Введение в компьютерное моделирование в программном комплексе OpenFOAM. Учебное пособие. Introduction to computer modeling in the OpenFOAM package. Tutorial.

Book · October 2021

CITATIONS
0

READS
6,866

3 authors:

Artem Nuriev
Kazan Federal University
64 PUBLICATIONS **246** CITATIONS
SEE PROFILE

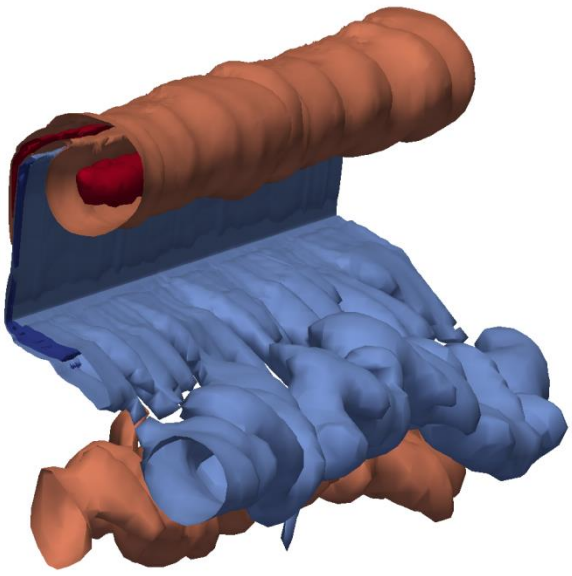
Airat Marsovich Kamalutdinov
Kazan Federal University
34 PUBLICATIONS **189** CITATIONS
SEE PROFILE

Olga Zaitseva
Kazan State Technological University
31 PUBLICATIONS **62** CITATIONS
SEE PROFILE

Введение в компьютерное моделирование в программном комплексе OpenFOAM

учебное пособие

Нуриев А.Н., Зайцева О.Н., Камалутдинов А.М.



Казань 2021

УДК 532.5

Публикуется по рекомендации Учебно-методической комиссии
Института математики и механики им. Н. И. Лобачевского
ФГАОУ ВО «Казанский (Приволжский) федеральный университет»
Протокол № ?? от ?? апреля 2021 года

Принято на заседании кафедры аэрогидромеханики
Протокол № ?? от ?? апреля 2021 года

Рецензенты
доктор физико-математических наук,
профессор кафедры аэрогидромеханики КФУ
Егоров Андрей Геннадьевич,
кандидат физико-математических наук,
???

Введение в компьютерное моделирование в программном комплексе OpenFOAM. Учебное пособие / А.Н. Нуриев, О.Н. Зайцева, А.М. Камалутдинов. – Казань: Казан. ун-т, 2021. – ?? С.

Учебное пособие основано на курсе лекций и практических занятий для студентов старших курсов и магистрантов, проводимых в Институте математики и механики КФУ и на кафедре информатики и прикладной математики КНИТУ. В нём изложены основы численного моделирования в программном комплексе OpenFOAM. Курс разработан в рамках проблемно-ориентированного подхода: студенты знакомятся с разными возможностями моделирования в OpenFOAM в ходе решения нескольких классических задач гидромеханики. Пособие может быть полезно также аспирантам и инженерам в качестве руководства для начинающего пользователя OpenFOAM.

Содержание

Введение

В настоящее время пакеты вычислительной гидродинамики (CFD-пакеты) и системы инженерного анализа (CAE-системы) представляют собой мощный и хорошо развитый аппарат для решения широкого класса физических задач. Они активно используются ведущими мировыми инженерными компаниями при проектировании и создании разнообразной высокотехнологичной продукции. Применение CFD-пакетов и CAE-систем позволяет экономить финансовые средства и время, заменяя множество дорогостоящих этапов разработки и тестирования численными экспериментами. Большинство CFD-пакетов и CAE-систем являются проприетарными программными продуктами и требуют крупных финансовых вложений на этапе их приобретения. Но более важно то, что они содержат закрытый программный код и, как следствие, представляют собой «черный ящик» для исследователя. Это ограничивает возможности по созданию, модификации и верификации новых численных моделей, мешает при оценке точности полученных результатов и т. д.

OpenFOAM – один из немногих прикладных программных пакетов с открытым исходным кодом (open-source software) для решения задач механики сплошных сред. Широкий инструментарий для формализации задачи, высокая эффективность реализации, а также хорошая масштабируемость под архитектуру вычислительной системы позволяют легко конструировать в пакете OpenFOAM численные модели разной сложности. Открытый исходный код, в свою очередь, дает возможность в деталях контролировать ход решения, начиная от построения сетки до выбора схем аппроксимации слагаемых управляющей системы и методов численного решения. Это позволяет использовать OpenFOAM как для инженерных, так и для научных исследований.

OpenFOAM – это, прежде всего, набор программных библиотек, написанных на языке C++, предоставляющих широкий инструментарий для решения систем дифференциальных уравнений в частных производных с помощью метода конечных объемов. Вместе с библиотеками программный комплекс содержит набор программ-решателей и программ-утилит, в которых реализованы различные математические модели механики сплошных сред (и не только) а также средства для начального конфигурирования и постобработки численных расчетов. Структура программного кода OpenFOAM, выполненная в рамках объектно-ориентированного подхода, позволяет пользователям, обладающим минимально необходимыми знаниями в областях численного моделирования и программирования на C++, создавать свои решатели и утилиты для численной реализации собственных моделей любой сложности.

В данном учебном пособии рассматриваются основы численного моделирования в программном комплексе OpenFOAM. Курс разработан в рамках проблемно-

ориентированного подхода: студенты знакомятся с разными возможностями моделирования в OpenFOAM в ходе решения нескольких классических задач гидромеханики.

В первой главе рассматривается задача о течении жидкости в квадратной каверне. Решения данной модельной задачи содержат важнейшие структурные элементы вязких течений, такие как сдвиговый поток, нарастающие на стенках пограничные слои, первичный, присоединенный и вторичные вихри. Разномасштабность и сложность этих структур предъявляет растущие с ростом числа Рейнольдса требования к качеству применяемых численных схем и размерности используемых сеток. Последнее делает задачу о стационарном циркуляционном течении в каверне классическим объектом для тестирования новых методов решения уравнений Навье-Стокса и программных пакетов. Простейшая конфигурация для решения этой задачи (готовый проект) представлена разработчиками OpenFOAM. На базе этого примера студенты знакомятся с общей структурой проекта OpenFOAM, с базовыми методами построения расчетных сеток, с основными настройками решателей, а также программами и методами для пост-обработки и визуализации результатов расчетов.

Во второй главе рассматривается численное решение задачи об обтекании круглого цилиндра потоком вязкой несжимаемой жидкости. Гидродинамика около цилиндра сложнее, чем в каверне: помимо тонких пограничных слоев и присоединенных вихревых структур, образующихся около тела, за цилиндром формируется нестационарный вихревой след, который распространяется на большие расстояния от цилиндра. Это накладывает принципиально более строгие требования к структуре расчетных сеток, к точности и скорости работы вычислительных алгоритмов. Разработка модели в OpenFOAM для этой задачи проводится «с нуля». В ходе создания модели студенты знакомятся с методами построения сложных структурированных блочных сеток, с критериями оценки их качества, на новом уровне изучают численные методы и алгоритмы решения задач в рамках конечно-объемного подхода, учатся правильно подбирать методы аппроксимации и расчетные схемы для решения конкретной задачи, знакомятся с методом распараллеливания расчетов.

При подготовке данного учебного пособия авторами использовалась версия OpenFOAM 5 (ветка openfoam.org), при работе в других версиях пакета могут потребоваться небольшие уточнения синтаксиса.

1. Исследование задачи о течении жидкости в квадратной каверне с подвижной верхней крышкой

1.1. Математическая постановка задачи

В данном разделе рассмотрим моделирование в OpenFOAM течения вязкой несжимаемой жидкости в квадратной каверне (квадратной области) с подвижной верхней крышкой. Геометрия задачи показана на рис. 1.1. Квадратная каверна, заполненная несжимаемой жидкостью, имеет характерный размер стороны d. Границы каверны представляют собой твердые непроницаемые стенки. Верхняя граница движется со скоростью $U = 1$ м/с, индуцируя циркуляционное движение жидкости в области.

Течение жидкости в каверне описывается нестационарной системой уравнений Навье – Стокса

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} = -\frac{1}{\rho}\nabla p + \nu \Delta \boldsymbol{u}, \quad \nabla \cdot \boldsymbol{u} = 0,$$

где $\boldsymbol{u} = (u_x, u_y)$ – скорость движения жидкости, p – давление, ρ, ν - плотность и кинематическая вязкость жидкости, t – время.

