МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ» (НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ, НГУ)

Факультет _____**Механико-Математический**_____

Кафедра _____**Дискретной математики и Информатики**_____

Направление подготовки __**Прикладная математика и Информатика**_____

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА МАГИСТРА

_____**Мулюкова Артёма Равильевича**_____

(Фамилия, Имя, Отчество автора)

Тема работы “**Иерархическая кластеризация базы данных индикатрис для решения обратной задачи светорассеяния”**

«К защите допущена»

Заведующий кафедрой

Докт. физ.-мат. наук,

Профессор, академик РАН

Гончаров С.С./……

(фамилия , И., О.) / (подпись, МП)

«……»……20…г.

Научный руководитель

Канд. физ.-мат. наук

С.н.с. ИХКГ СО РАН

Юркин М.А./……

(фамилия , И., О.) / (подпись, МП)

«……»……20…г.

Дата защиты: «……»……20…г.

Новосибирск, 2020

Введение

Задача измерения характеристик клеток крайне важна в очень широком спектре областей. Она важна как для научных исследований биологических явлений, так и в прикладных областях, таких как медицина. В частности, развитие методов характеризации одиночных клеток и их популяций имеет огромную важность для улучшения методов диагностики пациентов. Данная задача уже давно решается в Лаборатории цитометрии и биокинетики Института химической кинетики и горения им. В.В. Воеводского СО РАН, в частности, благодаря технологии сканирующей проточной цитометрии [1].

Сканирующий проточный цитометр (СПЦ) измеряет интенсивность рассеяния света биологическими клетками в диапазоне углов (например, от 10° до 70°, с интервалом в 1°), таким образом, сигнал представляет из себя серию измерений для каждой клетки. С достаточно высокой скоростью, порядка 100 клеток в секунду [2], мы можем получить измерения большого количества клеток популяции, что позволяет нам делать статистические выводы о биологической системе [3–5].

Обратная задача определения характеристик клеток по имеющемуся сигналу рассеянного света достаточно сложна и не имеет точного решения, хотя и некоторые методы были предложены и исследованы в ранних работах. Например, стандартный метод нелинейной регрессии, применимый только для сферических частиц [6], который так же требовал некоторых оптимизаций для эффективной и быстрой работы [7].

Для частиц произвольной формы, решение прямой задачи, например с помощью метода дискретных диполей [8], занимает огромное вычислительное время (порядка минуты), что делает невозможным применение методов регрессии полученного сигнала. В качестве альтернативы, в настоящий момент, используется предварительно построенная база сигналов для случайно распределенных наборов характеристик частицы и поиск оптимального решения с помощью интерполяции методом ближайшего соседа в пространстве сигналов [9]. Для решения задачи характеризации частицы с 4мя характеристиками (например, тромбоцит), размер базы данных должен быть уже порядка ста тысяч элементов, что делает процесс поиска ближайшего элемента в базе данных очень времязатратным. Для оптимизации этого процесса уже рассматривались различные методы кластерных структур данных и были показаны реальные ускорения поиска ближайших сигналов в базе данных [10], но реализация не была доведена до инструмента, готового к применению для обработки реальных экспериментов.

Важной модификацией стандартного метода ближайшего соседа является возможность получения не только ближайшего решения, но и его статистическая оценка, например, с использованием расстояний до достаточно большого количества теоретических сигналов, чьи характеристики плотно распределены пространстве характеристик [6,9]. Вычисление или аппроксимация этих расстояний требует модификации имеющихся реализаций поиска ближайшего и поднимает вопрос подбора оптимальных параметров перебора кластеров. Сама идея использования расстояний до просмотренных кластеров уже

рассматривалась в работе [11], но не была доведена до работающей реализации. В данной работе я предлагаю использовать другой алгоритм кластерной структуры, а именно бинарное дерево, типа kd-дерева, что позволяет значительно сэкономить используемую память. Также, здесь будет закончена реализация программы для использования другими исследователями и проведен анализ полученных зависимостей с точки зрения применимости алгоритма, а также природы полученных зависимостей ускорения. Дополнительно, обсуждается возможность применения полученного решения для использования в более широком спектре задач, не ограниченных задачами светорассеяния или биологии.

Основная часть

Описание области

Сканирующий проточный цитометр

Цитометрия – наука о фенотипических и функциональных свойств клеток в сложных системах [12]. Основным прибором для измерения характеристик клеток служит цитометр, который может иметь различное устройство и способ работы, базирующиеся на тех или иных физических явлениях. Как пример, можно вспомнить фото-цитометрию [13], где принцип работы основан на анализе микроскопических цифровых фотографий клеток с помощью цифровых методов анализа, вплоть до нейронных сетей [14].

Более классическим методом, активно используемым как в научных исследованиях, так и медицинских практиках является проточный цитометр. Основа его устройства состоит в пропускании клеток-образцов через измерительный элемент, измерении и последующем анализе сигнала, различающегося для разных клеток. Измерения могут происходить как с помощью отслеживания изменения электрического потенциала в области пролета клетки, так и с помощью освещения пространства в области пролета. Иногда клетки для этой цели окрашивают специальными флуоресцентными красителями, иногда – нет [15,16]. Возможны и другие способы измерения сигнала, неизменным остаётся лишь положение клеток в потоке жидкости, проходящем через измеряющий элемент.

Исследования данной работы базируются на измерениях, произведенных продвинутой формой проточного цитометра, называемой сканирующий проточный цитометр [17]. Основа его работы состоит в измерении сигнала от клетки в течении некоторого времени при пролёте через область измерения (не в один момент времени, как в классическом цитометре). В течение всего пролета клетки через область, сонаправленно с движением клетки мы светим на нее лазером (частота лазера может быть различной, например, для данных, на которых проводились тесты в этой работе, она была равна 660 нм). Клетка рассеивает падающее излучение во все направления, а, благодаря сферическому зеркалу-линзе, противопоставленному ее движению, в каждый конкретный момент времени только свет, направленный в небольшой телесный угол, проходит через диафрагму фотоэлектрического умножителя. Подробнее об устройстве СПЦ можно почитать в работе [18], здесь же приведена его схема на Рис. 1.

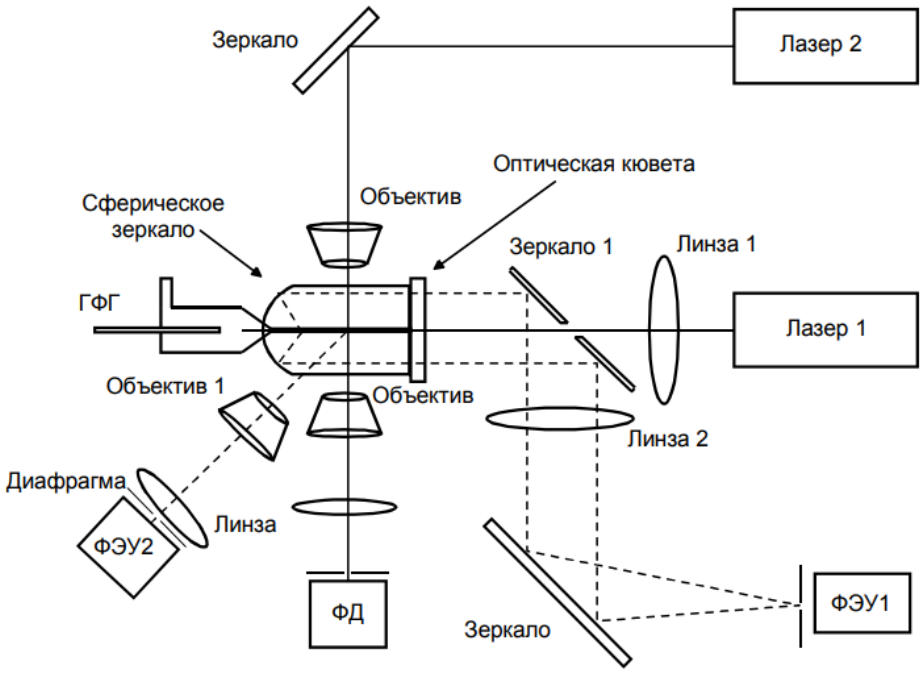

Рис. 1 Инженерная схема сканирующего проточного цитометра [18]

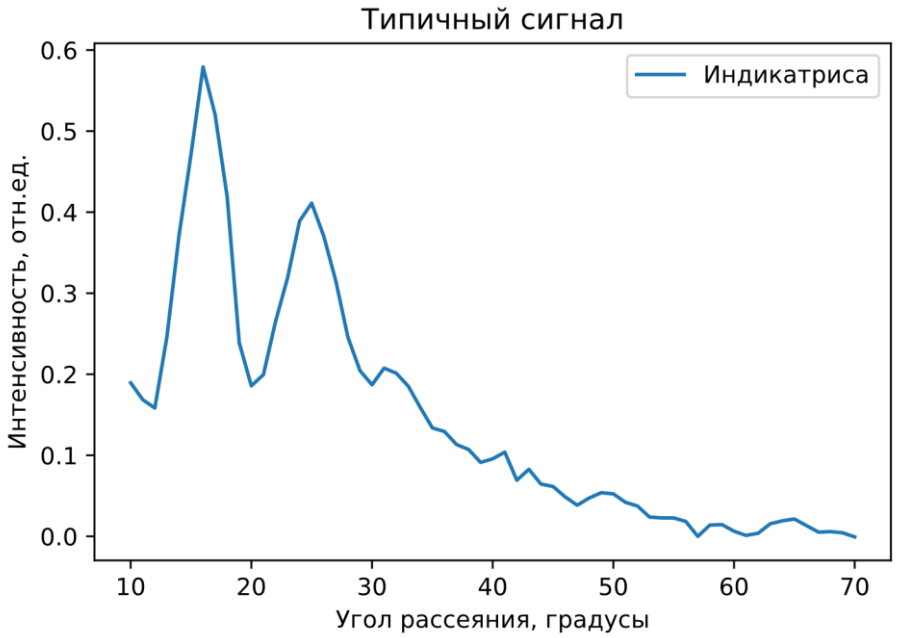

Рис. 2 Типичный сигнал, полученный для тромбоцита, измеренного на СПЦ.

