Документация по OpenFOAM для начинающих пользователей

Автор: Рассадин А.А. студент группы 601-01
email: rassadin_aa@edu.surgu.ru

7 июня 2024 г.

Введение

Данный документ это краткое руководство по программному комплексу OpenFOAM. Материал составлен так, что его не обязательно читать от начали и до конца. Его нужно воспринимать скорее как справочник, в котором можно быстро найти решение своего вопроса. Данная работа разделена на три главы.

В первой главе сделан обзор на библиотеку OpenFOAM, объясняется процесс использования этого пакета. В конце этой части есть раздел, где приведены ссылки на работы других авторов, откуда можно получить дополнительную информацию. Тем, кто впервые знакомится с OpenFOAM, автор рекомендует прочитать эту главу **полностью**.

Вторая глава, фактически, является тем самым справочником. В ней приведены шаблоны для выполнения стандартных этапов решения задачи: создания расчетной сетки, определения начальных и граничных условий, добавления источников в уравнения. В конце этой главы можно найти ссылки на документацию по обработке результатов вычислений.

В третьей главе сделан разбор решения задачи об эволюции температуры в твердом теле. На примере этой задачи показано выполнение всех промежуточных действий от постановки задачи, до анализа данных после её решения. После рассмотрения этого примера должно сложиться общее представление о том, как используется OpenFOAM на практике.

Для данного руководства создан общедоступный *репозиторий*, в котором *хранятся все материалы и примеры.*

Здесь основное внимание уделено *использованию платформы* OpenFOAM. В *работе*[1] автор данного руководства подробнее рассмотрел численные методы, которые используются для решения задач гидродинамики, а также особенности их реализации в программной платформе OpenFOAM.

[1] Если при переходе по ссылке документ не открывается, то его можно загрузить, кликнув на кнопку на той же странице. В этом случае точно получится посмотреть документ.

Оглавление

Глава 1

Обзор программного комплекса OpenFOAM

1.1 Что такое OpenFOAM

OpenFOAM это библиотека с открытым исходным кодом, написанная на языке C++, которая разработана для численного решения задач гидродинамики. У этого проекта есть *несколько ответвлений*. В данной работе рассмотрена *ветка*, выпускаемая компанией ESI Group. Проект кросс-платформенный, то есть OpenFOAM можно установить как на Windows, так и на Linux. Хотя изначально проект разработан под Linux, поэтому разработчики гарантируют выполнение всего функционала только для этой платформы.
В OpenFOAM численное решение задач происходит с помощью метода контрольных объемов (подробнее МКО описан в пособии [1], или *данную работу автора*[1], посвященную численному решению задач методом контрольных объемов.). При использовании этого метода исходную задачу заменяют её дискретным аналогом. Непрерывную расчетную область разбивают на конечное число ячеек (контрольных объемов). Затем ставится задача определения значений искомой величины в центрах этих ячеек. Дифференциальные уравнения интегрируют по каждой ячейке. При аппроксимации интегралов применяют различные схемы. В результате получается система линейных уравнений (СЛАУ). Искомые значения определяют, после решения полученной системы уравнений.
В состав библиотеки входит *набор готовых программ (утилит)* для выполнения всех этапов решения задачи данным методом. Есть программы для составления расчетных сеток, реализовано множество схем аппроксимации дифференциальных операторов, есть несколько алгоритмов решения систем линейных уравнений, средства для обработки результатов вычислений. Также вместе с комплексом OpenFOAM предоставляется собрание *демонстрационных задач*. Среди этих задач есть *верификационные*, они показывают качество и точность решений, получаемых с помощью средств OpenFOAM. Некоторые задачи-примеры рассмотрены в отдельном *руководстве*, выпущенном разработчиками этой библиотеки.

[1] Если при переходе по ссылке документ не открывается, то его можно загрузить, кликнув на кнопку на той же странице. В этом случае точно получится посмотреть документ.

1.2 Как использовать OpenFOAM для решения задач

1.2.1 Установка программного комплекса

Процесс установки библиотеки OpenFOAM для операционных систем Windows и Linux отличается. Для пользователей, использующих Windows посмотреть процесс установки можно в *данном источнике*. Также на сайте другой ветки OpenFOAM есть *раздел*, посвященный установке под Windows. Для платформы Linux (под разные дистрибутивы) и той версии OpenFOAM, которую использовал автор, загрузка пакета и его установка описаны на *данном сайте*.
Для работы с OpenFOAM есть много сторонних программ, в каком-то смысле главная среди них это ParaView. При этом чаще всего её нужно *устанавливать отдельно*. Но уже этом случае все довольно просто. В дистрибутиве Linux Debian программа ParaView это просто пакет, который устанавливается через пакетный менеджер **apt** одной командой:

```
sudo apt-get install paraview
```

1.2.2 Файловая структура директории с задачей

Программный комплекс OpenFOAM не имеет графического интерфейса пользователя. Это значит, что его нельзя запустить как обычную программу. *Работа с ним осуществляется через терминал (командную строку).* В процессе использования этого пакета человек вызывает через терминал нужные ему утилиты. Их поведение регулируется на основании текстовых файлов, которые принято называть *словарями*.
Для каждой задачи создаётся *отдельная одноименная директория*, её имя задает пользователь. В ней расположены необходимые словари с настройками для утилит. Существует специальный формат как для директории с задачей, так и для самих словарей. Это определенный стандарт, которого нужно придерживаться, чтобы не было ошибок во время работы утилит. Общие правила синтаксиса словарей OpenFOAM описаны в *этом источнике*, а формат директории кратко рассмотрен ниже, более подробно об этом можно узнать в *данном разделе* документации.
На рис. 1.1 приведена схема, где на примере директории **case** показана общая структура любой директории с задачей.

В данной схеме директория для задачи называется **case**. В неё вложены как минимум ещё 3 директории:

- Директория **system** содержит следующие файлы с настройками:
  - **fvSchemes** – Схемы аппроксимации дифференциальных операторов (слагаемых в уравнениях).
  - **fvSolution** – Настройки для решения СЛАУ.
  - **controlDict** – Шаг по времени, конечное время, способ сохранения временных слоев.

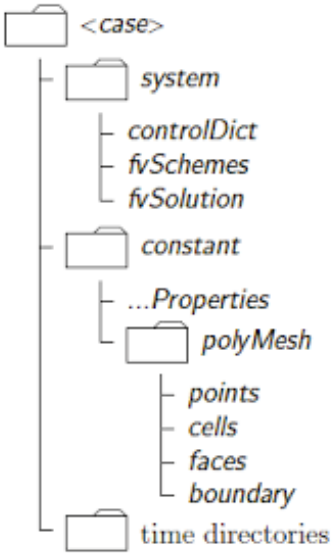


Рис. 1.1: Файловая структура директории с задачей

  - Дополнительные файлы с настройками для утилит, используемых при решении задачи.

- Директория **constant** содержит величины неизменяемые в процессе решения:
  - Поддиректория **polyMesh** – хранит данные о расчетной сетке, в виде нескольких текстовых файлов.
  - Прочие словари, например, в **transportProperties** записаны физические константы, а в **turbulenceProperties** выбирается модель турбулентности.
- Остальные каталоги, содержащие расчетные данные *различных временных слоях*. Имя каждого такого каталога соответствует значению временного слоя. В самих этих каталогах находятся файлы со значениями рассчитываемых физических (скалярных и векторных) полей. До начала расчетов обязательно *хотя бы один* такой каталог должен быть, он содержит начальные и граничные условия задачи. Чаще всего это момент времени $t = 0$ и поэтому директория называется **0**. Внутри каталога с данными каждому полю соответствует отдельный одноименный текстовый файл. В таких файлах записаны размерность поля, значения во внутренних и граничных точках сетки.

1.2.3 Запуск утилит через терминал

Чтобы использовать библиотеку OpenFOAM, нужно запускать её утилиты через терминал (командную строку). Это относится ко всем платформам: Windows, Linux, MacOS. Для доступа к этим программам нужно, чтобы *«было активировано окружение OpenFOAM>*. В последних версиях пакета для активации окружения достаточно выполнить в терминале команду:

```
openfoam<version>
```

Вместо <version> нужно указать установленную версию пакета. На текущий момент последняя версия это 2306. Запуск команды выше подключит необходимые перемен-

ные окружения и сделает остальные действия, после которых станут доступны все утилиты OpenFOAM. Но это будет относиться только *текущему* сеансу работы с терминалом. Для вновь открытого терминала придётся повторять эту процедуру. Подробнее о том, как работать с терминалом для применения OpenFOAM можно прочитать в *этом источнике*.