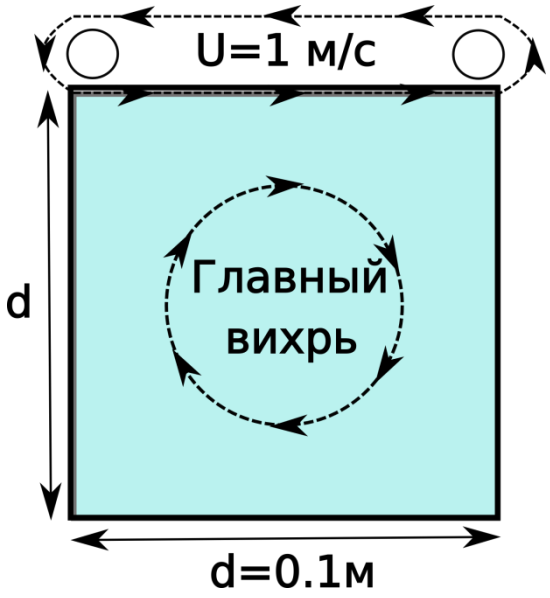


Рис. 1.1. Схема течения жидкости в квадратной каверне.

На границах каверны задаются условия прилипания:

на верхней границе: $u_x = U,\ u_y = 0,$

на неподвижных границах: $u_x = 0\ ,\ u_y = 0.$

В начальный момент времени жидкость в каверне считается неподвижной:

в жидкости при $t = 0$: $u_x = 0\ ,\ u_y = 0.$

Для дальнейшего решения проведем обезразмеривание задачи с целью выявления основных параметров, определяющих структуру течений в области. Обезразмерим

скорость на U, пространственные координаты на d, время на d/U . В этом случае уравнение движения перепишется как

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} = -\nabla \mathrm{p} + \frac{1}{\mathrm{Re}} \Delta \boldsymbol{u}, \qquad \nabla \cdot \boldsymbol{u} = 0, \qquad \mathrm{Re} = \frac{U\, d}{\nu},$$

где Re – число Ренольдса (безразмерный параметр). Таким образом, течение зависит только от этого безразмерного комплекса.

В OpenFOAM все расчеты проводятся в размерных переменных. Для дальнейшего численного решения примем, что характерная скорость движения верхней границы 1 м/с, длина стороны каверны 0.1м, как и в размерной задаче. Кинематическую вязкость ν будем изменять так, чтобы получить желаемое число Рейнольдса:

$$\nu = \frac{0.1}{Re} \text{ м}^2/\text{с}.$$

1.2. Создание проекта OpenFOAM

Задача о течении вязкой несжимаемой жидкости в квадратной каверне - это классическая задача вычислительной гидромеханики, она традиционно используется для тестирования точности вычислительных алгоритмов. Простейшая конфигурация для решения этой задачи (готовый проект OpenFOAM) представлена разработчиками пакета. Директорию примера $FOAM_TUTORIALS/incompressible/icoFoam/cavity/cavity нужно скопировать в любое удобное место, например в домашнюю папку. В ОС на базе Linux это можно выполнить командой

```
cp –r  $FOAM_TUTORIALS/incompressible/icoFoam/cavity/cavity ~/
```

Директория *cavity* содержит три основных папки: **0**, **constant** и **system** (см. рис. 1.2). В папке 0 содержатся файлы, описывающие начальные и граничные условия, папка **constant** содержит файлы, описывающие геометрию задачи и свойства жидкости, папка **system** – конфигурационные файлы, описывающие настройки решателя и утилит, используемых для решения задачи. Все перечисленные файлы будут подробно рассмотрены ниже. Для редактирования этих файлов можно использовать любой текстовый редактор, например gedit для Linux, или WordPad для Windows .

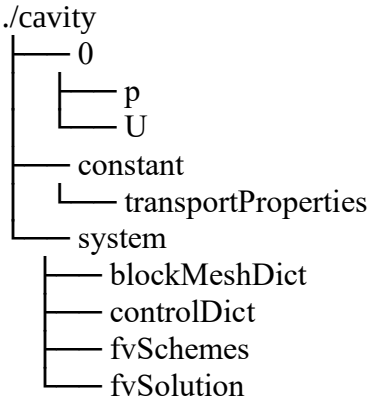


Рис. 1.2. Дерево каталогов проекта OpenFOAM для примера *cavity*

1.3 Выполнение расчета.

Поскольку проект *cavity* уже сконфигурирован, можно сразу перейти к выполнению расчетов и посмотреть результаты моделирования. Для этого в консоли в директории проекта необходимо выполнить следующую последовательность команд:

```
blockMesh
icoFoam
paraFoam
```

Первая команда вызывает утилиту **blockMesh** для генерации сетки, вторая команда вызывает решатель **icoFoam,** третья команда вызывает приложение **paraview** для визуализации результатов решения задачи. Далее в пунктах 1.4-1.9 мы подробно рассмотрим, как конфигурировать и управлять работой blockMesh и icoFoam. Визуализация и анализ гидродинамических характеристик в **paraview** описаны в пункте 1.10. Изучение этого материала может быть проведено в произвольном порядке.

1.4. Построение области и расчетной сетки

Пакет OpenFOAM основан на методе конечных объемов. Поэтому в качестве первого шага для численного решения задачи помимо задания геометрии расчетной области необходимо создать расчетную сетку. Построение геометрии и генерация расчетной сетки может быть выполнена как с помощью утилит OpenFOAM, так и с помощью сторонних приложений. В данном разделе мы рассмотрим генерацию сетки с помощью утилиты blockMesh, входящей в пакет OpenFOAM. Она позволяет строить структурированные блочные сетки.

OpenFOAM всегда работает в трехмерной декартовой системе координат. Все геометрические конфигурации производятся в трех измерениях. Для решения двухмерной задачи необходимо определить специальное пустое граничное условие **empty** на границах,

чья нормаль совпадает с направлением оси, определяющей третье измерение (в примере cavity это ось совпадает с осью z), для которого решение проводить не требуется.

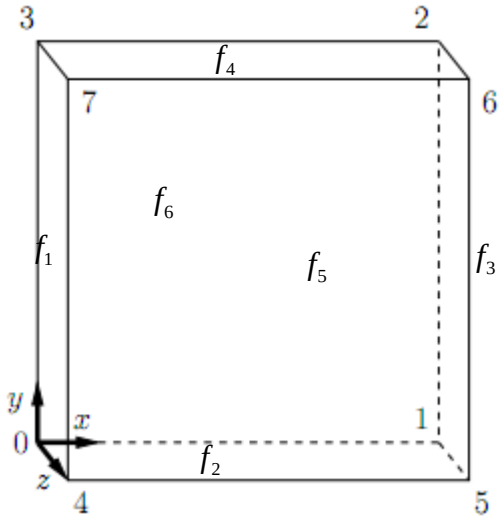


Рис. 1.3. Блочная структура расчетной сетки

Рассмотрим файл с конфигурацией расчетной сетки задачи **cavity.** Для этого откроем файл **blockMeshDict,** находящийся в папке **system** (в ранних версиях OpenFAOM находился в папке **constant/polyMesh),** с помощью текстового редактора. В заголовке содержится информация о файле (см. рис. 1.4.), а именно:

- **version** определяет версию формата входного/выходного файла
- **format** определяет формат файла ascii (текстовой) или binary (бинарный).
- **class** определяет класс файла. Класс файла указывает утилитам, каким образом следует воспринимать данный файл. Например, класс **dictionary** – является классом конфигурационных файлов.
- **object** определяет тип файла. В данном примере это файл типа **blockMeshDict,** то есть конфигурационный файл для утилиты **blockMesh.**

Подобную информацию содержит каждый файл программы.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.0                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

Рис. 1.4. Заголовок конфигурационного файла OpenFOAM

Постановка и решение задачи в пакете OpenFOAM осуществляется в размерных переменных. В таблице 1.1 приведены основные единицы измерения, используемые в пакете OpenFOAM.

Таблица 1.1. Основные единицы измерения в OpenFOAM

Единица измерения	Название	Определение
кг	килограмм	единица измерения массы в Международной системе единиц (СИ).
м	метр	единица измерения длины и расстояния в Международной системе единиц (СИ)
с	секунда	единица измерения времени в Международной системе единиц (СИ) и системы СГС
К	Кельвин	единица температуры в Международной системе единиц (СИ)
кмоль	килограмм-моль	единица измерения количества вещества в Международной системе единиц (СИ)
А	Ампер	единица измерения силы электрического тока в Международной системе единиц (СИ)
кд	канде́ла	единица силы света в Международной системе единиц (СИ)

Основные размерные переменные и константы, встречающиеся при решении задач гидродинамики в пакете OpenFOAM, приведены в таблице 1.2. Геометрические параметры сетки должны определяться в метрах.