Таким образом, мы можем получить зависимость интенсивности собранного излучения, рассеянного клеткой, от времени. Используя известные геометрические параметры оптической схемы, этот сигнал можно привести к формату зависимости интенсивности рассеяния от угла, который называется индикатрисой. При этом мы используем следующую функциональную форму индикатрисы, которая применима для осесимметричных частиц в любой ориентации (как все модели клеток в данной работе) [9]:

$$I(\theta) = \frac{w(\theta)}{2\pi} \int_0^{2\pi} S_{11}(\theta, \varphi) d\varphi, \tag{1}$$

где S_{ij} – элемент матрицы Мюллера, а θ и φ – полярный и азимутальный углы соответственно, и в нее включена $w(\theta)$ – эмпирическая весовая функция, задаваемая формулой

$$w(\theta) = \frac{1\ \text{deg}}{\theta} \exp[-2 \ln^2(\frac{\theta}{54\ \text{deg}})]. \tag{2}$$

Использование весовой функции примерно выравнивает амплитуду экспериментального шума для разных значений θ, поэтому использование обычной эвклидовой нормы в качестве близости индикатрис (см. параграф «Текущий метод решения задачи») оправдано именно для такого определения $I(\theta)$. Детальнее о приведении зависимости от времени к форме индикатрисы можно почитать, например, в статье [6]. Пример индикатрисы, измеренной на СПЦ и используемой при тестировании алгоритма в этой работе, вы можете видеть на Рис. 2. Здесь индикатриса показана в интервале от 10° до 70° и с интервалом 1°, в формате, как она и будет использоваться в работе.

Моделирование рассеяния света клетками

Поиск характеристик пролетевшей клетки по измеренному сигналу является достаточно сложной обратной задачей. Для ее решения рассмотрим сначала более простую прямую задачу, а именно расчет сигнала светорассеяния при известных характеристиках клетки.

Каждая клетка является достаточно сложным биохимическим комплексом и ее моделирование обычно осуществляется в некотором приближении, где самые важные для текущей задачи части моделируются с помощью простых физических примитивов с заданными физическими свойствами (такими как показатель преломления материала или геометрическая форма объекта) [20,21].

Задача моделирования взаимодействия клетки и электромагнитного излучения (включая и светорассеяние) может решаться набором различных методов. В данной работе, используется база данных, полученная с помощью метода аппроксимации структуры клетки через набор дискретных диполей

примитивной формы [22]. Данный подход мог быть реализован и путем использования любого другого известного метода, таких как теория Ми [23] или метод Т-матриц.

Для получения базы данных использовалась одна из популярных реализаций метода дискретных диполей – программный пакет ADDA [8]. Его разработка велась международной командой разработчиков с участием сотрудников лаборатории Цитометрии и Биокинетики ИХКГ СО РАН. Одним из преимуществ этой реализации является полное использование принципа открытого исходного кода.

Для тестов работы алгоритма использовалась большая база данных, состоящая из 200 тысяч индикатрис, аналогичная используемой в статье [9], подробное описание используемой базы данных можно найти в параграфе «Описание используемых для тестирования данных».

Описание задачи

Текущий метод решения задачи

В настоящее время, для решения описанной задачи поиска ближайшей индикатрисы в лабораторных условиях, применяется достаточно простой метод: для каждого измеренного сигнала светорассеяния мы перебираем все имеющиеся элементы из предварительно построенной базы данных и находим элемент, наиболее близкий по сумме квадратов невязок по всем углам измерения [2] (стандартная евклидова метрика – логичное решение при минимизации нормальных ошибок, полученных из-за неточности модели и/или различных шумов измерения).

После этого, расстояния до теоретических индикатрис, применяя байесовский подход, могут быть использованы для определения вероятностей принятия того или иного значения для экспериментальной индикатрисы. Это можно использовать для оценки адекватности используемой модели, например, через получение статистических оценок полученного значения, таких как математическое ожидание (которое может отличаться от ближайшего) или дисперсия [9].

Вычисление этих параметров является неотъемлемой частью задачи, как для научных дисциплин (что говорит о достоверности полученного в эксперименте значения), так и для медицинской диагностики (оценка группы риска пациента). Потому, любая модификация стандартного алгоритма должна сохранить возможность вычисления этих параметров и, желательно, без значительных дополнительных временных затрат.

Вычисление статистических параметров измеренного значения

Как было сказано в прошлой части, вычисление статистических параметров — очень важная часть задачи, потому отдельно обсудим формулы, используемые для этих вычислений. Для демонстрации формул позаимствуем их из работы [9] с некоторым изменением обозначений согласно нотации, используемой в этой работе. Сначала посмотрим на формулу для вычисления произвольной интересующей нас вероятностной статистики (хоть математического ожидания, хоть дисперсии) при переходе от непрерывного интеграла к дискретному суммированию в предположении достаточно высокой плотности базы данных:

$$\langle f(\boldsymbol{\beta})\rangle \stackrel{\text{def}}{=} \int_{\beta} \mathrm{d}\boldsymbol{\beta} f(\boldsymbol{\beta}) P(\|\mathbf{y}-\mathbf{g}(\boldsymbol{\beta})\|) \approx \sum_{i} f(\boldsymbol{\beta}_i) P(d_i)\,. \tag{3}$$

Тут f – это некоторая функция, которую мы хотим оценить для интересующих нас характеристик частицы $\boldsymbol{\beta}$, а $P(d_i)$ – условная вероятность частицы иметь характеристики $\boldsymbol{\beta}_i$, зависящая от расстояния d_i от экспериментального сигнала y до соответствующего теоретического сигнала $\mathbf{g}(\boldsymbol{\beta}_i)$. Она вычисляется следующим образом:

$$P(d_i) = \kappa {d_i}^{-k_{\text{eff}}/2}, \qquad \kappa = \frac{1}{\sum_i {d_i}^{-k_{\text{eff}}/2}}, \tag{4}$$

где параметр k_{eff} играет роль эффективного количество степеней свободы рассматриваемого пространства. Он необходим, так как, например, для базы данных тромбоцитов, описанной ранее, рассматриваемые измерения света в разные углы не являются независимыми с точки зрения отклонений экспериментального сигнала от теоретического (т.е. отклонения не вызваны

только белым шумом). Поэтому ранее было предложено [2] приближать эти отклонения белым шумом, но с меньшим числом степеней свободы. Формула для вычисления k_{eff} опирается на невязку экспериментальной индикатрисы (отклонение от ближайшей теоретической), и в тестовой задаче приблизительно равна 10. Подробнее о вычислении этого коэффициента можно почитать в работах [2,9].

Цель работы

Цель работы состоит в ускорении процесса перебора элементов предварительно построенной базы данных, путем использования иерархической структуры данных. С точки зрения простоты реализации и экономии памяти для хранения элементов предлагается использовать классическое kd-дерево [24] (хотя имеется возможность использования других видов деревьев, что может является темой отдельного исследования), которое позволяет нам хранить необходимые элементы в базе данных изначального размера и индексироваться по ней в бинарном массиве (псевдо-дереве). Простота и элегантность этого решения позволяет нам обрабатывать базы данных большего размера.

Таким образом, при вычислении расстояний, мы считаем расстояние не до каждого элемента, а до вложенных друг в друга кластеров, постоянно опускаясь на уровень ниже (далее процесс будет объяснен в деталях), что будет позволять нам исключать из просмотра точки целыми кластерами. Таким образом, мы сокращаем перебор элементов, уменьшая количество сравнений и заменяя в формуле вычисления статистик вероятности на взвешенные вероятности кластеров по количеству точек в них. В итоге, мы получаем формулу:

$$\sum_{i\in A_k} f(\boldsymbol{\beta}_i)P(d_i) \approx M_k f(\boldsymbol{\beta}'_k)P(d'_k), \qquad \boldsymbol{\beta}'_k \stackrel{\text{def}}{=} \frac{1}{M_k}\sum_{i\in A_k}\boldsymbol{\beta}_i\,, \tag{5}$$

где A_k – отброшенный кластер, а M_k – количество его элементов. Далее мы в деталях рассмотрим процесс построения и обхода дерева. Здесь и далее используется обозначение

$$d'_j \stackrel{\text{def}}{=} \|\mathbf{c}_j - \mathbf{y}\|, \tag{6}$$

используемое для обозначения расстояния от измеренного сигнала $\mathbf{y}$ до центра некоторого кластера $\mathbf{c}_j$.

Ускорение за счет кластеризации.

Описание нового метода

Каждая индикатриса может быть рассмотрена как точка в p-мерном пространстве (например, для базы данных с p измерениями, где $p = 61$, мы имеем точки в пространстве $\mathbb{R}^{61}$). Все полученные с помощью моделирования точки занимают некоторое многообразие значительно меньшей размерности (в случае базы данных тромбоцитов, определяемых четырьмя характеристиками, размерность будет 4). Наша задача – поиск расстояния от измеренной точки до этого многообразия и оценка достоверности полученного результата. Желательно при этом делать это как можно быстрее и без потери в точности полученного решения. Как и было сказано ранее, для ускорения поиска ближайшей индикатрисы предлагается построение дерево-подобной иерархической структуры данных на теоретически смоделированных индикатрисах.

Рассмотрим построение данной структуры. Каждый узел построенного дерева будет содержать некоторое количество привязанных к нему индикатрис – точек многомерного пространства, центр и радиус шара, в который они помещаются в $\mathbb{R}^p$. Центр кластера вычисляется как среднее всех точек, а его радиус – как расстояние от центра до самой дальней точки. Каждая точка дерева (т. е. кластер точек из базы данных) имеет 2 потомка (кластера меньшего размера) с количествами точек вдвое меньшим (± 1 точка). Построение kd-дерева проводится сверху вниз с разделением пополам всех точек текущего кластера по координате, имеющей наибольший разброс для этих точек.