Рекомендация для опытных пользователей Linux Deiban:
Если у вас есть доступ к правам администратора на Linux, вы можете сделать так чтобы окружение OpenFOAM было ***всегда активно****. Для этого нужно добавить содержимое файла* `/usr/lib/openfoam/openfoam2306/etc/bashrc` *в конец файла* `/etc/bash.bashrc`*. Будьте осторожны при редактировании файла* `/etc/bash.bashrc`*, так как он регулирует всю работу терминала на Linux.*

1.3 Обзор основных файлов для любой задачи

1.3.1 Файл system/fvSchemes для способа аппроксимации слагаемых

В файле **fvSchemes** пользователь указывает схемы аппроксимации слагаемых, входящих в дифференциальные уравнения. Если решается несколько уравнений, то можно для отдельных величин выбрать разные схемы аппроксимации, или задать схему по умолчанию, которая будет применяться ко всем слагаемым.
Пример заполнения этого файла можно посмотреть в демонстрационной задаче в разделе 3.3.2.
OpenFOAM содержит большой набор численных схем, их описание и реализацию можно посмотреть в *этом разделе* документации.

1.3.2 Файл system/fvSolution для настройки решения СЛАУ

Файл **fvSolution** определяет параметры решения системы линейных уравнений (СЛАУ), а также он содержит настройки для алгоритмов решения уравнения Навье-Стокса: SIMPLE, PISO, PIMPLE.
Для каждого рассчитываемого поля нужно указать: метод решения СЛАУ и критерий остановки итерационного решения.
В общем виде СЛАУ полученная после дискретизации уравнения записывается так:

$$A\boldsymbol{x} = \boldsymbol{b} \tag{1.1}$$

При итерационном подходе решение получается найти с заданной погрешностью за некоторое число шагов. На первом этапе выбирается начальное приближение – вектор $\boldsymbol{x}^{\mathbf{1}}$ю Для этого вектора (на этой итерации) *абсолютная невязка* R^1_{abs} в общем виде определяется как значение следующей нормы:

$$R^1_{abs} = ||\boldsymbol{b} - A\boldsymbol{x}^{\mathbf{1}}|| \tag{1.2}$$

Критерий остановки по значению *абсолютной невязки* означает, что на n-ой итерации норма R^n_{abs} стала достаточно малой.

Для n-ой итерации *относительная невязка* R^n_{ref} определяется как отношение текущего значения невязки R^n_{abs} к начальному R^1_{abs}, то есть по формуле:

$$R^n_{ref} = \frac{R^n_{abs}}{R^1_{abs}}, \qquad 0 \leq R^n_{ref} \leq 1 \tag{1.3}$$

Остановка по значению *относительной невязки* означает, что на n-ой итерации величина R^n_{ref} стала достаточно малой.
Пример заполнения файла **fvSolution** можно посмотреть в демонстрационной задаче в разделе 3.3.3.
Более подробное объяснение предназначение файла **fvSolution** можно найти в Данному файлу отведен следующий *этом разделе* документации.

1.3.3 Файл system/controlDict для контроля проведения расчетов

Словарь **controlDict** используется для общего контроля решения. В нем задается значение шага по времени, начальное и конечное время расчета, параметры сохранения значений. Об этом файле можно узнать больше в *этом источнике*.
Пример заполнения файла **controlDict** можно посмотреть в демонстрационной задаче в разделе 3.3.4.

1.4 Рекомендуемые источники с документацией

О численных методах для гидродинамики:
Основные этапы численного решения задач гидродинамики рассмотрены в *данной работе*[2], которая выполнена автором данного справочника.
Метод контрольных объемов описан, например, в учебном пособии [1]. О численном решении задач гидродинамики написано в также учебнике [2]. Различным способам аппроксимации слагаемых посвящена диссертация [3] одного из основателей проекта OpenFOAM. Также есть собрание конспектов [4] по гидродинамике от других разработчиков этого пакета.

Официальная документация по OpenFOAM:
Новым пользователям автор рекомендует ознакомиться с руководством пользователя [5], которое создано разработчиками библиотеки OpenFOAM. В этом источнике дана базовая информация, имея которую можно начинать осознанное использование этого инструмента. Также полезно посмотреть разборы обучающих задач [6]. Детали реализации библиотеки можно узнать в источниках: [7] и [8].

Материалы по OpenFOAM от других авторов:
В открытом доступе находится собрание уроков [9] от компании WolfDynamics. Есть обучающий материал от организации CFD-Support [10]. Среди материалов на *русском языке* можно выделить два пособия [11] и [12].

[2]Если при переходе по ссылке документ не открывается, то его можно загрузить, кликнув на кнопку на той же странице. В этом случае точно получится посмотреть документ.

Глава 2

Примеры заполнения словарей

Эта глава является своеобразным справочником. Примеры из этой части можно использовать как шаблоны для своих задач. Для каждого примера есть отдельная ссылка, по которой его можно открыть и загрузить. Здесь представлена *ссылка* на общую директорию со всеми примерами, представленными в этой главе. *Примеры были проверялись на операционной системе Linux Debian.*

2.1 Подготовка расчетной сетки

В OpenFOAM есть утилиты как для создания, так и для конвертации сеток, созданных в сторонних программах. В *данном разделе* документации можно найти информацию о том, как «устроена» расчетная сетка в OpenFOAM, а также описание утилит для её создания.

2.1.1 Блочно-структурированная сетка (утилита `blockMesh`)

Программа `blockMesh` используется для создания блочных структурированных сеток. *Документация для этой программы*. Можно сказать, что её можно применять для создания областей с относительно простой геометрией. Все нужные настройки записываются в одном файле **blockMeshDict**, располагаемом в директории **system**.
В словаре **blockMeshDict** нужно заполнить следующие разделы:

- **verticies**
- **blocks**
- **edges**
- **boundary**

Сначала перечисляют множество *вершин* в секции **verticies**, там каждая вершина нумеруется, начиная с нуля. Ключевое слово **scale** определяет масштаб, это означает, что все координаты вершин будут умножены на число, записанное в **scale**.
В следующей секции **blocks** нужно перечислить *блоки*, находящиеся в расчетной области. Каждый блок определяется 8-ю вершинами через их индексы. В этой же секции для каждого блока задаётся число разбиений вдоль осей Ox, Oy, Oz и метод изменения делений вдоль этих областей (параметр simpleGrading или edgeGrading).

После этого идёт секция **edges**, в которой при необходимости можно задать способ соединения вершин. Их можно соединять, например, дугами окружностей или сплайнами. По умолчанию эту секцию оставляют пустой и тогда вершины соединяются отрезками.

Последняя секция **boundary** отвечает за «разметку» границы. В этой секции частям границы присваивают имена, типы. Чтобы выделить часть границы нужно перечислить грани, входящие в эту часть. Это делается в списке граней, сама же грань это набор из 4-х вершин. При описании грани нужно придерживаться правила нумерации вершин: порядок следования вершин такой же, как при обходе против часовой стрелки, если смотреть с вершины внешней нормали. Границу можно разделить на несколько участков или объединить все грани в один участок. Имя участку можно задать произвольно, в дальнейшем это имя используется при определении граничных условий в файлах в директории **0**. Тип области может влиять на вычисления, например, тип границы **empty** исключает эту границу из расчетов. С помощью такого приема в OpenFOAM решают двухмерные и одномерные задачи. Также есть и другие типы.

