МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г.ЧЕРНЫШЕВСКОГО»**

Кафедра Математического и компьютерного моделирования

Автоматизация построения расчётных областей для метода конечных объёмов

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 413 группы

направление 01.03.02 — Прикладная математика и информатика

механико-математического факультета

Иванова Дмитрия Алексеевича

Научный руководитель
доцент, к.т.н. И.А. Панкратов

Зав. кафедрой
зав. каф., д.ф.-м.н., доцент Ю.А. Блинков

Саратов 2019

Введение. Динамичная история практического применения методов вычислительной гидрогазодинамики насчитывает всего полвека. За это время возник и достиг высокого совершенства метод конечных объемов. Этот метод занимает лидирующее положение в разработках вычислительных средств, ориентированных на решение задач гидрогазодинамики и конвективного теплообмена в областях сложной геометрии. Отправной точкой метода конечных объемов является интегральная формулировка законов сохранения массы, импульса, энергии и др. Балансовые соотношения записываются для небольшого контрольного объема; их дискретный аналог получается суммированием по всем граням выделенного объема потоков массы, импульса и т.д., вычисленных по каким-либо квадратурным формулам. Поскольку интегральная формулировка законов сохранения не накладывает ограничений на форму контрольного объема, метод конечных объемов пригоден для дискретизации уравнений гидрогазодинамики как на структурированных, так и на неструктурированных сетках с различной формой ячеек, что, в принципе, полностью решает проблему сложной геометрии расчетной области.

Лидирующее положение метода конечных объемов по отношению к другим способам дискретизации уравнений гидрогазодинамики подтверждается тенденциями современного рынка программного обеспечения.

В данной работе в качестве инструмента для моделирования различных процессов используется пакет прикладных программ OpenFOAM (Open Source Field Operation And Manipulation CFD ToolBox). Это открытая интегрируемая платформа для численного моделирования задач механики сплошной среды. Она представляет собой свободно распространяемый инструментарий вычислительной гидродинамики для операций с полями (скалярными, векторными и тензорными). На сегодня является одним из наиболее известных приложений, предназначенных для вычислений по методу конечных объемов. В состав пакета OpenFOAM входит набор утилит, решателей и средств отображения результатов.

Целью данной выпускной бакалаврской работы является автоматизация построения расчетных областей для метода конечных объемов.

Данная работа актуальна, т.к. хотя среди проанализированных источников встречаются инструменты для автоматизации и манипулирования слова-

рями, тем не менее они не реализуют в полной мере функционала необходимого для автоматизации рассмотренных задач.

Работа состоит из четырех разделов:

- В первом разделе будут рассмотрены утилиты для построения расчетных областей, входящие в пакет OpenFOAM, такие как blockMesh и snappyHexMesh, а также информация о STL-файлах.
- Второй раздел связан с определением рутинных функций и выбором средств их автоматизации.
- Третий раздел содержит описание реализации программы, выбранными средствами автоматизации.
- В четвертом разделе будут даны примеры решения задачи построения расчетных областей.

Платформа для численного моделирования OpenFOAM и краткая информация о blockMesh. OpenFOAM — свободно распространяемый инструментарий вычислительной гидродинамики для операций с полями (скалярными, векторными и тензорными). Является одним из «законченных» и известных приложений, предназначенных для вычислений методом конечных объемов (finite volume method).

Код OpenFOAM, изначально разрабатываемый в Великобритании компанией OpenCFD Limited, в настоящее время поддерживается и развивается усилиями некоммерческой организации The OpenFOAM Foundation.

OpenFOAM состоит из двух частей: библиотеки классов, решающих дифференциальные уравнения в частных производных, которые необходимы при численном моделировании, и библиотеки «программ-решателей» (solvers), которые используют данные классы для решения конкретных задач моделирования. Пакет OpenFOAM реализован в рамках объектно-ориентированного подхода на языке C++, что обеспечивает полную инкапсуляцию, иначе разделение, процессов построения сетки для дискретизации сплошной среды и методов аппроксимации уравнений в частных производных, алгебраических уравнений. Также отдельным блоком выступают различные утилиты необходимые для обработки данных.

Триангулированная геометрия в формате стереолитографии(STL). STL – формат файла, широко используемый для хранения

трехмерных моделей объектов. Информация об объекте хранится как список описывающих поверхность треугольных граней и их нормалей. STL-файл может быть текстовым (ASCII) или двоичным. Свое название получил от сокращения термина «Stereolithography», поскольку изначально применялся именно в этой технологии трехмерной печати.

