МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

МЕХАНИКО-МАТЕМАТИЧЕСКИЙ ФАКУЛЬТЕТ

Кафедра дифференциальных уравнений и системного анализа

ПОСТРОЕНИЕ РАСЧЕТНЫХ СЕТОК ДЛЯ ТРЕХМЕРНЫХ ЗАДАЧ С ПОДВИЖНЫМИ ОБЛАСТЯМИ

Курсовая работа

Жемойтяк Натальи Павловны
студента 2-го курса
специальности 1-31 03 09
«Компьютерная математика
и системный анализ»

Научный руководитель:
кандидат физ.-мат. наук,
доцент О.А.Лаврова

Минск, 2021

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В современном мире очень много разных подходов к моделированию и анализу физических процессов. Главной задачей всегда является вычисление и отслеживание изменений некоторых параметров модели и ее объектов. Но, когда объект представляет собой сложную фигуру, чаще всего посчитать какой-то параметр довольно сложно. Приходится находить методы для упрощения вычислений.

Расчетные сетки являются одним из методом анализа объектов. Они представльют собой разбиение на маленькие части, которые, как правило, проще анализировать. Расчетные сетки либо заполняют, либо покрывают объект, и таким образом вычисления параметров по рассчетным сеткам будут аппроксимациями вычислений параметров самого объекта.

В данной работе будут изучены программные возможности для генерации, визуализации и работы с сетками. Будут построены сетки, для нескольких фигур в качестве примеров. Будут разработаны и реализованы алгоритмы для расчета центра масс и сферичности, объема, ограниченного поверхностью, на основании анализа двух несогласованных сеток (сетка для объема и сетка для поверхности).

ГЛАВА 1

ТЕОРЕТИЧЕСКОЕ ВВЕДЕНИЕ В РАСЧЕТНЫЕ СЕТКИ

1.1 Основные определения.

Расчетной сеткой в области D в $\mathbb{R}^n$ является множество геометрических фигур, которые покрывают или заполняют n- мерную область область или поверность [5, с. 5].

Триангуляция — разбиение области $\mathbb{R}^n$ на k-мерные симплексы, $1 \leq k \leq n$.

Симплекс — k-мерный тетраэдр.

Пусть Q — двухмерный симплекс (треугольник), C — окружность на плоскости, описывающая Q. Триангуляция Делоне на двухмерной плоскости — разбиение на симплексы таким образом, чтоб все точки этой сетки, кроме вершин, лежат вне окружности Q [2, с. 2].

Триангуляция Делоне в трехмерном пространстве — разбиение на тетраэдры таким образом, что все точки этой сетки лежат вне сфере, кроме вершин тетраэдра, которого описывает данная сфера.

1.2 Вычисление характеристик фигуры в $\mathbb{R}^3$ с помощью расчетных сеток

Пусть дана фигура G, граница которой задана замкнутой поверхностью S, которая задается формулой $F(x, y, z) = 0$, а $D \in \mathbb{R}^3$ это область внутри фигуры. Необходимо вычислить четыри следующие характеристики фигуры:

- Объем
- Площадь поверхности
- Центр масс
- Оценку сферичности

Данные характеристики легко вычислить для простых фигур, как сфера и призма, но для моделирования разных задач необходимо сформулировать общий случай для вычисления этих характеристик, зная уравнение поверхности.

Разобъем область D на тетраэдры $t_i \in T = \{t_i | i = \{1, n\}\}$, где T — множество тетраэдров. Затем посчитаем объем каждого тетраэдра по формуле:

$$v(t) = \frac{1}{6} \begin{vmatrix} x_2 - x_1 & y_2 - y_1 & z_2 - z_1 \\ x_3 - x_1 & y_3 - y_1 & z_3 - z_1 \\ x_4 - x_1 & y_4 - y_1 & z_4 - z_1 \end{vmatrix},$$

где $\{x_i, y_i, z_i, i = \{1, 4\}\}$ — координаты вершин тетраэдра. Объем фигуры равен сумме объемов всех тетраэдров $t_i \in T$:

$$\mathscr{V} = \sum_{i=1}^{n} v(t_i).$$

Для вычисления площади поверхности необходимо найти те тетраэдры t_j, которые лежат на границе S. То есть координаты трех вершин тетраэдра удовлетворяют уравнению $F(x, y, z) = 0$ (неявная функция, которая задает границу фигуры). Эти три вершины являются вершинами треугольника, который лежит на границе. Площадь треугольника можно посчитать по формуле:

$$s = \frac{1}{2} \vec{a} \cdot \vec{b},$$

где $a = (x_2 - x_1, y_2 - y_1, z_2 - z1), b = (x_3 - x_1, y_3 - y_1, z_3 - z_1), (x_i, y_i, z_i), i = \{1, 3\}$ —вершины треугольника.

Зная площади всех треугольников, покрывающих границу, можно посчитать площадь поверхности всей фигуры по формуле:

$$\mathscr{S} = \sum_{j=1}^{m} s_j.$$

Одна координата центр масс считается по формуле:

$$x_c = \frac{1}{\mathscr{V}} \int\limits_D x \; d\mathscr{V} = \frac{1}{\mathscr{V}} \iiint\limits_D x \; dx \; dy \; dz \approx \frac{1}{\mathscr{V}} \sum_T \overline{x_j} * v(t_i),$$

где

$$\overline{x_j} = \frac{1}{4}\sum_{j=1}^{4} x_j$$

—среднее значение этой координаты в данном тетраэдре [6, с. 34].Остальные координаты считаются по этой же формуле.

Оценку сферичности можно посчитать по формуле:

$$\mathscr{C} = \frac{\pi^{\frac{1}{3}}(6\mathscr{V})^{\frac{2}{3}}}{\mathscr{S}},$$

где $\mathscr{V}$ - объем тела, $\mathscr{S}$ — площадь поверхности [6, с. 34].

Оценка сферичности для сферы равняется 1.

$$\mathscr{V} = \frac{4}{3}\pi r^3$$

$$\mathscr{S} = 4\pi r^2$$

$$\mathscr{C} = \frac{\pi^{\frac{1}{3}}(6\mathscr{V})^{\frac{2}{3}}}{\mathscr{S}} = \frac{\pi^{\frac{1}{3}}(6 * \frac{4}{3}\pi r^3)^{\frac{2}{3}}}{4\pi r^2} = \frac{\pi^{\frac{1}{3}}(8\pi r^3)^{\frac{2}{3}}}{4\pi r^2} = \frac{\pi^{\frac{1}{3}}4\pi^{\frac{2}{3}}r^2}{4\pi r^2} = 1$$

Альтернативно можно построить двухмерную сетку на трехмерной фигуре, т.е. построить сетку для ее границы. Площадь поверхности будет считаться по той же формуле, только в данном случае уже известны треугольники на границах. Для объема можно взять любую точку $a \in D$ внутри фигуры. Если взять любой треугольник на поверхности и соединить все вершины с этой точкой, получится тетраэдр. Эти тетраэдры уже можно использовать для вычисления объема фигуры.

Теоретически это кажется легче, но многие программы строят только сетки с n-мерными симплексами в $\mathbb{R}^n$. Единственный подход к такой задаче это построить выпуклую оболочку, которая будет являтся двухмерной сеткой.

ГЛАВА 2

ПРОГРАМНЫЕ ПАКЕТЫ ДЛЯ ПОСТРОЕНИЕ И ВИЗУАЛИЗАЦИИ РАСЧЕТНЫХ СЕТОК

2.1 Alberta

Alberta это адаптивный пакет для решение дискретных задач. Он разработан Альфредом Шмидтом, Кунибертом Сибертом и другими. Шмидт и Сиберт являются авторами документации[1]. Данная программа написана на C и компилируется на компьютерах с операционной системой Linux.

Всего существует три версии и последняя.Третья была выпущена в 2014 году, но на сайте она не качается и доступна вторая версия 2006 года[11]. Данная программа была скачана на компьютер с операционной системой MacOS и комплировалась, но демонстрации и встроенные примеры не запускались. Терминал выдавал ошибку и печатал, что не хватает определенного модуля. Модули были скачаны, но все равно примеры не запустились. Хотя в работе о задаче про двухфазное течение авторы продемонстрировали построение трехмерных сеток в alberta [6, с. 35]. Здесь есть несколько потенциальных причин:

- Операционная система MacOS не подходит, хотя она считается Linux-based (основанна на Linux).
- Данная версия программы устаревшая. Скачана была вторая версия 2006 года, а компьютер 2019. За тринадцать лет могла изменится структрура информации, язык C, на котором написана данная программа, и дополнительные модули, которые использует alberta.