Ниже приведёно содержимое файла **blockMeshDict**, который задает расчетную сетку форме *параллелепипеда* с измерениями длинами Lx, Ly, Lz.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2306                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

scale   1.0; // Масштабный множитель, на который умножаются значения вершин

// Длины сторон параллелипипеда
Lx 1.0;
Ly 2.0;
Lz 3.0;
// $Lx и т.д. это макро-подстановка, т.е. вместо $Lx подставляется значение


// Набор вершин
vertices
(
    (0    0   0) // Индекс вершины: 0
    ($Lx  0   0) // 1
    ($Lx $Ly  0) // 2
    (0   $Ly  0) // 3
```

```
    (0    0  $Lz) // 4
    ($Lx  0  $Lz) // 5
    ($Lx $Ly $Lz) // 6
    (0   $Ly $Lz) // 7
);

// По перечисленным в 'vertices' вершинам можно сформировать блоки
// Каждый блок будет разделен на ячейки
blocks
(
    // (10 10 10) - число разбиений по осям Ox, Oy, Oz соответственно

    // simpleGrading - используется, чтобы задавать отношение:
    // [длины ребра последней ячейки к длине ребра первой ячейки]
    // вдоль соответствующей оси (Ox, Oy, Oz)
    // Например, здесь ребёра вдоль оси (Ox) от начала к концу
    // изменятся в 5 раз
    hex (0 1 2 3 4 5 6 7) (10 10 10) simpleGrading (5 1 1)
);

// Выбор способа соединения вершин
// Не заполнен, поэтому вершины соединяются отрезками
// Это способ по умолчанию
edges
(
);

// "Разметка" граничной области (именование граней параллелипипеда)
boundary
(
    leftWall // Грань x = 0
    {
        type patch; // Тип граничной области
        faces ( (0 3 7 4) ); // Список граней, составляющих этот участок границы
        // Каждая грань определяется набором из 4-х вершин
        // При описании грани порядок перечисления вершин такой
        // Как при обходе этих вершин против часовой стрелки
        // Если смотреть с конца вектота внешней нормали к этой грани
    }

    rightWall // Грань x = Lx
    {
        type patch;
        faces ( (2 1 5 6) ); // В списке 'faces' может быть большей одной грани
    }

    bottomWall // Грань z = 0
    {
        type patch;
        faces ( (0 1 2 3) );
    }

    topWall // Грань z = Lz
    {
        type patch;
```

```
        faces ( (4 7 6 5) );
    }

    backWall // Грань y = 0
    {
        type patch;
        faces ( (1 0 4 5) );
    }

    frontWall // Грань y = Ly
    {
        type patch;
        faces ( (3 2 6 7) );
    }

);

// ************************************************************************* //
```

Данный файл можно посмотреть и загрузить перейдя по *ссылке*.
Для создания сетки нужно внутри директории с задачей выполнить команду:

`blockMesh`

Примеры более сложных сеток, которые можно создавать с помощью `blockMesh` можно получить по ссылкам:

- *Создание цилиндра*.
- *Создание цилиндра с «вырезом» по центру*.

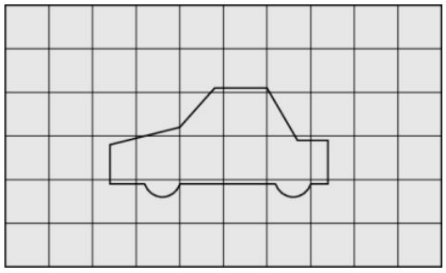
2.1.2 Неструктурированная сетка (утилита `snappyHexMesh`)

Утилита `snappyHexMesh` — стандартная программа пакета OpenFOAM, позволяющая создавать *неструктурированную* стеку для областей со сложной геометрией. Подробное описание данной утилиты сделано в *этом разделе* документации. Процедура создания сетки делится на три этапа:

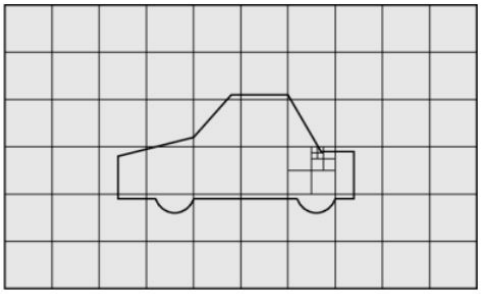
1. Аппроксимация области кубиками.
2. Уточнение граней путем деформации кубиков (создаются многогранники).
3. Добавление дополнительных слоев (дальнейшее измельчение многогранников).

Настройка этих этапов происходит в файле **snappyHexMeshDict**, который как и все конфигурационные файлы должен находиться в директории **system**.
Сначала пользователь должен создать файл в формате **.stl**, определяющий **замкнутую** ограничивающую поверхность. Его можно создать в любом из специализированных редакторов, либо можно использовать стандартные геометрические примитивы OpenFOAM.
Затем нужно создать «объемлющее» пространство в виде большого куба (или параллелепипеда), **целиком содержащего граничную поверхность внутри себя**. Этот этап выполняется с помощью утилиты `blockMesh` (см. раздел 2.1.1).

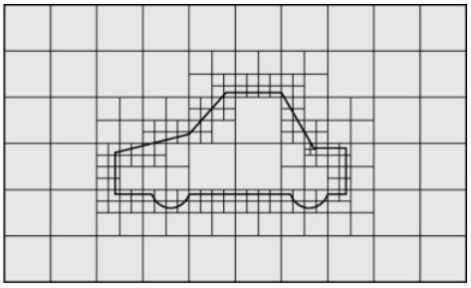
После этого нужно вызвать утилиту `snappyHexMesh`. В процессе работы она будет аппроксимировать ограничивающую поверхность и измельчать внутренние многогранники в соответствии с настройками в **snappyHexMeshDict**. Во время этого процесса лишние блоки будут удаляться, а граничные ячейки будут нужным образом деформироваться. После этого размер ячеек уменьшается вдвое и процесс повторяется, заданное число раз. Схематично это изображено на рисунках (2.1(a) – 2.1(d)).



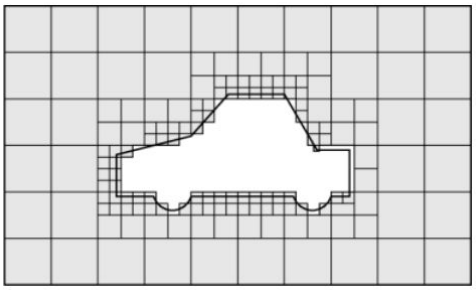
(a) граничная поверхность (машинка) внутри объемлющей блочной сетки



(b) Первое приближение границы



(c) Уточнение особенностей поверхности



(d) Исключение лишних блоков

Для создания сетки нужно последовательно выполнить две команды:

```
blockMesh
```

```
snappyHexMesh -overwrite
```

Файлы для создания сетки в виде шара доступны по *ссылке*.

2.2 Задание начальных условий (internalField)

При решении любой задачи с помощью OpenFOAM в директории должен быть каталог с файлам, где записаны начальные и граничные условия для рассматриваемых физических (скалярных и векторных) полей. Каждому полю соответствует свой файл, имеющий следующую структуру:

- Заголовок, где указано имя поля и его тип (скалярное или векторное).
- Размерность (dimensions).
- Внутренние значения (internalField.)
- Граничные значения (boundartField).

При задании *начальных условий* задаются значения *внутри области* на начальный момент времени, т.е. нужно заполнить значения в секции «internalField».
Подробно синтаксис файлов с данными описан в *этом разделе* документации.

2.2.1 Равномерное (постоянное) значение

Равномерные значения как начальные, так и граничные определяются ключевым словом `uniform`, после которого следует само значение. Для скалярной величины это одно значение, а для векторной это три значения, заключенные в круглые скобки (внутри скобок запятые не ставятся). Ниже приведен пример словаря для скалярной величины T, в котором *начальные значения этого поля равны единице во всей области*:

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2306                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField; // Скалярное поле 'T'
    object      T;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

// Выбор размерности для поля 'T'
//                кг м с К моль А кд
dimensions      [0  0 0 1   0  0 0]; // Размерность поля 'T' это К-Кельвины

// Начальные условия равномерные
internalField   uniform 1.0; // Всюду в области в начальный момент времени T=0

// Граничные условя (первого рода, равномерные и нулевые)
boundaryField
{
```

```
    boundaryRegion // Имя граничной области (см. файл blockMeshDict)
    {
        type    fixedValue;
        value   uniform 0.0;
    }
 }
// ************************************************************************* //
```

Доступ к данному файлу можно получить по *ссылке*.