Таблица 1.2. Размерности основных переменных и констант

Переменная	Обозначение в OpenFOAM	Единица измерения
Кинематическое давление	p	$м^2с^{-2}$
Скорость	U	$мс^{-1}$
Температура	T	K
Кинематическая вязкость	nu	$м^2с^{-1}$
Поток (flux)	phi	$м^3с^{-1}$
Время	time	$с$
Завихренность	vorticity	$с^{-1}$
Модуль завихренности	magVorticity	$с^{-1}$

Следующая строка в файле **blockMeshDict**

```
convertToMeters 0.1
```

устанавливает соответствие между единицами измерения, используемыми при построении геометрии, и реальными единицами измерения (в представленной конфигурации 1 ед $=$ 0.1 м) или, иными словами, устанавливает масштаб.

Далее идет список вершин **vertices**, где перечисляются все координаты вершин (*x*, *y*, *z* координаты) геометрии задачи.

```
vertices
(
    (0 0 0)  //0
    (1 0 0)  //1
    (1 1 0)  //2
    (0 1 0)  //3
    (0 0 0.1)//4
    (1 0 0.1)//5
    (1 1 0.1)//6
    (0 1 0.1)//7
);
```

Как упоминалось выше, геометрию задачи необходимо строить в трех измерениях. Для рассматриваемой задачи внешние границы области будут представлять собой параллелепипед с шириной равной 1, длиной равной 1 и высотой равной 0.1 (см. Рис. 1.3). Итого в списке вершин необходимо обозначить 8 точек. Для удобства каждую вершину можно пронумеровать в комментариях. Пакет OpenFOAM поддерживает комментарии в стиле языков программирования C/C++. Нумерация вершин проводиться с 0, далее обращение к вершинам проводится по их порядковому номеру.

В следующем списке **blocks** указываются составляющие блоки расчетной сетки задачи (в нашем случае блок всего один).

```
blocks
(
    hex (0 1 2 3 4 5 6 7) (100 100 1) simpleGrading (1 1 1)
);
```

Блок расчетной сетки – это подобласть геометрии задачи. Описание блоков производится в следующем формате:

```
hex (v1 v2 v3 v4 v5 v6 v7 v8)(nx,ny,nz) simpleGrading(ax,ay,az)
```

Параметр hex – определяет тип блока, а именно шестигранник (для версии пакета OpenFOAM 6 других типов доступных для пользователя нет). Параметры v1, v2, v3, v4, v5, v6, v7, v8 - это номера вершин, определяющие блок.

Отметим, что перечисление вершин блока должно выполняться в определенном порядке. Рекомендуем пользоваться следующим правилом для определения порядка вершин:

1. Сначала указываются первые 4 вершины с минимальным значением координаты по третьему измерению (для этой задачи $z = 0$), согласно следующему правилу:
   a. Из четырех вершин первой записывается вершина с минимальным значением координаты по второму и первому измерению (для этой задачи $y = 0$ и $x = 0$).
   b. Далее перечисляем вершины таким образом, чтобы обход области происходил против часовой стрелки, если смотреть в направление оси третьего измерения (z).
2. Переходим к следующему значению по третьему измерению (z).

3. Повторяем пункты 1, 2

Параметры **nx, ny, nz** – определяют количество ячеек в блоке по направлениям *x, y, z* соответственно. Если решается двухмерная задача, то в направление третьего измерения должна быть только одна ячейка.

Параметр **simpleGrading(ax,ay,az)** - отвечает за линейное сгущение сетки, **ax ay az** - параметры сгущения, которые определяют отношение длин ячеек по следующей формуле (см. также рис. 1.5):

$$\text{ax}=\frac{\delta x_E}{\delta x_S};\text{ay}=\frac{\delta y_E}{\delta y_S};\text{az}=\frac{\delta z_E}{\delta z_S}.$$

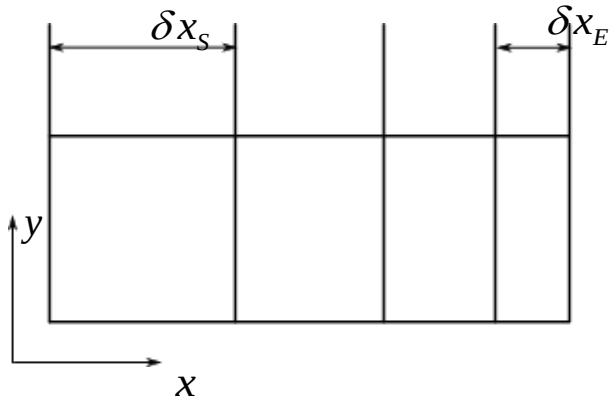


Рис. 1.5. Метод сгущения

Также можно воспользоваться более «продвинутой» командой:

```
edgeGrading(a1 a2 a3 a4 a5 a6 a7 a8 a9 a10 a11 a12),
```

которая позволяет задавать параметры сгущения для каждого ребра блока, где a1, a2, a3, a4, a5, a6, a7, a8, a9, a10, a11, a12 - номера ребер блока.

Далее идет список **edges** в котором описывается тип линий, которыми соединяются вершины:

```
edges
(
);
```

Формат описания линий следующий:

```
type v1 v2 ( (p1x p1y p1z) (p2x p2y p2z)...)
```

Параметр **type** – отвечает за способ соединения. Возможны следующие варианты:

- **arc** – дуга окружности;
- **spline** – кубический сплайн;
- **polyLine** – линейный сплайн.

Параметры **v1 v2** задают номера вершин, которые необходимо соединить. Параметры **((p1x p1y p1z) (p2x p2y p2z)...)** задают список точек, через которые должна пройти линия, соединяющая вершины. В случае соединения вершин с помощью дуги

окружности, необходимо указать только одну дополнительную точку. По умолчанию, соединение происходит прямыми линиями. В данном примере список пустой, поскольку все ребра - прямые линии.

В следующем списке **boundary** определяются границы, на которых будут задаваться граничные условия.

```
boundary
(
    movingWall
    {
        type wall;
        faces
        (
            (3 7 6 2)
        );
    }
    fixedWalls
    {
        type wall;
        faces
        (
            (0 4 7 3)
            (2 6 5 1)
            (1 5 4 0)
        );
    }
    frontAndBack
    {
        type empty;
        faces
        (
            (0 3 2 1)
            (4 5 6 7)
        );
    }
);
```

Формат описания границы следующий:

```
type name
(
(b11 b12 b13 b14)
…
(bn1 bn2 bn3 bn4)
)
```

Параметр **type** - отвечает за тип границы. Возможны следующие типы:

- **patch** – базовый тип, позволяющий использовать граничные условия, в которых не учитывается особенности геометрии.
- **symmetryPlane** – плоскость симметрии
- **empty –** пустая граница, используется для перехода от решения трехмерной к решению двухмерной задачи.

- **wedge** - используется для описания осесимметричной геометрии, например, для клина;
- **cyclic** – периодическая граница;
- **wall** – стенка (твердая граница);
- **processor** - используется для внутрипроцессорных границ (определяется пакетом автоматически при многопроцессорных расчетах);

Параметр **name** определяет название границы, например, movingWal;. Параметры **(b11 b12 b13 b14), …, (bn1 bn2 bn3 bn4)** определяют грани блоков, из которых состоит рассматриваемая граница. **bij** - номера вершин, описывающих грань блока. Граница может состоять из нескольких граней разных блоков. Например, граница области fixedWalls состоит из трех граней (0 4 7 3), (2 6 5 1) и (1 5 4 0) одного блока.

В списке **mergePatchPairs** описываются параметры сращивания поверхностей, если это необходимо. В данном случае этот блок остается пустым.

```
mergePatchPairs
(
);
```

Для генерации расчетной сетки необходимо, находясь в директории задачи в терминале набрать команду **blockMesh.** Данная команда запускает утилиту **blockMesh**, которая генерирует расчетную сетку согласно конфигурационному файлу **blockMeshDict**. Текущее состояние работы утилиты **blockMesh** сообщается в окне терминала.

В случае обнаружения ошибки в файле **blockMeshDict** утилита выдаст сообщение с номером строки, в которой обнаружена эта ошибка. Для успешной генерации сетки ошибок быть не должно.

Типичные ошибки, связанные с генерацией сетки:

1. Ошибка, связанная с установкой неправильного обхода поверхности блока.

```
--> FOAM FATAL ERROR:
Face 1 in patch 2 does not have neighbor cell face: 4(4 8 24 20)
```

Ошибку следует искать в модуле **blocks** или в модуле **boundary** в порядке перечисления вершин.