Алгоритм обхода дерева для измеренной точки начинается с корня дерева, который мы помещаем в стек нерассмотренных узлов. Далее для каждого узла-кластера в стеке мы проверяем, пересекает ли измеренная точка с возможным максимальным радиусом (изначально – бесконечным) границы кластера. Если нет, то кластер точно не содержит ближайшую к измеренной точку, поэтому мы отмечаем этот кластер как просмотренный и сохраняем расстояние до него и количество точек в нем для дальнейшего статистического анализа. То же мы делаем, если кластер имеет всего одну точку. Кластеры, состоящие более чем из одной точки и имеющие шанс содержать искомую ближайшую точку, мы разбиваем на 2 кластера меньшего размера и кладем эти новые кластеры в стек. Также мы пересчитываем максимальную возможную дистанцию до точки как расстояние до дальней границы кластера.

Так мы исключаем некоторые точки из рассмотрения целыми кластерами, значительно ускоряя процесс поиска ближайшего. Кроме того, полученный массив из сохраненных кластеров и единичных точек должен быть значительно меньше первоначальной базы данных, и может быть использовано как некоторая аппроксимация рассмотренного пространства, которая нам позволит провести более быстрый расчет статистических параметров.

Теоретическое обобщение

Вводим множество, состоящее из всех заранее известных теоретических сигналов $X = \{\mathbf{x}_1, \mathbf{x}_2, \dots \mathbf{x}_n\}$, где n – размер базы данных. Решение обратной задачи мы будем искать для некоторого $\mathbf{y}$ – измеренного сигнала. Каждый из объектов $\mathbf{x}$ и $\mathbf{y}$ состоят из p вещественных чисел, т. е. $\mathbf{x}, \mathbf{y} \in \mathbb{R}^p$ (типичный

пример приведен на Рис. 2). Наша цель – для каждого $\mathbf{y}$ найти ближайший из $\mathbf{x}_i$ и соответствующее расстояние между ними, т.е.

$$h_{\min} \stackrel{\text{def}}{=} \min_i d_i\,, \qquad d_i \stackrel{\text{def}}{=} \|\mathbf{x}_i - \mathbf{y}\|. \tag{7}$$

Также, имея все значения d_i (или их аппроксимации d'_j до неких $\mathbf{c}_j$, полученных как центры сохраненных кластеров), мы считаем статистические параметры в соответствии с Ур. (3). Рассмотрим построение самого kd-дерева.

Построение:

Пусть дерево-граф T имеет множество узлов V. Дерево T предполагает бинарную структуру. Каждый узел дерева v_j имеет привязанные к нему индексы $S_j = \left\{k_1^j, k_2^j, \ldots k_{n_j}^j\right\}$ (номера сигналов из X в количестве n_j), а также $\mathbf{c}_j$ и r_j. $\mathbf{c}_j$ – это центр кластера, а r_j – радиус кластера. Для их вычисления, в случае использования простого kd-дерева, применяются следующие формулы:

$$\mathbf{c}_j = \frac{1}{n_j} \sum_{k \in S_j} \mathbf{x}_k, \tag{8}$$

$$r_j = \max_{k \in S_j} \left\| \mathbf{c}_j - \mathbf{x}_k \right\|. \tag{9}$$

Дерево строится от корня вниз итеративно. Для каждого следующего узла мы повторяются те же операции, что для его родителя, кроме кластеров-узлов из одного элемента.

Первым создается корень дерева v_0, для которого $S_0 = \{1,2, \ldots, n\}$, т.е. $n_0 = n$, и вычисляем $\mathbf{c}_0$ и r_0. Для всех остальных узлов дерева мы имеем бинарную структуру, где потомки каждого узла v_j – это v_{2j+1} и v_{2j+2}, для которых множества S_{2j+1} и S_{2j+2} удовлетворяют следующим формулам:

$$S_{2j+1} \cup S_{2j+2} = S_j, \qquad S_{2j+1} \cap S_{2j+2} = \emptyset, \tag{10}$$

так, что $\left|n_{2j+1} - n_{2j+2}\right| \leq 1$.

Способ разделения объектов S_j осуществляется следующим образом. Для всех объектов индексируемых элементами S_j мы смотрим на каждую из координат (обозначенные нижним индексом l) и среди них выбираем $l_{j,\max} = \underset{l}{\operatorname{argmax}}[\max_{k \in s_j} x_{k,l} - \min_{k \in s_j} x_{k,l}]$, т. е. ту координату, для которой разброс значений по ней среди всех объектов кластера наибольший ($x_{k,l}$ – это l-я координата сигнала $\mathbf{x}_k$). А далее находим значение медианы

$$m_j = \operatorname{med}_{k \in S_j} x_{k,l_{\max}} \tag{11}$$

и проводим разбиение множества значений $x_{k,l_{\max}}$ по этому медианному значению. Проводимую процедуру можно описать следующим правилом:

$$S_{2j+1} = \left\{k \in S_j \middle| x_{k,l_{\max}} \leq m_j\right\}, \;\; S_{2j+2} = \left\{k \in S_j \middle| x_{k,l_{\max}} > m_j\right\}. \tag{12}$$

Для S_k состоящих из одного элемента ($n_q = 1$) мы получаем $\mathbf{c}_k = \mathbf{x}_k$, а $r_k = 0$. Эти значения согласуются и с формулами выше.

Обход дерева:

Решение задачи происходит независимо для каждого **y**, т.е. ищем решение, определенное в Ур. (7). Так для каждого обхода мы вводим множество запланированных к просмотру вершин графа Q (изначально в нем только v_0), множество сохраненных для вычисления вероятностей кластеров W (изначально пустое) и текущее минимальное расстояние h до элементов из X (изначально ∞, по ходу поиска сходится к искомому минимуму $h_{\min}$).

Мы запускаем итеративный алгоритм просмотра объектов из Q и заполнения W новыми элементами. Далее я опишу процедуру для следующего рассматриваемого элемента из Q. Его выбор может быть реализован разными способами, как FIFO (first in, first out), FILO (first in, last out), так и элемента, с наименьшим расстоянием d_j'. В работе я остановился на последнем варианте, как на потенциально быстрее находящем наименьшее значение, и поэтому быстрее отбрасывающем кластеры, а значит, более быстрым.

Таким образом, мы берем $v_j \in Q$ (с удалением его из этого множества), если для него выполняется выражение

$$h < d_j' - r_j, \tag{13}$$

то есть ближайший элемент из X точно ближе, чем весь просматриваемый кластер. В этом случае мы сохраняем множество S_j и расстояние до центра d_j' в массив W для будущего расчета вероятностей.

В обратном случае, для $h \geq d_j' - r_j$, если просмотренный узел имеет детей ($n_j > 1$), то мы добавляем его детей (v_{2j+1} и v_{2j+2}) в Q для дальнейшего просмотра; если же у него нет детей, это кластер из одного элемента и мы его добавляем в массив W для будущих вычислений вероятностей.

Дополнительно, если выполняется $h > d_j' + r_j$, мы присваиваем $h = d_j' + r_j$ (точно есть хотя бы одна точка на расстоянии не более h) и сохраняем кластер как ближайший.

В ходе этого процесса мы получим массив W из элементов, необходимых для расчета вероятностей в пространстве, и ближайший сохраненный кластер (из одного элемента, т.е. ближайшую точку).

Про степень отброса кластеров

Отдельно хочется обсудить идею возможности захода в кластеры, которые могли бы быть отброшены целиком, игнорируя правила отброса кластеров в Ур. (13). Анализ таких кластеров не поможет получить более точного решения, но поможет получить статистические параметры решения с меньшей ошибкой.

Глубину обхода кластеров можно задавать разными способами. В этой работе предлагается использовать в качестве такого уровня коэффициент $q \in [0, 1]$, который влияет на алгоритм следующим образом: помимо формулы (13) мы дополнительно проверяем и следующее неравенство:

$$qd_j' < r_j, \tag{14}$$

где r_j – радиус просматриваемого сейчас кластера. При соблюдении этого неравенства мы, игнорируя прошлое правило отброса, заходим в рассматриваемый кластер. Так, при значении $q = 0$, мы просматриваем каждый из кластеров, а при $q = 1$ – Ур. (14) всегда мажорируется неравенством $h \leq d_j' + r_j$, а потому не ставит дополнительных ограничений на алгоритм.

Логически это неравенство выступает в качестве контроллера работы с числами одного масштаба. Так, при наличии кластера большого радиуса, но при известном небольшом порядке расстояния до измеренной точки, кластер не будет отбрасываться целиком, а разобьется на кластеры меньшего размера. Таким образом, мы не вносим погрешности в связи с огрублением вычислений

на порядки (разумеется, это не убирает все подобные ошибки, но значительно уменьшает их количество). В параграфе «Анализ глубины обхода» мы рассмотрим влияние этого коэффициента на точность и скорость работы алгоритма.

Реализация и тестирование

Выбор технологий и компромиссы

Алгоритмически я использовал реализацию kd-дерева, базирующуюся на коде открытой библиотеки sklearn [25] (стандартная библиотека для обработки данных и для задач машинного обучения в языке Python). К сожалению, ее прямое использование было невозможно, в связи с необходимостью иметь расширенный вывод данных (с расстояниями до кластеров) и совместимость с кодом из среды разработки LabView IDE, используемого в лаборатории. А также, хотелось иметь возможность гибко отбрасывать кластеры, контролируя степень отброса, что позволяет нам подбирать оптимальный баланс между точностью статистических параметров и временем работы алгоритма.