2.2.2 Неравномерные значения

Для установки неравномерных начальных условий «универсальным» решением будет использование технологии **codeStream** (см. *документацию*). Эта технология позволяет вычислить начальные значения во внутренней области *во время запуска* программы. Используется так называемый динамический код, подробнее см. *источник*. Для демонстрации данной технологии показано, как задать следующие начальные условия:

$$D: \quad T(x,y,z)\Big|_{t=0} = x + \cos y + e^z \tag{2.1}$$

Ниже привед листинг словаря **0/T**, в котором задаются начальные условия, согласно формуле (2.1)

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2306                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      T;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

// Выбор размерности для поля 'T'
//                кг м с К моль А кд
dimensions      [0  0 0 1  0   0  0];

// Почти весь код ниже можно считать "шаблоном"
// Для разных формул изменяется в только содержимое цикла 'forAll'

// С помощью codeStream значения в начальный момент времени
// Будут вычислены во время запуска расчетов
internalField   #codeStream
{
    codeInclude // Подключение основного заголовочного файла
    #{ #include "fvCFD.H" #};
```

```
    codeOptions // Добавление новых путей для поиска подключаемых файлов при компиляции
    #{
        -I$(LIB_SRC)/finiteVolume/lnInclude \
        -I$(LIB_SRC)/meshTools/lnInclude
    #};

    codeLibs // Подключение библиотек
    #{
        -lmeshTools \
        -lfiniteVolume
    #};
    // Секция с кодом C++
    code
    #{
        // Получение данных о сетке (объект 'mesh')
        const IOdictionary& d = static_cast<const IOdictionary&>(dict);
        const fvMesh& mesh = refCast<const fvMesh>(d.db());

        // Создание списка скалярных значений ('T')
        // Эти значения будут записаны в центры ячеек, как начальное условие
        scalarField T(mesh.nCells()); // n.Cells() возвращает кол-во ячеек

        // 'CC' это список векторных значений -- центры ячеек
        const vectorField& CC = mesh.C();

        forAll(CC, i) // Цикл по всем центрам ячеек
        {
            scalar x = CC[i].x(); // 'X' - координата очередной ячейки
            scalar y = CC[i].y(); // 'Y' - координата очередной ячейки
            scalar z = CC[i].z(); // 'Z' - координата очередной ячейки

            // Формула, определяющая начальные условия на поле 'T'
            T[i] = x + Foam::cos(y) + Foam::exp(z);
        }

        T.writeEntry("", os); // Запись значений (в оперативную память)
    #};
};


// Граничные условя (первого рода, равномерные и нулевые)
boundaryField
{
    boundaryRegion // Имя граничной области (см. файл blockMeshDict)
    {
        type    fixedValue;
        value   uniform 0.0;
    }
}

// ************************************************************************* //
```

Доступ к данному файлу можно получить по *ссылке*.

2.3 Задание граничных условий (boundaryField)

Граничным условиям выделен *этот раздел* документации. Нужно отметить, что в OpenFOAM границу можно разделить на несколько регионов и для разных регионов задать разные виды условий.
Для простоты в следующих примерах расчетная сетка это параллелепипед. Его граница состоит из 6 граней, все они объединены в единственный граничный регион, который назван «boundaryRegion».

2.3.1 Условия первого рода (условие Дирихле)

Граничные условия первого рода означают, что значение искомой величины известно на границе. Для скалярной величины T эти условия в общем виде записываются как:

$$\partial D: \quad T(x,y,z,t)\Big|_{t>0} = \varphi(x,y,z,t) \tag{2.2}$$

В формуле выше значения поля T вычисляются на границе области (множество ∂D), в моменты времени после начала расчетов ($t > 0$) и функция $\varphi(x,y,z,t)$ известна.

2.3.1.1 Равномерное значение (fixedValue)

В OpenFOAM тип граничных условий **fixedValue** используется для задания равномерных (постоянных) значений искомой функции на выбранном участке границы, т.е. это условия первого рода. Для примера, скалярному полю T задаются граничные условия первого рода формулой:

$$\partial D = \text{bondaryRegion}: \quad T = 3.14 \tag{2.3}$$

Ниже приведен соответствующий файл **0/T**, в котором заданы граничные условия, определённые формулой (2.3).

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2306                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      T;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

              //[kg m s K kgmol A cd]
dimensions      [0 0 0 1 0 0 0]; // размерность скалярного поля 'T'
```

```
// Начальные условия
internalField   uniform 0.0; // всюду ноль

// Граничные условия
boundaryField
{
    boundaryRegion // имя участка границы
    {
        type    fixedValue; // тип граничного условия
        value   uniform 3.14; // равномерное значение поля 'T' на границе
    }
}
// ************************************************************************* //
```

Доступ к данному файлу можно получить по *ссылке*.

2.3.1.2 Неравномерные значения (codedFixedValue)

Для определения неравномерных граничных условий первого рода в OpenFOAM существует тип **codeFixedValue**. Для примера, скалярному полю T задаются граничные условия первого рода формулой:

$$\partial D = \text{bondaryRegion}: \quad T(x, y, z, t) = \sin x + \cos y + e^z + 10t \tag{2.4}$$

Ниже приведен соответствующий файл $\mathbf{0}/\mathbf{T}$, в котором заданы граничные условия, определённые формулой (2.4).

```
1  /*--------------*- C++ -*----------------*\
2  | =========                 |                                                 |
3  | \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
4  |  \\    /   O peration     | Version:  v2306                                 |
5  |   \\  /    A nd           | Website:  www.openfoam.com                      |
6  |    \\/     M anipulation  |                                                 |
7  \*---------------------------------*/
8  FoamFile
9  {
10     version     2.0;
11     format      ascii;
12     class       volScalarField;
13     object      T;
14 }
15 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
16
17 dimensions      [0 0 0 1 0 0 0]; // размерность поля 'T'
18
19 // Начальные условия
20 internalField   uniform 0.0;
21
22 // Граничные условия
23 boundaryField
24 {
25     boundaryRegion // имя участка границы
26     {
27         type  codedFixedValue; // тип неравномерных граничных условий первого рода
28         value   uniform 0; // значения для инициализации (произвольное)
```

```
29          name    codedBC; // 'name' имя созданного граничного условия
30
31          code // участок с кодом C++
32          #{
33              // 'boundaryRegion' участок сетки, относящийся к границ
34              const fvPatch & boundaryRegion = this->patch();
35
36              // центры граничных граней граней
37              const vectorField& facesCenters = boundaryRegion.Cf();
38
39              // значение времени
40              const scalar t = this->db().time().value();
41
42              // значение поля 'T' в центрах граней, участка границы boundaryRegion
43              scalarField& Tb = *this;
44
45              forAll(facesCenters, i) // цикл по всем граням границы
46              {
47                  scalar x = facesCenters[i].x();
48                  scalar y = facesCenters[i].y();
49                  scalar z = facesCenters[i].z();
50
51                  // формула для граничных значений поля 'T'
52                  // на каждом временном слое значения обновляются
53                  Tb[i] = Foam::sin(x) + Foam::cos(y) + Foam::exp(z) + 10.0*t;
54              }
55          #};
56      }
57  }
58  // ************************************************************************* //
```

Доступ к данному файлу можно получить по *ссылке*.