2. Ошибка, связанная с несоответствием количества ячеек между двумя соседними блоками.

```
--> FOAM FATAL ERROR:
Inconsistent number of faces between block pair 0 and 3
```

Ошибку следует искать в модуле **blocks** в параметрах, задающих количество ячеек.

3. Ошибка, связанная с несоответствием положения ячеек между двумя соседними блоками.

```
--> FOAM FATAL ERROR:
Inconsistent point locations between block pair 0 and 1
```

```
probably due to inconsistent grading.
```

Ошибку следует искать в модуле **blocks** в параметрах сгущения.

4. Предупреждение о наличии поверхности, не вошедшей в описание известных границ.

```
--> FOAM Warning :
    From function polyMesh: :polyMesh (… construct from shapes …)
    in file meshes/polyMesh/polyMeshFromShapeMesh.C at line 888
    Found 17 undefined faces in mesh; adding to default patch.
```

Если поверхность не входит в описание границы, то она автоматически будет входить в описание границы по умолчанию **defaultFaces**. Не включение поверхности в описание границы может привести к некорректным результатам.

5. Ошибка, возникающая при совпадении вершин.

```
***Max skewness = 4.90416, 596 highly skew faces detected which
may impair the quality of the results
```

Ошибку следует искать в списках **vertices** (проверить координаты), **blocks** (проверить перечисление вершин) и **boundary** (проверить перечисление вершин).

1.5. Граничные и начальные условия

После создания сетки необходимо определить граничные и начальные условия. В задаче **cavity** необходимо задать граничные и начальные условия для давления и скорости. Соответствующие конфигурационные файлы находятся в папке **0**. В файлах **p** и **U** описываются начальные и граничные условия для давления и скорости соответственно.

Разберем содержимое файла **p**:

Параметр **dimensions** задает размерность поля.

```
dimensions      [0 2 -2 0 0 0 0];
```

В общем случае размерность задается по следующему правилу:

$$[Var] = кг^{n_1} м^{n_2} с^{n_3} K^{n_4} кмоль^{n_5} A^{n_6} кд^{n_7}$$

$$[n_1\ n_2\ n_3\ n_4\ n_5\ n_6\ n_7]$$

Кинематическое давление **p** имеет размерность $м^2 с^{-2}$, поэтому параметр **dimensions** задается следующим образом: **[0 2 -2 0 0 0 0]**:

Параметр **internalField** задает значение поля внутри расчетной области, которое может быть задано однородным (**uniform**), задаваемым одной цифровой величиной; или неоднородным, когда все значения поля должны быть заданы с помощью списка.

```
internalField   uniform 0;
```

Для начального момента времени **internalField** можно трактовать как начальное условие.

В списке **boundaryField** определяются граничные условия. Формат задания граничных условий следующий:

```
BoundaryName
```

```
{
     type    typename;
}
```

Название границы **BoundaryName** должно быть из списка boundary, определенного ранее в файле **blockMeshDict**. Параметр **typename** определяет тип граничных условий. Необходимо определить граничные условия для всех границ из списка boundary.

В примере **cavity** определены три границы.

```
boundaryField
{
    movingWall
    {
        type            zeroGradient;
    }

    fixedWalls
    {
        type            zeroGradient;
    }

    frontAndBack
    {
        type            empty;
    }
}
```

Граница **fixedWalls** содержит неподвижные боковые стенки и основание каверны, граница **frontAndBack** содержит переднюю и заднюю стенки каверны, граница **movingWall** содержит подвижную крышку каверны. На границах **fixedWalls** и **movingWall** для давления определяются однородные условия Неймана, для этого в качестве типа граничных условий указывается **zeroGradient.** На границах **frontAndBack** задаются пустые граничные условия **empty** (так как моделируется двумерная задача)**.**

Далее рассмотрим файл **U**.

```
dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0 0 0);

boundaryField
{
    movingWall
    {
        type            fixedValue;
        value           uniform (1 0 0);
    }

    fixedWalls
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
```

```
    frontAndBack
    {
        type            empty;
    }
}
```

Размерность задается как **(dimensions)** *м/с* [0 1 -1 0 0 0 0]. Внутреннее поле инициализируется как однородное (**uniform**). Однородное векторное поле должно быть определено тремя компонентами вектора, в данном случае (0 0 0). Скорость во всех направлениях отсутствует.

Граничные условия для скорости (для двухмерной задачи) на границе **frontAndBack** определяются как **empty**. На границе **fixedWalls** задается условие прилипания, то есть определяются нулевые значения для всех компонент скорости fixedValue uniform (0 0 0). На границе **movingWall** также задается условие прилипания, но здесь горизонтальная компонента скорости задается равной единице fixedValue uniform (1 0 0).

1.6. Физические свойства среды

В качестве решателя **(solver)** в примере **cavity** используется программный модуль **icoFoam**, который решает уравнение Навье-Стокса для вязкой несжимаемой жидкости, используя алгоритм **PISO**. Для данного решателя единственное свойство среды, которое следует задать - это кинематическая вязкость. Она задается в файле **transportProperties**, находящемся в папке **constant**. Рассмотрим этот файл. Вязкость задается с помощью параметра **nu**

```
nu              [0 2 -1 0 0 0 0] 0.0025;
```

Здесь в квадратных скобках, как и для других физических величин, задается размерность. Последнее значение в строке определяет величину кинематической вязкости. Напомним, что в примере cavity мы с помощью изменения параметра nu будем изменять число Рейнольдса.

1.7. Конфигурация параметров расчета

Параметры, относящиеся к контролю времени, чтению и записи данных решения описываются в файле **controlDict,** находящемся в папке **system**. Параметр **application** указывает на утилиту, реализующую решение задачи, в данном примере это **icoFoam.** Для запуска расчета необходимо установить время начала и конца расчета, а также шаг по времени. Пакет **OpenFOAM** предоставляет большую гибкость в управлении временем.

Расчеты в задаче **cavity** будут проводиться с момента времени $t = 0$. Это означает, что пакету **OpenFOAM** необходимо считывать массивы данных из каталога с именем **0**. Поэтому мы устанавливаем для **startFrom** значение **startTime**, а затем назначаем

startTime значение равное **0**. Если требуется продолжить расчеты с другого момента времени, например **0.1** то в качестве **startTime** необходимо поставить **0.1** и в корне программы должен присутствовать каталог с именем **0.1,** содержащий файлы с данными о полях. Для задания времени конца расчета необходимо указать значение **stopAt** как **EndTime**, а затем установить значение **EndTime,** равное, например, **0.5**. Обычно, время конца расчета определяется из физических соображений или специфических нужд пользователя.

Теперь необходимо установить шаг по времени, который задается значением **deltaT**. Следует отметить, что для достижения временной точности и численной стабильности во многих расчетных схемах требуется, чтобы число Куранта было не больше чем единица.

Число Куранта определяется для одной ячейки, как:

$$Co = \frac{\delta t|U|}{\delta x},$$

где δt - шаг по времени, $|U|$ представляет собой величину скорости в этой ячейке, а δx – это размер ячейки в направлении скорости. Скорость потока варьируется в пределах области, поэтому необходимо соблюдать не условие $Co \leq 1$, а условие $\max(Co) \leq 1$, где максимум вычисляется по всей области.

В данной задаче размер ячейки постоянный, поэтому максимальное значение Co будет в ячейке, которая находится вблизи подвижной верхней границы, где скорость близка к 1 м/с. Размер ячейки определяется по формуле: $\delta x = d/n = 0.1/n$. Для того, чтобы получить число Куранта меньше или равное единицы во всей области, шаг по времени **deltaT** должен выбираться как: $\delta t \leq 0.1/n$.

