developed for the analysis of PL/SQL program blocks with author's participation. It includes the proofs of the partial order and lattice defined on the set of security policy expressions.

Keywords: information flow control, security policy language, logical semantics, formal verification, model checking, program units

REFERENCES

1. *Devyanin P.N. and others.* Integration of mandatory and role-based access control and mandatory control in a verified hierarchical security model of LED systems // Proceedings of the Institute of System Programming of the Russian Academy of Sciences. – 2020. – T. 32. – No. 1. – P. 7–26.
2. *Lamport L.* Specifying systems: the TLA+ language and tools for hardware and software engineers. – 2002.
3. *Denning E.D.* A lattice model of secure information flow // Communications of the ACM, 1976, No 5. P. 236–243.
4. *Broberg N., Sands D.* Paralocks: Role-based information flow control and beyond // Conference Record of the Annual ACM Symposium on Principles of Programming Languages, 2010. P. 431–444.
5. *Myers C.A., Liskov B.* A decentralized model for information flow control // ACM SIGOPS Operating Systems Review, 1997, No 5. p. 129–142.
6. *Timakov A.A.* PLIF. 2021. GitHub. https://github.com/timimin/plif.
7. *Blanchette J.C., Bulwahn L., Nipkow T.* Automatic proof and disproof in Isabelle/HOL //Frontiers of Combining Systems: 8th International Symposium, FroCoS 2011, Saarbrucken, Germany, October 5-7, 2011. Proceedings 8. – Springer Berlin Heidelberg, 2011. – C. 12–27.
8. *Bonichon R., Delahaye D., Doligez D.Z.* An extensible automated theorem prover producing checkable proofs // LPAR. – 2007. – T. 4790. – C. 151–165.
9. *Lamport L.* How to write a proof //The American mathematical monthly. – 1995. – T. 102. – №. 7. – C. 600–608.
10. *Kukovec J., Konnov I.* Type Inference for TLA in Apalache.
11. *Broberg N.* Thesis for the Degree of Doctor of Engineering Practical, Flexible Programming with Information Flow Control. 2011.
12. *Korablin Yu.P.* Equivalence of program schemes based on the algebraic approach to specifying the semantics of programming languages // Russian Technological Journal, 2022, 10(1), P. 18–27.
13. *Broberg N., van Delft B., Sands D.* Paragon for practical programming with information-flow control // Programming Languages and Systems: 11th Asian Symposium, APLAS 2013, Melbourne, VIC, Australia, December 9–11, 2013. Proceedings 11. – Springer International Publishing, 2013. – C. 217–232.
14. *Clarkson M.R. et al.* Temporal logics for hyperproperties // Principles of Security and Trust: Third International Conference, POST 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5–13, 2014, Proceedings 3. – Springer Berlin Heidelberg, 2014. – C. 265–284.

ПРОГРАММИРОВАНИЕ, *2024*, *№ 1*, *с. 100–112*

КОМПЬЮТЕРНАЯ ГРАФИКА И ВИЗУАЛИЗАЦИЯ

УДК 004.925.3

ИНТЕРАКТИВНОЕ ВЫЧИСЛЕНИЕ ПРЕЛОМЛЕНИЯ СВЕТА И КАУСТИК С ПРИМЕНЕНИЕМ ГРАФИЧЕСКОГО ПРОЦЕССОРА

С. И. Вяткин[a],[*] (ORCID: 0000-0002-1591-3588),
Б. С. Долговесов[a],[**] (ORCID: 0000-0002-6255-9315)

[a] *Институт автоматики и электрометрии Сибирского отделения Российской академии наук, 630090 Новосибирск, пр. Академика Коптюга, д. 1, Россия*

**E-mail: sivser@mail.ru*
***E-mail: bsd@iae.nsk.su*

Поступила в редакцию: 20.01.2023
После доработки: 27.02.2023
Принята к публикации: 28.02.2023

В то время как современные системы рендеринга эффективны при моделировании сложных световых путей в сложных средах, рендеринг преломляющих каустик по-прежнему занимает много времени. Каустики — это световые узоры, возникающие, когда свет преломляется и отражается от поверхности. Из-за резкого распределения плотности этих зеркальных событий алгоритмы рендеринга в основном полагаются на прямую выборку функции распределения двунаправленного рассеяния на этих поверхностях для построения траекторий. Это требует больших вычислений. Также применяются фотонные карты. Однако есть проблемы, ограничивающие применимость карт каустик. Так как каждый фотон в фотонном буфере должен быть обработан, поэтому приходится выбирать между сильно заниженной дискретизацией каустики и большим снижением скорости, чтобы использовать достаточное количество фотонов для каустики, с целью получения качественных изображений. Сложные зеркальные взаимодействия вызывают передискретизацию в ярких фокальных областях, в то время как другие области карты каустик остаются недостаточно выбранными и шумными. В то же время скорость имеет приоритет над реализмом в большинстве интерактивных приложений. Однако желание улучшить качество графики побудило к разработке различных быстрых приближений для реалистичного освещения. В данной работе представлен комбинированный метод визуализации преломления света и каустик с использованием обратного интегрирования для освещения и прямого интегрирования для просмотра лучей. Используется подход для одновременного распространения света и отслеживания лучей в объеме, и, следовательно, он не требует хранения данных промежуточного объема освещения. В реализации метода расстояние между световыми плоскостями задается равным одному вокселю, что обеспечивает минимум одну выборку на воксель для всех ориентаций. В методе не используются предварительные вычисления, все параметры рендеринга могут быть изменены в интерактивном режиме.
В результате с использованием предлагаемого метода можно создавать правдоподобные приближения сложных явлений, таких как преломления и каустики. Показано влияние преломлений на тень. Демонстрируются сложные световые узоры из-за изогнутой геометрии объектов. Результаты визуализации показывают важность преломления для внешнего вида прозрачных объектов. Например, искажения, вызванные преломлением, и преломление на границе между средами. Разница в показателях преломления между отдельными средами вызывает сложное взаимодействие между светлыми и теневыми областями. Показано, как преломление и каустика улучшают визуализацию функционально заданных объектов, предоставляя дополнительную информацию о форме и местоположении.

Ключевые слова: функционально заданные объекты, функции возмущения, освещение, преломление, отражение, тени, каустика, графический процессор

DOI: 10.31857/S0132347424010086 **EDN:** HJAGHQ

1. ВВЕДЕНИЕ

Одной из сложных проблем является визуализация зеркальных и прозрачных материалов [1]. Для таких материалов свет отражается или преломляется в каждой точке поверхности, поэтому приближения, повышающие скорость за счет размытия, неприемлемы. Эффекты преломления могут сильно влиять на восприятие прозрачных объектов. С другой стороны, скорость имеет приоритет над реализмом в большинстве интерактивных приложений. Желание улучшить качество графики побудило к разработке различных быстрых приближений для реалистичного освещения. Также важным является возможность эффективной реализации методов на GPU с учетом все возрастающей аппаратной поддержки.