2.3.2 Условия второго рода (условие Неймана)

Граничные условия второго рода означают, что для искомой величины известно значение её производной по *внешней нормали* на границе. Для скалярной величины T эти условия в общем виде записываются как:

$$\partial D: \quad \left.\frac{\partial T(x,y,z,t)}{\partial \boldsymbol{n}}\right|_{t>0} = \psi(x,y,z,t) \tag{2.5}$$

В формуле выше значения производной $\frac{\partial T}{\partial \boldsymbol{n}}$ вычисляются на границе области (множество ∂D), в моменты времени после начала расчетов ($t > 0$) и функция $\psi(x,y,z,t)$ известна. Вектор $\boldsymbol{n}$ – единичный и направлен «наружу» из области D.
Тип **fixedGradient** используется, чтобы устанавливать на выбранном участке границы постоянное значение производной по внешней нормали выбранному поля. Например, чтобы задать условие, определяемое формулой:

$$\partial D = \text{bondaryRegion}: \quad \frac{\partial T}{\partial \boldsymbol{n}} = 10 \tag{2.6}$$

Нужно таким образом заполнить словарь $\mathbf{0/T}$:

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2306                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField; // Скалярное поле 'T'
    object      T;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

// Выбор размерности для поля 'T'
//               кг м с K моль A кд
dimensions      [0  0 0 1   0  0 0]; // Размерность поля 'T' это K-Кельвины

// начальные условия
internalField   uniform 0.0;

// граничные условия
boundaryField
{
    boundaryRegion // имя участка границы
    {
        type    fixedGradient; // равномерные граничные условия второго рода
        gradient    uniform 10.0;
        // здесь 'gradient' означает значение производной по внешней нормали

        // Однородные (нулевые) условия можно задать всего одной строчкой:
        //type  zeroGradient;
    }
}
// ************************************************************************* //
```

Доступ к данному файлу можно получить по *ссылке*.

Для часто встречающихся случаев, когда нужно задать *нулевое значение* производной по нормали, можно вместо **fixedGradient** со значением `value = 0`, использовать *сокращенный тип*: **zeroGradient**. Для данного типа не нужно определять дополнительные значения. То есть он задает нулевую производную по нормали «в одну строчку».

2.3.3 Условия третьего рода (условие Робина)

Условия третьего рода означают, что на границе известно чему равна сумма искомой величины и её производной по внешней нормали к границе.

$$\partial D: \quad T(x,y,z,t) + \left.\frac{\partial T(x,y,z,t)}{\partial \boldsymbol{n}}\right|_{t>0} = f(x,y,z,t) \tag{2.7}$$

В формуле выше значения величин T и $\frac{\partial T}{\partial \boldsymbol{n}}$ вычисляются на границе области (множество ∂D), в моменты времени после начала расчетов ($t > 0$) и функция $f(x, y, z, t)$ известна. Вектор $\boldsymbol{n}$ – единичный и направлен «наружу» из области D.
В OpenFOAM тип граничных условий **codedMixed** для некоторой скалярной величины T в общем виде определяется формулой:

$$T_f = \omega T_{ref} + (1 - \omega)(T_c + \nabla_n \cdot T_{ref}\delta) \tag{2.8}$$

Для лучшего понимания условий **codedMixed** можно посмотреть *документацию* или *видео урок*.

- T_f – значение в центре очередной грани.
- $0 \leq \omega \leq 1$ – весовой коэффициент (в коде `valueFraction`).
- $\nabla \cdot T_{ref}$ – величина *пропорциональная* внешнему потоку (в коде `refGrad()`).
- T_c – значение в центре граничной ячейки (в коде `refValue()`).
- δ – расстояние от центра ячейки до центра грани.

Так как в названии присутствует слово «coded» это означает, что формула (2.8) будет применена для каждой грани выбранного участка границы, причем значение коэффициентов в формуле можно изменять в зависимости от положения грани. Правильный выбор значений в формуле (2.8) позволяет задать граничные условия третьего рода, определяемые формулой (2.7).
Например, для условий третьего рода такого вида:

$$\partial D = \text{bondaryRegion} : \quad T + \frac{\partial T}{\partial \boldsymbol{n}} = x + y + z + \sin(t) \tag{2.9}$$

Нужно будет выбрать такие значения весовых коэффициентов:

- $\omega = \frac{1}{1+\delta}$ (в коде `valueFraction`)
- $\nabla \cdot T_{ref} = 0$ (в коде `refGrad()`).
- $T_c = x + y + z + \sin(t)$ – значение в центре граничной ячейки (в коде `refValue()`)
- δ – определяется автоматически.

Объяснение такого выбора значениям весовых коэффициентов сделано в *данной статье*.
Ниже приведён словарь **0/T**, в котором задаётся условие третьего рода, согласно формуле (2.9) с помощью типа **codedMixed**.

```
/*--------------------------------*- C++ -*----------------------------------*\
 | =========                 |                                                 |*
 | \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
 |  \\    /   O peration     | Version:  v2306                                 |
 |   \\  /    A nd           | Website:  www.openfoam.com                      |
 |    \\/     M anipulation  |                                                 |
 \*---------------------------------------------------------------------------*/
FoamFile
{
```

```
    version     2.0;
    format      ascii;
    class       volScalarField; //Скалярное поле температур
    object      T;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 1 0 0 0]; //Размерность К-Кельвины

// Начальные условия
internalField   uniform 0; //Начальное распределение температуры

// Граничные условия
boundaryField
{
    boundaryRegion // Имя участка границы, для которого задается условие
    {
        type            codedMixed;
        refValue        uniform 0; // Произвольное значение для инициализации
        refGradient     uniform 0; // Произвольное значение для инициализации
        valueFraction   uniform 0; // Произвольное значение для инициализации
        name            codedRobinBC;

        code
        #{
            const fvPatch& boundaryPatch   = this->patch(); // Доступ к границе сетки
            const vectorField& CF = boundaryPatch.Cf();//Массив центров граней границы

            //!!! delta ОБРАТНЫЕ расстояния от центров яччек до граней !!!
            const scalarField& delta      = spherePatch.deltaCoeffs();

            const scalar h = 1.0;
            const scalar k = 1.0;

            forAll(CF, i) // Цикл по всем граням учатска границы 'boundaryRegion'
            {
                scalar x = CF[i].x();
                scalar y = CF[i].y();
                scalar z = CF[i].z();

                // "Изменяемая" часть
                this->valueFraction()[i] = 1.0 / (1.0 + (k*delta[i])/h);
                this->refGrad()[i] = 0.0;
                this->refValue()[i] = x + y + z + Foam::sin(t); // Условие 3-го рода
            }
        #};
    }
}
// ************************************************************************* //
```

Доступ к данному файлу можно получить по *ссылке*.

2.4 Добавление источника в уравнение

В OpenFOAM есть специальный файл **fvOptions**, который располагается в каталоге **constant** в директории для любой задачи (см. главу 1). С помощью этого файла можно добавлять источниковые слагаемые в уравнения. Для разных случаев есть множество специальных *типов источников*. В раздел документации о задании источников можно перейти *по ссылке*.

2.4.1 Линеаризованное представление источника

Линеаризованное представление источника S определяется формулой:

$$S = S_u + S_p T \tag{2.10}$$

В формуле выше величины S_u и S_p являются *константами*. Значение S_u вносит вклад в правую часть СЛАУ, а S_T в диагональные значения матрицы. Для данного способа представления источника есть *шаблонный тип* <**type**>SemiImplicitSource, где <type> может быть либо scalar – для скалярного, либо vector – для векторного поля. Для данного типа источника есть также *хорошая статья* другого автора.
Ниже приведен пример файла, в котором к основным уравнениям добавляются источники. Один источник для скалярного поля (T), а второй для векторного ($\boldsymbol{U}$).
Источники определяются формулами:

$$\begin{aligned} S_1 &= 1 - 2T \\ S_2 &= [\mathbf{1}, \mathbf{2}, \mathbf{3}] - 10\boldsymbol{U} \end{aligned} \tag{2.11}$$

Файл **constant/fvOptions**, в котором задается источник согласно формуле (2.11):

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2306                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      fvOptions;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

scalarLinerizedSourceTerm
{
    type            scalarSemiImplicitSource;
    active          true;

    scalarSemiImplicitSourceCoeffs
```

```
    {
        // Здесь источник задаётся для всей расчетной области.
        // Но, можно задавать источник в заданных участках области.
        selectionMode   all; // all, cellSet, cellZone, points
        volumeMode      absolute; // specific, absolute;
        injectionRateSuSp
        {
            T     (1 -2);
            //    (Su Sp), Su - 'свободный коэффициент',
            //             Sp - 'коэффициент' при неизвестной (перед 'T')
            // То есть здесь задан источник: S = 1 - 2*T
        }
    }
}

vectorLinerizedSourceTerm
{
    type            vectorSemiImplicitSource;
    active          true;

    vectorSemiImplicitSourceCoeffs
    {
        selectionMode   all; // all, cellSet, cellZone, points
        volumeMode      absolute; // specific, absolute;
        injectionRateSuSp
        {
            U     ((1 2 3) -10);
            //    (Su Sp), Su - 'свободный вектор',
            //             Sp - 'коэффициент' при неизвестной (перед 'U')
            // То есть здесь задан источник: S = (1,2,3) - 10*(Ux, Uy, Uz)
        }
    }
}
```

Доступ к данному файлу можно получить по *ссылке*.