Альтернативным методом построения схожей кластерной структурой могло бы являться использование методов индексации стандартных реляционных базах данных (вроде MySQL). Но их использование достаточно сложно, как для построения индексации (в этой задаче индексация проводится не по одному столбцу базы данных, а по функции от всех столбцов), так и для проведения запросов (они должны выдавать не один найденный элемент, а все отброшенные). Так же мы лишаемся возможности гибкой настройки уровня обхода графа. Но, теоретически, при использовании некоторых кустарных методов решения описанных проблем (например, построения индексации, не базирующейся на стандартных функциях SQL) и принятии некоторых компромиссов (таких, как отказ от гибкости обхода), этот метод является потенциальным расширением работы, для обработки баз данных размера,

превышающего размеры оперативной памяти и имеющих дорогой случайный доступ к памяти.

Для реализации моего решения использовался язык C++ со средой разработки Microsoft Visual Studio. Решение оформлено в виде dll библиотеки для вызова необходимых функций (эффективно реализованных на языке низкого уровня) через интерфейс LabView IDE. Код программы организован в виде 2х функций, а именно – функции построения дерева по имеющейся базе данных и функции для его обхода.

Первым, перед началом работы алгоритма, строится дерево с переданными в dll значениями функции (также имеется возможность использовать ранее построенное дерево из памяти на диске компьютера). Далее, построенное дерево, без постоянной выгрузки и загрузки в долгосрочную память, используется второй функцией, что возможно осуществлять в рамках одного вызова dll внутри процесса. Вторая функция занимается обходом дерева и возвращает в заранее подготовленный массив информацию о сохраненных ей кластерах (отключаемая функция, что убыстряет вычисления) и ближайшую найденную точку.

Описание используемых для тестирования данных

Для тестирования алгоритма использовались 2 набора данных: большая база данных для построения кластерной структуры данных и набор тестовых данных.

Большая база данных – это база данных посчитанная с помощью ADDA[8], используемая в настоящий момент в лаб. Цитометрии и Биокинетики для решения обратной задачи светорассеяния на тромбоцитах. Ранее эта база данных

использовалась при выполнении исследований [16]. В этой базе данных, для описания теоретических моделей частиц, используется их аппроксимация физическими сфероидами с заданными четырьмя случайными характеристиками, взятыми из равномерного распределения: радиус шара того же объема (от 0.5 до 2.12 мкм), отношение полуосей (от 0.1 до 1), показатель преломления (от 1.37 до 1.39) и угол ориентации клетки в момент ее облучения (от 0 до 90 градусов). Данная база данных была разбита на 2 части: основная база данных из 198 847 индикатрис и тестовый набор из 1000 индикатрис. Первая использовалась для построения иерархической структуры и решения задачи характеризации во всех дальнейших тестах, а вторая в качестве теоретических индикатрис для тестирования алгоритмов.

Решение задачи находилось на двух небольших наборах данных, состоящих из 1000 индикатрис каждый. Теоретические (тестовые) индикатрисы были выделены из основной базы данных, но они независимы от оставшейся части (описанной выше) ввиду случайной процедуры ее построения. Экспериментальные индикатрисы были получены с помощью измерения клеток донора на СПЦ. При проведении измерений в установке использовался лазер с длиной волны 660 нм. Эти же тестовые экспериментальные индикатрисы использовалась в работе [10].

Предварительные тесты

Первым делом была проверена корректность прохождения алгоритма по базе данных и реальное уменьшение количества сравнений (вычислений расстояний между точками). Для этого мы использовали готовую реализацию

библиотеки на языке Python 3 (конкретно – kd-tree из библиотеки sklearn [25]), в которой, путем фиксации параметра просмотра всех элементов из дерева, можно заставить алгоритм как активно отбрасывать кластеры, так и проверять каждый из них, как в классической версии оптимизируемого алгоритма.

Рис. 3 показывает полученное ускорение количества просмотренных элементов. Тестирование проводилось отдельно для поиска тромбоцитов, полученных путем моделирования и для измеренных на реальной установке. Как видим, количество сравнений зависит как от типа тестовых индикатрис, так и от размера базы данных. Полученные зависимости сложно объяснить, хотя и получилось найти для обоих рассмотренных случаев приблизительно коренную зависимость, которая, однако, остается гипотетической.

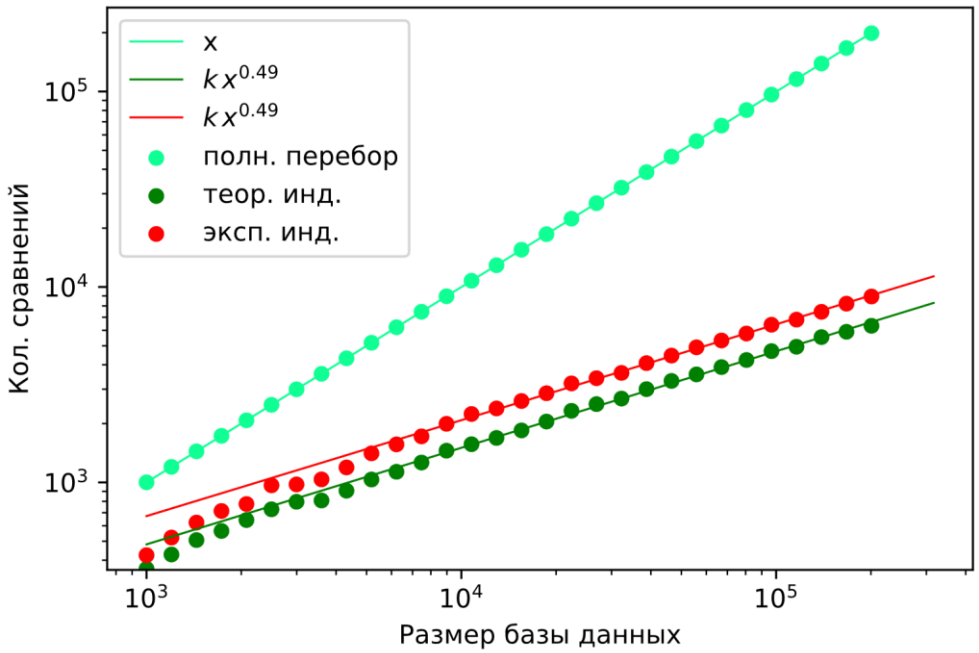

Рис. 3 Зависимость количества сравнений, необходимых для поиска ближайшей индикатрисы, от размера используемой базы данных теоретических индикатрис и ее аппроксимация степенной функцией. Результаты усреднены по 1000 тестовых индикатрис, график в логарифмическом масштабе по обеим осям.

Так же было проведено тестирование реального ускорения, благодаря использованию kd-дерева с готовым кодом из sklearn, но без вычисления статистических параметров и не в реальной системе анализа лаборатории (LabView). Их вы можете видеть на Рис. 4. Замеры были проведены несколько раз, дабы получить усредненное значение временных затрат (уменьшить влияние планировщика задач и текущей загруженности вычислительной машины за счет фоновых задач).

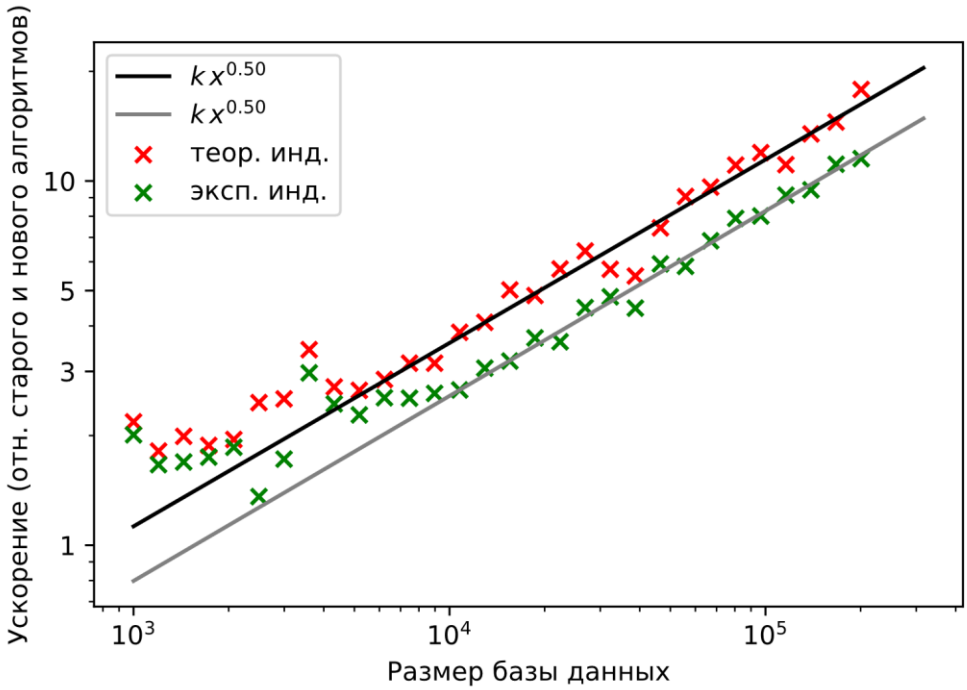

Рис. 4 Отношение времени работы и его аппроксимация степенной зависимостью (подгонка методом наименьших квадратов) в логарифмическом масштабе по обеим осям. Результаты усреднены по 1000 тестовых индикатрис.

Для более полного понимания зависимостей полученных скоростей, я предлагаю взглянуть на сравнительный график полученного ускорения по времени и по количеству проведенных сравнений. Как вы можете видеть, предварительные тесты для базы данных размера 2×10^5 индикатрис

предсказывают нам ускорение по количеству сравнений около 20 раз и около 10 раз – реальное ускорение работы (Рис. 5).