В работе [2] представлен метод рендеринга каустики в реальном времени, который использует DirectX Raytracing API и интегрирован в конвейер

рендеринга. Использованы карты прямой каустической видимости и карты обратной каустической видимости, которые создаются для источников света и виртуальной камеры соответственно.

Каустика — важный визуальный эффект, способствующий восприятию реалистичности сцен с использованием отражающих и преломляющих поверхностей [3, 4].

Последние достижения в области рендеринга значительно снижают стоимость глобального освещения. Но даже с аппаратным ускорением сложные световые пути с множеством взаимодействий с глянцем по-прежнему требуют многих вычислений. В статье [5] описан метод, который в сочетании с традиционными световыми картами для рассеянного освещения интерактивно отображает все световые пути в статичных сценах с непрозрачными объектами. Чтобы минимизировать объем памяти, вводится адаптивная параметризация, которая обеспечивает повышенное разрешение для более блестящих поверхностей и областей с более высокой геометрической сложностью.

Эффекты, такие как отражения, создают существенную проблему для алгоритмов визуализации на основе изображений и нейронных сетей. Прежде всего, это изогнутые отражатели, поскольку они приводят к сильно нелинейным потокам отражения при движении камеры. В статье [6] вводится представление на основе точек для вычисления нейронной точечной катакаустики, позволяющей синтезировать сцены с изогнутыми отражателями в новом ракурсе из набора случайно снятых входных фотографий. В основе метода лежит нейронное деформирующее поле, которое моделирует катакаустические траектории отражений. Поэтому сложные зеркальные эффекты могут быть визуализированы с использованием эффективного точечного сплаттинга в сочетании с нейронным визуализатором.

В данной работе, в отличие от метода [7] высокореалистичной визуализации нереального времени, предлагается интерактивное вычисление преломления света и каустик с применением графического процессора.

Целью представленной работы является разработка метода визуализации преломляющих сред с интерактивной частотой кадров без предварительных вычислений.

Главными отличительными особенностями предложенного метода являются:

1) объекты на основе функций возмущения;

2) распространение света в виде параллельных плоскостей.

2. ПОСТАНОВКА ЗАДАЧИ

Визуализация сложных пространственных структур является затратной в вычислительном отношении, так как из-за отсутствия дискретных однородных объектов вклад освещенности должен оцениваться и распространяться в каждой точке пространства.

Хотя разработаны эффективные высококачественные методы интерактивного рендеринга объемных данных с улучшенными эффектами освещения, с целью уменьшения вычислений пренебрегают эффектом рефракции. То есть изменением направления распространения света из-за различий в скорости света между передающими средами. Так как эффекты преломления игнорируются, в результате лучи рассматриваются как прямые. А это значительное упрощение.

В то время как рефракция отвечает за широкий спектр оптических явлений, которые сильно влияют на внешний вид полупрозрачных материалов. Например, световой луч внутри стеклянной линзы, окруженной воздухом, распространяется с более высокой скоростью. Каустики представляют собой сложные узоры сфокусированного света, окруженного затененными областями из-за преломления изогнутыми объектами.

То есть скорость света является непрерывно изменяющимся свойством, вызывающим изменение направления в каждой точке пространства, и эффекты преломления сильно влияют на восприятие прозрачных объектов. Преломление вносит важные дополнительные эффекты форм при визуализации.

Физически корректный рендеринг выполняется с помощью таких методов, как описано в [8]. Однако они медленные и результирующие изображения получаются с заметными артефактами.

Для целей визуализации нам требуется интерактивная производительность, позволяющая выполнять такие операции, как смена положения камеры, модификация передаточной функции и клиппирование.

Поскольку искривленные световые лучи значительно усложняют выборку на регулярной сетке, необходимо избежать явного формирования объема освещения, так как для вычисления преломления такая дискретизация проблематична. Кроме этого, необходимо указывать источник света относительно

положения камеры, чтобы его не перенастраивать всякий раз, когда сцена поворачивается.

В этой статье представлен подход визуализации преломляющего объемного освещения, включающий каустики, с интерактивной частотой кадров. С помощью совмещения вычислений света и распространения лучей метод не требует запоминания информации освещенности и предварительных вычислений, что позволяет полностью динамично манипулировать всеми параметрами рендеринга. В результате не требуется хранения данных промежуточного объема освещения. Параметры, такие как положение света и передаточная функция, могут изменяться в интерактивном режиме без снижения производительности.

3. ОПИСАНИЕ МЕТОДА

3.1. Объекты на основе функций возмущения

Задание объектов (свободных форм) происходит с применением поверхностей второго порядка — квадрик и функций возмущения второго порядка [9]:

$$F'(x,y,z) = F(x,y,z) + \sum_{i=1}^{N} f_i R_i(x,y,z), \qquad (1)$$

где $F'(x,y,z)$ — функция свободной формы; $F(x,y,z)$ — функция базовой квадрики; f_i — форм-фактор; $R_i(x,y,z)$ — функция возмущения, $i = 1 \ldots N$,

$$R_i(x,y,z) = \begin{cases} Q_i^3(x,y,z), & \text{если } Q_i(x,y,z) \ge 0; \\ 0, & \text{если } Q_i(x,y,z) < 0, \end{cases} \qquad (2)$$

где $Q(x, y, z)$ — возмущающая квадрика.

Объект при функциональном задании целиком определен с помощью вещественной непрерывной описывающей функции трех переменных (x, y, z) в виде $F'(x,y,z) \ge 0$. Объекты рассматриваются как замкнутые подмножества евклидова пространства E^3, определяемые описывающей функцией $F'(x,y,z) \ge 0$, где F — непрерывная вещественная функция и (x, y, z) — задаваемая координатными переменными точка в E^3. Здесь $F'(x,y,z) > 0$ задает точки внутри объекта, $F'(x,y,z) = 0$ — точки на границе и $F'(x,y,z) < 0$ — точки, лежащие снаружи и не принадлежащие объекту.

Решая описывающую функцию в виде неравенства $F'(x,y,z) \ge 0$, мы можем визуализировать не только поверхность, но и внутреннюю структуру объекта [10].

Важной частью является эффективное нахождение первого пересечения луча с поверхностью. Данная задача напоминает методы визуализации объемных данных, которые часто применяются, например, в томографии. В подобных методах задана функция плотности. Основным отличием является то, что в подобных подходах мы имеем дело с дискретными данными. А в нашем случае есть аналитически заданная функция плотности. Это позволяет более эффективно осуществлять поиск пересечения луча с поверхностью [11]. Так как условные переходы — это дорогие операции для геометрического процессора, чтобы не использовать рекурсию, возмущения задаются явно.