На попытки запуска примеров было потрачено много времени, в ходе которого была изучена документация. В ней содержатся алгоритмы и методы построения сеток, а так же теоретическая база[1]. Данный пакет строит симплексные сетки, но не методом триангуляции Делоне. Данный метод триангуляции называется иерархическим. С помощью рекурсии находится новый тетраэдр, используя нотацию Косакского[1, с. 4].

2.2 Triangle

Triangle — программа написанная Джоннатеном Ричардом Счууиком на C для генерации расчетных сеток методом триангуляции Делоне в двух мерном пространстве. В документации написано, что главный метод построения— это алгоритм Руперта [3, с. 1,3]. На вход падается список точек и отрезков, которые соединяют точки между собой. Данный алгоритм можно описать двумя стадиями:

1. Найти треугольники, соответствующие определению триангуляции Делоне. Таким образом соединяются точки между собой, чтобы образовать треугольники.

2. Добавить точки, необходимые для завершения генерации сетки.

Данный алгоритм считается одним из самых быстрых на практике[3, с. 3].

Автор отметил, что в данном пакете так же существует комбинация следующих алгоритмов:

- incremental insertion(инкрементальное вставление)
- divide and conquer (разделяй и властвуй)
- plane-sweep (алгоритм заметающей прямой)

которые ранее были написаны разными людьми в разных программах и пакетах [3, с. 1].

Несмотря на комбинацию различных известных в данной области алгоритмов эта программа, как и alberta, давно не обновлялась. На официальном сайте написано, что последняя версия выпущена была в 2005 году[10].

2.3 TetGen

TetGen — это пакет, написанный Hang Si на C++, который генерирует трехмерные сетки Делоне. По словам разработчика, он использовал алгоритм написанный разработчиком triagle Счуиком [2, с.]. Если посмотреть на метод построения сетки, то программы отличаются лишь размерностью пространства

и алгоритмом. В обеих программах на вход подается список точек и ребер или граней, которые Хан Си называет Piecewise Linear Complex(PLC — кусочные линейные комплекции)[2, с. 7].

Один из главных возможностей tetgen это возможность задание на входе большого количества опций, таких как:

- максимальный объем для увеличения концентрации тетраэдров
- возможность построить сетку заново
- возможность оптимизации и улучшения качества через добаления новых точек

TetGen вполне неплохой продукт, за исключением того, что на вход нужно самому задать границы поверхности, и достаточно современный(2020 года) [2].

2.4 Gmsh

Gmsh— это еще один удобный пакет для построения сеток. В этой программе сетки строятся обычные триангуляции и по принципу BRep(граничное представление). В добавок gmsh может создавать сетки в основе которой не треуголники, а например четырехугольники и шестиугольники[5, с. 8]

Наверное самое большое преимущество gmsh в том, что разработчики сами разработали API для возможности пользоваться с помощью других языков:

- C++
- C
- Python
- Julia

[5, с. 249]. Это дает большое преимество так как иногда проще разобрается с программой в языке как питоне, а затем ускорить алгоритм на другом, как например C++.

Gmsh последний раз был обновлен в 2021, что говорит о том, что программа обновленная и современная[5].

2.5 MeshPy

MeshPy— это библиотека, написанная на питоне, которая включает в себя Triangle для генерации двухмерных сеток и TetGen для генерации трехмерных. В ранних версиях этот пакет включал в себя Gmsh, но разработчики решили перенести код в отдельный пакет gmsh_interop [7].

Необходимые данные для построение сеток такие же как и в Triangle и TetGen:

- для трехмерной сетки координаты и отрезки,
- для двухмерной сетки координаты и грани.

Дополнительной возможностью в meshpy является возможность создавать фигуру с помощью встроенных функций.

- окружность радиуса r
- сфера с радиусом r
- прямоугольная призма
- цилиндр
- фигура вращения
- призма с основанием s

[7]

Этих базовых функций вполне достаточны для введения в генерацию триангуляции Делоне. Но для более сложных сеток MeshPy не очень удобен, потому что нужно самостоятельно задавать грани фигуры (самостоятельно разбить ее на многогранники).

Еще один плюс данного пакета в топ, что в есть классы фигуры и сетки. У класса(и объекта) сетки есть два очень важных поля: точки и элементы(симплексы), которые упрощает анализ сетки после ее построения.

Последнее обновление было в 2020 году, что означает, что эта программа достаточно новая.

2.6 Qhull

Qhull — это универсальная программа для нахождения выпуклой оболочки, генерации триангуляции Делоне и др. В данной работе этот пакет был изучен поверхностно, как программа, на основе которой питоновская scipy библиотека строит выпуклые оболочки. Но случайно было обнаружено, что данная програма интегрирована в Wolfram Mathematica и MatLab [8].

2.7 Paraview

Paraview — это программа для визуализации и симуляции. В ней можно построить фигуру или несколько фигур, наложить на них функции и смотреть как фигуры меняются в течение времени. В программе есть встроенный питон, через который задаются функции и настраиваются параметры [4].

В данной работе Paraview был использован для визуализации сеток. В нее загружается файл с расширением .vtk с данными сетки(симплексы и точки) и затем они визуализируются в трехмерном пространстве.

Если в Paraview просто загрузить фигуру или построить ее там, то можно воспользоваться встроенной функцией, которая строит расчетную треугольную сетку прямо в программе. Другими словами, у Paraview есть собственная функция, которая может построить сетку триангуляции Делоне, но потом с этой сеткой сложно работать и анализировать самостоятельно, потому что изначально надо настроить программу и изучить тонкости.

Так как Paraview для данной работы был скачан для визуализации, методы настройки фигур не были изучены, но для построения графика это хорошая программа. В ней есть возможность отображать несколько сеток сразу.

ГЛАВА 3
ПОСТРОЕНИЕ И АНАЛИЗ СЕТКИ

3.1 Построение сетки в MeshPy

Перед тем как построить сетку нужно задать начальные данные. Так как фля генерации трехмерной сетки MeshPy использует TetGen, то необходимыми параметрами на вход являются точки и грани, которые соединяют точки между собой. Грани представляют собой список индексов точек, которые являются вершинами данной грани. В общем случае надо будет посчитать самому точки и соединить их таким образом, чтоб получить грань. Это не очень удобно, но в MeshPy есть встроенные функции для частных случаев:

- окружность радиуса r
- сфера с радиусом r
- призма с основанием s
- фигура вращения

[7].Если взять фигуру вращения, то вполне можно построить не мало разных сеток. Если взять сферу, можно потом изменить координаты на коэффициент, чтоб получить эллипсоид.

Для начала построим призму. Можно воспользоваться встроенной функцией, которая вернет грани и точки, или сделать это самостоятельно, чтоб понять, какие объекты подаются на вход. Если самостоятельно, то необходимо задать сначала координаты. Так как у призмы понятно, где грани, вручную составляется список индексов точек каждой грани.

Затем список граней и точек подается в генератор сетки, и на вход получается объект, у которого есть поля:

- точки(их координаты), составляющие сетку
- элементы (тетраэдры)- список индексов точек, которые составляют данный тетраэдр.

С этими объектами можно работать и считать характеристики фигуры. Для визуализации необходимо экспортировать данные в файл с раширением .vtk. Файл такого формата принимается в Paraview.

При построение есть еще дополнительные возможности, чтоб задать сетку. Например, можно задать максимальный объем одного элемента, чтоб уменьшить размер тетраэдра. Это можно сделать двумя способами:

- при построение сетки указать необходимый флаг. В пакете tetgen при построение есть возможность задать определенные условие через так называемые флаги. Они подаются на вход в аргумент Options в генератор сетки[7].

- после построения можно в поле volume задать максимальнуп величину и построить сетку заново.