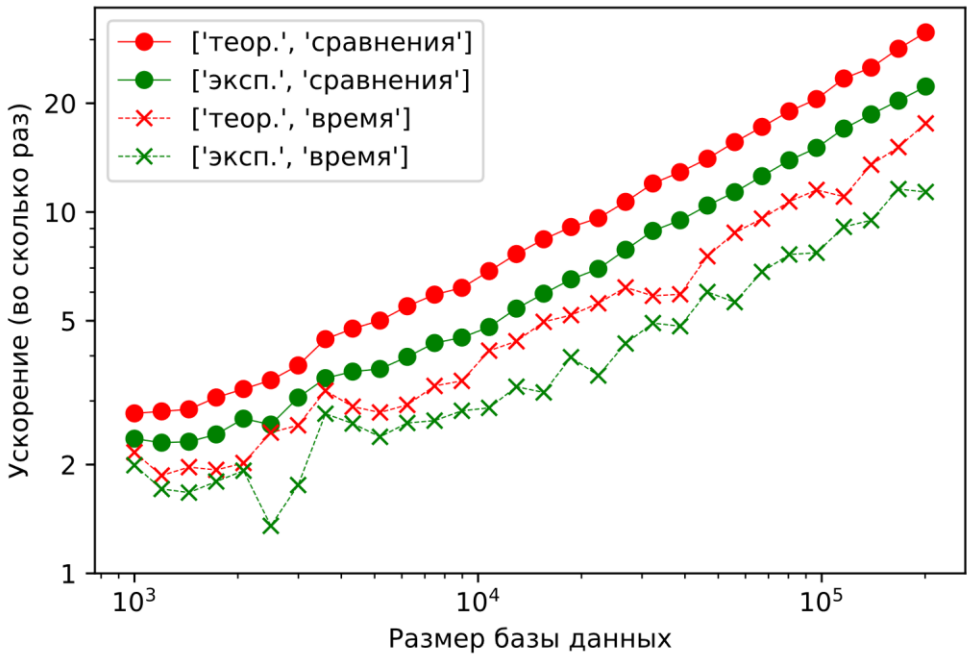

Рис. 5 Полученное ускорение в зависимости от размера базы данных, в логарифмическом масштабе по обеим осям.

Тестирование нового алгоритма

Изначально, моя реализация библиотеки была сравнена с результатами из Python/sklearn. Полученные на одних и тех же данных результаты поиска ближайших элементов с этой библиотекой и результаты моего алгоритма полностью совпадают.

Аналогичное тестирование статистических параметров было невозможно, т.к. в реализации sklearn отсутствует вывод расстояний до отброшенных кластеров, но этот функционал был протестирован сравнительно с исходным кодом классического алгоритма и моего алгоритма с установленным минимальным уровнем отброса кластеров, что обеспечивало заход во все кластеры ненулевого радиуса (во все кластеры больше одной точки, т. к. точки

не повторяются). Таким образом, я протестировал совпадения статистических оценок решения старого алгоритма и нового алгоритма в предельном случае, сохранив совместимость решений. Дополнительно я проверил, что при значении коэффициента отброса кластеров (подробнее об анализе этого коэффициента в параграфе «Анализ глубины обхода») количество выданных кластеров равно количеству имеющихся точек в базе данных.

Далее я протестировал возможность интеграции написанного на C++ кода в среду LabView. Полученный алгоритм всегда совпадает в выданной ближайшей индикатрисе с оригинальным методом, как и должно было быть теоретически, т. к. при обходе отбрасываются только заведомо удаленные кластеры.

Реальное ускорение системы для решения обратной задачи светорассеяния, путем добавления в решение иерархической структуры для обхода данных можно посмотреть на Рис. 6. Там вы можете видеть ускорение для двух основных режимов работы программы: с вычислением статистических погрешностей и без него. Дополнительно замечу, что функционал программы более широк (например, мы можем строить графики для найденных решений, что будет дополнительно увеличивать время работы) и потребности исследователя могут быть значительно шире, но мы проверяем ускорение программы при минимальном функционале. Итоговое ускорение при максимальном размере текущей используемой базы данных 2×10^5 индикатрис достигает, практически, 40 раз. Обратим внимание, что полученные зависимости имеют зависимость близкую к линейной, что говорит об ускорении реального

кода на еще более значимую величину, чем это предсказывалось по количествам сравнений в предыдущем разделе. Возможно, это достигается за счет большей скорости работы dll относительно кода из LabVIEW, так как LabVIEW является языком достаточно высокого уровня и, потенциально, может иметь более низкий уровень оптимизации работы с памятью.

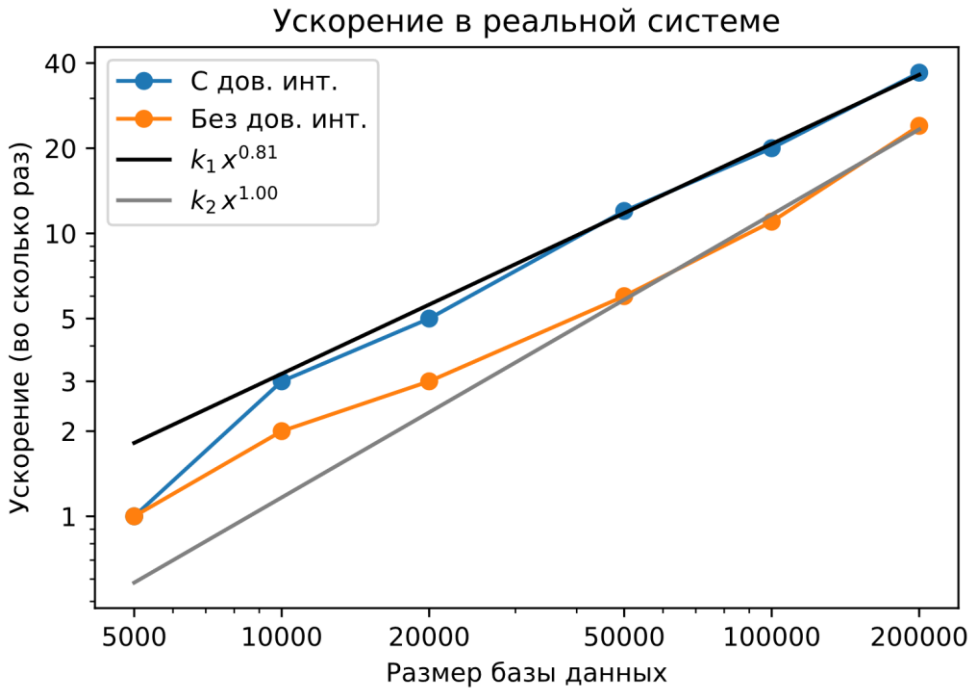

Рис. 6 Ускорение работы алгоритма при использовании иерархической структуры в программе LabVIEW для обработки 1000 экспериментальных индикатрис, график в логарифмическом масштабе по обеим осям.

Сравнение реальных и теоретических времен работы

Посмотрим на результаты работы алгоритма с точки зрения фактического времени его работы. Для получения теоретических временных оценок использовалась следующая формула:

$$t = \frac{pk_{\text{op}}n_{\text{comp}}}{\omega}. \tag{15}$$

Где p – размерность пространства (61), n_{comp} – количество сравнений, ω – частота процессора компьютера, а k_{op} – время, затрачиваемое на работу с одним числом при сравнении расстояния между двумя векторами, включая все необходимые операции, в том числе и чтение из памяти/кеша. Данная формула предполагает, что основные потери времени работы алгоритма происходят именно из-за большого количества сравнений и связанной с этим работы. Также, мы, не зная реального количества тактов требуемого для операции вычитания, возведения в квадрат, а так же, что самое долгое, чтения из кеша, оценим параметр k_{op} числом 15, что является средним количеством тактов, необходимых для чтения информации из кеша второго уровня (10 тактов) и проведения 5и простых операций по одному такту (5 – тоже оценочное число операций на одно сравнение) [26]. Разумеется, это число оценивает лишь порядок необходимого количества операций.

На Рис. 7 мы видим эти оценки, а также и время выполнения алгоритма на ноутбуке Dell Inspiron 5555 с процессором AMD A10-8700P частотой 1.8 Ггц с 8Гб оперативной памяти. Время работы, около десятка миллисекунд, подводит нас к возможности проведения экспериментов в режиме онлайн (при измерении на СПЦ не более 100 частиц в секунду), даже при проведении экспериментов на недорогой технике.

Дополнительно стоит заметить, что в описанных тестах не учитывалось время, потраченное на предварительное построение дерева (до 30 секунд) и/или время, затраченное загрузку уже построенного дерева из памяти компьютера

(около 2 секунд). Последнее пренебрежимо мало при обработке 1000 и более индикатрис за раз.