2.5 Об обработка результатов расчетов

Для обработки результатов расчетов в OpenFOAM есть различные инструменты. Это и специальные утилиты, и «функциональные объекты», и сторонняя программа ParaView. В официальной документации есть целый *раздел*, посвященный обработке данных. **У автора этого руководства есть также *краткая справка* для программы ParaView.** Функциональные объекты это шаблонные функции, предназначенные для выполнения типичных задач. С их помощью можно получать дополнительную информацию о решении. Возможностей у функциональных объектов очень много, например, их можно запускать во время расчетов или отдельно после них. Узнать больше об этой технологии можно *в документации*.
Простейшая обработка данных показана в главе 3 в разделе 3.5.

Глава 3

Решение демонстрационной задачи о теплопроводности

Перед изучением данного раздела рекомендуется прочитать главу 1.
В этой части показан полный процесс решения физической задачи с помощью средств, которые предоставляет пакет OpenFOAM. Разобрав этот пример, можно понять, как OpenFOAM применяется на практике. Значительную часть текста занимают файлы, используемые утилитами при выполнении разных этапов рассматриваемой задачи. Это может выглядеть громоздко, но так кажется только по-началу. Новичкам будут полезны пояснения, сделанные в этих файлах. Это должно помочь им усвоить предназначение файлов и принцип их заполнения.
Для многих задач основная часть файлов с настройками сохраняется, меняются лишь «некоторые детали». Поэтому данный пример, можно использовать как шаблон при решении других задач средствами пакета OpenFOAM.
В качестве демонстрационной задачи рассмотрен процесс распределения температуры в твердом теле, который описывается уравнением теплопроводности.
Полное решение этого примера (все файлы и команды для запуска утилит), можно получить по *ссылке*.

3.1 Постановка задачи

Расчетная область – множество D:

$$D = \{\ 0 \le x \le 1,\ 0 \le y \le 2,\ 0 \le z \le 3\ \} \tag{3.1}$$

Внутри области D рассматриваемый физический процесс описывается уравнением:

$$\frac{\partial T}{\partial t} = \Delta T \tag{3.2}$$

Начальные условия для поля T в момент времени $t = 0$ определяются формулой:

$$D: \quad T(x, y, z, 0) = x^2 \tag{3.3}$$

Граничные условия задаются на гранях области (множество ∂D) следующим образом:

$$\partial D,\ t > 0 : \begin{cases} T\Big|_{x=0} = 2t, \quad T\Big|_{x=1} = 1 + 2t & \text{условия первого рода} \\ \frac{\partial T}{\partial \boldsymbol{n}}\Big|_{y=0} = 0, \quad \frac{\partial T}{\partial \boldsymbol{n}}\Big|_{y=2} = 0 & \text{условия второго рода} \\ \frac{\partial T}{\partial \boldsymbol{n}}\Big|_{z=0} = 0, \quad \frac{\partial T}{\partial \boldsymbol{n}}\Big|_{z=3} = 0 & \text{условия второго рода} \end{cases} \tag{3.4}$$

В граничных условиях второго рода производная по нормали вычисляется как:
$\frac{\partial T}{\partial \boldsymbol{n}} := \nabla T \cdot \boldsymbol{n},\ |\boldsymbol{n}| = 1$, вектор нормали направлен «наружу».
Для этой задачи существует аналитическое решение:

$$T_{an} = x^2 + 2t \tag{3.5}$$

С помощью «проверки в уме» можно убедиться в том, что формула (3.5) действительно удовлетворяет уравнению (3.2) и условиям (3.3), (3.4) в выбранной области.
В данной задаче нужно:

- Получить численное решение задачи, определённой условиями: (3.1) - (3.4).
- Оценить погрешность решения.

3.2 Выбор утилит для решения задачи

Для создания сетки согласно определению в формуле (3.1) подойдет утилита `blockMesh`, которая создает блочные-структурированные сетки. Краткий обзор на неё сделан в разделе 2.1.1.
Для решения уравнения теплопроводности в форме, представленной в выражении (3.2) можно воспользоваться программой `laplacianFoam`. Согласно *документации* данная утилита решает следующее уравнение:

$$\frac{\partial T}{\partial t} = D_T \Delta T + F_v \tag{3.6}$$

Здесь D_T это *постоянный* коэффициент температуропроводности ($\frac{k}{\rho c}$), а F_v это источниковый член, который при необходимости можно задать в файле **constant/fvOptions**. В данной задаче в уравнении (3.2) *нет источника*.
В коде утилиты `laplacianFoam` для каждого временного слоя составляется матрица дискретного аналога уравнения теплопроводности. На конкретный вид этой матрицы влияют выбранные схемы аппроксимации слагаемых (определяются файлом **system/fvSchemes**), а также шаг по времени и параметры сетки (задаются в файле **system/blockMeshDict**). Метод решения полученной системы уравнений указывается в словаре **system/fvSolution**.

3.3 Подготовка файлов с настройками

В этом разделе приведены все файлы, используемые при решении задачи поставленной в разделе 3.1. Внутри каждого файла сделаны разъясняющие комментарии. Доступ к директории с этим файлами можно получить по *ссылке*.

3.3.1 Файл system/blockMeshDict (создание сетки)

Этот файл определяет настройки для утилиты `blockMesh`, которая создает расчетную сетку. Создаваемая сетка будет неравномерной, но ортогональной.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2306                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

scale   1.0; // Масштабный множитель, на который умножаются значения вершин

// Длины сторон параллелипипеда 'D'
Lx 1.0;
Ly 2.0;
Lz 3.0;
// Набор вершин параллелипипеда 'D'
// $Lx и т.д. это макро-подстановка
vertices
(
    (0    0   0)
    ($Lx  0   0)
    ($Lx $Ly  0)
    (0   $Ly  0)
    (0    0  $Lz)
    ($Lx  0  $Lz)
    ($Lx $Ly $Lz)
    (0   $Ly $Lz)
);

// (10 10 10) - Число разбиений по осям Ox, Oy, Oz соответственно
// Сетка получается неравномерная, но ортогональная
blocks
(
    hex (0 1 2 3 4 5 6 7) (10 10 10) simpleGrading (1 1 1)
);
```

```
// Выбор способа соединения вершин
// Не заполнен, т.к. по умолчанию установлены отерзки - то что нужно
edges
();

// "Разметка" граничной области (именование граней параллелипипеда)
boundary
(
    minX // x = 0
    {
        type patch; // стандартный тип граничной области
        faces ( (0 3 7 4) ); // список входящих граней
    }

    maxX // x = Lx
    {
        type patch;
        faces ( (2 1 5 6) );
    }

    minY // y = 0
    {
        type patch;
        faces ( (1 0 4 5) );
    }

    maxY // y = Ly
    {
        type patch;
        faces ( (3 2 6 7) );
    }

    minZ // z = 0
    {
        type patch;
        faces ( (0 1 2 3) );
    }

    maxZ // z = Lz
    {
        type patch;
        faces ( (4 7 6 5) );
    }
);
// ************************************************************************* //
```

3.3.2 Файл system/fvSchemes (схемы аппроксимации слагаемых)

В этом файле определяются схемы дискретизации слагаемых в уравнении теплопроводности. Выбор именно этих схем объясняется характером рассматриваемого процесса и строением расчетной сетки.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2306                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      fvSchemes;
}
// ************************************************************************* //
// Схема, указанная в 'default' применяется ко всем слагаемым такого вида
ddtSchemes // Аппроксимация производной по времени
{
    default         none;
        ddt(T)                  Euler;
}

gradSchemes // Аппроксимация градиента
{
    default         Gauss linear;
}

divSchemes // Аппроксимация дивергенции
{
    default         Gauss linear;
}

laplacianSchemes // Аппроксимация оператора Лапласа
{
    default         none; // Схема по умолчанию не определена
    laplacian(DT,T)    Gauss linear orthogonal;
}

interpolationSchemes // Аппроксимация значений на гранях контрольных объемов
{
    default         linear;
}

snGradSchemes // Аппроксимация потока на гранях контрольных объёмов
{
    default         orthogonal;
}
// ************************************************************************* //
```