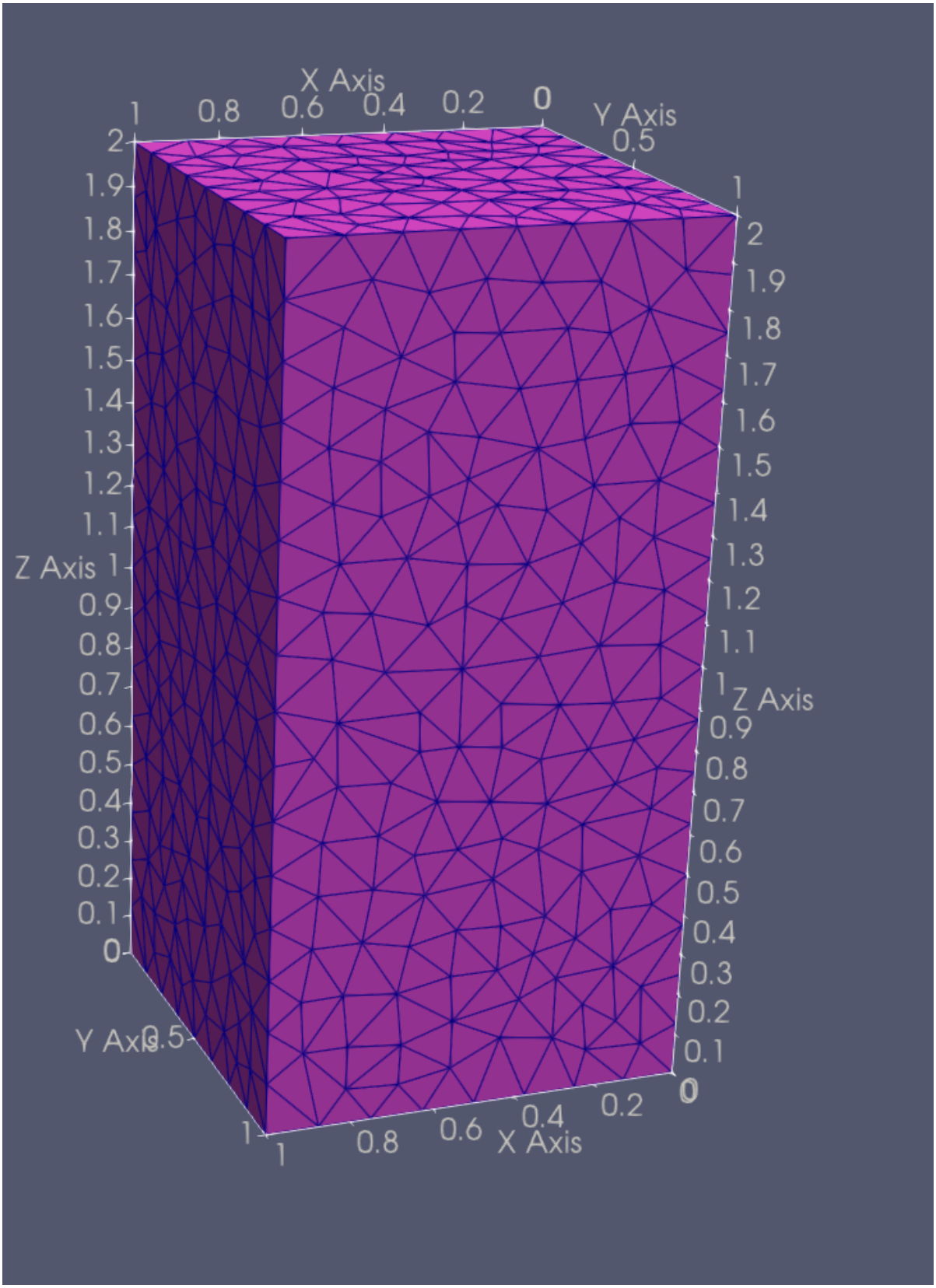


Рисунок 3.1 Сетка, с ограничением объема для тетраэдра 0.01. Визуализация в ParaView.

Таким образом работает построение сетки в MeshPy.

Рассмотрим теперь возможность перемещение объекта. Зададим сферу S с помощью встроенной функции meshpy. Эта функция строит сферу с центром в (0,0,0). На вход можно изменить только два параметра сферы:

- радиус
- количество разделений: в основу сферы берется окружность, которая вращается. Берется количесто разделений(кусков) k и делится на 2π. Получаем один интервал с полярной системе координат. Затем по окружности с шагом $\frac{2\pi}{k}$. Получается k точек, принадлежащих окружности.

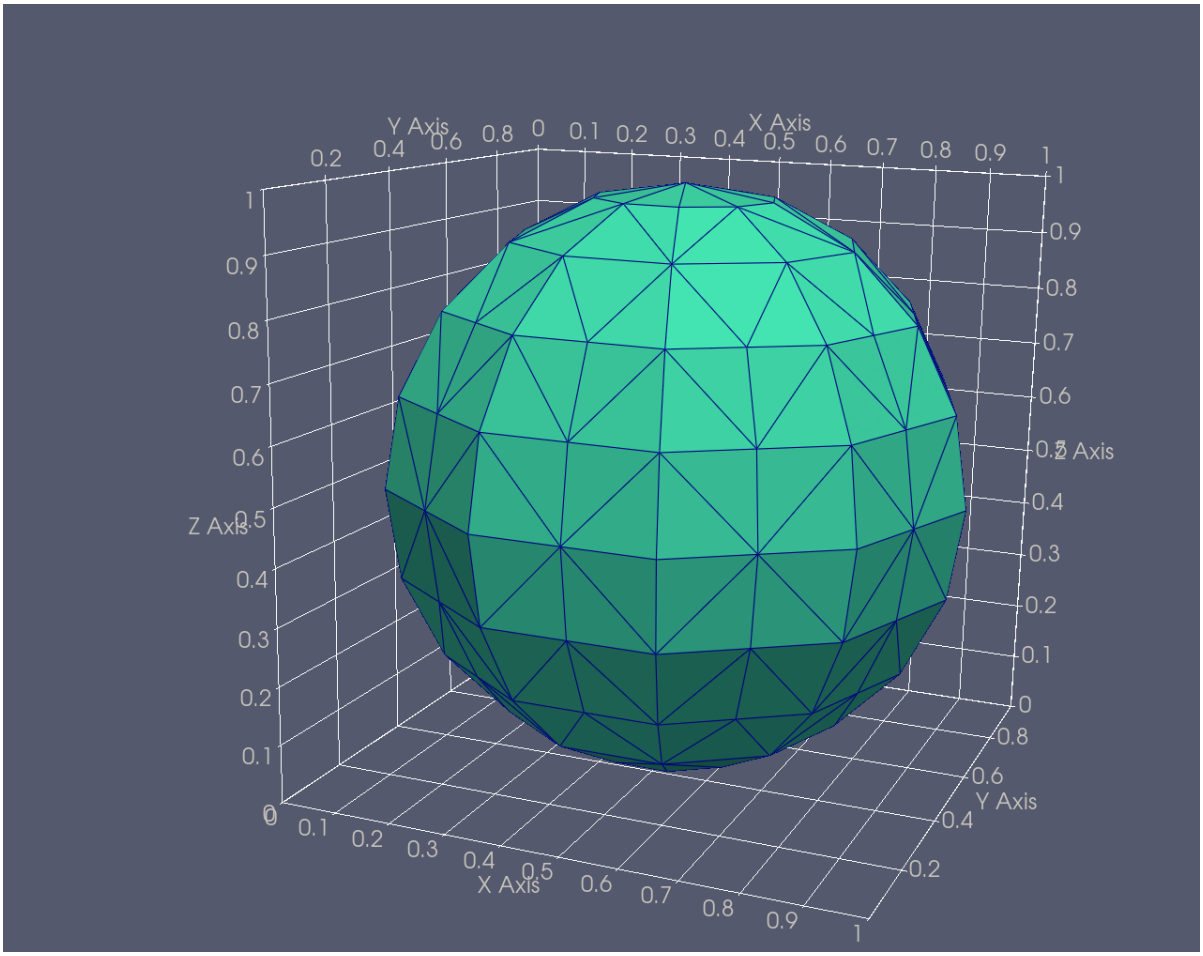


Рисунок 3.2 Сетка, заполняющая сферу. Визуализация в ParaView. [Приложение Б]

У нас есть множество точек $\{point_i = (x_i, y_i, z_i)\}_{i=1}^{n}$ и множество граней $\{facet_j\}_{j=1}^{m}$. Можно переместить сферу таким образом, чтоб ее центр находился

в точке $a = (x_a, y_a, z_a)$. Другими словами применить к множеству точек такую функцию:

$$f(x, y, z) = (x, y, z) + (x_a, y_a, z_a).$$

Функция $f : \mathbb{R}^3 \to \mathbb{R}^3$ линейна и не влияет никак на грани. Отношение и растояние между точками остается таким же. Поэтому можно переносить сферу в другие системы координат с другим центром.