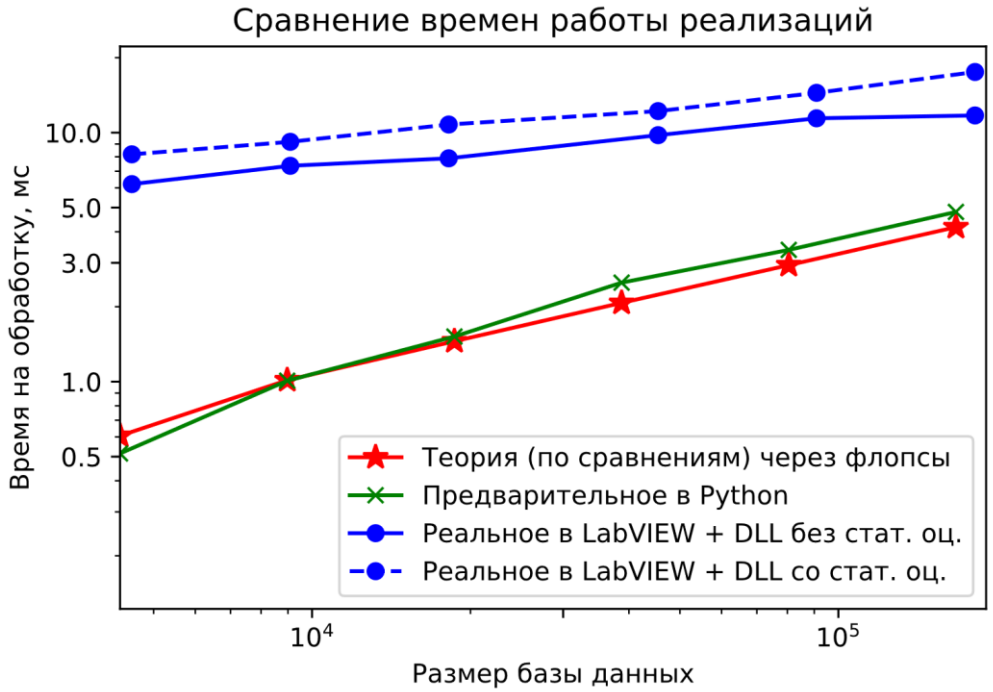

Рис. 7 Время работы нового алгоритма без учета вспомогательных операций: теоретическое (через оценку числа сравнений), предварительное (получено в Python) и реальное, на экспериментальных индикатрисах (в LabVIEW). График построен в логарифмических координатах по обеим осям,

Скорость работы алгоритма в минимальном режиме (без вычисления статистик и построения графиков) совпадает в реализации Python и при оценке по сравнениям через скорость работы процессора. Это дает нам право предположить (хотя и гипотетически, в связи с приближенностью таких оценок), что это и является реальным возможным временем работы алгоритма.

В самой же используемой при тестах системе, полученное время работы программы для небольшого размера базы данных различается от теоретического примерно в 10 раз, а при базах данных большого размера уже всего в 4 раза, что говорит о том, что, с ростом базы данных, мы приближаемся к реально

возможной скорости работы алгоритма. Более того, при еще большем размере базы данных, потенциально, продолжая тренд полученных графиков, можно предположить, что итоговое ускорение сравнится, с полученным в Python. Различие в скорости работы реализации в LabVIEW возможно происходит из-за неоптимальности исполнения кода языком высокого уровня – LabVIEW, в том числе при вызове из него dll.

Анализ глубины обхода и уровня ошибок алгоритма

Вычисление среднего значения и доверительного интервала для найденной индикатрисы – вторая важная часть задачи. Вычисление этих параметров проводится с использованием вероятностей для частицы иметь то или иное значение характеристики, которые можно получить с помощью Ур. (3). Разумеется, с использованием аппроксимации через кластеры, мы имеем несколько искаженную картину пространства возможных значений для точек (все K точек из кластера заменяются на K точек, находящихся в центре кластера, т. к. все индикатрисы заменяются на усредненную индикатрису и на ее усредненные характеристики, согласно Ур. (5)). По этой причине мы имеем некоторую погрешность в искомых статистических параметрах. Рассмотрим их величину подробнее в зависимости от глубины обхода дерева.

На графиках (Рис. 8 и Рис. 9) зависимости ошибки вычисленных математического ожидания и стандартного отклонения для разных характеристик тромбоцитов для 1000 экспериментальных индикатрис (приведены средние значения и полный интервал, куда попадают все полученные значения).

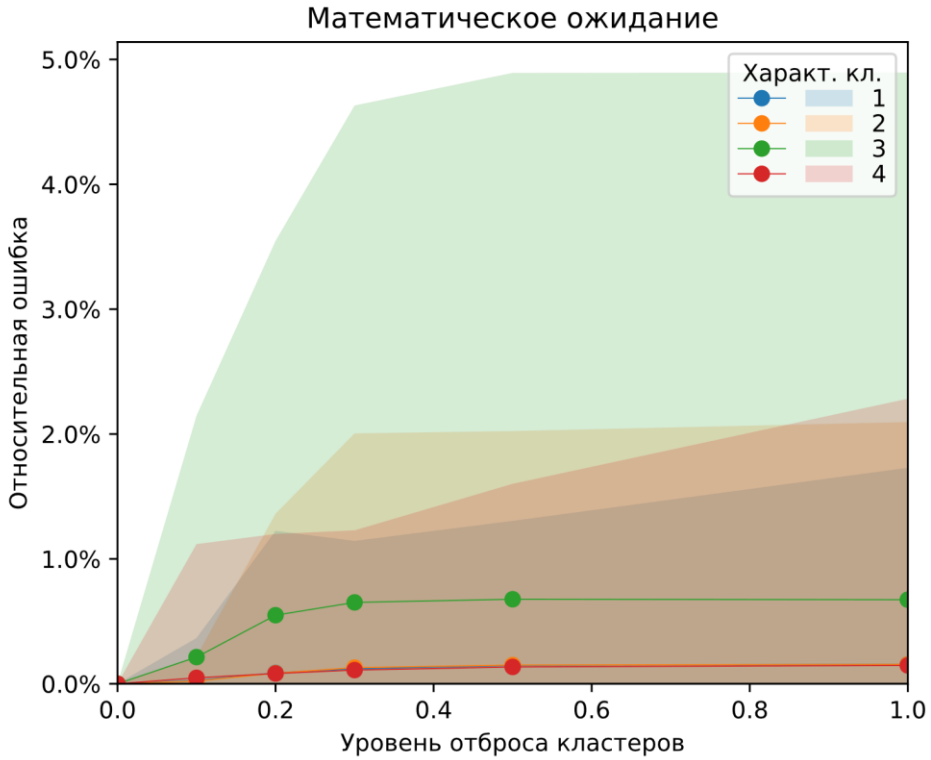

Рис. 8 Относительная ошибка (ее среднее и полный интервал по 1000 экспериментальным индикатрисам) при вычислении математического ожидания 4 характеристик от уровня отброса кластеров q. Характеристики: 1 – радиус шара равного объема, 2 – отношение полуосей сфероида, 3 – показатель преломления, 4 – ориентация клетки в пространстве. (аналогично на рисунке ниже).

На рисунках эти статистические параметры представлены для четырех характеристик частиц, таких как радиус шара равного объема, отношение полуосей, показатель преломления и угол ориентации клетки в момент ее облучения. Их значения расположены в промежутках, имеющих физический смысл при моделировании тромбоцитов человека. Подробное описание базы данных было в параграфе «Описание используемых для тестирования данных»). Как мы видим, имеется некоторое отклонение в результатах вычисления статистических параметров полученного решения от классического алгоритма.

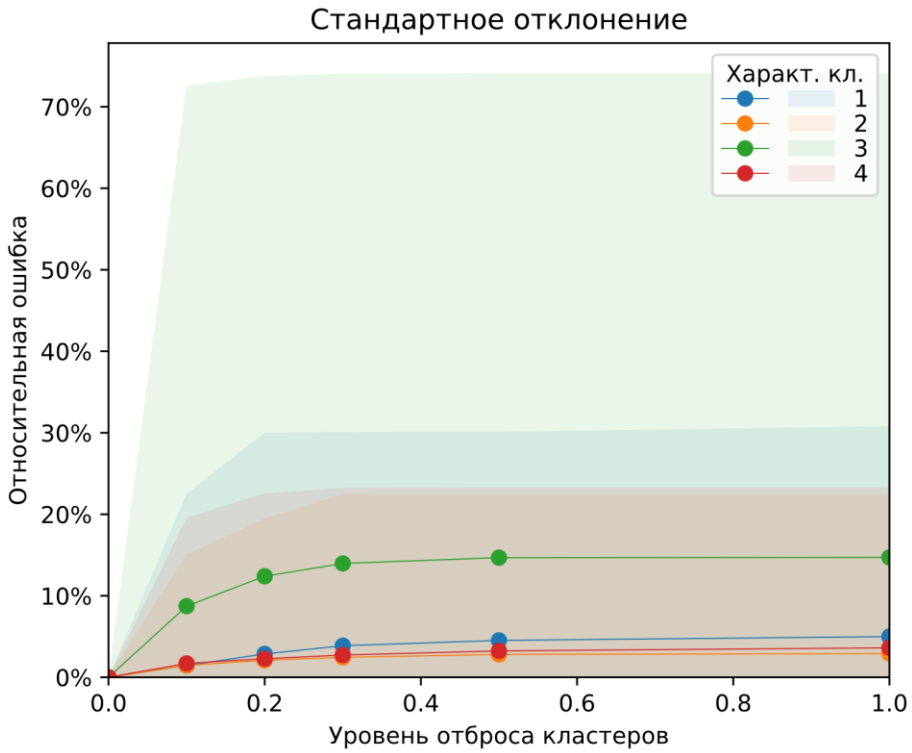

Рис. 9 Относительная ошибка (ее среднее и полный интервал по 1000 экспериментальным индикатрисам) при вычислении стандартного отклонения характеристик в зависимости от уровня отброса кластеров q.

Средние ошибки при вычислении математического ожидания характеристик составляют не более 1%, для стандартного отклонения – не более 15%. Стоит учесть, однако, что полученные измерения имеют достаточно широкий диапазон. Но даже с учетом этого, погрешность для математического ожидания меньше 5%, что уже дает неплохую точность (также см. Рис. 12 ниже); а для стандартного отклонения не более 75%, что дает нам, в свою очередь, верный порядок оценки (достаточная точность для некоторых задач). Заметим также, что это экстремальные значения для характеристики 3 (показатель преломления), которая в принципе не определяется надежно в данной экспериментальной задаче [9,16], а для других характеристик мы имеем более надежную работу полученного алгоритма.

