ИНФОРМАТИКА

УДК 519.683

ОПТИМИЗАЦИЯ ПОСТРОЕНИЯ РАСЧЕТНОЙ СЕТКИ ДЛЯ РЕШЕНИЯ ЗАДАЧИ ЛОКАЛЬНОГО КРИОВОЗДЕЙСТВИЯ С ИСПОЛЬЗОВАНИЕМ МНОГОМЕРНОГО ГЕОМЕТРИЧЕСКОГО ХЕШИРОВАНИЯ НА ОСНОВЕ ПАКЕТА NumPy

В. А. Клячин

Доктор физико-математических наук, доцент кафедры компьютерных наук и экспериментальной математики, Волгоградский государственный университет, klchnv@mail.ru

В работе, на примере решения задачи построения температурного поля при криовоздействии показывается эффективность использования геометрического хеширования выполненного на основе пакета NumPy для построения соответствующей расчетной сетки. Такое построение предполагает для каждого узла сетки определение его местоположения относительно полигональной области неправильной формы. Именно такие формы чаще всего моделируют поверхности внутренних органов. Решение построения расчетной сетки позволит осуществить 3D визуализацию температурного поля в окрестности точки криовоздействия, что будет способствовать своевременному контролю температуры биоткани в заданный момент времени.

Ключевые слова: криобиология, вычислительная геометрия, геометрическое хеширование.

НАУЧНЫЙ ОТДЕЛ

ВВЕДЕНИЕ

За последнее время в различных областях медицины большое распространение получил криогенный метод лечения [1]. Интерес к локальному низкотемпературному воздействию в основном связан с тем, что лечение криогенными методами проходит практически безболезненно [2].

Математическая модель для численного исследования процессов протекающих в биотканях при таком методе лечения описана в монографии [1]. При указанных в этой работе допущениях температурное поле в биоткани описывается с помощью уравнения теплопроводности вида

$$c \cdot \rho(x,T)\frac{\partial T}{\partial t} = \operatorname{div}\left(\lambda(x,T)\nabla T(x)\right) - \chi(x)\frac{\partial f}{\partial t}, \qquad (1)$$

где T — тепература, а коэффициенты уравнения зависят от свойств биоткани. Можно отметить работу [3], в которой предложены определенные оригинальные методы численного решения с соответствующими краевыми условиями (задача Стефана). Однако в указанных работах данная задача решается в осесимметричном случае и соответствующая сетка для численного решения задачи строится как прямоугольная сетка в цилиндрических координатах. Данный подход по нашему мнению является весьма приближенным и не может охватить все возможные биоткани, геометрическая форма которых

достаточно разнообразна. В качестве примера можно привести биоткань поджелудочной железы, форма которой не имеет явной симметрии, а ее поверхность обладает в определенной степени ячеистой структурой.

Численное решение приведенного уравнения можно осуществить с использованием как прямоугольной сетки, так и с использованием нерегулярной сетки, например в виде триангуляции расчетной области. Наиболее популярным методом триангуляции является триангуляция Делоне [4, 5]. В монографии [6] приведены различные алгоритмы построения таких триангуляций на плоскости. Отметим также, что в работах [7, 8] получены условия на симплексы триангуляции, выполнение которых гарантирует соответствующую сходимость первых производных. Однако использование триангуляции в многомерном случае сопряжено с рядом принципиальных трудностей, описанных в работе [9]. Поэтому, нами был выбран другой подход, связанный с построением прямоугольной сетки, не имеющей препятствия с аппроксимацией производных первого порядка.

Для расчета решений уравнения с использованием прямоугольной сетки необходимо дополнительно решать задачу определения для каждого ее узла принадлежности расчетной области. Это классическая задача вычислительной геометрии о локализации точки [10, 11]. При достаточно большом количестве граней модели расчетной области может потребоваться значительное время для решения задачи локализации всех узлов расчетной сетки с помощью даже линейных по числу граней модели алгоритмов. Поэтому применяют некоторую оптимизацию. Одним из способов оптимизации является так называемое геометрическое хеширование, суть которого хорошо описана, в частности, в статьях [12–14], в которых указанный метод применяется в задачах распознавания объектов на изображениях. Цель настоящей статьи – показать каким образом метод многомерного геометрического хеширования можно применить к решению задачи локализации точки относительно замкнутых многогранных поверхностей. При этом задача будет решена для произвольной размерности, а для управления многомерными данными мы воспользуемся пакетом NumPy [15, 16].