Аналогично можно превратить сферу в эллипсоид. Зададим функцию $g : \mathbb{R}^3 \to \mathbb{R}^3$ таким образом:

$$g(x, y, z) = (x, y, k\ z),$$

где k это какой-то коэффициент. Получим эллипсоид с радиусами $(r, r, k\ r)$. Это тоже не изменяет грани, потому что порядок точек остается таким же.

Таким образом можно манипулировать сферой, чтоб получить другую фигуру и ее проанализировать.

3.2 Анализ фигуры

Получув сетку фигуры, необходимо ее проанализировать, вычисляя следующие параметры:

- Объем
- Площадь поверхности
- Центр масс
- Оценку сферичности

Для вычисления характеристик необходимо иметь координаты вершин тетраэдра. Как уже было сказано, после генерации сетки получаем объект с двумя полями: координатами и тетраэдрами. В списке тетраэдров хранятся индексы точек(индекс из списка точек).

Пусть $ind_k, k = 1, 4$ — индекс k-ой вершины тетраэдра, а $\{point_i =\}_{i=1}^{n}$ — список точек. Тогда координата k-ой вершины можно найти таким образом:

$$(x_k, y_k, z_k) = point_{ind_k}.$$

Зная координаты вершин, можно вычислить необходимые характеристики, подставляя их в функции [Приложение А].

Но есть один метод вычисления площади поверхности, который можно применить именно в MeshPy. Как известно, для построение сетки необходимо задать грани фигуры. Грани это области принадлежащие $\mathbb{R}^2$. Так как сетки будут касаться этих границ, то площадь поверхности

$$\mathscr{S}_{triangle} = \sum_{j=1}^{m} s(triangle_j)$$

по треугольникам будет равнятся площади поверхности по граням

$$\mathscr{S}_{facet} = \sum_{k=1}^{b} s(facet_k).$$

Если посмотреть на визуализацию сетки и фигуры, то можно увидеть, что грани разделяются на треугольники, которые являются гранями тетраэдров сетки. Таким образом это утверждение имеет место.

Площадь грани можно посчитать, разбив грань на треугольники и сложив их площади. В данном контексте рассматривается сфера, у которой грани являются треугольники и четырехугольники. Таким образом можно найти площадь поверхности, разбивая каждый четырехугольник на два треугольника. Важно учесть то, что трегульники должны всего лишь иметь общее ребро, поэтому нужно знать как MeshPy задает такие грани.

Такие грани задаются следующим образом: последовательно по часовой стрелке.

Пусть $pt = \{A, B, C, D\}$ - последовательные вершины четырехугольника, то есть ребра $\{AB, BC, CD, DA\}$ задают четырехугольник. Тогда площадь четырех угольника

$$\mathscr{S}_{ABCD} = \mathscr{S}_{\triangle ABC} + \mathscr{S}_{\triangle ACD}.$$

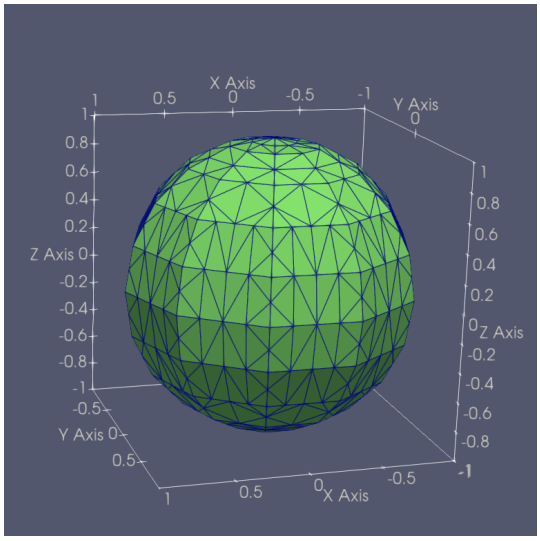
Треугольники $\triangle ABC$ и $\triangle ACD$ не пересекаются и имеют общее ребро.

3.3 Анализ последовательности эллипсоидов

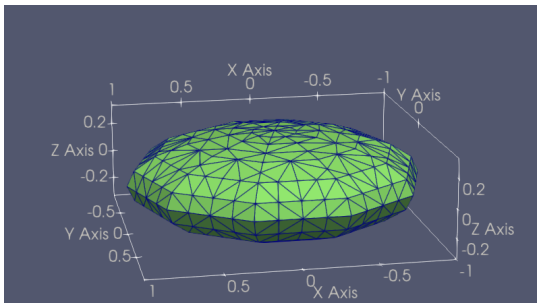
Зададим последовательность из десяти эллипсоидов с центром в (0,0,0) таким образом:

$$E_k : \frac{x^2}{a^2} + \frac{y^2}{a^2} + \frac{z^2}{b^2} - 1 = 0, b = \frac{a}{k}, k = \{1, 10\}.$$

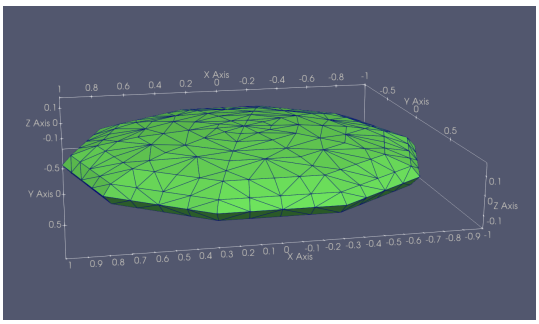
Получается, что сфера сжимается с каждым шагом.



(a)



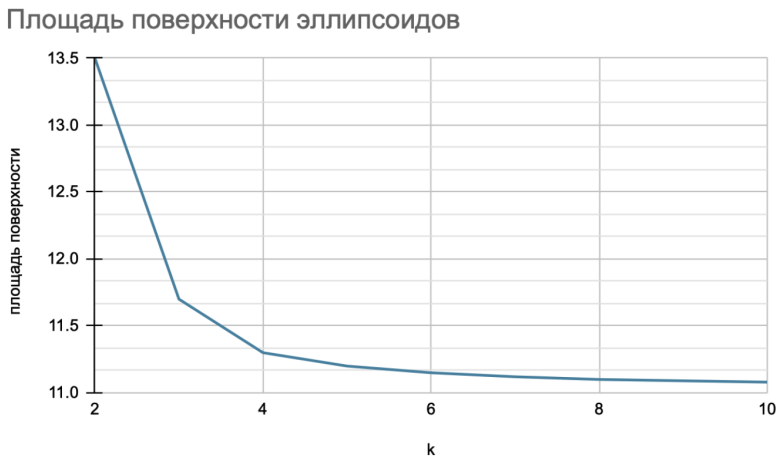
(b)



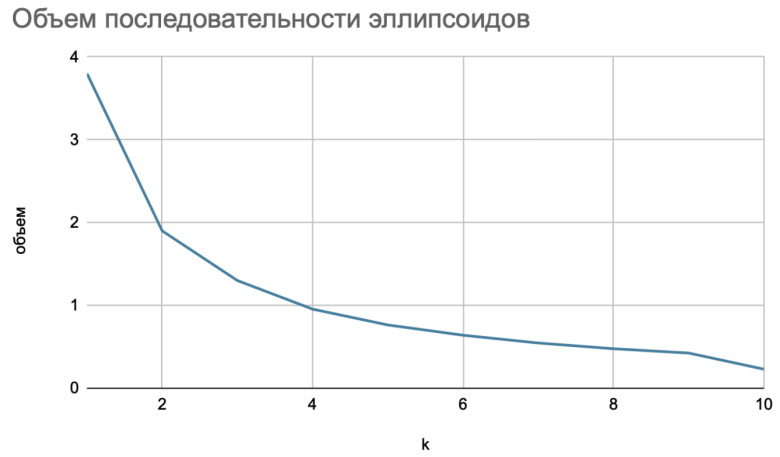
(c)

Рисунок 3.3 Эллипсоиды при k = 1(a),k = 3 (b),k = 6 (c).