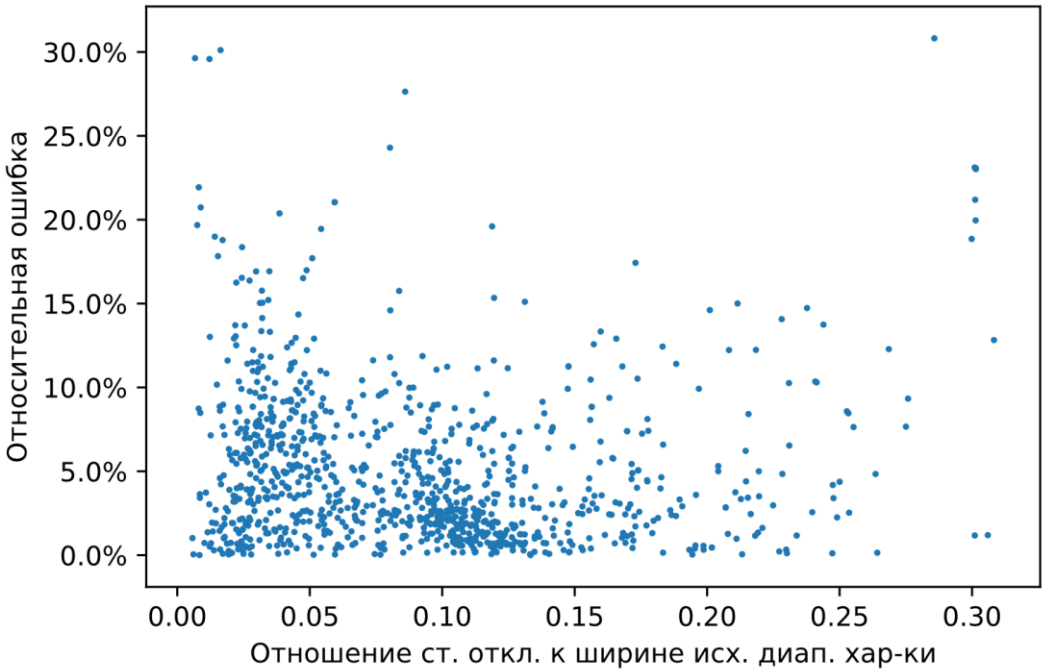

Рис. 10 Зависимость относительной ошибки стандартного отклонения характеристики 1 (радиус шара равного объема) от отношения этого стандартного отклонения к ширине исходного диапазона данной характеристики. Каждая точка соответствует одной из 1000 экспериментальных индикатрис.

Для количественной демонстрации этой особенности предлагаю взглянуть на графики распределения всего тестового набора по полученным ошибкам стандартного отклонения в зависимости от отношения самой величины стандартного отклонения к ширине исходного диапазона рассматриваемой характеристики, используемого при создании базы данных (Рис. 10 и Рис. 11). Значение последней величины 0.25 соответствует ширине доверительного интервала при определении этой характеристики (грубо оцениваемая как 4 стандартных отклонения) равной ширине исходного диапазона. Т.е. это характерное значение, когда данная характеристика надежно не определяется при решении обратной задачи.

На Рис. 10 вы можете увидеть, что для первой характеристики (имеющей малые относительные стандартные отклонения для большинства точек, а потому и определяемой достаточно надежно), значения ошибок стандартного отклонения не больше 30% (а для 95% точек – менее 13%), что уже достаточно для большинства возможных применений этих оценок. Важно отметить, что оценки стандартных отклонений характеристик отдельных частиц применяются лишь полу-количественно для контроля надежности решения обратной задачи.

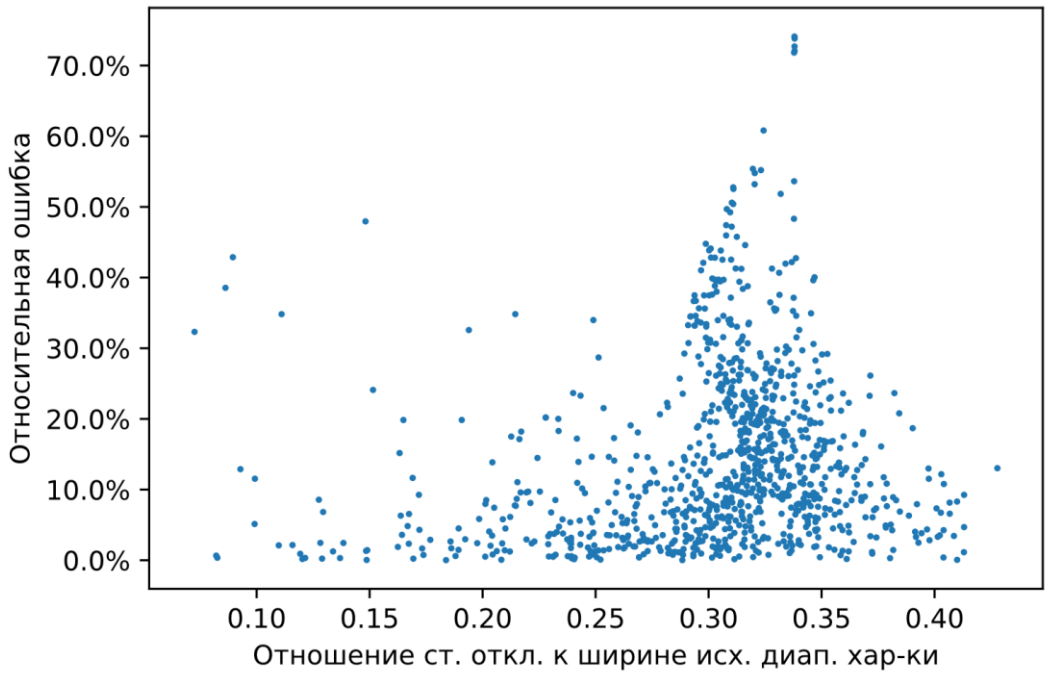

Рис. 11 То же, что и Рис. 10, но для третьей характеристики (показателя преломления).

Но для таких характеристик, как показатель преломления (Рис. 11), большая часть частиц имеет очень большое относительное стандартное отклонение. Именно для этих частиц достигаются наибольшие значения ошибок в вычислении стандартного отклонения (до 75%), что, с одной стороны, объяснимо ввиду общей ненадежности решения обратной задачи для этой характеристики, а с другой стороны, неважно, так как для этих частиц дальнейшее использование оценки данной характеристики не имеет смысла.

Если же мы ограничимся частицами в левой части Рис. 11 (относительное стандартное отклонение < 0.25), то ошибки менее 50% (а для 95% из этих удовлетворительных частиц – менее 32%), что сравнимо с результатами для других характеристик.

Также важно, что даже небольшие отклонения в мат. ожидании (как на Рис. 8) могут быть заметны, если характеристики частицы определяются очень точно, т.е. соответствующее стандартное отклонение мало. Поэтому дополнительно посмотрим на зависимость отношения абсолютной ошибки в математическом ожидании и величины стандартного отклонения на Рис. 12. При этом все значения попадают в интервал до 0.16. Значения, полученные в доверительном интервале 95% меньше 0.08, а в среднем меньше 0.02, что говорит о высокой надежности вычисления математического ожидания для всех возможных применений.

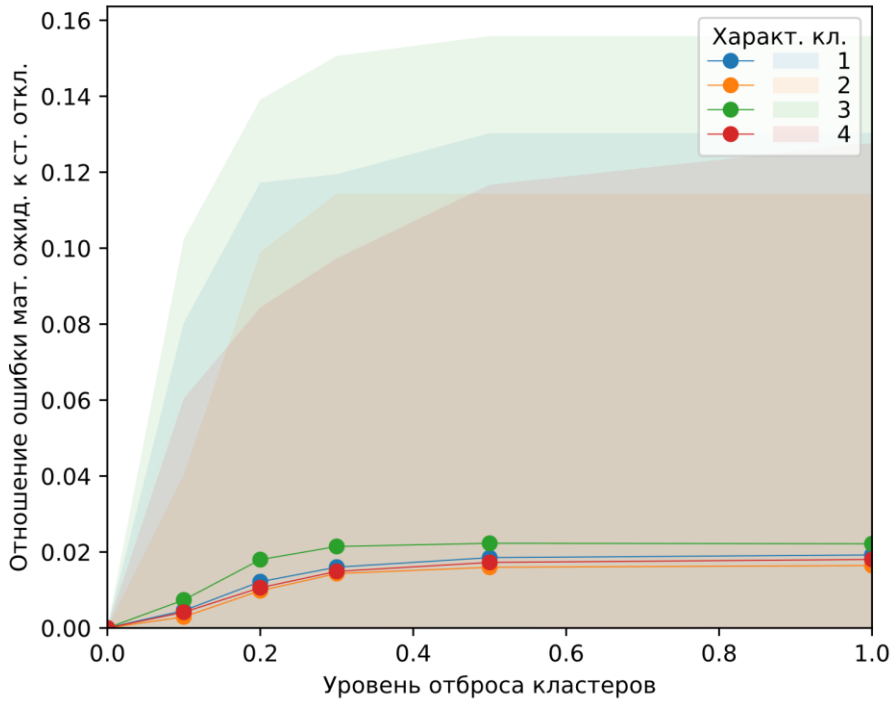

Рис. 12 Отношение мат. ожидания и стандартного отклонения (среднее и полный интервал по 1000 экспериментальным индикатрисам) в зависимости от уровня отброса кластеров q.