1. МАТЕМАТИЧЕСКАЯ ОСНОВА ГЕОМЕТРИЧЕСКОГО ХЕШИРОВАНИЯ

Пусть X — произвольное множество и K обозначает единичный квадрат $K = [0,1] \times [0,1]$. Каждой точке $(a,b) \in K$ можно сопоставить отрезок $[a,b] \subset [0,1]$, а каждой точке $p = (a_1, b_1, a_2, b_2, \ldots, a_n, b_n) \in K^n$, можно сопоставить многомерный параллилепипед $Q(p) = [a_1, b_1] \times \ldots \times [a_n, b_n]$, $n \geq 1$.

Пусть дано некоторое отображение

$$s : X \to K^n, \qquad n \geq 1.$$

Непосредственно в задаче локализации точки относительно многогранника в качестве X рассматривается множество всех пространственных треугольников, расположенных, например, в единичном кубе $[0,1] \times [0,1] \times [0,1]$. Для случая $n = 1$ отображением s в этом случае может служить отображение, которое каждому треугольнику ставит в соответствие отрезок проекции этого треугольника на ось Ox. При $n = 2$ мы можем взять два отрезка проекций треугольника — один на ось Ox, другой — на ось Oy.

Помимо отображения s, предположим задана матрица T размерности $n \times m, m \geq 1$. При этом предполагаем выполнение условий

$$0 = T_{i1} < T_{i2} < T_{i3} < \ldots < T_{im} = 1, \qquad \forall\ i = 1, \ldots, n.$$

Элементы этой матрицы задают сетку, по которой будет вестись хеширование. В простейшем случае можно использовать равномерную сетку, для которой $T_{ik} - T_{ik-1} \equiv \text{const}$.

Каждой точке $(a,b) \in K$, $a \leq b$ и натуральному $i = 1, \ldots, n$ сопоставим два натуральных числа k_i, l_i таких, что $a \in [T_{ik_i}, T_{ik_i+1})$, $b \in [T_{il_i}, T_{il_i+1})$. Заметим, что эти два числа определяют отрезки вида $[T_{ij}, T_{ij+1}]$, которые имеют непустое пересечние с отрезком $[a,b] \subset [0,1]$ для всех $j = k_i, \ldots, l_i$. Например, в задаче локализации точки числа k_i, l_i определяют те полуинтервалы сетки хеширования, которые будет пересекать проекция треугольника на ось Ox в случае $n = 1$ и эти числа определяют номера прямоугольных ячеек сетки хеширования, которые пересекает проекция треугольника на плоскость Oxy при $n = 2$.

Опишем математическую суть процесса хеширования.

Пусть задано некоторое конечное подмножество $Y \subset X, Y = \{y_1, y_2, \ldots, y_N\}$ и некоторый вектор $t = (t_1, \ldots, t_n) \in [0,1]^n$. Требуется найти такое подмножество $Y' \subset Y$, для элементов которого будет выполнено условие

$$y \in Y' \Leftrightarrow t_i \in [a_i, b_i], i = 1, \ldots, n, \tag{2}$$

где $s(y) = (a_1, b_1, \ldots, a_n, b_n)$. В нашем примере в качестве подмножества Y выбираем множество треугольных граней многогранника, а записанное условие означает принадлежность точки t проекции треугольной грани. Другими словами поиск подмножества Y' — это поиск тех граней многогранника, проекции которых содержат заданную точку t. Для решения задачи будем использовать следующую конструкцию. Для каждого мультииндекса $J = (j_1, \ldots, j_n)$, $j_i = 0, 1, \ldots, m-1$ построим набор натуральных чисел $n_J = \{n_{J1}, n_{J2}, \ldots, n_{Jr_I}\}$ такой, что

$$Q(s(y_k)) \cap [T_{1j_1}, T_{1j_1+1}] \times \ldots \times [T_{nj_n}, T_{nj_n+1}] \neq \emptyset \quad \forall k \in n_J.$$

Построение таких наборов номеров n_J позволяет ускорить обработку целого потока данных $t^1, \ldots, t^k, \ldots \in [0,1]^n$, поскольку не требует полного перебора всего множества Y.