Поскольку STL-файл в формате ASCII может быть очень большим, существует двоичная версия данного формата. Файл начинается с заголовка из 80 символов (который обычно игнорируется, но не должен начинаться с 'solid', так как с этой последовательности начинается ASCII STL файл). После заголовка идет 4 байтовое беззнаковое целое число, указывающее количество треугольных граней в данном файле. После этого идут данные, характеризующие каждый треугольник.

Каждый треугольник описывается двенадцатью 32 битными числами с плавающей точкой: 3 числа для нормали и по 3 числа на каждую из трёх вершин для X/Y/Z координат. После идут 2 байта беззнакового 'short', который называется 'attribute byte count'.

Подробное рассмотрение утилиты для построения сеток snappyHexMesh. Перед запуском утилиты snappyHexMesh, пользователь должен выполнить следующие действия:

- Подготовить STL-файлы, которые будут предметом улучшения «фоновой» сетки в бинарном или ASCII формате. Данные файлы необходимо разместить в директории /constant/triSurface примера.
- Создать «фоновую» гексаэдральную сетку, которая определяет расчетную область и представляет базовый уровень плотности будущей сетки.
- Создать словарь snappyHexMeshDict с соответствующими данными и поместить его в директорию system данного примера.

Для работы утилиты snappyHexMesh необходима предварительно созданная «фоновая» сетка. Одна из самых простых утилит которую можно использовать для этой цели – blockMesh. «Фоновая» сетка определяет первоначальное разбиение рассматриваемой области. Если в дальнейшем не задать параметры для улучшения сетки, то «фоновая» сетка станет итоговой.

Определение рутинных функций и выбор средств автоматизации. Вид задачи не требует значительного быстродействия от программы.

Ввиду потенциально не большого числа строк в коде динамическая типизация не представляется недостатком, а множество библиотек, переносимость и присутствие парадигмы объектно-ориентированного программирования облегчает разработку программы.

При решении задачи будут использованы сторонние библиотеки для экономии времени и для того чтобы не повторять уже написанные функции.

Numpy-stl – простая библиотека для работы с STL-файлами (и 3D объектами в общем случае). Благодаря тому, что все операции в основном базируются на библиотеке numpy, numpy-stl одна из самых быстродействующих библиотек среди доступных на Python для редактирования STL-файлов.

PyFoam – библиотека для запуска и редактирования словарей OpenFOAM.

Реализация программы выбранными средствами автоматизации.

Данная задача автоматизации легко ложится на парадигму объектно-ориентированного программирования.

Были созданы три основных класса:

- SnappyHexMesh
- BlockMesh
- Stl

В них были реализованы методы, позволяющие манипулировать словарями, применяя несложные алгоритмы, комбинируя функции данных классов, добиваться автоматизации некоторых рутинных процессов.

Класс SnappyHexMesh. В классе SnappyHexMesh были реализованы следующие *основные* функции:

- `__init__ (self, stlFile, path="")`
  Конструктор экземпляра класса. Принимает на вход экземпляр класса Stl и абсолютный путь до словаря SnappyHexMeshDict.
- `setStl(self, path, stlFile)`
  Данная функция по заданному пути до SnappyHexMeshDict и экземпляру класса Stl изменяет имя рассматриваемого в словаре STL-файла.
- `getLocationInMesh(self)`

Возвращает вектор, который указывает, какая область сетки будет разбиваться на элементы.

- `setLocationInMesh(self, value)`
  Принимает значение value, которое задает вектор, указывающий на область разбития сетки на элементы.
- `toggleTargetMesh(self, innerMesh, bmFile, stlFile, LocInMeshType=0)`
  Данная функция служит для переключения, без задания конкретных координат вектора locationInMesh, между внутренней и внешней областью построения разбиения сетки.
- `saveDict(self, path="")`
  Сохраняет словарь snappyHexMeshDict по указанному пути(path). По умолчанию, при пустой переменной path, данный словарь сохраняется вместо заданного при инициализации экземпляра класса исходного файла.