В процессе моделирования мы будем записывать результаты через определенные промежутки времени. Параметр **writeControl** представляет собой набор опций по настройке вывода результата. В примере используется вариант **timeStep**, который определяет, какие результаты будут записываться каждый *n-ый* шаг по времени, где значение *n* определяется в соответствии со значением **writeInterval**. Предположим, что нам необходимо записывать результаты в 0.1, 0.2, . . ., 0.5 сек. с шагом 0.005 сек., то есть выводить результаты на каждом 20-ом шаге. Для этого надо задать значение **writeInterval** равное **20**. OpenFOAM создает новую папку с именем текущего времени, например, 0.10, содержащую полный набор данных о полях (в данном примере о полях скорости и давления).

```
application     icoFoam;
startFrom       startTime;
startTime       0;
stopAt          endTime;
endTime         0.5;
```

```
deltaT          0.005;
writeControl    timeStep;
writeInterval   20;
```

Параметр **purgeWrite** позволяет ограничить количество создаваемых каталогов с данными о полях. Например, мы начинаем расчет с момента времени t = 5 с шагом сохранения 1 и используем команду **purgeWrite 2.** В этом случае сначала создадутся каталоги с именами 6 и 7, затем эти каталоги будут заменены на 8 и 9 соответственно, и будут содержать только поля, соответствующие их именам. То есть, в корневой папке программы всегда будут только три каталога с данными о полях. Для отключения ограничения необходимо в качестве аргумента задать 0. Как показано в примере.

```
purgeWrite      0;
```

Параметр **writeFormat** задает формат файла с данными. Возможны 2 формата ascii (ASCII) и binary (Binary).

```
writeFormat     ascii;
```

Параметр **writePrecision** задает количество знаков после запятой для выводимых в файл чисел.

```
writePrecision  6;
```

Параметр **writeCompression** определяет, нужно ли архивировать выходные данные. Архивация производится в формате **gzip**. Если архивация требуется, то в качестве аргумента указываем **on** в противном случае **off**.

```
writeCompression off;
```

Параметр **timeFormat** задает формат названия каталогов с данными. Возможны следующие варианты:

- **fixed** – ± m.dddddd – число знаков после запятой определяется командой TimePrecison
- **scientific** – ± m.dddddd ± xx- число знаков после запятой определяется командой TimePrecison , xx – показатель степени.
- **general** – если точность вывода $<10^{-4}$ то формат стандартный m.d и число знаков после запятой определяется автоматически в противном случае формат аналогичен **fixed**

```
timeFormat      general;
```

Параметр **TimePrecison** Задает число знаков после запятой в формате названия каталогов с данными.

```
timePrecision   6;
```

Параметр **runTimeModifiable** определяет необходимость считывания файла для каждого момента времени.

```
runTimeModifiable true;
```

1.8. Выбор методов дискретизации и методов решения уравнения движения жидкости

В файле **fvSchemes,** находящимся в папке **system,** задаются схемы аппроксимации для каждого члена уравнений исходной математической модели.

Блок **ddtSchemes** отвечает за аппроксимацию первых производных по времени.

```
ddtSchemes
{
    default         Euler;
}
```

В блоке **gradSchemes** задаются схемы аппроксимации оператора градиента (в данном случае градиента давления).

```
gradSchemes
{
    default         Gauss linear;
    grad(p)         Gauss linear;
}
```

В блоке **divSchemes** задаются схемы аппроксимации оператора дивергенции (конвективных слагаемых).

```
divSchemes
{
    default         none;
    div(phi,U)      Gauss linear;
}
```

В блоке **laplacianSchemes** задаются схемы аппроксимации оператора Лапласа (диффузионных слагаемых).

```
laplacianSchemes
{
    default         Gauss linear orthogonal;
}
```

В блоке **interpolationShemes** задаются схемы интерполяции значений, определенных в центрах ячеек расчетной сетки, на грани ячеек.

```
interpolationSchemes
{
    default         linear;
}
```

В блоке **snGradSchemes** задаются схемы аппроксимации нормальных производных на границах расчетной области (surface normal gradient schemes).

```
snGradSchemes
{
    default         orthogonal;
}
```

Блок **fluxRequired** отвечает за вычисление потоков (**flux**)

```
fluxRequired
```

```
{
    default         no;
    p               ;
}
```

Последний блок не используется в версиях OpenFOAM старше 3.0.

Все возможные схемы для аппроксимации слагаемых можно узнать с помощью встроенного поиска. Например, в OpenFOAM 4 все схемы для аппроксимации производной по времени (представленные в примерах) можно увидеть с помощью команды

```
foamSearch $FOAM_TUTORIALS ddtSchemes.default fvSchemes
```

В OpenFOAM 8 синтаксис этой команды несколько изменен

```
foamSearch $FOAM_TUTORIALS fvSchemes ddtSchemes.default
```

Выбор решателей линейных уравнений и других алгоритмов решения задачи проводится в файле **fvSolution,** находящемся так же в директории **system**.

Рассмотрим файл **fvSolution** примера **cavity**. В блоке **solvers** указаны решатели для давления и скорости. Параметр **solver** указывает на тип решателя. Для давления выбирается метод **PCG** (preconditioned conjugate gradient - метод сопряженных градиентов с предобуславлевателем), а для скорости - **smoothSolver** (метод релаксации). Параметр **preconditioner** задает тип предобуславлевателя. Для давления это **DIC** (Diagonal incomplete-Cholesky (symmetric) - диагональное неполное разложение Холецкого). Для скорости задается метод релаксации (параметр **smoother**) это symGaussSeidel (метод Гаусса-Зейделя). Параметр **tolerance** задает точность, которую должен достичь итерационный процесс. Параметр **relTol** задает относительную точность решателя.

```
solvers
{
    p
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-06;
        relTol          0.05;
    }

    pFinal
    {
        $p;
        relTol          0;
    }

    U
    {
        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-05;
```

```
        relTol          0;
    }
}
```

Далее идет блок описания алгоритма решения задачи. В данном примере используется алгоритм **PISO** (Pressure-Implicit with Splitting of Operators**)**. Параметр **nCorrectors** определяет число коррекций поля давления, параметр **nNonOrthogonalCorrectors** определяет количество коррекций на неортогональность (в силу ортогональности используемой сетки для решения задачи о течении в квадратной каверне такие коррекции не требуются).

```
PISO
{
    nCorrectors     2;
    nNonOrthogonalCorrectors 0;
    pRefCell        0;
    pRefValue       0;
}
```

В задаче давление относительно, то есть давление определяется с точностью до произвольной константы. В таких случаях решатель (solver) устанавливает относительный уровень давления с помощью **pRefValue** в ячейке **pRefCell**. В этом примере оба значения равны нулю. Изменение любого из этих значений будет изменять область абсолютного давления, но не область относительной скорости или давления.

1.9. Запуск расчета

Для запуска расчета необходимо, находясь внутри каталога с задачей, вызвать решатель (в случае с **cavity** это **icoFoam)**. Ход решения записывается в окне терминала. Он сообщает пользователю текущее расчетное время (`time`), среднее и максимальное значения числа Куранта (`courant number mean, max`), начальные и конечные значения, невязки (`residual`) на каждом этапе решения задачи.

```
Time = 0.495

Courant Number mean: 0.222158 max: 0.852134
DILUPBiCG: Solving for Ux, Initial residual = 1.99664e-07 ,Final
residual = 1.99665e-07, No Iterations 0
DILUPBiCG: Solving for Uy, Initial residual = 4.36311e-07 ,Final
residual = 4.36311e-07, No Iterations 0
DICPCG: Solving for p, Initial residual = 1.0746e-06 ,Final
residual = 3.53797e-07, No Iterations 1
time step continuity errors: sum local = 5.37651e-09, global = -
6.05083e-19, cumulative = 1.44504e-18
DICPCG: Solving for p, Initial residual = 6.81574e-07 ,Final
residual = 6.81574e-07, No Iterations 0
time step continuity errors: sum local = 8.06059e-09, global =
1.8016e-19, cumulative = 1.6252e-18
ExecutionTime = 0.13 s ClockTime = 1 s
```

После расчета в директории проекта должны появиться новые папки (например, 0.1, 0.2 и т.д.), в которых хранятся результаты расчетов в заданные моменты времени (см. рис. 1.6).

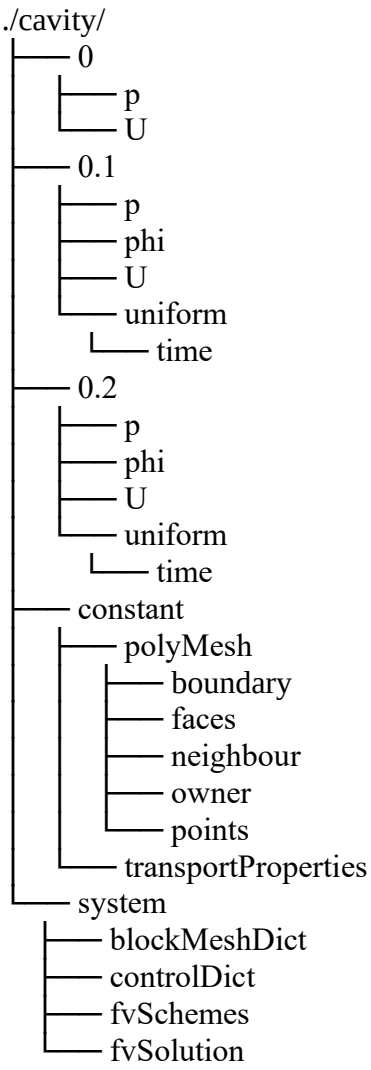


Рис. 1.6. Дерево каталогов в папке проекта после расчета

1.10. Визуализация результатов расчета

1.10.1. Визуализация расчетной сетки и физических полей

Визуализацию полученных результатов можно провести в программе ParaView, которая устанавливается вместе с пакетом OpenFOAM. Это свободно распространяемое программное обеспечение можно также установить отдельно, скачав на сайте https://www.paraview.org/ (или из репозиториев в дистрибутивах на базе ядра Linux). На сайте также можно найти подробную техническую документацию на английском языке.