2. ОПИСАНИЕ АЛГОРИТМА И РЕАЛИЗАЦИИ

Опишем подробнее алгоритм хеширования. Прежде всего, до начала основного цикла необходимо определить n-мерный массив $H[\ \ ]$, имеющий m элементов по каждой размерности. Так, что элементы этого массива будут нумероваться мультииндексами вида $J = (j_1, \ldots, j_n)$ с $j_i = 0, 1, \ldots, m-1$. Элементами же этого массива будут наборы целых чисел, которые в процессе хеширования будут вычисляться. Входными данными для алгоритма является массив объектов – элементов подмножества $Y \subset X$.

1. В основном цикле алгоритма перебираются элементы массива объектов, для которых вычисляется отображение $s : X \to K^n$. Результатом вычисления является массив пар чисел (a_i, b_i), $i = 1, \ldots, n$, лежащих в отрезке $[0, 1]$. При этом будем считать, что переменная q является номером текущего объекта.

2. Далее для всякой пары (a_i, b_i) вычисляются целые числа k_i, l_i так, что

$$a_i \in [T_{ik_i}, T_{ik_i+1}) \quad b_i \in [T_{il_i}, T_{il_i+1}) \ .$$

Геометрически эти числа задают отрезки вида $[T_{ji}, T_{ji+1}]$ такие, что $[T_{ij}, T_{ij+1}] \cap [a_i, b_i] \neq \emptyset, j = k_i, \ldots, l_i$.

3. Для всякого мультииндекса $J = (j_1, \ldots, j_n)$ с $k_i \leq j_i \leq l_i$, $i = 1, \ldots, n$ добавляем номер текущего объекта q в список $H[J]$.

Основная сложность реализации многомерного хеширования состоит в необходимости перебирать элементы массива произвольной размерности. Наиболее подходящее для этого средство, по всей видимости, это — пакет **NumPy** для языка программирования Python. Этот пакет, совместно с пакетами **SymPy**, **SciPy** является программным обеспечением с открытым исходным кодом для математики, естественных наук и инженерии. В частности, **NumPy** –— это расширение языка Python, добавляющее поддержку больших многомерных массивов и матриц, вместе с большой библиотекой высокоуровневых математических функций для операций с этими массивами. Отметим ряд работ, в которых этот пакет испоьзуется в научных исследованиях в области кристаллографии [17], экологии [18], гидродинамики пористых сред [19]. Мы опишем собственную реализацию многомерного хеширования на основе пакета **NumPy**. Для этого реализуем класс **MDHash**, в котором основная процедура будет обозначена как метод **MDHash. hash**. Метод **MDHash.segments** является чисто виртуальным и должен быть определен в подклассах. Этот метод является программной реализацией отображения $s : X \to K^n$. Весь код метода **MDHash.hash** не сложен и его полностью можно привести.

```
def hash(s):
    q=0
    for x in s.input:
        sel=[]
        co=1
        for ab in s.segments(x):
```

```
            kl=s.mu(ab,co)
            sel.append(slice(kl[0],kl[1]+1,1))
            co+=1
        B=s.H[sel]
        if(B.size):
            it = nditer(B, flags=['multi_index','refs_ok'])
            while not it.finished:
                mi=it.multi_index
                B[mi].append(q)
                it.iternext()
        q+=1
```

Сделаем соответствующие пояснения. Предварительная подготовка, связанная с определением результирующего массива выполняется в конструкторе класса и выглядит так

```
def __init__(s,input,n,t):
    s.n=n
    s.T=t
    s.input=input
    shp=[len(ti) for ti in t]
    s.tree=ndarray(shape=shp,dtype=object)
    it = nditer(s.H, flags=['multi_index','refs_ok'])
    while not it.finished:
        mi=it.multi_index
        s.H[mi]=[]
        it.iternext()
```

Цикл **while** заполняет каждый элемент массива $H[\ ]$ пустыми списками. При этом используется обход всего массива основанный на вычислении мультииндекса **mi**, являющегося атрибутом объекта итератора **it**. Этот итератор строится с помощью функции **nditer()**, принадлежащей пакету **NumPy**. Также отметим, что искомый массив $H[\ \ ]$ создается с помощью встроенный в пакет **NumPy** функции **ndarray**. При этом указываются размерности массива и тип его элементов.