Класс BlockMesh В классе BlockMesh были реализованы следующие функции:

- `__init__(self, path="")`
  Конструктор экземпляра класса. Атрибут path передают в функцию абсолютный путь до файла blockMeshDict. Инициализирует словарь, содержащий разделы файла и их содержимое.
- `__normalizeVerticesOrder(self)`
  Данная функция расставляет вершины в «естественном порядке», который необходим для автоматического формирования «фоновой» сетки для STL-файла. Таким образом, файл blockMeshDict должен содержать прямоугольный параллелепипед.
- `findMinsMaxs(self)`
  Данная функция возвращает кортеж (tuple) из 6 переменных. По 2 переменные для каждой оси OX, OY, OZ. Каждая пара представляет собой два вектора, определяющие ограничивающий параллелепипед для «фоновой» сетки соответствующей оси.
- `editVertices(self, factors, offset=(0, 0, 0), figureOffset=(0, 0, 0))`

Для работы данной функции файл blockMeshDict должен содержать прямоугольный параллелепипед. Функция редактирует или изменяет размеры параллелепипеда, а конкретнее расстояния между вершинами сетки.

- `saveDict(self, path="")`
  Сохраняет словарь blockMeshDict по указанному пути(path). По умолчанию, при пустой переменной path, данный словарь сохраняется вместо исходного файла, заданного при инициализации экземпляра класса.
- `adjustBlockMeshToStl(self, stlFile, difBtwStlAndMesh)`
  Данная функция при помощи метода editVertices реализует автоматическую подгонку «фоновой» сетки под заданный STL-файл.
- `verifyingBlockMeshSize(self, stlFile, difBtwStlAndMesh=0)`
  Данная функция необходима для проверки соответствия размеров «фоновой» сетки и области STL-модели.
- `refineMesh(self, factors, offset=(0, 0, 0))`
  Данная функция уточняет сетку, увеличивая число ячеек в параллелепипеде.
- `convertToMeters_ToUnit(self)`
  Функция, переводящая все записанные в blockMeshDict данные в метры. Иными словами метод, устанавливающий значение поля convertToMeters равным единице.

Класс Stl.

- `__init__(self, path="")`
  Конструктор экземпляра класса. Атрибут path передает в функцию абсолютный путь до файла содержащего STL-геометрию. Инициализирует переменную класса numpy-stl, необходимую для работы основной программы.
- `getFileName(self, splitExt=False)`
  Возвращает имя STL-файла. Содержит необязательный параметр slitExt. По умолчанию (False) возвращает строку, содержащую, имя файла вместе с его расширением. При значении True возвращает массив из двух элементов: имя файла и его расширение.
- `findMinsMaxs(self)`

Данная функция возвращает кортеж (tuple) из 6 переменных. По 2 переменные для каждой оси OX, OY, OZ. Каждая пара представляет собой два вектора, определяющие ограничивающий параллелепипед для STL-области соответствующей оси.

Примеры решения задач построения расчетных областей. Рассмотрим некоторые примеры решения конкретных задач с помощью построенных функций.

Рассмотрим директорию примера flange из пакета OpenFOAM. Подключим нужные библиотеки:

```
import numpy as np
import sys
import SnappyHexMesh
import BlockMesh
import Stl
```

Инициализируем пути для данного примера:

```
bm_path = ".../flange/system/blockMeshDict"
stl_path = ".../constant/triSurface/star.stl"
shm_path = ".../flange/system/snappyHexMeshDict"
```

Таким образом, был указан путь к словарям для «фоновой» сетки и для построения области вокруг STL-файла.

Далее изменим «фоновую» сетку таким образом, чтобы она соответствовала данному STL-файлу. Для этого применим следующие функции:

```
bmFile.convertToMeters_ToUnit()
bmFile.adjustBlockMeshToStl(stlFile, 1)
bmFile.saveDict()
```

Сначала была использована функция для перевода всех значений файла blockMeshDict в метры. Затем функция, покрывшая «фоновой» сеткой STL-модель с разницей между ограничивающими параллелепипедами в один метр. После этого перезаписываем файл blockMeshDict.

Теперь изменим файл snappyHexMeshDict:

```
shmFile.toggleTargetMesh(True, bmFile, stlFile)
shmFile.saveDict()
```

Первой строчкой задана область разбиения внутри STL-области. Второй строчкой словарь был перезаписан.

Результат работы показан в соответствии с рисунком 1. При этом поле locationInMesh в словаре snappyHexMeshDict приняло значение (0 0 50).