3.2. Уравнение переноса излучения

Уравнение в дифференциальной форме имеет следующий вид [12]:

$$(\omega \cdot \nabla) L(\vec{x},\omega) = -\mu_t(\vec{x}) L(\vec{x},\omega) + \\ + \mu_a(\vec{x}) L_e(\vec{x},\omega) + \mu_s(\vec{x}) \int_{S^2} f_p(\omega,\bar{\omega}) L(\vec{x},\bar{\omega}) d\bar{\omega}, \qquad (3)$$

где $L(\vec{x},\omega)$ — излучение, параметризованное точкой $\vec{x}$ и направлением движения ω:

$\mu_t(\vec{x})L(\vec{x},\omega)$ — потери, вызванные поглощением и рассеянием;

$\mu_a(\vec{x}) L_e(\vec{x},\omega)$ и $\mu_s(\vec{x}) \int_{S^2} f_p(\omega,\bar{\omega}) L(\vec{x},\bar{\omega}) d\bar{\omega}$ —

выигрыши, обусловленные излучением $L_e(\vec{x},\omega)$ и рассеиванием соответственно; $f_p(\omega,\bar{\omega})$ — функция фазы, количественно определяет направленную плотность рассеянного света (направление движения $\omega \in S^2$); S^2 — единичная сфера.

С помощью решения интеграла Дирака получается интегральная форма уравнения переноса излучения, включающая прямое рассеяние:

$$L(\vec{x},\omega) = \int_0^{\infty} \exp\left(-\int_0^t \vec{\mu}_t(\vec{x}_s) + \vec{\mu}_n(\vec{x}_s) ds \right) \times \\ \times \left[\vec{\mu}_a(\vec{x}_t) L_e(\vec{x}_t,\omega) + \vec{\mu}_s(\vec{x}_t) L_e(\vec{x}_t,\omega) \right] dt, \qquad (4)$$

где $\vec{x}_t = \vec{x} - t\omega$ и $\vec{x}_s = \vec{x} - s\omega$.

3.3. Подповерхностное рассеяние

Количество энергии, проходящей в нижние слои поверхности, равно [13]:

$$E = (1 - F(\alpha_i))(1 - F(\alpha_o)), \qquad (5)$$

где F — коэффициент отражения Френеля для отражения. Для пластика коэффициент отражения зависит от коэффициента отражения ρ_d диффузного слоя [12]:

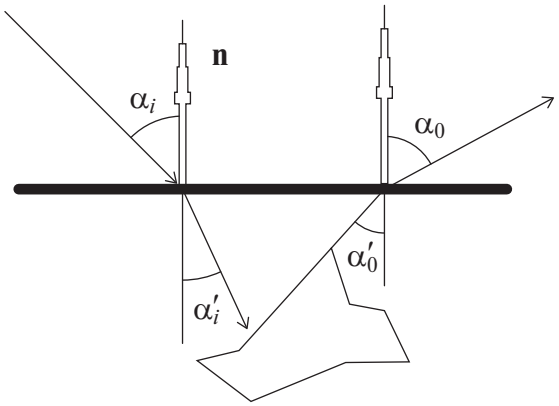


Рис. 1. Подповерхностное рассеяние света.

$$\rho_p = E\frac{\rho_d}{\pi}\left(1-\frac{\rho_d}{\pi}F_{dr}\left(\frac{1}{\sigma}\right)\right)^{-1}, \quad (6)$$

где F_{dr} — коэффициент диффузного отражения Френеля.

Коэффициент диффузного отражения Френеля соответствует интегралу коэффициента отражения Френеля, умноженному на косинус угла по входящим направлениям.

Для подповерхностного слоя это зависит от альбедо a рассеивающего материала [13]:

$$\rho_{subsurf} = E\frac{a}{4\pi}\left(\frac{1}{\cos\alpha_i' + \cos\alpha_o'}\right) + E\frac{a}{2\pi}e^{-\sqrt{3(1-a)}}\left(e^{-\frac{4}{3}\sqrt{3(1-a)}\left(\frac{1-F_{dr}}{1+F_{dr}}\right)}\right), \quad (7)$$

где α_i' и α_o' — углы преломления лучей с нормалью к поверхности (рис. 1).

Таким образом, при преломлении света лучи как от источника освещения, так и лучи, видимые наблюдателем, изменяют направление в зависимости от свойств материала.

3.4. Передача света

Перенос света в общем виде описывается с использованием интеграла пути [14]. Интенсивность I определяется в виде интеграла

$$I = \int_\Omega f(\bar{z})d\mu(\bar{z}), \quad (8)$$

где Ω — пространство путей переноса света $\bar{z}, f(\bar{z})$ — пропускная способность пути.

Интеграл по траектории аппроксимируется оценкой

$$I \approx \frac{1}{p}\sum_{i=1}^{p}\frac{f(\bar{z}_i)}{d(\bar{z}_i)}, \quad (9)$$

где p — вклад путей распространения света, отобранных с плотностью вероятности $d(\bar{z})$.

Пути распространения света $\bar{z}=\bar{x}_n\bar{y}_m$ разлагаются на фотонные субпути $\bar{x}_n$ и субпути камеры $\bar{y}_m$.

Вычисление плотности выполняется в паре вершин $\bar{x}_n\bar{y}_m$.

3.4.1. Вычисление плотности

Приближенное вычисление плотности полной пропускной способности пути

$$\frac{f(\bar{z})}{d(\bar{z})} \approx C(\varpi_n)C(\bar{t}_{n-1})\langle D\rangle^{n,m}C(\bar{s}_{m-1})C(\varpi_m'), \quad (10)$$

здесь $\langle D\rangle^{n,m}$ — оценка плотности в паре вершин $\bar{x}_n$ и $\bar{y}_m$.

3.4.2. Вычисление излучения

Введем трехмерные оценки плотности. Метод может быть повторно применен к еще более высоким измерениям.

Оценка плотности для лучей фотонов вычисляется следующим образом [15, 16]:

$$\langle D\rangle_r^{n,m} = \int_{s_{m-}}^{s_{m+}} f(\bar{t}_n)\left\{K_2(x_n,\bar{y}_m)f_\omega^{n,m}\right\}f(s)ds, \quad (11)$$

где $\bar{t}_n, \bar{y}_m$ — функции интегрирующей переменной s, K_2 — 2D-ядро размытия.

Интегральные границы определяются пересечением луча камеры и фотонного луча.

Лучи объединяются в непрерывную фотонную плоскость с вкладом

$$\langle D\rangle_{pl}^{n-1,m} = \int_{s_{m-}}^{s_{m+}} f(\bar{t}_{n-1})f(\bar{t}_n)\times \left\{\frac{K_1(x_n,\bar{y}_m)}{J_{pl}^{n-1,n}}f_\omega^{n,m}\right\}f(s)ds, \quad (12)$$

где $J_{pl}^{n-1,n} = \|\omega_{n-1}\times\omega_n\|$ — якобиан.