Выполнение алгоритма хеширования начинается внутри метода **hash()** циклом **for x in s.input**. Для каждого объекта массива создается объект среза массива $H[\ ]$ по значениям функции **MDHash.mu()**.

```
def mu(s,ab,co):
    [a,b]=ab
    k=s.bsearch(s.T[co-1],a)
    l=s.bsearch(s.T[co-1],b)
    return [k,l]
```

Этот объект представляет собой массив n пар целых чисел $(k_i, l_i), i = 1, \ldots, n$. Используемый далее массив $B[\ ]$ представляет собой часть массива $H[\ ]$ соответствующий всем мультииндексам $J = (j_1, \ldots, j_n)$ с $k_i \le j_i \le l_i, i = 1, \ldots, n$. Используя встроенную систему итераторов **NumPy**, добавляем номер q в соответствующие списки **B[mi].append(q)**. Заметим, что для ускорения поиска чисел k_i, l_i используем функцию бинарного поиска

```
def bsearch(s,input,x):
    n=len(input)
    b=0
    e=n
    while(e-b>1):
        k=int((e+b)/2)
        if(x<=input[k]): e=k
        if(x>input[k]): b=k
    return b
```

Наконец, для определения подмножества $Y' \subset Y$ используется процедура

```
def into(s,x,co):
    mi=[]
    for xi in x:
        k=s.bsearch(s.T[co-1],xi)
        mi.append(k)

    return s.H[tuple(mi)]
```

Результат работы этой процедуры список — элемент массива $H[\,]$, содержащий номера тех объектов, которые удовлетворяют условию (2).

3. ОЦЕНКА АЛГОРИТМИЧЕСКОЙ СЛОЖНОСТИ И РЕЗУЛЬТАТЫ ЧИСЛЕННЫХ ЭКСПЕРИМЕНТОВ

Построение расчетной сетки для ограниченной области $D \subset \mathbb{R}^3$ сводится к вычислению трехмерной матрицы M_{ijk} вида

$$M_{ijk} = \begin{cases} 1, (x_i, y_j, z_k) \in D \\ 0, (x_i, y_j, z_k) \notin D. \end{cases}$$

Здесь (x_i, y_j, z_k) — соответствующий узел прямоугольной сетки. Принадлежность точки p области D с многогранной границей определяется по простому праилу. Если луч, выходящий из точки p имеет нечетное число точек пересечения с границей, то точка $p \in D$. В противном случае — $p \notin D$. Если каждой грани границы приписать число 1, при условии ее пересечения с лучем и 0 в противном случае, то число точек пересечения с границей расчитывается как сумма этих величин по всем граням. Так, что если m — число узлов сетки по каждой координате, а N – число граней модели, то всего элементарных операций вычисления факта пересечения луча с пространственным треугольником будет ровно $m^3 \cdot N$. Однако, можно не рассматривать те грани, которые заведомо не пересекают луча. Здесь как раз и помогает предобработка в виде геометрического хеширования. Результат хеширования — это двумерный массив **H[ ]**, элементами которого являются наборы натуральных чисел. Предположим, что из каждого узла сетки выпускается луч в положительном направлении оси Oz. Хеширование будем вести по сетке, совпадающей с расчетной. Тогда каждой паре натуральных чисел $0 \le i, j < m - 1$ соответствует набор **H[i][j]** номеров тех граней, проекции которых вдоль оси Oz пересекают ячейку сетки оперделяемую узлами $(x_i, y_j), (x_{i+1}, y_{j+1})$. Оценим число таких граней. Предположим, что каждая грань имеет диаметр, не превосходящий $d > 0$. Тогда число ячеек сетки, которые могут пересекаться с проекцией этой грани не превосходит числа $[md] + 1$. Введем такие обозначения. Пусть m_k — число ячеек, которые пересекаются с проекцией k-й грани, а n_{ij} — число граней, чьи проекции пересекают (i, j)-ю ячейку сетки. Кроме того, пусть L — максимальное значение кратности проекции. Тогда, для построения вышеуказанной матрицы M потребуется число операций вычисления пересечения луча с пространственным треугольником равное