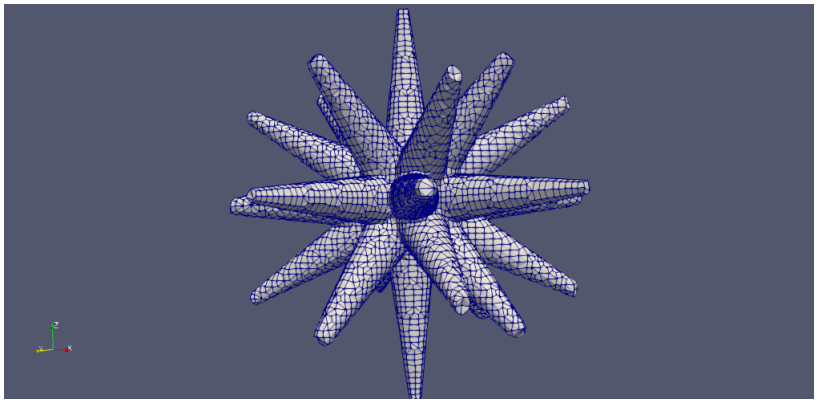

Рисунок 1 — Результат работы примера

Теперь построим внешнюю область сетки, просто изменив параметр в функции с True на False:

```
shmFile.toggleTargetMesh(False, bmFile, stlFile)
```

Получим результат в соответствии с рисунком 2.

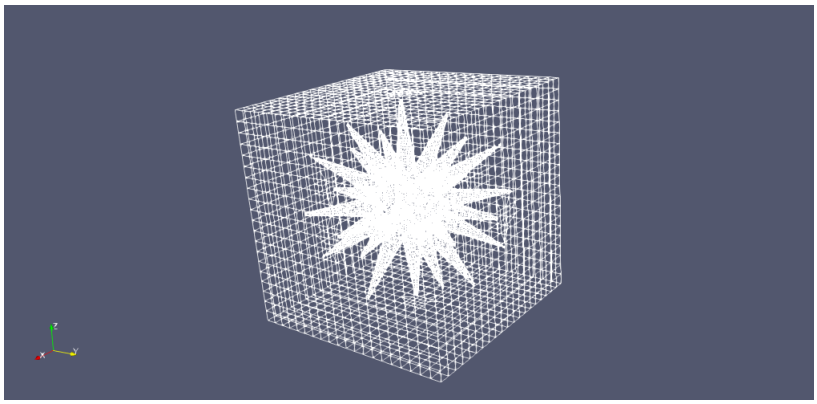

Рисунок 2 — Результат работы примера с использованием функции вида toggleTargetMesh(False, bmFile, stlFile)

При этом поле locationInMesh приняло значение (-150.25 -150.25 -100.25). Как видно из приведенного примера расстояние между «фоновой» сеткой и

STL-областью не велико. Это приводит к возникновению артефактов. Например, искажению граничных ячеек «фоновой» сетки вблизи лучей звезды.

Избежать этого можно изменив рассматриваемую функцию следующим образом:

```
bmFile.adjustBlockMeshToStl(stlFile, 5)
```

Увеличив таким образом расстояние между сеткой и STL-областью.

Заключение. На основании проделанного теоретико-практического исследования в данной бакалаврской работе был изучен пакет OpenFOAM, а также некоторые утилиты из данного пакета: blockMesh, snappyHexMesh.

Названные выше утилиты позволяют строить расчетные области для произвольных STL-файлов. Утилита blockMesh обеспечивает построение «фоновой» сетки, snappyHexMesh используется для внедрения STL-модели в «фоновую» сетку.

Цель данной бакалаврской работы состояла в автоматизации построения расчетных областей для метода конечных объемов. Для ее достижения были поставлены и выполнены ряд задач:

- Были рассмотрены утилиты, входящие в пакеты OpenFOAM: blockMesh, snappyHexMesh, а также структура STL-файла.
- Были определенны рутинные функции, связанные с работой с этими утилитами. Например, нахождение точки locationInMesh для выбора построения области сетки или увеличение (уменьшение) точности заданной сетки. В качестве средств автоматизации был выбран язык программирования Python.
- В целях расширяемости и удобства освоения была описана реализация автоматизации данных функций.
- Приведены примеры решения задач построения расчетных областей с их помощью.

Таким образом, поставленная во введении цель выпускной бакалаврской работы была достигнута.