Вставим и расширим оценку (12) в уравнение (10), для $\langle D\rangle_{pl}^{n-1,m}$:

$$\frac{f(t_{n-2})}{d(t_{n-2})}\int_{s_{m-}}^{s_{m+}} f(\bar{t}_{n-1})f(\bar{t}_n)\left\{\frac{K_1(x_n,\bar{y}_m)}{J_{pl}^{n-1,n}}f_\omega^{n,m}\right\}f(s)ds, \quad (13)$$

Заменим выборку расстояния вдоль t_{n-2} детерминированной плоскостью.

$K_1(x_n,y_m) = u^{-1}$ — однородное ядро размытия. $\bar{u} = (\omega_{n-1}\times\omega_n)/\|\omega_{n-1}\times\omega_n\|$ — нормаль к плоскости (рис. 2).

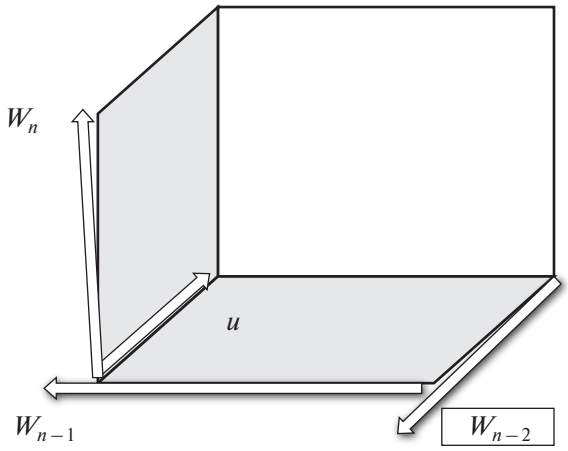


Рис. 2. Вклад плоскостей в объем.

Вклад всех плоскостей в объем составит:

$$\sum_{i=0} f(t_{n-2}^{(i)}) \int_{s_{m-}}^{s_{m+}} f(\bar{t}_{n-1}) f(\bar{t}_n) \left\{ \frac{u^{-1}}{J_{pl}^{n-1,n}} f_{\omega}^{n,m} \right\} f(s)\, ds, \quad (14)$$

Взятие предела даст непрерывный объем фотонов.

Однако подходы нереального времени [15, 16] не поддерживают рефракцию. Чтобы учесть преломление, модифицируем метод фотонных плоскостей. Искривленные световые лучи значительно усложняют выборку на регулярной сетке. Для визуализации в нереальном времени сцен с преломляющими средами используются методы вычисления освещенности с помощью фотонных карт. Эти методы требуют очень много вычислений, поскольку необходимо огромное количество фотонов для качественного изображения, то есть чтобы изображение казалось непрерывным [17].

Вместо того чтобы испускать отдельные фотоны, в предлагаемом методе распространяется свет плоскость за плоскостью, одновременно с вычислением лучей от наблюдателя. Объем пересекается плоскостями, параллельными плоскости изображения.

При визуализации желательно указывать источник света относительно положения камеры, чтобы его не перенастраивать при геометрических преобразованиях.

Для этого мы использует один удаленный источник света, расположенный в том же полушарии, что и наблюдатель.

Свет распространяется в виде параллельных плоскостей, с сохранением его направления и излучения в 2D-буферах. Для каждой точки в световом буфере происходит интегрирование в обратном направлении вдоль светового пути. Вычисляется пересечение луча, исходящего из текущей точки на световой плоскости, с предыдущей плоскостью света в отрицательном направлении света. Используется билинейная интерполяция для получения значений яркости и направления в точке пересечения.

Нарезка плоскостей происходит, как описано в работе [18].

Входящее излучение смешивается с вкладами частиц и среды между двумя плоскостями. Направление света обновляется на основе градиента поля показателя преломления. Для каждого положения пикселя в текущем световом буфере вычисляем:

$$L_j = L_{j-1} I_j (1 - \alpha) c, \quad (15)$$

здесь L_j — новый цвет света, L_{j-1} — фильтрованный предыдущий цвет света, I_j – интенсивность света, α — непрозрачность, c — средний цвет между плоскостями $j-1$ и j.

Направление света вычисляется следующим образом:

$$dl_j = dl_{j-1} + \Delta p \lambda, \quad (16)$$

здесь dl_j — новое направления света, dl_{j-1} — фильтрованное предыдущее направление света, Δp — расстояние между световыми плоскостями, λ — показатель преломления.

Необходимо не только распространять и рассеивать свет, но и находить каустики.

Используем следующее уравнение интенсивности:

$$I_j = I_{j-1} A_{j-1} / A_j, \quad (17)$$

здесь I_j — интенсивность света на элементе площадью A_j, передающаяся из элемента с площадью A_{j-1} с интенсивностью I_{j-1}.

Необходимо теперь вычислить требуемые области. Так как графические процессоры имеют встроенные функции для вычисления частных производных, оптимальным решением является аппроксимация областей с использованием частных производных в пространстве экрана. Для этого вычисляются пересечения светового луча с текущей и предыдущей плоскостями.

Области A_j и A_{j-1} (17) задаются:

$$A_j \approx \left| \frac{d}{dx} p_i \right| \left| \frac{d}{dy} p_i \right|, \quad (18)$$

здесь p_i — точка пересечения луча и соответствующей плоскости.

Эта поправка затем используется в уравнении (17) для восстановления каустики.

Входящее рассеянное освещение определяется путем поиска значения в световом буфере в текущем местоположении луча обзора. Вычисляется зеркальная составляющая аналитически с использованием коэффициента Кука–Торренса в модели BRDF [19]. Зеркальная составляющая вычисляется с помощью направления света, луча обзора и нормали. С помощью нормали вычисляется отражательная способность границы раздела на основе относительного показателя преломления между текущей и предыдущей плоскостями освещения.

Отражения и передача света вычисляются следующим образом.

Цвет:

$$C_j = C_{j-1} + (1 - \alpha_{j-1}) \cdot c_{j-1} \cdot (\varsigma \cdot \sigma \cdot i_d + i_s), \quad (19)$$

непрозрачность:

$$\alpha_j = \alpha_{j-1} + (1 - \alpha_{j-1}) \cdot \varsigma, \quad (20)$$

среднее значение цвета:

$$c_j = c_{j-1} \cdot \upsilon, \quad (21)$$

где C_j — новое значение цвета частиц, C_{j-1} — предыдущее значение цвета частиц, α_j — новое значение непрозрачности, α_{j-1} — предыдущее значение непрозрачности, c_j — новое среднее значение цвета частиц, c_{j-1} — предыдущее среднее значение цвета частиц, i_d — вклад рассеянного освещения, i_s — вклад зеркального освещения, ς, σ, υ — вклады сегмента луча.