$$\sum_{ij} n_{ij} = \sum_{k} m_k \le N \cdot L \cdot ([md] + 1).$$

Таким образом, при простом переборе всех граней при измельчении сетки (с возрастанием m) число элементарных операций растет порядка $O(m^3)$, тогда как двумерное хеширование дает оценку порядка $O(m)$. Приведем результаты численных экспериментов.

Первые эксперименты проведены для области, граница которой представляет собой выпуклый многогранник с 80 треугольными гранями. Время построения сетки с m узлами по каждой координате в табл. 1 указано в секундах.

Во втором эксперименте в качестве области была взята геометрическая модель поджелудочной железы (рис. 1, 2). В табл. 2 представлены результаты измерения времени построения сетки для различных значений числа узлов разбиений m по каждой координате. Указано время в секундах. Число граней на границе области равно 27648.

Таблица 1

m	Количество узлов сетки	Без хеширования	Одномерное хеширование	Двумерное хеширование
10	1 000	3.156	0.813	0.281
20	8 000	23.078	4.297	0.938
30	27 000	74.218	12.970	3.250
40	64 000	172.734	31.344	6.187
50	125 000	332.203	57.250	11.969
100	1 000 000	2 594.499	434.110	90.609
200	8 000 000	19 815.125	3 352.312	688.110

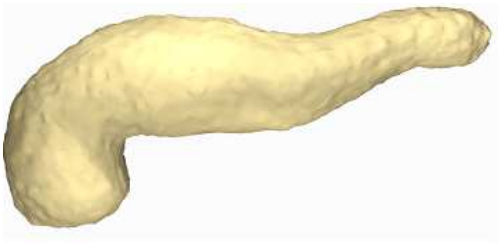

Рис. 1. 3D модель поджелудочной железы

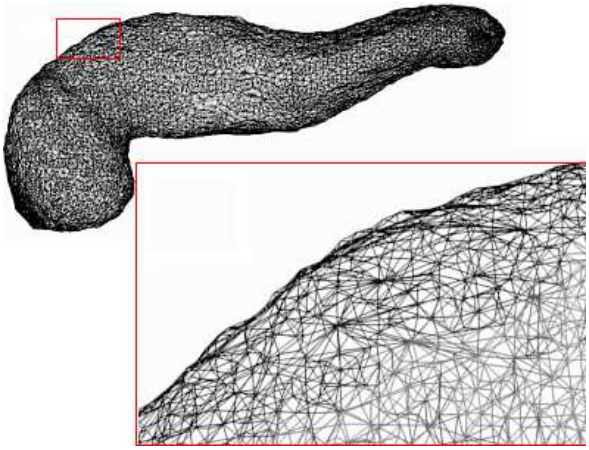

Рис. 2. Геометрическая модель поджелудочной железы, содержащая 27648 треугольников

Таблица 2

m	Количество узлов сетки	Без хеширования	Одномерное хеширование	Двумерное хеширование
10	1 000	1112.89	125.43	13.08
20	8 000	*	480.25	25.69
30	27 000	*	960.97	39.27
40	64 000	*	2022.75	62.08
50	125 000	*	*	78.34
100	1 000 000	*	*	263.75
200	8 000 000	*	*	1242.20

*время, превышающее 3600 с

Работа выполнена при финансовой поддержке РФФИ (*проект* 13-01-97034).

Библиографический список

1. *Цыганов Д. И.* Процессы криовоздействия, аппараты и крио-СВЧ технологии деструкции новообразований. М. : РМАПО, 2004. 88 с.