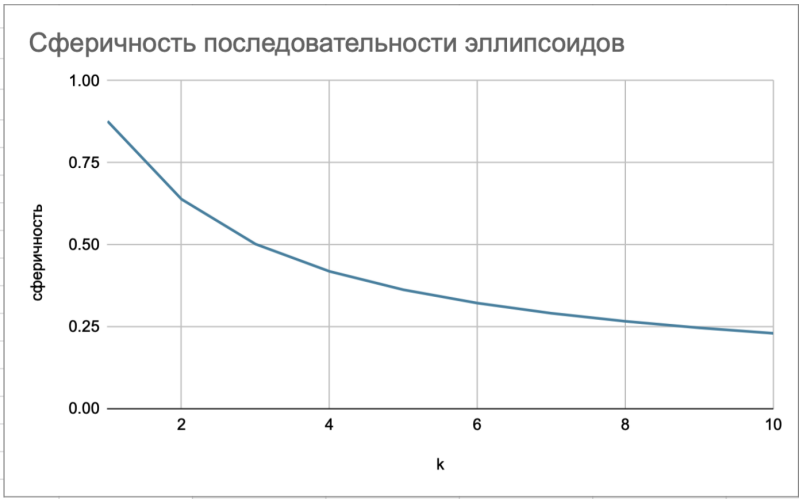
Построим для каждого эллипсоида сетку и вычислим необходимые характеристики [Приложение B].



(a)



(b)



(c)

Рисунок 3.4 Площадь поверхности(a), объем(b) и сферичность(c) последовательности эллипсоидов.

Как можно заметить, все параметры уменьшаются почти с той же динамикой. Центр масс для каждой фигуры в окрестности нуля. Так как данные были посчитаны питоном, возможно что там (0,0,0), но питон представляет это число как достаточно маленькое. Но, не исключено, что такая погрешность из-за неточности в объеме. Данная фигура вписана в эллипсоид и ее объем меньше объема самого эллипсоида.

Возьмем для примера сферу, то есть первый эллипсоид, где $k = 1$ и радиус равен a(в данном случае 1). Вычислим объем и площадь поверхности.

$$\mathscr{V}_{sphere} = \frac{4}{3}\pi r^3 = \frac{4}{3}\pi \approx 4.1889$$
$$\mathscr{S}_{sphere} = 4\pi r^2 \approx 12.5664$$

Объем больше, чем у полученного, а площадь меньше (см. рис.3.4(a),3.4(b)).

ГЛАВА 4
АНАЛИЗ НЕСОГЛАСОВАННЫХ СЕТОК

4.1 Общая формулировка задачи

Пусть известны два объекта:

- Сетка состоящая из тетраэдров Ω.
- Другая сетка Γ, которая находится внутри первой сетки(она задана уравнением $f(x, y, z) = 0$). Ее внутренняя область обозначается как Ω

Необходимо изучить внутреннюю сетку и вычислить 4 характеристики по формулам с помощью сеток. В качестве примера рассмотрим начальное положение капли в двухфазном течении [6, с. 34].

$$\Gamma_0 = \{(x, y, z) \in \mathbb{R}^3, \|(x, y, z) - (\frac{1}{2}, \frac{1}{2}, \frac{1}{2})\| = \frac{1}{4}\}$$

— сфера,

$$\Omega = [0, 1] \times [0, 1] \times [0, 2] \in \mathbb{R}^3$$

— прямоугольная призма. Так как известно, как строить и анализировать сферу саму по себе, то можно построить две отдельные сетки. Для сферы посчитаны ее параметры.

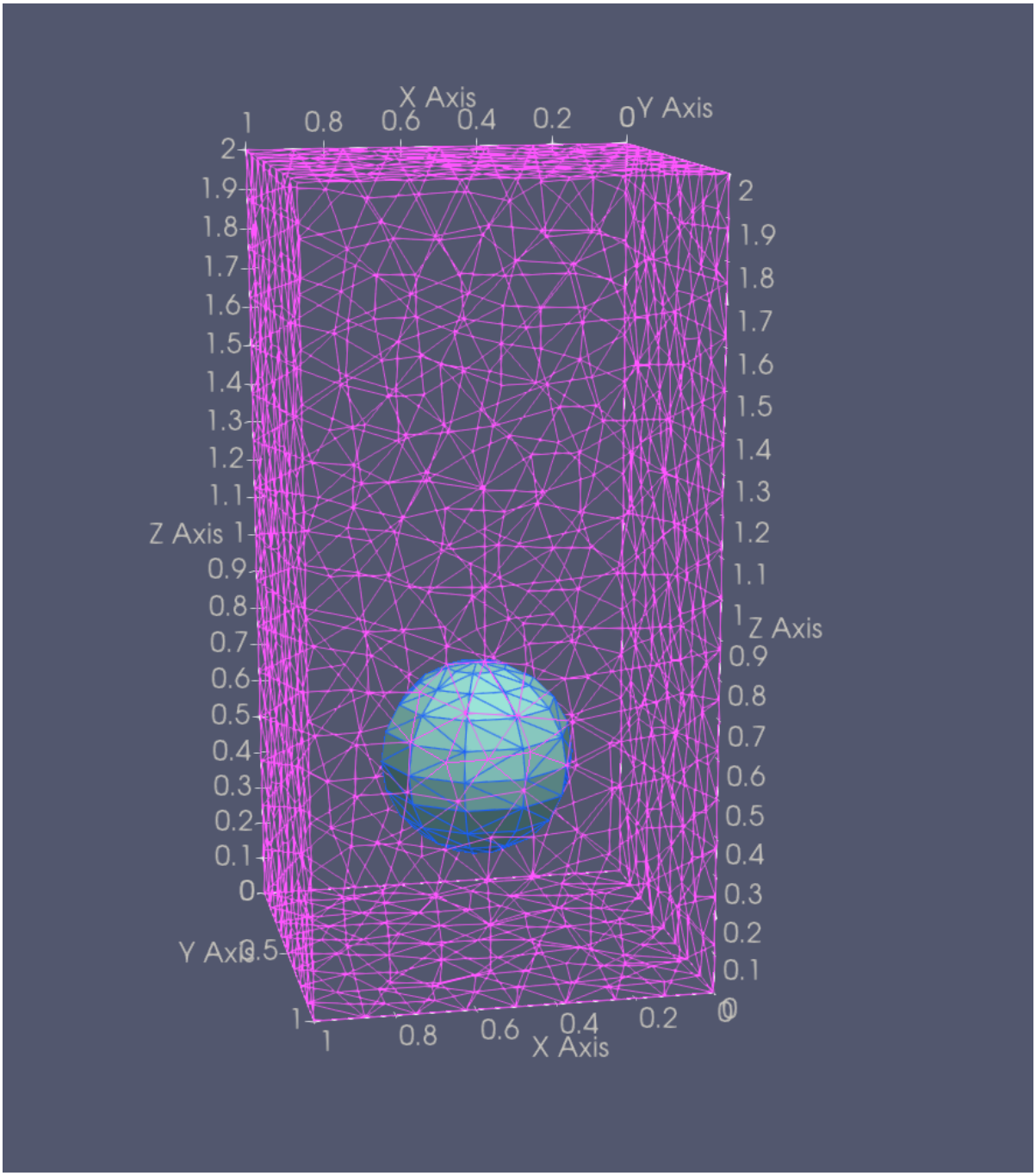


Рисунок 4.1 Геометрическое представление задачи. Построены две сетки. Визуализация в ParaView.

Но по условии задачи Γ меняется со временем и фигура преобразится из сферы в другую. Необходима придумать решение для данных задач:

- Как, зная уравнение поверхности Γ, найти область $\Omega_- \in \Omega$, которая является внутренней областью внутренней сетки?

- Как посчитать характеристики для Г?

4.2 Метод выбора тетраэдров

Так как известно, что фигура, границу которой задает Г находится в Ω, значит в фигуре содержатся тетраэдры, которые заполняют Ω. Эти тетраэдры $t \in \Omega_-$.

Рассмотрим пример со сферой в призме. Радиус нашей сферы равен $r = \frac{1}{4}$, центр находится в точке $a = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$. Тогда можно Г уравнением

$$f(x, y, z) = (x - \frac{1}{2})^2 + (y - \frac{1}{2})^2 + (z - \frac{1}{2})^2 - \frac{1}{16} = 0.$$

По определению, сфера это множество точек, которые равноудаленны от фиксированной точки. Получается, что те точки, которые лежат в сфере удалены от фиксированной точки (x_c, y_c, z_c) на растоянии $d < r$. Тогда

$$f(x, y, z) = (x - x_c)^2 + (y - y_c)^2 + (z - z_c)^2 - r^2 = d^2 - r^2 = (d - r)(d + r) < 0.$$

Следовательно, для того, чтоб тетраэдр полностью лежат во внутренней области Ω_-:

$$\forall i \in \{1, 4\} : f(x_i, y_i, z_i) < 0.$$

Если допустить, что тетраэдр касается границы Г, то знак будет нестрогий.