Открыть проект OpenFOAM в графическом интерфейсе ParaView можно двумя способами:

а) набрать команду **paraFoam** в терминале внутри каталога с расчетом (если ParaView установлен одновременно с OpenFOAM);

б) создать пустой файл 1.foam внутри каталога с расчетом и открыть его с помощью команды

```
paraview 1.foam
```

или два раза кликнув на него в файловом менеджере.

Общий вид окна программы ParaView представлен на рис. 1.7. Основные элементы управления paraview сосредоточены на панели инструментов, панели обзора объектов (**Pipeline Browser**), панели свойств объекта (**Properties, Information**), панели анимации (**Animation View**) и в окне визуализации. Настроить (открыть) элементы управления можно в меню **View.**

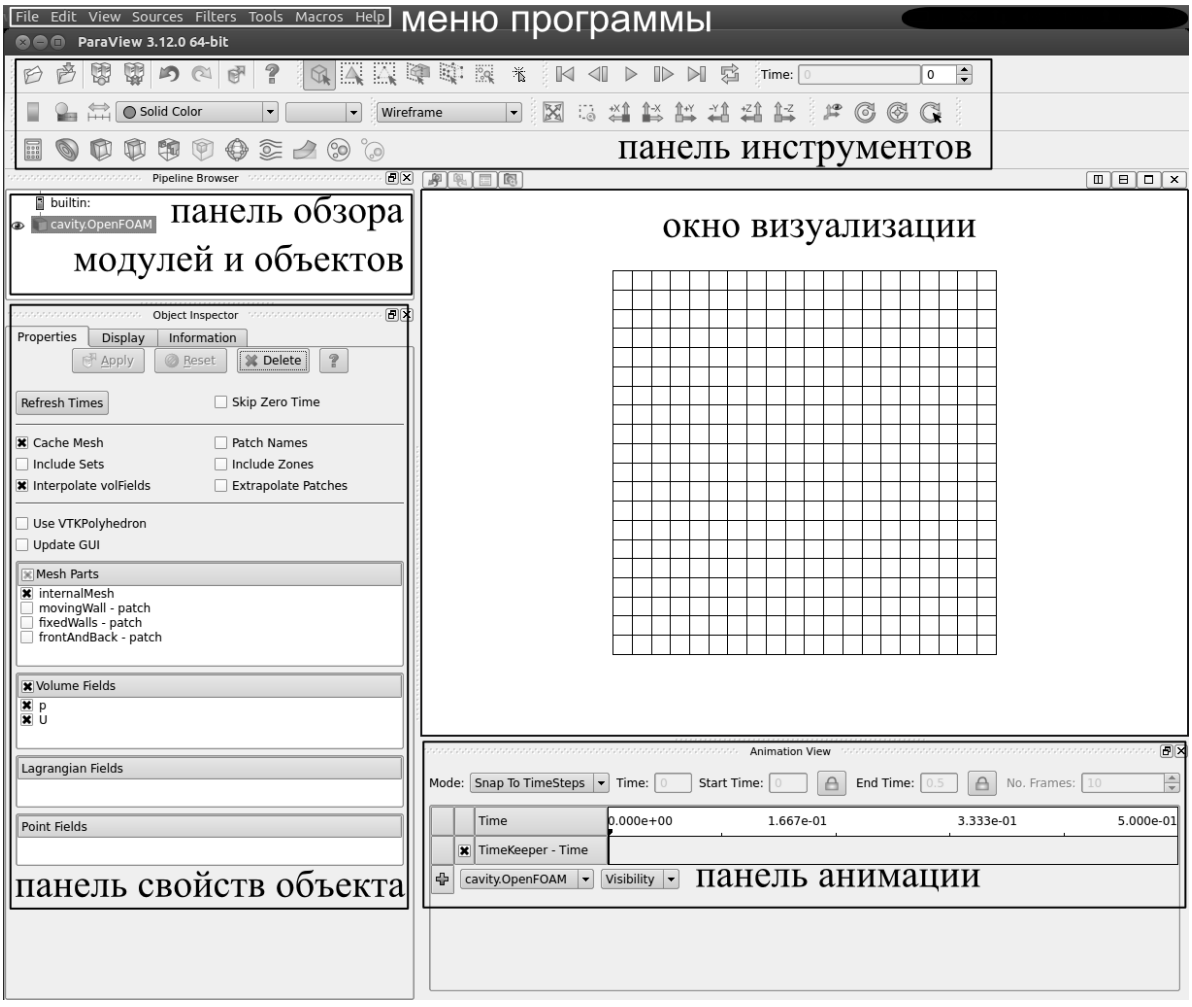


Рис. 1.7. Общий вид окна ParaView

В **Pipeline Browser** указываются, какие объекты (источники данных и фильтры) используются в данный момент. На рис. 1.7. используется только один источник данных **cavityOpenFOAM,** который содержит все данные о полях, полученных в ходе расчета и постобработки в OpenFOAM**.**

В панели **Properties** перечислены свойства и настройки модулей, объектов и фильтров (рис. 1.8).

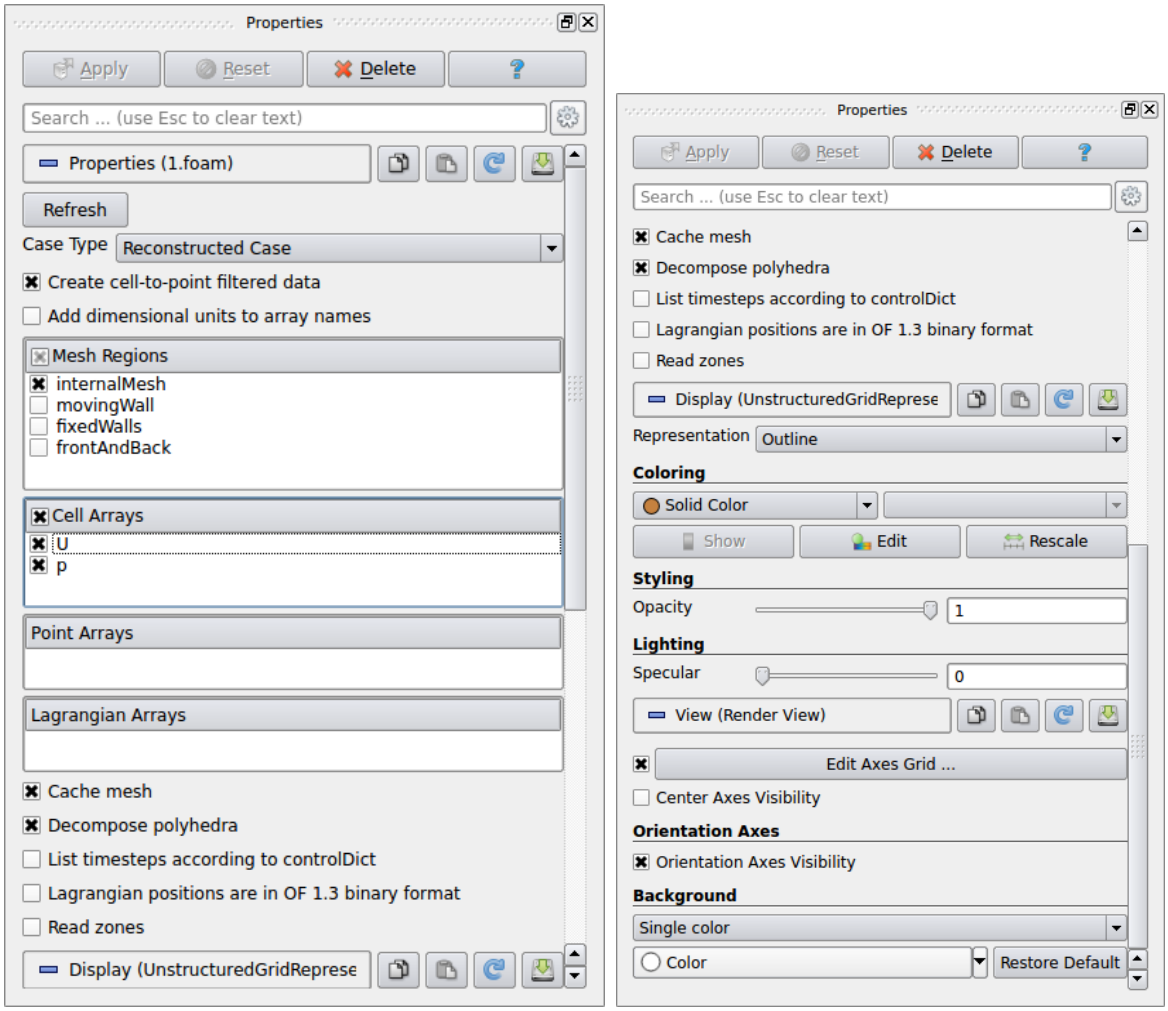


Рис. 1.8. Панель свойств объекта (**Properties**)

В меню **Mesh Regions** перечислены части геометрии задачи, которые необходимо отобразить. Далее будем рассматривать внутреннюю часть расчетной области, она обозначается как **internalMesh**. В меню **CellArrays** указываются поля, которые будут доступны для визуализации. В данном примере это поля давления и скорости.