2. *Терновский К. С., Гассанов Л. Г.* Низкие температуры в медицине. Киев : Наук. думка, 1988. 280 с.

3. *Буздов Б. К.* Моделирование криодеструкции биологической ткани // Мат. моделирование. 2011. Т. 23, № 3. С. 27–37.

4. *Альес М. Ю., Копысов С. П., Новиков А. К.* Построение и адаптация конечно-элементной сетки при решении эллиптической задачи второго порядка // Мат. моделирование. 1997. Т. 9, № 2. С. 43–45.

5. *Боровиков С. Н., Крюков И. А., Иванов И. Э.* Построение нерегулярных треугольных сеток на криволинейных гранях на основе триангуляции Делоне // Мат. моделирование. 2005. Т. 17, № 8. С. 31–45.

6. *Скворцов А. В., Мирза Н. С.* Алгоритмы построения и анализа триангуляции. Томск : Изд-во Том. ун-та, 2006.

7. *Клячин В. А., Широкий А. А.* Триангуляция Делоне многомерных поверхностей и ее аппроксимационные свойства // Изв. вузов. Математика. 2012. № 1. С. 31–39.

8. *Клячин В. А., Пабат Е. А.* C^1-аппроксимация поверхностей уровня функций, заданных на нерегулярных сетках // Сиб. журн. индустр. мат. 2010. Т. 13, № 2. С. 69–78.

9. *Клячин В. А.* О многомерном аналоге примера Шварца // Изв. РАН. Сер. математическая. 2012. Т. 76, № 4. С. 41–48.

10. *Препарата Ф. Р., Шеймос М.* Вычислительная геометрия : введение. М. : Наука, 1989.

11. *Berg M., Cheong O., Kreveld M., Overmars M.* Computational Geometry. Algorithms and Applications. Berlin ; Heidelberg : Springer-Verlag, 2008.

12. *Wolfson H. J., Rigoutsos I.* Geometric Hashing : An Overview // IEEE Computational Science and Engineering. 1997. Vol. 4, № 4. P. 10–21.

13. *Ling M., Yumin L., Huiqin J., Zhongyong W., Haofei Z.* An Improved Method of Geometric Hashing Pattern Recognition // Intern. J. Modern Education and Computer Science. 2011. № 3. P. 1–7.

14. *Mian A. S., Bennamoun M., Owens R.* Three-dimensional model-based object recognition and segmentation in cluttered scenes // IEEE Transactions on Pattern Analysis and Machine Intelligence. 2006. Vol. 28. P. 1584–601.

15. Официальный сайт NumPy. URL : http://numpy.org (дата обращения : 21.12.2013).

16. Официальный сайт SciPy. URL : http://scipy.org/ (дата обращения : 21.12.2013).

17. *Newton M. C., Nishino Y., Robinson I. K.* BONSU : the interactive phase retrieval suite // J. of Applied Crystallography. 2012. Vol. 45, № 4. P. 840–843.

18. *Bryan B. A.* High-performance computing tools for the integrated assessment and modelling of social-ecological systems // Environmental Modelling & Software. 2013. Vol. 39. P. 295–303.

19. *Никольский Д. Н.* Разработка программного обеспечения для численного решения задач эволюции границы раздела различных жидкостей в пористых средах сложной геологической структуры с использованием пакета NumPy // Учен. зап. Орлов. гос. ун-та. Сер. Естественные, технические и медицинские науки. 2012. № 6–1. С. 42–47.

Optimization of Calculus Mesh for Cryobiology Problem Based on Multidimensional Hashing Using NumPy

V. A. Klyachin

Volgograd State University, 100, University ave., Volgograd, 400062, Russia, klchnv@mail.ru

In this paper, by the example of solving the problem of constructing the temperature field in cryotherapy shows efficiency of geometric hashing performed on the basis of the NumPy package for constructing appropriate computational grid . Such an arrangement implies for each node to determine its position relative to the polygonal area of irregular shape. Such forms often modeled surfaces of internal organs. Solution build computational grid will allow for 3D visualization of the temperature field in the vicinity of cryotherapy, which will facilitate the timely temperature control.