Таким образом можно найти множестви тетраэдров $T_{and} = \{t | t \in \Omega_-\}$.

К этому множеству можно применить формулу для вычисления характеристик. Но будут значительные погрешности, потому что сетки несогласованны. Это значит, что есть еще и тетраэдры, которые пересекают Г, но не обязательно лежат в области Ω_-.

Таким образом можно найти еще одно множество тетраэдров по данному критерию: тетраэдр пересекается с областью Ω_-, если хотя б одна его вершина удовлетворяет равенству $f(x, y, z) < 0$.

$$\exists i \in \{1, 4\} : f(x_i, y_i, z_i) < 0.$$

Назовем такое множество T_{or}.

Индекс and означает то, что в условном блоке кода, где проверяется тет-

раэдр, все значения $f(x_i, y_i, z_i), i = 1, 4$ должны удовлетворять условию, чтоб тетраэдр принадлежал множеству(and используется как союз в блоке if) [Приложение Г].

Индекс or означает то, что в условном блоке кода, где проверяется тетраэдр, хотя б одно значение $f(x_i, y_i, z_i), i = 1, 4$ должно удовлетворять условию, чтоб тетраэдр принадлежал множеству (or используется как союз в блоке if) [Приложение Г].

В обоих множествах будут значительные погрешности:

- у T_{and} объем будет меньше.
- у T_{and} объем будет больше.

Получается, что объем $\mathscr{V}(T_{or})$ является верхим пределом, а $\mathscr{V}(T_{and})$ — нижним.

При уменьшении объема(размера) тетраэдров в сетке Ω

$$v \to 0,$$

количество тетраедров будет возрастать

$$n \to 0,$$

а объемы множеств T_{or} и T_{and} будут стремится к объему области Ω_-.

$$\lim_{v \to 0} \mathscr{V}(T_{and}) = \mathscr{V}(\Omega_-)$$

$$\lim_{v \to 0} \mathscr{V}(T_{or}) = \mathscr{V}(\Omega_-)$$

Единственная проблема увеличения количества тетраэдров — увеличится время вычислений, потому что питон очень медленный в огромных итерациях.

Метод выбора тетраэдров не самый лучший, потому что те тетраэдры, которые пересекаются с поверхностью Г, либо вообще не учитываются, либо учитываются полностью. Это неправильно, потому что скорее всего там лежит часть тетраэдра. Таким методом еще очень сложно посчитать площадь поверхности. Тут не будут находится треугольники на поверхности, потому что тут тетраэдры пересекаются. Поэтому для данных высчислений было взято значение по формуле.

4.3 Метод выпуклой оболочки

Для более точный вычислений, необходимо придумать как учесть часть пересекающих тетраэдров. Один из методов это метод нахождения выпуклой оболочки по точкам пересечения сетки Ω с поверхностью Γ.

Метод выплуклой оболочки состоит из следующих шагов:

1. Находим тетраэдры, которые пересекаются с данной поверностью Γ
2. Для каждого тетрэадра, найденного в предыдущем шаге, находим точки пересечения с Γ
3. Находим выпуклую оболочку для множества точек пересечения
4. Используем выпуклую оболочку как двух мерную сетку и вычисляем необходимые характеристики

Возьмем критерий пересечения из Никукилна: многогранних пересекается с плоскостью тогда и только тогда когда две его точки лежат по разные стороны [9, с. 81]. Пусть задана функция $f(x, y, z) = 0$, которая задает поверхность Γ. Тогда две точки лежат на противоположных сторонах поверхности:

$$sign(f(x_1, y_1, z_1)) \neq sign(f(x_2, y_2, z_2)).$$

Чтоб проверить на пересечение тетраэдр, необходимо вывести список его ребер. Всего у тетраэдра 6 ребер, поэтому просто сделать это вручную, задав индексы точек в каждом ребре. Затем каждое ребро проверяем на пересечение. Если одно ребро пересекается с Γ, значит тетраэдр пересекается. Эти тетраэдры(или их индексы) необходимо занести в отдельный список, чтоб потом ими воспользоваться.

Каждое ребро тетраэдра является отрезком какой-то прямой. Зададим параметрическое уравнение прямой $\mathscr{L}$ по двух точкам (x_1, y_1, z_1) и (x_2, y_2, z_2):

$$\mathscr{L} = \begin{cases} x = (x_2 - x_1)t + x_1 \\ y = (y_2 - y_1)t + y_1 \\ z = (z_2 - z_1)t + z_1 \end{cases}.$$

Подставляя вместо x,y,z эти выражения получаем уравнение с неизвестной переменной t:

$$\Gamma(t) = f((x_2 - x_1)t + x_1, (y_2 - y_1)t + y_1, (z_2 - z_1)t + z_1) = 0 \ .$$

Для сферы это будет выглядеть следующим образом:

$$((x_2 - x_1)t + x_1 - \frac{1}{2})^2 + ((y_2 - y_1)t + y_1 - \frac{1}{2})^2 + ((z_2 - z_1)t + z_1 - \frac{1}{2})^2 - \frac{1}{16}.$$

Получается квадратное уравнение с двумя решениями. Его можно решить используя библиотеку sympy. Задается сначала переменная, потом в функцию передеается уравнение. В данном случае будет два решения но для ребра необходима одна потому что тетраэдр значительно меньше сферы и не может одно ребро пересекать ее дважды. Поэтому нужно ввести дополнительное условие для нахождения точки, которая лежит на отрезке [Приложение Д].

Критерий принадлежности точки к отрезку: Пусть т. c принадлежит отрезку ab. Тогда:

$$\vec{ca} \cdot \vec{cb} < 0.$$

где a и b это концы отрезка. Таким образом находим множество точек пересечения [Приложение Д]. Для сокращения время вычисления можно совместить проверку на пересечение и нахождение точек пересечение: Если ребро тетраэдра пересекается с поверхностью, находится точка пересечения.

Далее необходимо построить выпуклую оболочку, которая будет двухмерной сеткой для Ω_-. Есть класс в пакете scipy, который называется scipy.spatial.ConvexHull. Он включает в себя QHull. С помощью этого класса можно построить выпуклую оболочку. Этот класс еще удобен тем, что у объекта есть два поля:

- объем
- площадь поверхности

Это означает, что уже посчитаны две характеристики для фигуры, которую покрывает двухмерная сетка. В формуля сферичности подставляем эти два значения и вычисляем.

Для вычисления центра масс необходимо иметь множество тетраэдров. Во время проверки на пересечение тетраэдры сетки делятся на две группы:

- пересекаются с Г
- не пересекаются

В качестве множества тетраэдров нужно взять тетраэдры, которые пересекаются. Но стоит обратить внимание, что нужно для этого множества сумму объемов посчитать. На каком-то этапе разработки была совершена ошибка, когда центр был поделен на объем фигуры в выпуклой оболочки. Центр масс оказался за пределами самой сетки. Чтоб исправить эту ошибку, необходимо посчитать объемов выбранных тетраэдров [Приложение Е].

Несмотря на время выполнение этот метод самый близкий к расчетом по известным гоеметрическим формулам. В таблице видно, что результат самый близким к значениям, посчитанным по известным геометрическими формулам для сферы 4.1.

Таблица 4.1 Результаты разных методов вычиселния

	Вычисление по формулам	Вычисление отдельной сети	Выбор внутренних	Выбор внутренних и пересекающих	метод выпуклой оболочки
Объем	0.065	0.06	0.015	0.134	0.06
Сферичность	1	0.88	0.38	1.61	0.986
Центр масс	(0.5,0.5, 0.5)	(0.5,0.5 ,0.5)	(0.506, 0.508, 0.47)	(0.51, 0.506, 0.47)	(0.509, 0.506, 0.47)
Время вычислений(сек)	0	0.18	0.51	0.57	192

ЗАКЛЮЧЕНИЕ

В ходе работы были изучены пакеты и программы для построение сеток. Также были изучены методы построение объектов и их сеток в модуле MeshPy. В конце была расмотрена изучена задача про несогласованные сетки и реализованы некоторые методы анализа сетки поверхности.