Для начала визуализации необходимо нажать кнопку **Apply**. Если требуется посмотреть сетку, то во вкладке **Display** (панель **Properties**) необходимо выбрать **Wireframe** в меню **Representation** (рис. 1.8.). Если все параметры заданы правильно, то в окне визуализации появится расчетная трехмерная сетка задачи (рис. 1.9.).

В окне визуализации можно вращать объекты. Для этого, удерживая нажатой левую кнопку мыши, необходимо перемещать мышь в сторону, куда следует повернуть объект. Также можно изменять масштаб с помощью колесика мыши.

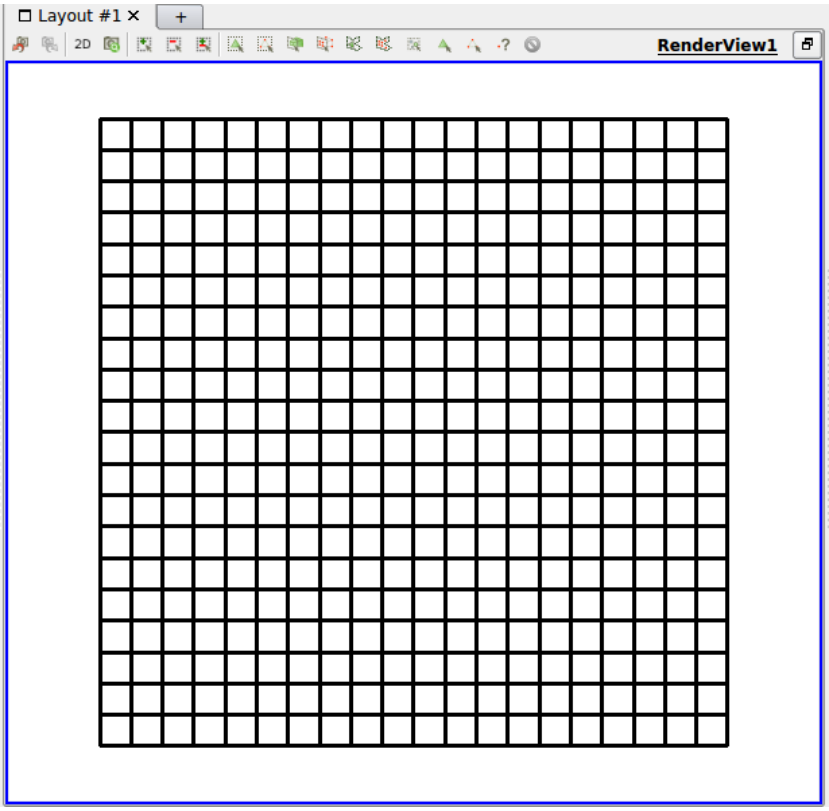


Рис.1.9. Общий вид расчетной сетки в окне ParaView

В качестве следующего шага проведем визуализацию гидродинамических полей, полученных в ходе расчета. Многие поля можно визуализировать с помощью цветовой палитры. Для визуализации поля давления таким способом, во вкладке **Display** в меню **Representation** выберете тип отображения **Surface**; в панели **Coloring** выберете поле даления **p** (см. рис. 1.10). Далее, регулируя время (**Time**) на панели инструментов (см. рис. 1.7), расположенной в верхней части окна, можно наблюдать за изменением давления с течением времени. Аналогично можно визуализировать различные компоненты поля скорости.

Результат визуализации можно сохранять в виде изображения или в специальном формате для хранения данных. Для этого в меню **File** следует выбрать опцию **Save Screenshot** или **Save Data** соответственно.

1.10.2. Визуализация функции тока и завихренности

Во многих гидродинамических задачах помимо стандартных физических полей – скорости и давления (U, p) – бывает удобно рассматривать функцию тока (psi) и завихренность (vorticity). Вычисление этих полей не происходит автоматически в ходе решения задачи, однако их можно вычислить в рамках постобработки результатов с помощью утилит OpenFOAM. Для вычисления функции тока (в сохраненные моменты времени) необходимо в папке выполнить команду streamFunction

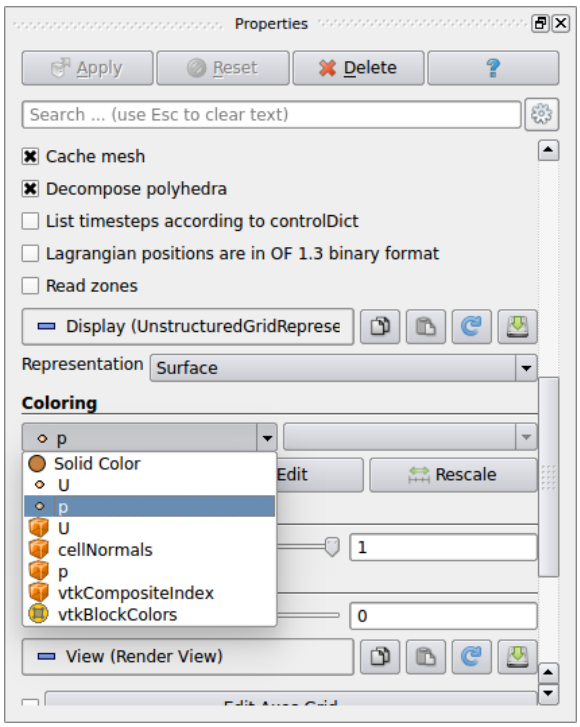


Рисунок 1.10. Выбор поля давления для визуализации

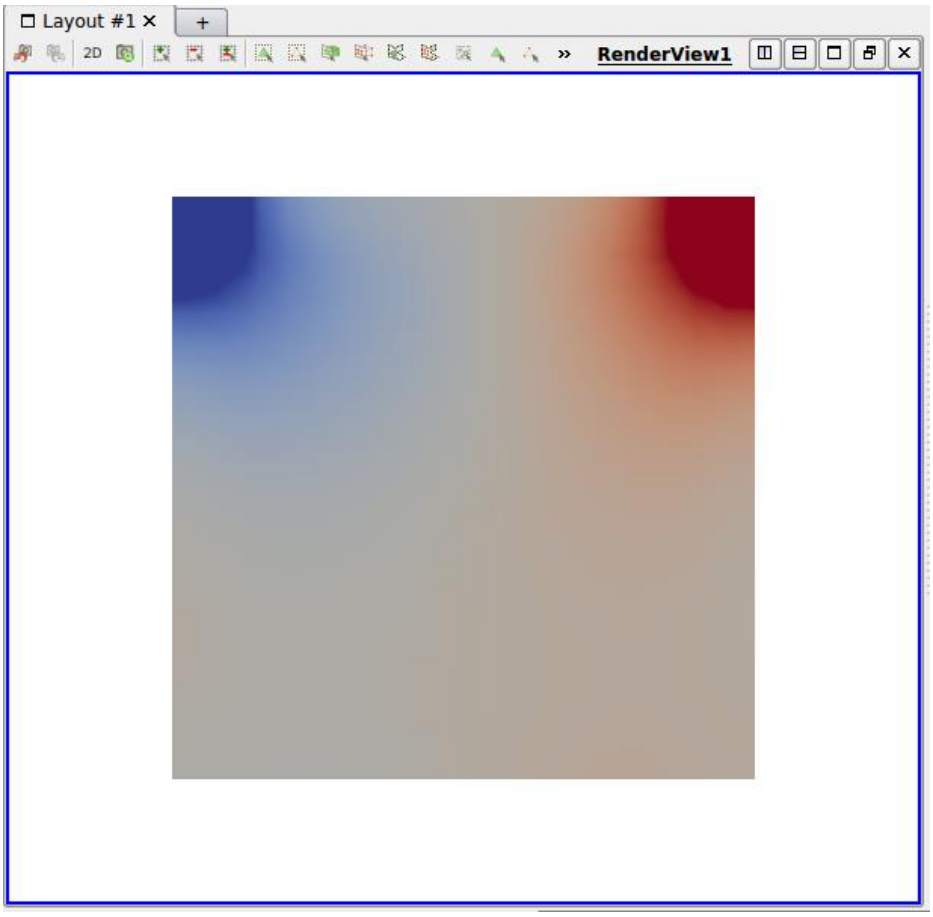


Рис.1.11. Визуализация распределение давления в расчетной области

Для расчета завихренности необходимо выполнить команду

```
postProcess -func vorticity
```

После расчета в папках, хранящих значения полей на разных временных слоях, появятся новые файлы psi и vorticity (см. рис. 1.12). Полученные новые данные можно визуализировать с помощью ParaView.

```
./0.2
├── p
├── phi
├── streamFunction
├── U
├── uniform
│   └── time
└── vorticity
```

Рис. 1.12. Дерево каталогов в папке 0.2

Поле завихренности можно визуализировать аналогично полю давления. Для визуализации линий тока нам потребуется использовать дополнительные фильтры.