Замечание. В случае таких материалов, как стекло, вклад зеркального освещения не умножается на непрозрачность, так как максимально пропускающие части объема без вклада непрозрачности могут иметь зеркальные отражения.

На рис. 3 показан алгоритм вычислений.

Используются следующие буферы: буфер освещения, буфер направления света, буферы положения и направления луча обзора, буферы цвета и окружающей среды (рис. 3).

Шаг 1. Буферы инициализируются с помощью шейдера. Шейдер инициализации очищает буферы цвета и окружающей среды, вычисляет направление луча от наблюдателя и начальные положения. Инициализируются буферы освещения, направления света с цветом и направления источника света. Сохраняются координаты направления лучей и положения в пространстве.

Шаг 2. Клиппирование функционально заданного объекта световой плоскостью, получение среза. Вычисляется контур $F'(x, y, z) = 0$ — точки на гра-

Рис. 3. Алгоритм визуализации.

нице (воксели с нормалями) и $F'(x,y,z) > 0$ — точки внутри объекта. Алгоритм приведен для одного среза, для остальных он повторяется.

Распространение света и лучей, исходящих от наблюдателя, выполняются с помощью программы фрагментного шейдера для каждой световой (клиппирующей) плоскости.

Шаг 3. Распространение света от предыдущей плоскости выполняется с использованием обратного интегрирования (формулы (15) и (16)). Моделируется направленный источник света.

Шаг 4. Фильтрация цвета и направления входящего света необходима, так как затенение будет становиться все более рассеянным по мере продвижения света от предыдущей плоскости. Используется ядро фильтрации, как описано в [20]. Это же ядро применяется для фильтрации направления лучей, чтобы устранить артефакты, вызванные обратным отображением.

Отфильтрованные цвет L_{j-1} и направление света dl_{j-1} затем используются в уравнениях (15) и (16).

Шаг 5. Коррекция интенсивности света выполняется с использованием конечно-разностных функций графического процессора, то есть частных производных в пространстве экрана от пересечения светового луча с текущей и предыдущей клиппирующими плоскостями, поскольку современные графические процессоры имеют встроенные функции для вычисления этих производных.

Вклад объема в текущей плоскости определяется с использованием таблиц предварительного интегрирования для цвета среды и частиц, непрозрачности и используется для вычисления нового цвета.

Шаг 6. Вычисление нового цвета и показателя градиента преломления, который используется для обновления направления света. Новый цвет и направление света сохраняются в буферах.

Шаг 7. Вычисление зеркального отражения и каустики. В отличие от распространения света, просмотр лучей от наблюдателя происходит с использованием прямого интегрирования. Сохраняется их положение, направление и накопленный цвет в наборе 2D-буферов. Для положения и направления лучей от наблюдателя предыдущие значения, а также накопленный цвет, непрозрачность и цвет окружающей среды извлекаются из соответствующих буферов. Входящее освещение извлекается из светового буфера и вычисляется зеркальное затенение с использованием градиента объема.

Шаг 8. Вклады текущего сегмента луча извлекаются из таблиц предварительной интеграции и используются для вычисления новой непрозрачности, цвета частиц и среды. Направление луча обновляется с использованием градиента показателя преломления. Определяется пересечение луча со следующей световой плоскостью.

Шаг 9. Новые значения сохраняются в буферах.

Для отображения изображений использован OpenGL 4.5 с многослойными вложениями буфера кадра (в режиме Flip-Flop), синхронизацией и т.д. Геометрический шейдер указывает, какой слой вложения записывается. Режим Flip-Flop означает, что можно одновременно в одну часть памяти записывать, а из другой считывать данные. Используется также встроенный контроль за записями текстур и кешированием.

4. РЕЗУЛЬТАТЫ

Тестирование производилось на компьютере с процессором Intel Core i7-4930K OEM и графическом процессоре GeForce GTX 980 Ti. Изображения были визуализированы с разрешением 1024 × 768 пикселей.

На рис. 4а показана каустика от стеклянного эллипсоида на поверхность с диффузно-зеркальной составляющей $\alpha = 0{,}1$. Диффузно-зеркальная составляющая поверхности изменяется от 1 (диффузная) до 0 (зеркальная). Параметр α определяет, является ли поверхность диффузной или зеркальной. Такая модель была представлена Шликом в [21]. Эта модель проста и обладает полезным свойством, заключающимся в том, что она обеспечивает непрерывный переход от ламбертовского отражения к глянцевому зеркальному отражению. На рис. 4б показан тот же объект, визуализированный с помощью метода [7], при этом некоторые детали внутренней структуры эллипсоида упущены.

На рис. 5а (слева) показан объект с преломлением и сочетанием пропускающих и отражающих свойств материала. Рисунок демонстрирует сложные световые узоры каустики из-за изогнутой геометрии объекта. На рис. 5б показан тот же объект, визуализированный с помощью метода [7], который демонстрирует меньшее изгибание световых лучей.

Рис. 6 демонстрирует важность преломления для внешнего вида прозрачных структур. В прозрачном сосуде содержится жидкость. Задаем различные показатели преломления для стекла и жидкости. Увеличиваем непрозрачность жидкости. Жидкость теперь поглощает большую часть поступающего света, в результате чего образуется темная тень, а внутри — каустика. Кроме того, становится виден рисунок, вызванный преломлением света.

Рис. 4. Слева: прозрачный эллипсоид с каустикой. Справа: тот же объект, визуализированный с помощью метода [7].

Рис. 5. Слева: прозрачный объект с каустикой. Справа: тот же объект, визуализированный с помощью метода [7].

На рис. 6б показан тот же объект, визуализированный с помощью метода [7], на острых краях сосуда видны артефакты, которые уменьшаются с увеличением количества выборок.

В табл. 1 приведены показатели производительности алгоритма. Время указано в миллисекундах.

Как видно из таблицы, визуализация осуществляется в интерактивном режиме. Это стало возможно благодаря некоторым упрощениям.

Таблица 1. Время визуализации

Изображение	Разрешение по глубине в воксе-лях	Время визуали-зации. Разре-шение 640×480	Время визуали-зации. Разре-шение 800×600	Время визуали-зации. Разреше-ние 1024×768
Рис. 4а	256	15 мс	26 мс	56 мс
Рис. 5а	512	60 мс	109 мс	176 мс
Рис. 6а	800	65 мс	126 мс	203 мс

В табл. 2 приведены времена вычисления этих изображений с помощью метода [7]. Время указано в минутах.

Таблица 2. Время визуализации метода [7]

Изображение	Время визуализации. Разрешение 1024×768
Рис. 4б	1.2 мин.
Рис. 5б	3.3 мин.
Рис. 6б	4.8 мин.