Конечным результатом работы стала программа, который вычисляет площадь поверхности, объем, сферичность и центр масс для тела внутри другого с использованием сетки внешнего объекта. Есть программа которая строит последовательность эллипсойдов и анализирует их. И есть несколько программ которые вычисляют параметры для сетки поверности методом выбора и методом выпуклой оболочки.

В перспективе ставятся задачи создать динамическую модель двух фазового течения и проанализировать необходимые характеристики с течением времени. Также ставится задача сравнить вычисления при построении сеток разными программами.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

[1] Schmidt Alfred. Design of Adaptive Finite Element Toolbox ALBERTA /Scmidt Alfred, Siebert Kunibert G. — Berlin:Springer, 2006.—324c.

[2] Han Si. TetGen. A Quality Tetrahedral Mesh Generator and 3D Delaunay Triangulator. — Berlin: WIAS , 2020 — 95c.

[3] Shewchuck Jonathan Richard. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. — Pittsburgh:Carnegie Mellon University,2005.—10c.

[4] Moreland Kenneth. The ParaView tutorial.—Mountain View: Sandia National Laboratories,2018.—158c.

[5] Guezaine Christophe. Gmsh Reference Manual/Guezaine Christophe,Remacle Jean-Francois.—International Journal for Numerical Methods in Engineering 79(11), 2021.—389c.

[6] Barrett J.W. A Stable Parametric Finite Element Discretization of Two-Phase Navier-Stokes Flow/Barrett J.W., Garcke H., Nuernberg R. // J. Sci. Comput., 2014. — 42c.

[7] MeshPy Documentation [Электронный ресурс] —
Режим доступа: https://documen.tician.de/meshpy/ —
Дата доступа: 05.12.2020

[8] QHull [Электронный ресурс] —
Режим доступа: http://www.qhull.org — Дата доступа: 03.05.2021

[9] Никулин. Е.А. Компьютерная геометрия и алгоритмы машинной графики. — Санкт- Петербург:БХВ Петербург,2003.—558c.

[10] Triangle Documentation [Электронный ресурс] —
Режим доступа: http://www.cs.cmu.edu/ quake/triangle.html —
Дата доступа: 15.05.2021

[11] alberta [Электронный ресурс] —
Режим доступа: http://www.alberta-fem.de — Дата доступа: 13.10.2020

ПРИЛОЖЕНИЕ А

Код программы. Функции для вычисления параметров

```
# volume
def volume(a,b,c,d):
    import numpy as np
    ab = np.array(b) - np.array(a)
    ac = np.array(c) - np.array(a)
    ad = np.array(d) - np.array(a)
    return 1/6 *np.linalg.det(np.matrix([ab,ac,ad]))


# center of mass part for one tetrahedron
def centre_of_mass(a,b,c,d, i):
    return 0.25*(a[i]+b[i]+c[i]+d[i])*volume(a,b,c,d)


# area of triangle
def area_of_triangle(a,b,c):
    import numpy as np
    ab = np.array(b) - np.array(a)
    ac = np.array(c) - np.array(a)
    return 0.5*np.dot(ab,ac)

# area of quadrilateral
def area_of_quadrelateral(a,b,c,d):
    return area_of_triangle(a,b,c) + area_of_triangle(a,c,d)


# sphericity
def sphericity(area, volume):
    import math
    return (math.pi**(1/3)) *((6*volume)**(2/3))/area
```

ПРИЛОЖЕНИЕ Б

Код программы.Построение и анализ сферы

```
def main():
    import numpy
    from meshpy.tet import MeshInfo, build, Options
    from meshpy.geometry import GeometryBuilder, Marker,make_box,make_ball
    import volume
    import time
    start_time = time.time()
    # mesh generation
    geom = GeometryBuilder();
    box_marker = Marker.FIRST_USER_MARKER
    points, facets, _, _ = make_ball(r = 0.25)
    for i in range(len(points)):
        a = [0,0,0]
        a = numpy.array(points[i]) + numpy.array([0.5,0.5,0.5])
        points[i] = tuple(a)
    area = 0
    # surface area count using facets
    for i in range(len(facets)):
        if(len(facets[i][0]) == 4):
            [[p1,p2,p3,p4]] = facets[i]
            [pt1,pt2,pt3,pt4] = [points[p1],points[p2],points[p3],points[p4]]
            area += volume.area_of_quadrelateral(pt1,pt2,pt3,pt4)
        elif(len(facets[i][0]) == 3):
            [[p1,p2,p3]] = facets[i]
            [pt1,pt2,pt3] = [points[p1],points[p2],points[p3]]
            area += volume.area_of_triangle(pt1,pt2,pt3)
    print("area: " + str(area))
    geom.add_geometry(points, facets, facet_markers=box_marker)
    mesh_info = MeshInfo()
    geom.set(mesh_info)
    mesh = build(mesh_info, max_volume=0.01,
                 volume_constraints=True, attributes=True)
    mesh.element_volumes.setup()
    tot = 0;
    #volume count
    for i in range(len(mesh.elements)):
        [p1,p2,p3,p4] = mesh.elements[i]
        [pt1,pt2,pt3,pt4] = [mesh.points[p1], mesh.points[p2],
        mesh.points[p3], mesh.points[p4]]
        tot += volume.volume(pt1,pt2,pt3,pt4)
    print("volume: " +  str(tot))
    d = [0,0,0]
    for i in range(len(mesh.elements)):
        [p1,p2,p3,p4] = mesh.elements[i]
        [pt1,pt2,pt3,pt4] = [mesh.points[p1], mesh.points[p2],
        mesh.points[p3], mesh.points[p4]]
        for i in range(3):
            d[i]+= volume.centre_of_mass(pt1,pt2,pt3,pt4,i)
```

```
    #center of mass, sphericity and time count
    print("center of mass: " + str(d/tot))
    print("sphericity: " + str(volume.sphericity(area,tot)))
    print("timing: %s seconds" % (time.time() - start_time))
    #export to vtk file
    mesh.write_vtk("G0.vtk")


if __name__ == "__main__":
    main()
```

ПРИЛОЖЕНИЕ В

Код программы.Построение и анализ последовательности эллипсоидов

```
def analysis(radius, scalar, center, n):
    import numpy
    from meshpy.tet import MeshInfo, build, Options
    from meshpy.geometry import GeometryBuilder, Marker, make_box, make_ball
    import volume
    geom = GeometryBuilder()
    box_marker = Marker.FIRST_USER_MARKER
    points, facets, _, _ = make_ball(r=radius)
    for i in range(len(points)):
        a = numpy.array(points[i])
        a[2] *= scalar
        a += numpy.array(center)
        points[i] = tuple(a)
    area = 0
    for i in range(len(facets)):
        if len(facets[i][0]) == 4:
            [[p1, p2, p3, p4]] = facets[i]
            [pt1, pt2, pt3, pt4] = [points[p1], points[p2],
            points[p3], points[p4]]
            area += volume.area_of_quadrelateral(pt1, pt2, pt3, pt4)
        elif len(facets[i][0]) == 3:
            [[p1, p2, p3]] = facets[i]
            [pt1, pt2, pt3] = [points[p1], points[p2], points[p3]]
            area += volume.area_of_triangle(pt1, pt2, pt3)
    print("area: " + str(area))
    geom.add_geometry(points, facets, facet_markers=box_marker)
    mesh_info = MeshInfo()
    geom.set(mesh_info)
    mesh = build(mesh_info, max_volume=0.01,
                 volume_constraints=True, attributes=True)
    mesh.element_volumes.setup()
    # volume count
    tot = 0
    for i in range(len(mesh.elements)):
        [p1, p2, p3, p4] = mesh.elements[i]
        [pt1, pt2, pt3, pt4] = [mesh.points[p1], mesh.points[p2],
        mesh.points[p3], mesh.points[p4]]
        tot += volume.volume(pt1, pt2, pt3, pt4)
    print("volume: " + str(tot))
    #center of mass count
    d = [0, 0, 0]
    for i in range(len(mesh.elements)):
        [p1, p2, p3, p4] = mesh.elements[i]
        [pt1, pt2, pt3, pt4] = [mesh.points[p1], mesh.points[p2],
        mesh.points[p3], mesh.points[p4]]
        for j in range(3):
```

```
            d[j] += volume.centre_of_mass(pt1, pt2, pt3, pt4, j)

    print("center of mass: " + str(d / tot))
    #sphericity count
    print("spherical: " + str(volume.sphericity(area, tot)))
    # file write
    # mesh.write_vtk("G"+str(n)+".vtk")


def main():
    a = []
    for i in range(10):
        a.append(1/(i+1))
    for i in range(len(a)):
        print("ellipse"+str(i+1))
        analysis(1,a[i],[0,0,0], i+1)
        print("==========================")


if __name__ == "__main__":
    main()
```