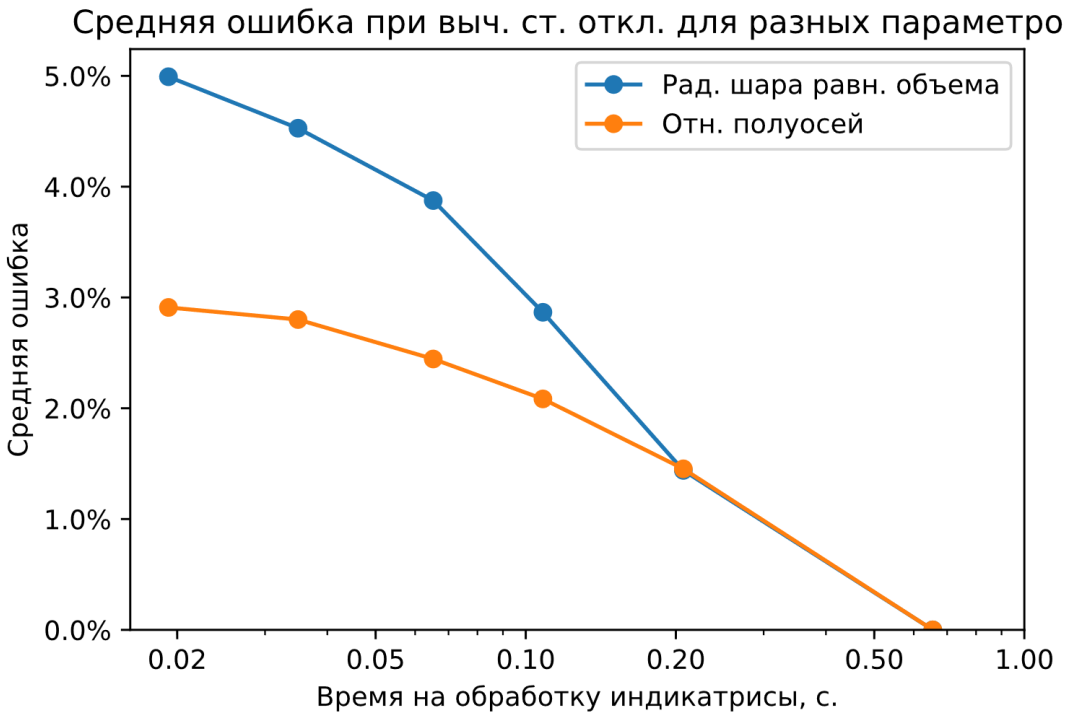

Рис. 13 Относительная ошибка при вычислении стандартного отклонения характеристик (средняя по 1000 экспериментальных индикатрис) в зависимости от времени работы программы, в логарифмическом масштабе по оси абсцисс.

Далее хотелось бы рассмотреть зависимость между точностью алгоритма (в вычислении статистик) и временем его работы. Для этого посмотрим на их зависимость, например, для вычисления стандартного отклонения некоторых характеристик индикатрис (Рис. 13). Как вы можете видеть, замечательное свойство предложенного алгоритма – это возможность выбора между точностью полученных оценок и временем работы алгоритма. Это решение может быть принято самим экспериментатором, в зависимости от необходимой точности решения и имеющихся ресурсов. Правда следует заметить, что график может значительно зависеть от конкретной модели частицы и диапазонов характеристик.

Заключение

В работе была применена иерархическая кластеризация для ускорения решения обратной задачи светорассеяния путем перебора решений прямой задачи (интерполяции методом ближайших соседей по предварительно насчитанной базе данных индикатрис). Предложенный алгоритм позволяет ускорить решение задачи при нахождении того же ближайшего элемента из базы данных и с контролируемыми погрешностями, в зависимости от ресурсного ограничения, для поиска его статистических оценок (таких как математическое ожидание и дисперсия отдельных характеристик). Определены диапазоны полученных погрешностей в зависимости от необходимой скорости работы программы, а также предложен способ контроля этой скорости с помощью введённого коэффициента отброса кластеров.

Полученное решение реализовано в виде кода на C++, собранного в dll библиотеку, успешно встроенную в программу на IDE LabView, используемую для реальной обработки экспериментов на сканирующем проточном цитометре. Работоспособность алгоритма протестирована на реальных данных в этой системе и продемонстрировала значительный прирост скорости, увеличивающийся с размером используемой базы данных. Для максимального размера тестируемой базы данных (2×10^5 индикатрис), ускорение времени работы составило от 20 до 40 раз в зависимости от режима работы.

Для базы данных максимального из рассматриваемых размеров реальное время работы на процессоре AMD A10-8700P с тактовой частотой 1.8 Ггц составило 12 мс для обработки одной индикатрисы в режиме без определения

статистических параметров и 17 мс с их определением. Это приближает нас к возможности проведения экспериментов в режиме онлайн (со скоростью до 100 частиц в секунду) даже при использовании обычного стационарного компьютера/ноутбука.

Полученные отклонения в вычислении статистических параметров достаточны для большинства практических применений. Математическое ожидание вычислялось с относительной точностью не хуже 5% (а в среднем – 1%). Стандартное отклонение определялось с точностью не хуже 75%, а для подавляющего большинства частиц, для которых данная характеристика определяется надежно, – не хуже 32%. Данный уровень точности является достаточным для большинства возможных применений оценок характеристик частиц и их стандартных отклонений.

Отдельно стоит заметить, что выполненный программный продукт позволяет получить решение более широкого спектра физических обратных задач (не ограничиваясь как сигналами светорассеяния, так и областью биологии) с необходимыми статистическими оценками результатов, достаточно высокой скоростью работы и прост в использовании в любой программной среде, позволяющем использование dll библиотек.

Список литературы

1. Chernyshev A.V. et al. Measurement of scattering properties of individual particles with a scanning flow cytometer // Appl. Opt. 1995. Vol. 34. P. 6301–6305.
2. Strokotov D.I. et al. Polarized light-scattering profile - advanced characterization of nonspherical particles with scanning flow cytometry // Cytometry A. 2011. Vol. 79A, № 7. P. 570–579.
3. Fiorani L. et al. Scanning flow cytometer modified to distinguish phytoplankton cells from their effective size, effective refractive index, depolarization, and fluorescence // Appl. Opt. 2008. Vol. 47, № 24. P. 4405–4412.
4. Konokhova A.I. et al. High-precision characterization of individual E. coli cell morphology by scanning flow cytometry // Cytometry A. 2013. Vol. 83, № 6. P. 568–575.
5. Konokhova A.I. et al. Enhanced characterisation of milk fat globules by their size, shape and refractive index with scanning flow cytometry // Int. Dairy J. 2014. Vol. 39, № 2. P. 316–323.
6. Strokotov D.I. et al. Is there a difference between T- and B-lymphocyte morphology? // J. Biomed. Opt. 2009. Vol. 14, № 6. P. 064036.
7. Dyatlov G.V. et al. An optimization method with precomputed starting points for solving the inverse Mie problem // Inv. Probl. 2012. Vol. 28. P. 045012.
8. Yurkin M.A., Hoekstra A.G. The discrete-dipole-approximation code ADDA: capabilities and known limitations // J. Quant. Spectrosc. Radiat. Transfer. 2011. Vol. 112, № 13. P. 2234–2247.

9. Moskalensky A.E. et al. Accurate measurement of volume and shape of resting and activated blood platelets from light scattering // J. Biomed. Opt. 2013. Vol. 18, № 1. P. 017001.
10. Боровкова С.В. Использование кластеризации при решении параметрической обратной задачи светорассеяния: квалиф. работа на соиск. степ. бакалавра. Новосибирск: Новосибирский государственный университет, 2014.
11. Можейко Е.А. Определение погрешностей характеристик объекта при решении обратной задачи светорассеяния с использованием кластеризации: квалиф. работа на соиск. степ. бакалавра. Новосибирск: Новосибирский государственный университет, 2016.
12. Pierzchalski A., Mittag A., Tárnok A. Chapter 1 - Introduction A: Recent Advances in Cytometry Instrumentation, Probes, and Methods: Review // Methods in Cell Biology / ed. Darzynkiewicz Z. et al. Academic Press, 2011. Vol. 102. P. 1–21.
13. Han Y. et al. Review: Imaging Technologies for Flow Cytometry // Lab Chip. 2016. Vol. 16, № 24. P. 4639–4647.
14. Gupta A. et al. Deep Learning in Image Cytometry: A Review // Cytometry Part A. 2019. Vol. 95, № 4. P. 366–380.
15. Vandenbem C., Froufe-Pérez L.S., Carminati R. Fluorescence signal of a single emitter coupled to a nanoparticle through a plasmonic film // J. Opt. A. 2009. Vol. 11, № 11. P. 114007.

16. Litvinenko A.L. et al. Fluorescence-free flow cytometry for measurement of shape index distribution of resting, partially activated, and fully activated platelets // Cytometry A. 2016. Vol. 89, № 11. P. 1010–1016.

17. Maltsev V.P. Scanning flow cytometry for individual particle analysis // Rev. Sci. Instrum. 2000. Vol. 71, № 1. P. 243–255.

18. Пичугин Ю.Г. Возомжности спектрального подхода в анализе индикатрис светорассеяния мононуклеарных клеток: квалиф. работа на соиск. степ. бакалавра. Новосибирск: Новосибирский государственный университет, 2008.

19. Maltsev V.P., Semyanov K.A. Characterisation of Bio-Particles from Light Scattering. Utrecht: VSP, 2004. 132 p.

20. Moskalensky A.E. et al. Method for the simulation of blood platelet shape and its evolution during activation // PLoS Comput. Biol. 2018. Vol. 14, № 3. P. e1005899.

21. Maltsev V.P., Hoekstra A.G., Yurkin M.A. Optics of white blood cells: optical models, simulations, and experiments // Advanced Optical Flow Cytometry: Methods and Disease Diagnoses / ed. Tuchin V.V. Weinheim: Wiley-WCH, 2011. P. 63–93.

22. Yurkin M.A., Maltsev V.P., Hoekstra A.G. The discrete dipole approximation for simulation of light scattering by particles much larger than the wavelength // J. Quant. Spectrosc. Radiat. Transfer. 2007. Vol. 106, № 1–3. P. 546–557.

23. van de Hulst H.C. Light Scattering by Small Particles. New York: Dover, 1981. 479 p.

24. Louis B. Multidimensional binary search trees used for associative searching // Communications of the ACM. 1975.

25. scikit-learn/scikit-learn: Python. scikit-learn, 2020.

26. Agner F. 4. Instruction tables: Lists of instruction latencies, throughputs and micro-operation breakdowns for Intel, AMD and VIA CPUs. Technical University of Denmark, 2019.