3.3.3 Файл system/fvSolution (настройки решения СЛАУ)

Этот словарь задает критерии итерационного решения системы линейных уравнений, которые получаются при решении задачи. На каждом временном слое СЛАУ считается решенной, если выполняется хотя бы один из критериев: `tolerance` или `relTol`.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2306                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      fvSolution;
}
// ************************************************************************* //

solvers
{
    T // Параметры решения СЛАУ для поля 'T'
    {
        solver          PCG; // Метод сопряженных градиентов
        preconditioner  DIC; // Предобуславливатель

        // Критерий сходимости решения СЛАУ (абсолютная ошибка)
        tolerance       1e-12;

        // Критерий сходимости решения СЛАУ (относительная ошибка)
        relTol          0.0;
    }
}
// ************************************************************************* //
```

3.3.4 Файл system/controlDict (контроль процесса решения)

В данном словаре выбраны параметры проведения расчетов. В эти параметры входят: начальное и конечное время, шаг разбиения по времени. Интервал сохранения данных, точность (количество знаков после запятой) записываемых значений. Согласно настройкам этого файла расчеты будут выполнены от момента времени $t = 0$ до времени $t = 1$, с шагом по времени $\delta t = 0.01$, данные с временных слоев сохраняются с интервалом в 100. Это означает, что будут сохранены значения поля температуры только с последнего временного слоя, то есть к моменту времени $t = 1$.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2306                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      controlDict;
}
// ************************************************************************* //

application     laplacianFoam; // Название утилиты OpenFOAM для решения задачи
// При 'ручном' вызове утилит опция игнорируется
// Она используется только в скриптах автоматизации
// (Allrun в примерах в директории $FOAM_TUTORIALS/)

startFrom       startTime; // Параметр, определяющий начало расчетов

startTime       0; // Начальное время расчетов
// Для заданного начального времени в директории должнен быть
// одноименный каталог с начальными и граничными условиями

deltaT          0.01; // Шаг по времени

stopAt          endTime; // Параметр, определяющий конец расчетов

endTime         1.0; // Конечное время расчёта

writeControl    timeStep; // Критерий для контроля сохранения данных
writeInterval   100; // Интервал сохранения временных слоев (здесь 100-й слой)


// Альтернативный вариант сохранения временных слоев (другой критерий)
//writeControl    runTime; // Критерий для контроля сохранения данных
//writeInterval   0.1; // Сохраняются временные слои с шагом в '0.1'
                       // то есть 0.1, 0.2, 0.3, ..., 1.0


purgeWrite      0; // Лимит на количество сохраняемых временных слоев
```

```
// Если "purgeWrite" не '0', то при превышении лимита будут удаляться более
// ранние временные слои, чтобы общее число сохраненных слоев не превосходило
// значения, записанного в "purgeWrite"

writeFormat     ascii; // Текстовый формат файлов с данными, есть ещё 'binary'

writePrecision  15; // Число знаков после запятой в десятичной записи значений

writeCompression off; // Если указать 'on', то значения будут в '.zip' формате

timeFormat      general;

timePrecision   6;

runTimeModifiable true;

// Дополнительный раздел для <<функциональных объектов>>
// Они используются для получения дополнительной информации о ходе решения
// Например, для вычисления числа Куранта, а также для пост-обработки данных
// https://www.openfoam.com/documentation/guides/latest/doc/guide-function-objects.html
functions
{
    // Function objects
}
// ************************************************************************* //
```

3.3.5 Файл constant/transportProperties

Это пример стандартного файла, в котором задаются физические постоянные в решаемых уравнениях. Обычно в любой задаче есть такой файл, он располагается в директории **constant**. Конкретно в этом случае в словаре **constant/transportProperties** задается единственная константа, обозначенная как DT и это коэффициент температуропроводности $(\frac{k}{\rho c})$.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2306                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

DT              1.0; // Коэффициент температуропроводности

// ************************************************************************* //
```

3.3.6 Файл 0/T (начальные и граничные условия)

В словаре **0/T** определяются начальные и граничные условия для поля T – температуры в заданной области. Он «связан» с файлом **system/blockMeshDict**. Потому что граница области разделена на несколько участков. Разделение границы (имена участков) определены как раз в файле **system/blockMeshDict**, приведенном в разделе 3.3.1.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2306                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      T;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

// Выбор размерности для поля 'T'
//               кг м с K моль A кд
dimensions      [0  0 0 1  0   0  0];


// Начальные условия, когда t=0: T = x*x
// С помощью codeStream значения в начальный момент времени
// Будут вычислены во время запуска расчетов
internalField   #codeStream
{
    codeInclude // Подключение основного заголовочного файла
    #{ #include "fvCFD.H" #};

    codeOptions // Добавление новых путей для поиска подключаемых файлов при компиляции
    #{
        -I$(LIB_SRC)/finiteVolume/lnInclude \
        -I$(LIB_SRC)/meshTools/lnInclude
    #};

    codeLibs // Подключение библиотек
    #{
        -lmeshTools \
        -lfiniteVolume
    #};

    // Секция с кодом C++
    code
    #{
        // Получение данных о сетке (объект 'mesh')
        const IOdictionary& d = static_cast<const IOdictionary&>(dict);
        const fvMesh& mesh = refCast<const fvMesh>(d.db());
```

```
        // Создание списка скалярных значений ('T')
        // Эти значения будут записаны в центры ячеек, как начальное условие
        scalarField T(mesh.nCells()); // n.Cells() возвращает кол-во ячеек

        // 'CC' это список векторных значений -- центры ячеек
        const vectorField& CC = mesh.C();

        forAll(CC, i) // Цикл по всем центрам ячеек
        {
            scalar x = CC[i].x(); // 'X' - координата очередной ячейки
            //scalar y = CC[i].y(); // 'Y' - координата очередной ячейки
            //scalar z = CC[i].z(); // 'Z' - координата очередной ячейки

            // Формула, определяющая начальные условия на поле 'T'
            T[i] = x*x;
        }

        T.writeEntry("", os); // Запись значений (в оперативную память)
    #};
};



// Граничные условия
boundaryField
{
    minX // t>0: T = 2t
    {
        type    codedFixedValue; // тип неравномерных граничных условий первого рода
        value   uniform 0; // значения для инициализации (произвольное)
        name    codedBC_1; // 'name' имя созданного граничного условия
        code // участок с кодом C++
        #{
            // 'boundaryRegion' участок сетки, относящийся к границ
            const fvPatch & boundaryRegion = this->patch();

            // центры граничных граней граней
            const vectorField& facesCenters = boundaryRegion.Cf();

            // значение времени
            const scalar t = this->db().time().value();

            // значение поля 'T' в центрах граней,
            // расположенных на участке 'minX' границы области
            scalarField& Tb = *this;

            forAll(facesCenters, i) // цикл по всем граням границы
            {
                // формула для граничных значений поля 'T'
                // на каждом временном слое значения обновляются
                Tb[i] = 2.0 * t;
            }
        #};
    }
```

```
    maxX // t>0: T = 1 + 2t
    {
        type    codedFixedValue;
        value   uniform 0;
        name    codedBC_2;
        code
        #{
            const fvPatch & boundaryRegion = this->patch();
            const vectorField& facesCenters = boundaryRegion.Cf();
            const scalar t = this->db().time().value();
            scalarField& Tb = *this;
            forAll(facesCenters, i) // цикл по всем граням границы
            {
                Tb[i] = 1.0 + 2.0*t;
            }
        #};
    }

    "minY|maxY|minZ|maxZ" // Имена частей границ можно объединять
    {
        //type  fixedGradient;
        //gradient      uniform 0.0;
        type    zeroGradient;
    }
}

// ************************************************************************* //
```

3.3.7 Файл system/setAnalyticSolution (построение поля аналитического решения)