ПРИЛОЖЕНИЕ Г

Код программы.Построение и анализ сетки методом выбора тетраэдров

```
# surface
def function(x,y,z):
    return (x-0.5)**2 + (y-0.5)**2 + (z-0.5)**2 - 1/16


def main():
    import numpy
    from meshpy.tet import MeshInfo, build, Options
    from meshpy.geometry import GeometryBuilder, Marker, make_box, make_ball
    import volume
    import math
    import time
    # mesh generation
    start_time = time.time()
    geom = GeometryBuilder();
    box_marker = Marker.FIRST_USER_MARKER
    points, facets, _, facet_markers = make_box(
        numpy.array([0, 0, 0]), numpy.array([1, 1, 2])
    )
    geom.add_geometry(points, facets, facet_markers=box_marker)
    mesh_info = MeshInfo()
    geom.set(mesh_info)
    mesh_info.regions.resize(1)
    mesh_info.regions[0] = ([0, 0, 0] + [1, 0.001, ])
    mesh = build(mesh_info, max_volume=0.01,
                 volume_constraints=True, attributes=True)

    in_region = []
    for i in range(len(mesh.elements)):
        [p1, p2, p3, p4] = mesh.elements[i]
        [pt1, pt2, pt3, pt4] = [mesh.points[p1], mesh.points[p2],
        mesh.points[p3], mesh.points[p4]]
        # selection of inside tetrahedra
        # if (function(pt1[0],pt1[1],pt1[2]) <= 0
        #         and function(pt2[0],pt2[1],pt2[2]) <= 0 and
        function(pt3[0],pt3[1],pt3[2]) <= 0
        #         and function(pt4[0],pt4[1],pt4[2] <= 0)):
        #     in_region.append(i)
        # selection of inside and intersecting tetrahedra
        if (function(pt1[0], pt1[1], pt1[2]) <= 0
                or function(pt2[0], pt2[1], pt2[2]) <= 0 or function(pt3[0],
                pt3[1], pt3[2]) <= 0
                or function(pt4[0], pt4[1], pt4[2] <= 0)):
            in_region.append(i)
    print(str(len(in_region)/len(mesh.elements)))
    tot = 0
```

```
    # volume
    for i in range(len(in_region)):
        [p1, p2, p3, p4] = mesh.elements[in_region[i]]
        [pt1, pt2, pt3, pt4] = [mesh.points[p1], mesh.points[p2],
        mesh.points[p3], mesh.points[p4]]
        tot += volume.volume(pt1, pt2, pt3, pt4)
    print("volume: " + str(tot))
    # center of mass
    d = [0, 0, 0]
    for i in range(len(in_region)):
        [p1, p2, p3, p4] = mesh.elements[in_region[i]]
        [pt1, pt2, pt3, pt4] = [mesh.points[p1], mesh.points[p2],
        mesh.points[p3], mesh.points[p4]]
        for j in range(3):
            d[j] += volume.centre_of_mass(pt1, pt2, pt3, pt4, j)
    print("center of mass: " + str(d / tot))
    # surface area for sphericity is counted by formula for a sphere
    print("sphericity: " + str(volume.sphericity(math.pi*0.25,tot)))
    print("timing: %s seconds" % (time.time() - start_time))


if __name__ == "__main__":
    main()
```

ПРИЛОЖЕНИЕ Д

Код программы.Функции для нахождения точек пересечения

```
#intersection test
def intersectQ(p1,p2,surface):
    a = surface(p1[0], p1[1],p1[2])
    b = surface(p2[0], p2[1], p2[2])
    if a * b == 0:
        return 0
    return a * b / abs(a * b)

# finding intersection points
def intersection_point(p1,p2,surface):
    from sympy import symbols, solve
    import numpy as np
    t = symbols('t')
    x = (p2[0] - p1[0]) * t + p1[0]
    y = (p2[1] - p1[1]) * t + p1[1]
    z = (p2[2] - p1[2]) * t + p1[2]
    sol = solve(surface(x,y,z))
    for i in sol:
        x = (p2[0] - p1[0]) * i + p1[0]
        y = (p2[1] - p1[1]) * i + p1[1]
        z = (p2[2] - p1[2]) * i + p1[2]
        if np.dot(np.array(p2) - np.array([x,y,z]),
        np.array([x,y,z]) - np.array(p1)) > 0:
            return [x,y,z]

# surface
def function(x,y,z):
    return (x-0.5)**2 + (y-0.5)**2 + (z-0.5)**2 - 1/16

# example
def main():
    from intersection import tetrahedral_division
    p1 = [0, 0, 0]
    p2 = [2, 0, 0]
    p3 = [0, 2, 0]
    p4 = [0, 0, 2]
    surface = function
    sides = tetrahedral_division(p1, p2, p3, p4)
    a = []
    for i in sides:
        if intersectQ(i[0], i[1], surface) == -1:
            print(intersection_point(i[0],i[1],surface))

if __name__ == "__main__":
    main()
```

ПРИЛОЖЕНИЕ Е

Код программы.Постоение и анализ сетки методом выпуклой оболочки

```
def main():
    import numpy
    from meshpy.tet import MeshInfo, build, Options
    from meshpy.geometry import GeometryBuilder, Marker, make_box
    import volume
    import math
    from scipy.spatial import ConvexHull
    import surface_intersection as si
    from intersection import tetrahedral_division
    import sys
    import time
    #mesh generation
    geom = GeometryBuilder()
    box_marker = Marker.FIRST_USER_MARKER
    points, facets, _, facet_markers = make_box(
        numpy.array([0, 0, 0]), numpy.array([1, 1, 2])
    )
    geom.add_geometry(points, facets, facet_markers=box_marker)
    mesh_info = MeshInfo()
    geom.set(mesh_info)
    mesh_info.regions.resize(1)
    mesh_info.regions[0] = ([0, 0, 0] + [1, 0.001, ])
    mesh = build(mesh_info, max_volume=0.01,
                 volume_constraints=True, attributes=True)
    #finding intersection points
    surface = si.function
    in_region = []
    in_region_tetr = []
    total = len(mesh.elements)
    start_time = time.time()
    for i in range(total):
        [p1, p2, p3, p4] = mesh.elements[i]
        [pt1, pt2, pt3, pt4] = [mesh.points[p1], mesh.points[p2],
        mesh.points[p3], mesh.points[p4]]
        sides = tetrahedral_division(pt1, pt2, pt3, pt4)
        print(str(int(100*i/total)) + '%')
        sys.stdout.write('\x1b[1A')
        sys.stdout.write('\x1b[2K')
        for j in sides:
            if si.intersectQ(j[0], j[1], surface) == -1:
                in_region.append(si.intersection_point(j[0],j[1],surface))
                in_region_tetr.append(i)
    print("timing: %s seconds" % (time.time() - start_time))
    #counting number of intersected tetrahedrons
    print(str(len(in_region))+"/"+ str(len(mesh.elements)))
    hull = ConvexHull(numpy.array(in_region))
```

```
    # volume, area and sphericity
    print("volume: " + str(hull.volume))
    print("area: " + str(hull.area))
    print("sphericity: " + str(volume.sphericity(hull.area,hull.volume)))
    tot = 0
    d = [0, 0, 0]
    for i in range(len(in_region_tetr)):
        [p1, p2, p3, p4] = mesh.elements[in_region_tetr[i]]
        [pt1, pt2, pt3, pt4] = [mesh.points[p1], mesh.points[p2],
        mesh.points[p3], mesh.points[p4]]
        for j in range(3):
            d[j] += volume.centre_of_mass(pt1, pt2, pt3, pt4, j)
        tot+= volume.volume(pt1,pt2,pt3,pt4)
    #center of mass
    print("center of mass: " + str(d/tot))
    print("timing: %s seconds" % (time.time() - start_time))


if __name__ == "__main__":
    main()
```