Key words: cryobiology, computation geometry, geometric hashing.

References

1. Cyganov D. I. *Processy kriovozdeistviya, apparaty i krio-SVC technologii destrukcii novoobrazovaniy* [Processes cryotherapy, apparatus and cryo-microwave technologies of destruction of tumors]. Moscow, RMAPO 2004, 88 p. (in Russian).

2. Ternovskiy K. S., Gassanov L. G. *Nizkie temperatury v medicine* [Low temperatures in medicine]. Kiev, Naukova Dumka, 1988, 280 p. (in Russian).

3. Buzdov B. K. Modelling of cryodestruction of biological tissues. *Mat. Model.*, 2011, vol. 23, no. 3, pp. 27–37 (in Russian).

4. Alies M. Yu., Kopysov S. P., Novikov A. K. Generation

and adaption of finite element mesh for elliptic problem of order two solution. *Mat. Model.*, 1997, vol. 9, no. 2, pp.43–45 (in Russian).

5. Borovikov S. N., Kryukov I. A., Ivanov I. E. Unstructured triangular mesh generation on curved faces based on Delauney triangulation. *Mat. Model.*, 2005, vol. 17, no. 8, pp. 31–45 (in Russian).

6. Skvortsov A. V., Mirza N. S.*Algoritmy postroeniya i analiza triangulacii* [Constructing and Analisys of Triangulation Algorithms]. Tomsk, Tomsk Univ. Press, 2006.

7. Klyachin V. A., Shirokii A. A. The Delaunay triangulation for multidimensional surfaces and its approximative properties. *Rus. Math.* [Izv. VUZ. Matematika], 2012, vol. 56, no. 1, pp. 27–34. DOI: 10.3103/S1066369X12010045.

8. Klyachin V. A., Pabat E. A. C^1-approximation of the level surfaces of functions defined on irregular grids. *Sib. Zh. Ind. Mat.*, 2010, vol. 13, no. 2, pp. 69–78 (in Russian).

9. Klyachin V. A. On a multidimensional analogue of the Schwarz example. *Izv. Math.*, 2012, vol. 76, no. 4, pp. 681–687. DOI: 10.1070/IM2012v076n04ABEH002601.

10. Preparata F. P., Shamos M. *Computational Geometry': An Introduction*. Berlin ; Heidelberg, Springer-Verlag, 1988.

11. Berg M., Cheong O., Kreveld M., Overmars M. *Computational Geometry. Algorithms and Applications*. Berlin ; Heidelberg, Springer-Verlag, 2008.

12. Wolfson H. J., Rigoutsos I. Geometric Hashing : An Overview. *IEEE Computational Science and Engineering*, 1997, vol. 4, no. 4, pp. 10–21.

13. Ling M., Yumin L., Huiqin J., Zhongyong W., Haofei Z. An Improved Method of Geometric Hashing Pattern Recognition. *I. J. Modern Education and Computer Science*, 2011, no. 3, pp. 1–7.

14. Mian A. S., Bennamoun M., Owens R. Three-dimensional model-based object recognition and segmentation in cluttered scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2006, vol. 28, pp. 1584–1601.

15. NumPy. URL: http://numpy.org/ (Accessed 21, Dec, 2013).

16. SciPy. URL: http://scipy.org/ (Accessed 21, Dec, 2013).

17. Newton M. C., Nishino Y., Robinson I. K. BONSU: the interactive phase retrieval suite. *Journal of Applied Crystallography*, 2012, vol. 45, no. 4, pp. 840–843.

18. Bryan B. A. High-performance computing tools for the integrated assessment and modelling of social-ecological systems. *Environmental Modelling & Software*, 2013, vol. 39, pp. 295–303.

19. Nikolsky D. N. Development of software for the numerical solution of the evolution of the interface of various fluids in porous media complex geological structures using the package NumPy. *Uchenye zapiski Orlovskogo gosudarstvennogo universiteta. Ser. Estestvennye, technicheskie i medicinskie nauki*, 2012, no. 6–1, pp. 42–47 (in Russian).