В табл. 1 показаны данные для одного источника света, расположенного в том же полушарии, что и камера. Каждый новый источник освещения требует дополнительных вычислений, которые растут линейно. Рост дополнительного времени вычислений (ордината, в миллисекундах) показан на рис. 7.

Кроме этого, наш подход рассматривает показатель преломления как скалярную величину, игнорируя тот факт, что физическое преломление также

Рис. 6. Слева: прозрачный сосуд с жидкостью и каустикой. Справа: тот же объект, визуализированный с помощью метода [7].

Рис. 7. Дополнительное время вычислений.

влияет на длину волны света. Это означает, что наш метод, следовательно, не способен улавливать явления, зависящие от длины волны. Можно было бы решить эту проблему, указав отдельные показатели преломления для спектральных образцов за счет создания их спецификации, или спектральный показатель преломления может быть получен в виде функции цвета среды. Но это требует также дополнительных вычислений.

Тем не менее предлагаемый метод имеет преимущества перед методами, использующими фотонные карты.

В работе [17] отмечено, что если важно уметь получать приближенное решение быстро, то можно использовать различные методы на основе фотонных карт [22]. Однако эти методы имеют недостатки, например, надо выбирать между сильно заниженной дискретизацией каустики и большим снижением скорости, чтобы использовать достаточное количество фотонов для каустики, чтобы получить качественные изображения.

В обзорной статье [23] представлен широкий ряд как интерактивных методов, использующих фотонные карты, так и методы нереального времени, которые включают в себя расширенные световые эффекты. Хотя было разработано много эффективных методов, они либо не поддерживают рефракцию [24], либо являются подходами нереального времени [25].

В статье [26] дан обзор прогрессивных методов переноса света, в том числе рассмотрены несколько подходов на основе фотонных карт, с применением графических процессоров. В ней исследованы три прогрессивных алгоритма, основанных на отображении фотонов, а именно, прогрессивное отображение фотонов (PPM), стохастическое прогрессивное отображение фотонов (SPPM) и прогрессивное двунаправленное отображение фотонов (PBPM). Общим элементом всех алгоритмов является оценка плотности, основанная на сборе фотонов в области с определенным радиусом от точки запроса. Рассмотрены три структуры данных, предназначенные для ускорения этого процесса: kD-дерево, полная хэш-сетка и стохастическая хэш-сетка. Авторы отмечают, что стандартная оценка плотности на глянцевых поверхностях дает результаты с шумом, который можно лишь уменьшить разными способами, но не устранить.

Поскольку и генерация фотонов, и запросы выполняются на графическом процессоре, важно, чтобы построение структуры данных также обрабатывалось графическим процессором, чтобы ограничить передачу данных из CPU в GPU.

Существуют следующие проблемы. Когда радиус значительно меньше размера ячейки, ячейка может содержать много фотонов, которые будут находиться

за пределами радиуса запроса и отбрасываться. Или ячейка может содержать много фотонов, которые будут находиться за пределами диапазона запроса. Другая проблема связана с неравномерным количеством фотонов в каждой ячейке — например, поверхности, расположенные вблизи источников света, могут иметь значительно более высокую плотность фотонов. Это означает, что количество фотонов, обработанных в каждом запросе, может значительно отличаться, снижая эффективность графического процессора. Есть проблемы балансировки. На графическом процессоре все потоки ожидают, пока каждый поток не обработает все фотоны в своей текущей ячейке, прежде чем обрабатывать следующую ячейку. Это означает, что даже если общее количество фотонов одинаково во всех потоках, некоторые потоки не работают, в то время как другие все еще обрабатывают свои фотоны из данной ячейки.

В статье также анализируются несколько тестовых сцен. Показан объект с зеркальными отражениями и преломлениями, состоящий из 918000 треугольников. При этом используются 175000 фотонов, однако не учитывается перенос преломляющего излучения, которое моделирует непрерывное изгибание световых лучей для физически корректного рендеринга участвующих сред с изменяющимся в пространстве показателем преломления, что является одной из важных возможностей предлагаемого нами метода.

5. ЗАКЛЮЧЕНИЕ

Представлен метод визуализации функционально заданных объектов с эффектами преломления, отражения света и каустики в интерактивном режиме. Использован комбинированный подход передачи света и отслеживания лучей. Для этого применены обратное интегрирование для освещения и прямое интегрирование для отслеживания лучей. Благодаря этому получаем динамическое объемное освещение с мягкими тенями и расширенными эффектами, такими как каустика. В методе не используются предварительные вычисления, что является положительной характеристикой.

ИСТОЧНИК ФИНАНСИРОВАНИЯ

Работа выполнена при поддержке Министерства науки и высшего образования в рамках выполнения работ по Государственному заданию № 1023032300221-9-1.2.1 в ИАиЭ СО РАН.

СПИСОК ЛИТЕРАТУРЫ

1. *Wang X., Zhang R.* Rendering Transparent Objects with Caustics using Real-Time Ray Tracing //March 2021Computers & Graphics. 2021. V. 96. № 3. DOI:10.1016/j.cag.2021.03.003
2. *Komarov E., Zhdanov D., Zhdanov A.* Rendering the Real-Time Caustics with DirectX Raytracing // 31th International Conference on Computer Graphics and Vision. (Graphicon -2021). С. 36–47. DOI:10.20948/graphicon-2021-3027-36-47
3. *Grittmann P., Pérard-Gayot A., Slusallek P., Krivanek J.* Efficient caustic rendering with lightweight photon mapping // Computer Graphics Forum. 2018. V. 37. № 4. P. 133-142. DOI:10.1111/cgf.13481
4. *Muller T., Gross M., Novak J.* Practical path guiding for efficient light-transport simulation // Computer Graphics Forum. 2017. V. 36. № 4. P. 91–100. DOI:10.1111/cgf.13227
5. *Rodriguez S., Leimkuhler T., Prakash S., Wyman C., Shirley P., Drettakis G.* Glossy Probe Reprojection for Interactive Global Illumination December 2020ACM Transactions on Graphics 39(6) P. 1-16. DOI:10.1145/3414685.3417823
6. *Kopanas G., Leimkuhler T., Rainer G., Jambon C., Drettakis G.* Neural Point Catacaustics for Novel-View Synthesis of Reflections // ACM Transactions on Graphics. 2022. V. 41. № 6. P. 1–15. DOI:10.1145/3550454.3555497
7. *Вяткин С.И., Долговесов Б.С.* Высокореалистичная визуализация каустик и шероховатых поверхностей // *Программирование*. 2022. № 5. С. 27–36. DOI: 10.31857/S0132347422050065
8. *Haber J., Magnor M., Seidel H.P.* Physically-based Simulation of Twilight Phenomena // ACM Transactions on Graphics. V. 24. № 4. 2005. P. 1353–1373. DOI:10.1145/1095878.1095884
9. *Вяткин С.И., Долговесов Б.С.* Физически корректная визуализация функционально заданных объектов // Автометрия. 2022. Т. 58. № 3. С. 98–105. DOI: 10.15372/AUT20220311
10. *Вяткин С.И., Долговесов Б.С.* Метод визуализации мультиобъемных данных и функционально заданных поверхностей с применением графических процессоров // Автометрия. 2021. Т. 57. № 2. С. 32–40. DOI: 10.15372/AUT20210204
11. *Вяткин С.И.* Метод бинарного поиска элементов изображения функционально заданных объектов с применением графических акселераторов // Автометрия. Т. 50. № 6. 2014. С. 89–96.
12. *Galtier M., Blanco S., Caliot C. et al.* Integral Formulation of Null Collision Monte Carlo Algorithms // Journal of Quantitative Spectroscopy and Radiative Transfer. 2013. V. 125. P. 57–68. DOI: 10.1016/j.jqsrt.2013.04.001
13. *Jensen H. W., Marschner S. R., Levoy M.* A practical model for subsurface light transport // SIGGRAPH ‘01