Этот файл не является обязательным при решении различных задач.
Данный словарь задаёт настройки для утилиты `setExprFields`. Она вычисляет значения в центрах контрольных объемов (ячейках). В результате создается новое поле $\boldsymbol{T_{an}}$, в котором будут записаны значения аналитического решения. Для лучшего понимания работы этой утилиты можно изучить этот *раздел* документации.
Утилита setExprFields может не работать на Windows.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2306                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
        version 2.0;
        format ascii;
        class dictionary;
        object setAnalyticSolutionDict;
}
// ************************************************************************* //

expressions
(
    T_an // Вложенный словарь
    {
        field T_an; // Имя для поля аналитического решения
        dimensions [0 0 0 1 0 0 0]; // Размерность поля
        constants {  }

        variables
        (
            "x = pos().x()" // доступ к 'x' комопненте центра каждой ячейки
            "y = pos().y()"
            "z = pos().z()"
            "t = time()"
        );

        expression // Внутри задается аналитическое решение
        #{
            x*x + 2.0*t // Формула применяется к каждой ячейке
        #};

        create yes; // Для создаваемых полей
    }
);
// ************************************************************************* //
```

3.3.8 Файл system/setAbsError (построение поля абсолютной погрешности)

Этот файл не является обязательным при решении различных задач.
Данный словарь задаёт настройки для утилиты `setExprFields`. Она вычисляет значения в центрах контрольных объемов (ячейках). В результате создается новое поле ***absErr***, в котором будут записаны значения абсолютной погрешности решения. Для лучшего понимания работы этой утилиты можно изучить данный *раздел* документации.
Утилита setExprFields может не работать на Windows.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2306                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      setAbsErrorFieldDict;
}
// ************************************************************************* //

readFields      ( T T_an ); // Получение доступа к значениям используемых полей

expressions
(
    absErr // Вложенный словарь
    {
        field   absErr; // Имя для поля абсолютных ошибок
        dimensions [0 0 0 1 0 0 0]; // Размерность поля
        constants   {  };
        variables   (  );

        expression  #{ mag(T_an - T) #}; //Формула, вычисляемая в каждой ячейке

        create  yes; // Для создаваемых полей
    }
);
// ************************************************************************* //
```

3.4 Запуск утилит для проведения расчетов

При запуске утилит OpenFOAM нужно учитывать, что их поведение зависит того из какой директории они вызваны. В основном их нужно вызывать из директории с задачей. Также должно быть **активно окружение OpenFOAM** более подробно об этом сказано в разделе 1.2.3.
Для выполнения всех этапов решения данной задачи нужны последовательно выполнить следующие команды:

```
blockMesh
laplacianFoam
setExprFields -noZero -dict system/setAnalyticSolutionDict
setExprFields -noZero -dict system/setAbsErrorDict
postProcess -func 'cellMax(absErr)' | grep "max(region0) of absErr"
paraFoam &
```

Последняя команда '`paraFoam`' вызывает программу ParaView для визуализации данных, а символ '`&`' означает запуск в фоновом режиме. То есть после выполнения этой команды можно продолжить работу с терминалом.

3.5 Обработка результатов

Визуализация данных

Изображение ниже получено с помощью программы ParaView, она не является внутренней частью пакета OpenFOAM, но поставляется вместе с ним.

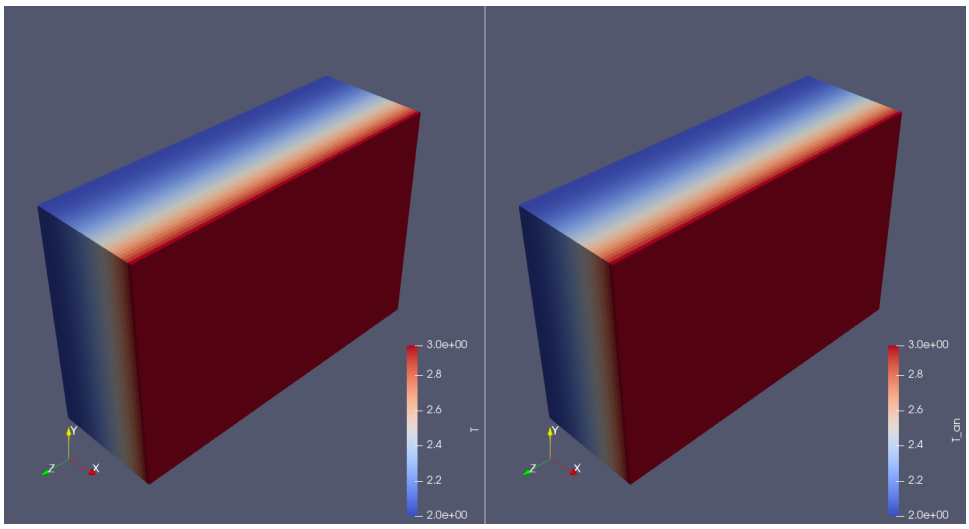


Рис. 3.1: Значения на момент времени $t = 1$, численное решение (слева), аналитическое решения (справа)

Определение погрешности

Для сравнения численного и аналитического решения вычислялась величина:

$$\Delta = \max_{i \in D} |T^i - T^i_{an}|, \quad \text{значения берутся в центрах ячеек} \tag{3.7}$$

Расчет был проведен на одной сетке. Это был параллелепипед D, определенный формулой (3.1), с числом разбиений вдоль осей $(Ox,\ Oy,\ Oz) = (10\ 10\ 10)$. Полное описание параметров сетки можно посмотреть в разделе 3.3.1.
Для данной сетки $\Delta = 0.00249$.

Список источников

[1] С. Патанкар. Численные методы решения задач теплообмена и динамики жидкости. — 1984. — Режим доступа: https://drive.google.com/file/d/1Xm3BORCa5WWlfP4SVk2lwrPwPsTb8Dut/view?usp=sharing.

[2] H. K. Versteeg W. Malalasekera. An Introduction to Computational Fluid Dynamics. — 2 изд. — 2007. — Режим доступа: https://drive.google.com/file/d/1PYr8M5PdBahck2dCnHlXSm1LbLR3aAnd/view?usp=sharing.

[3] Jasak Hrvoje. Error Analysis and Estimation for the Finite Volume Method With Applications to Fluid Flows. — 1996. — Режим доступа: https://www.researchgate.net/publication/230605842_Error_Analysis_and_Estimation_for_the_Finite_Volume_Method_With_Applications_to_Fluid_Flows.

[4] Greenshields Christopher, Weller Henry. Notes on Computational Fluid Dynamics: General Principles. — 2022. — Режим доступа: https://doc.cfd.direct/notes/cfd-general-principles/.

[5] OpenFOAM User Guide. — Режим доступа: https://www.openfoam.com/documentation/user-guide.

[6] OpenFOAM Tutorial Guide. — Режим доступа: https://www.openfoam.com/documentation/tutorial-guide.

[7] API Documentation for OpenFOAM. — Режим доступа: https://www.openfoam.com/documentation/guides/latest/doc/index.html.

[8] OpenFOAM Programmer's Guide. — Режим доступа: https://altushost-swe.dl.sourceforge.net/project/openfoam/v2312/ProgrammersGuide.pdf.

[9] WolfDynamics. — OpenFOAM Introductory Training. — Режим доступа: https://figshare.com/articles/media/OpenFOAM_Introductory_Training/16783657.

[10] OpenFOAM Training by CFD-Support. — Режим доступа: https://drive.google.com/file/d/1pfFTO825EwJcdcgiJGvcx2-m0qxjCVbO/view?usp=drive_link.

[11] Нуриев А. Марсович А. Зайцева О. — Введение в компьютерное моделирование в программном комплексе OpenFOAM. — Режим доступа: https://www.researchgate.net/publication/355187240_Vvedenie_v_komputernoe_modelirovanie_v_programmnom_komplekse_OpenFOAM_Ucebnoe_posobie_Introduction_to_computer_modeling_in_the_OpenFOAM_package_Tutorial.

[12] Нуриев А. Марсович А. Зайцева О. — Основы моделирования газодинамических процессов и конвективного теплообмена в программном комплексе OpenFOAM. — Режим доступа: https://dspace.www1.vlsu.ru/bitstream/123456789/11132/1/02648.pdf.