In: Proceedings of the 28th annual conference on Computer graphics and interactive techniques. ACM. New York, August, 2001, P. 511–518. DOI:10.1145/383259.383319
14. *Loube G., Zeltner T., Holzschuch N.* Slope-space integrals for specular next event estimation// ACM Transactions on Graphics. 2020. V. 39. № 6. P. 1–13. DOI:10.1145/3414685.3417811
15. *Deng X., Jiao S., Bitterli B., Jarosz W.* Photon surfaces for robust, unbiased volumetric density estimation", July 2019ACM Transactions on Graphics. 2019. V. 38, № 4. P. 1–12. DOI:10.1145/3306346.3323041
16. *Bitterli B., Jarosz W.* Beyond points and beams: Higher dimensional photon samples for volumetric light transport //ACM Transactions on Graphics (TOG). 2017. T. 36. №. 4. P. 1-12.
17. *Фролов В.А., Волобой А.Г., Ершов С.В., Галактионов В.А.* Современное состояние методов расчета глобальной освещенности в задачах реалистичной компьютерной графики // Труды Института системного программирования РАН. 2021. Т. 33. № 2. С. 7–48. DOI: https://doi.org/10.15514/ISPRAS-2021-33(2)-1
18. *Вяткин С.И., Долговесов Б.С.* Функционально заданные модели для аддитивного производства// Исследования. Инновации. Практика. 2022. № 4(4). С. 16–25. DOI: 10.18411/iip -08-2022-04
19. *Lavoue G., Bonneel N., Farrugia J.-P., Soler C.* Perceptual quality of BRDF approximations: Dataset and metrics // Comp. Graph. Forum. 2021. V. 40. № 2. P. 327-338. DOI: 10.1111/cgf.142636.
20. *Вяткин С.И., Долговесов Б.С.* Сглаживание функционально заданных объектов в сценах с глобальной освещенностью // Journal of Advanced Research in Technical Science. 2022. № 30. P. 96–103. DOI: 10.26160/2474-5901-2022-30-96-103.
21. *Schlick C.* A Customizable Reflectance Model for Everyday Rendering // In proceedings of four Eurographics Workshop on Rendering. 1993. P. 73–84. Corpus ID: 18967314
22. *Jensen H.W.* Realistic image synthesis using photon mapping. Publisher: A. K. Peters, Ltd.63 South Avenue Natick, MA United States. 2001. 181 p. ISBN:978-1-56881-147-5
23. *Kang C-M., Wang L., Xu Y., Meng X.* A survey of photon mapping state-of-the-art research and future challenges // Frontiers of Information Technology & Electronic Engineering. 2016. V. 17. № 3. P. 185–199. DOI:10.1631/FITEE.1500251
24. *Fabianowski B., Dingliana J.* Interactive global photon mapping // Computer Graphics Forum. 2009. V. 28. № 4. P. 1151–1159. http://dx.doi.org/10.1111/j.1467-8659.2009.01492.x
25. *Pediredla A., Chalmiani Y.K., Scopelliti M.G., Chamanzar M., Narasimhan S.* Path tracing estimators for refractive radiative transfer // ACM Transactions on Graphics. 2020. V. 39. № 6. P. 1–15. DOI:10.1145/3414685.3417793
26. *Davidovic T., Krivanek J., Hasan M., Slussalek P.* Progressive light transport simulation on the GPU: survey and improvements // ACM Trans. Graph. 2014. V. 33. № 3. P. 1–19. http://dx.doi.org/10.1145/2602144

INTERACTIVE CALCULATION OF LIGHT REFRACTION AND CAUSTICS USING A GRAPHICS PROCESSOR

S. I. Vyatkin[a], B. S. Dolgovesov[a]

[a]*Synthesizing Visualization Systems Laboratory, Institute of Automation and Electrometry, Siberian Branch, Russian Academy of Sciences,*

Academician Koptyug Avenue, 1, Novosibirsk, Novosibirsk region, Russia, 630090

While modern rendering systems are effective at modeling complex light paths in complex environments, rendering refractive caustics still takes a long time. Caustics are light patterns that occur when light is refracted and reflected from a surface. Due to the sharp density distribution of these mirror events, rendering algorithms mainly rely on direct sampling of the bidirectional scattering distribution function on these surfaces to plot trajectories. This requires many calculations. Photonic maps are also used. However, there are problems limiting the applicability of caustic maps. Since each photon in the photon buffer must be processed, therefore, one has to choose between a strongly underestimated caustic sampling and a large decrease in speed in order to use a sufficient number of photons for caustics in order to obtain high-quality images. Complex mirror interactions cause oversampling in bright focal areas, while other areas of the caustic map remain under-selected and noisy. At the same time, speed takes precedence over realism in most interactive applications. However, the desire to improve the quality of graphics prompted the development of various fast approximations for realistic lighting.

This paper presents a combined method for visualizing refraction of light and caustics using reverse integration for illumination and direct integration for viewing rays. An approach is used for simultaneous propagation of light and for tracking rays in volume and, therefore, it does not require storing data of an intermediate volume of illumination. In the implementation of the method, the distance between the light planes is set to one voxel, which provides at least one sample per voxel for all orientations. The method does not use preliminary calculations; all rendering parameters can be changed interactively.

As a result, using the proposed method, it is possible to create plausible approximations of complex phenomena such as refractions and caustics. The effect of refraction on the shadow is shown. Complex light patterns are demonstrated due to the curved geometry of the objects. The visualization results show the importance of refraction for the appearance of transparent objects. For example, distortions caused by refraction and refraction at the interface between media. The difference in refractive indices between individual media causes a complex interaction between light and shadow areas. It is shown how refraction and caustics improve the visualization of functionally defined objects by providing additional information about shape and location.

Keywords: functionally defined objects, perturbation functions, illumination, refraction, reflection, shadows, caustics, graphics processor

REFERENCES

1. *Wang X., Zhang R.* Rendering transparent objects with caustics using real-time ray tracing, Comput. & Graph., 2021, vol. 96, no. 3. DOI:10.1016/j.cag.2021.03.003.
2. *Komarov E., Zhdanov D., Zhdanov A.* Rendering the real-time caustics with DirectX raytracing, 31th Int. Conference on Computer Graphics and Vision (Graphicon-2021), pp. 36–47. DOI:10.20948/graphicon-2021-3027-36-47.
3. *Grittmann P., Pérard-Gayot A., Slusallek P., Krivanek J.* Efficient caustic rendering with lightweight photon mapping, Comput. Graph. Forum, 2018, vol. 37, no. 4, pp. 133–142. DOI:10.1111/cgf.13481.
4. *Muller T., Gross M., Novak J.* Practical path guiding for efficient light-transport simulation, Comput. Graph. Forum, 2017, vol. 36, no. 4, pp. 91–100. DOI:10.1111/cgf.13227.
5. *Rodriguez S., Leimkuhler T., Prakash S., Wyman C., Shirley P., Drettakis G.* Glossy probe reprojection for interactive global illumination, ACM Trans. Graph., 2020, vol. 39, no. 6, pp. 1–16. DOI:10.1145/3414685.3417823.
6. *Kopanas G., Leimkuhler T., Rainer G., Jambon C. Drettakis G.* Neural point catacaustics for novel-view synthesis of reflections, ACM Trans. Graph., 2022, vol. 41, no. 6, pp. 1–15. DOI:10.1145/3550454.3555497.
7. *Vyatkin S.I., Dolgovesov B.S.* Highly Realistic Visualization of Caustics and Rough Surfaces, Program. Comput. Software, 2022, vol. 48, no. 5, pp. 322–330. DOI: 10.31857/S0132347422050065.
8. *Haber J., Magnor M., Seidel H.P.* Physically-based Simulation of Twilight Phenomena, ACM Trans. Graph., vol. 24, no. 4, 2005, pp. 1353–1373. DOI:10.1145/1095878.1095884.
9. *Vyatkin S.I., Dolgovesov B.S.* Physically based rendering of functionally defined objects, Optoelectron., Instrum. Data Process., 2022, vol. 58, no. 3, pp. 291–297. DOI: 10.15372/AUT20220311.
10. *Vyatkin S.I., Dolgovesov B.S.* A Method for visualizing multivolume data and functionally defined surfaces using GPUs, Optoelectron., Instrum. Data Process., 2021, vol. 57, no. 2, pp. 32–40. DOI: 10.15372/AUT20210204.
11. *Vyatkin S.I.* Method of binary search for image elements of functionally defined objects using graphics processing units, Optoelectron., Instrum. Data Process., 2914, vol. 50, no. 6, pp. 606–612.
12. *Galtier M., Blanco S., Caliot C., et al.* Integral formulation of null collision Monte Carlo algorithms, J. Quantitative Spectrosc. Radiative Transfer., 2013, vol. 125, pp. 57–68. DOI: 10.1016/j.jqsrt.2013.04.001.
13. *Jensen H.W., Marschner S.R., Levoy M.* A practical model for subsurface light transport, Proc. of the 28th annual Conference on Computer Graphics and Interactive Techniques. SIGGRAPH '01, New York: ACM, 2001, pp. 511–518. DOI:10.1145/383259.383319.
14. *Loube G., Zeltner T., Holzschuch N.* Slope-space integrals for specular next event estimation, ACM Trans. Graph., 2020, vol. 39, no. 6. pp. 1–13. DOI:10.1145/3414685.3417811.
15. *Deng X., Jiao S., Bitterli B., Jarosz W.* Photon surfaces for robust, unbiased volumetric density estimation", ACM Trans. Graph., 2019, vol. 38, no. 4, pp. 1–12. DOI:10.1145/3306346.3323041.
16. *Bitterli B., Jarosz W.* Beyond points and beams: Higher dimensional photon samples for volumetric light transport, ACM Trans. Graph., 2017, vol. 36, no. 4, pp. 1–12.
17. *Frolov V.A., Voloboy A.G., Ershov S.V., Galaktionov V.A.* State-of-the art of methods for global illumination calculation in realistic computer graphics, Trudy ISP RAN, 2021, vol. 33, no. 2, pp. 7–48. DOI:https://doi.org/10.15514/ISPRAS-2021-33(2)-1.
18. *Vyatkin S.I., Dolgovesov B.S.* Functionally defined models for additive production, Issled. Innovatsii, Praktika, 2022, vol. 4, no. 4, pp. 16–25. DOI: 10.18411/iip -08-2022-04.
19. *Lavoue G., Bonneel N., Farrugia J.-P., Soler C.* Perceptual quality of BRDF approximations: Dataset and metrics, Comp. Graph. Forum, 2021, vol. 40, no. 2, pp. 327–338. DOI: 10.1111/cgf.142636.
20. *Vyatkin S.I., Dolgovesov B.S.* Smoothing functionally specified objects in scenes with global illumination, J. Adv. Res. Techn. Sci., 2022, no. 30, pp. 96–103. DOI: 10.26160/2474-5901-2022-30-96-103.
21. *Schlick C.* A Customizable reflectance model for everyday rendering, Proc. of the Fourth Eurographics Workshop on Rendering, 1993, pp. 73–84. Corpus ID: 18967314.
22. *Jensen H.W.* Realistic Image Synthesis Using Photon Mapping, Natick, Mass.: Peters, 2001.

23. *Kang C-M., Wang L., Xu Y., Meng X.* A survey of photon mapping state-of-the-art research and future challenges, Frontiers Inf. Technol. & Electron. Eng., 2016, vol. 17, no. 3, pp. 185–199. DOI:10.1631/FITEE.1500251.
24. *Fabianowski B., Dingliana J.* Interactive global photon mapping, Comput. Graph. Forum, 2009, vol. 28, no. 4, pp. 1151–1159. http://dx.doi.org/10.1111/j.1467-8659.2009.01492.x.
25. *Pediredla A., Chalmiani Y.K., Scopelliti M.G., Chamanzar M., Narasimhan S.* Path tracing estimators for refractive radiative transfer, ACM Trans. Graph., 2020, vol. 39, no. 6, pp. 1–15. DOI:10.1145/3414685.3417793.
26. *Davidovic T., Krivanek J., Hasan M., Slusallek P.* Progressive light transport simulation on the GPU: Survey and improvements, ACM Trans. Graph., 2014, vol. 33, no. 3, pp. 1–19. http://dx.doi.org/10.1145/2602144.