Прежде всего, воспользуемся фильтром Slice. Этот фильтр позволяет визуализировать скалярные и векторные поля в сечениях исходной области (напомним, что каверна фактически является трехмерной). Фильтр можно найти на панели инструментов (см. рис. 1.13) или в главном меню программы во вкладке Filters. Далее на панели Properties необходимо выбрать тип сечения (Slice Type) Plane.

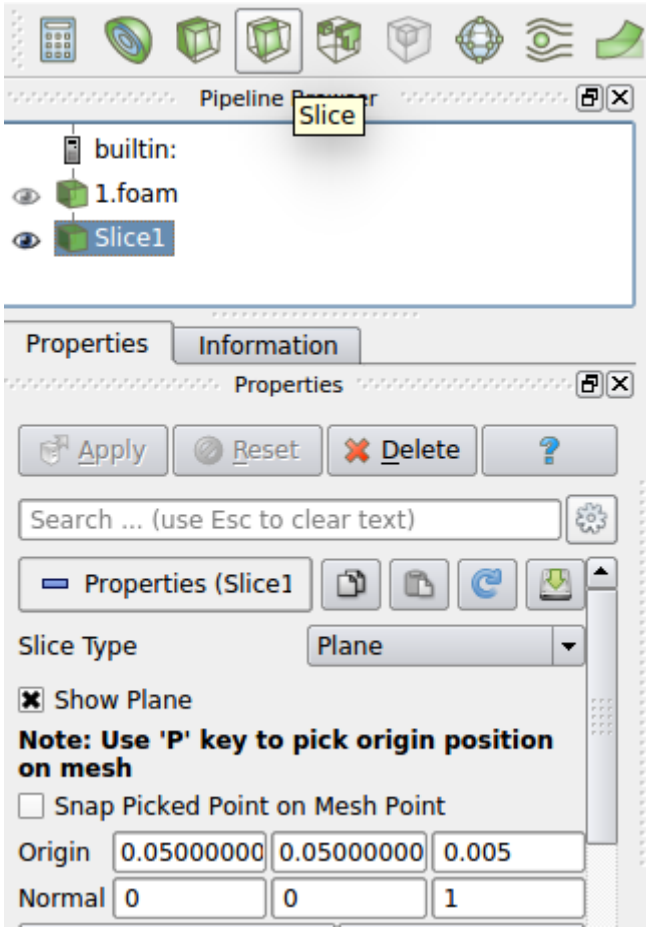


Рис. 1.13. Создание фильтра Slice и настройка его свойств в панели Properties

Далее тут же необходимо задать координаты центра плоскости (Origin) и координаты вектора нормали (Normal) к этой плоскости (см. рисунок). Для рассматриваемого случая это (0.05 0.05 0.005) и (0 0 1).

После этого применяем второй фильтр Contour. Фильтр так же можно найти на панели инструментов или в главном меню программы во вкладке Filters. Важно, что перед выбором фильтра Contour на панели Pipeline Browser должен быть выделен предыдущий фильтр Slice.

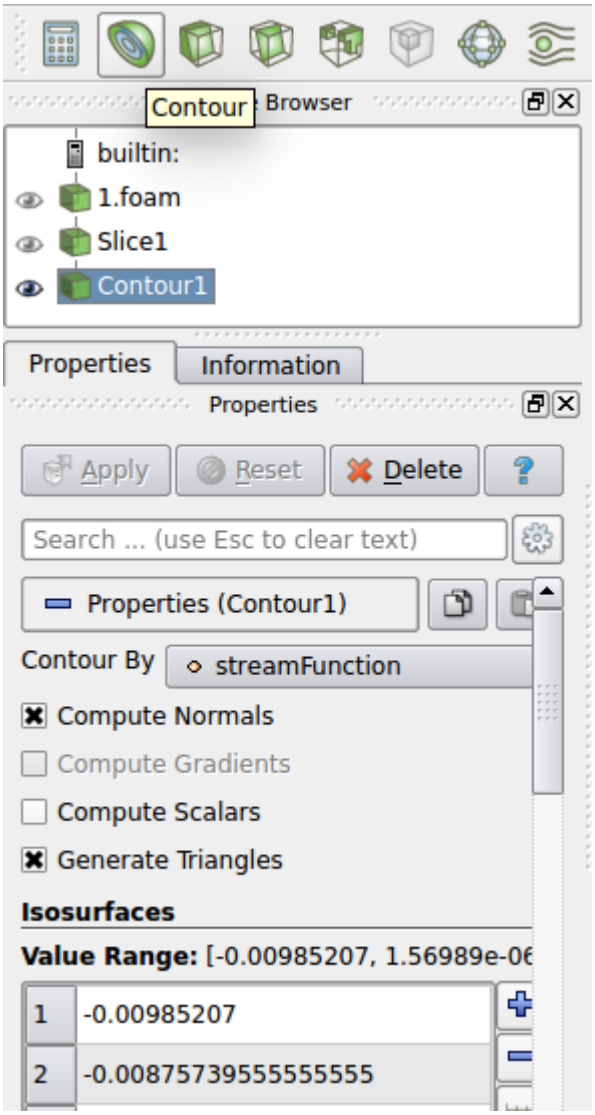


Рис. 1.14. Создание фильтра Contour и настройка его свойств в панели Properties

После этого, необходимо выбрать по какому полю строить изолинии. Для этого на панели Properties фильтра Contour выбираем из выпадающего списка Countour By поле streamFunction. Далее выбираем изолинии для отображения, для этого в пункте Isosurfaces нажимаем кнопку Add a range of values (рис. 1.15).

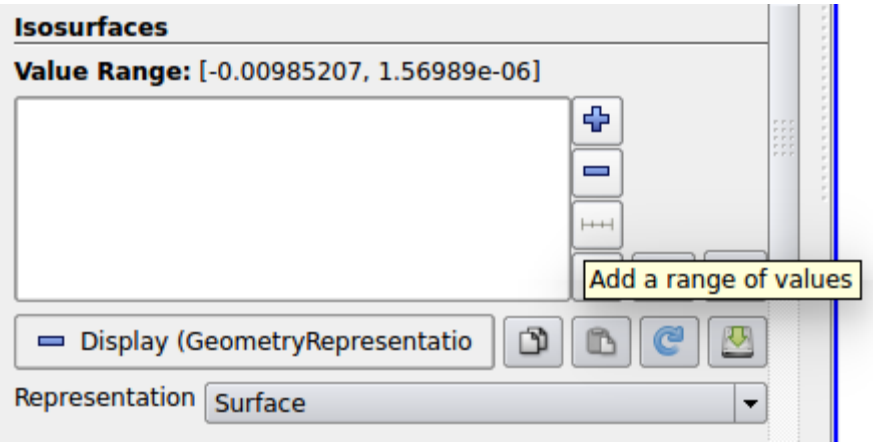


Рис. 1.15. Выбор линий тока для визуализации

В открывшемся диалоговом окне необходимо определить три значения. Первые два значения (From и To) определяют границы диапазона, в котором отображаются линии тока. Последнее значение определяет количество линий тока для отображения, они будут либо равномерно распределены по диапазону, либо логарифмически, если активирован флажок Use Logarithmic Scale (см. рис. 1.16). После нажатия на кнопку OK, обозначенные линии тока появятся в окне визуализации. В случае каверны удобно сначала выбрать диапазон отрицательных значений (-0.00985207 0), а затем вновь открыть диалоговое окно Add Range и выбрать диапазон положительных значений (0 1.56989e-06). Обратите внимание, что paraview сам определяет максимальные и минимальные значения выбранного поля (в данном случае streamFunction) для текущего момента времени.

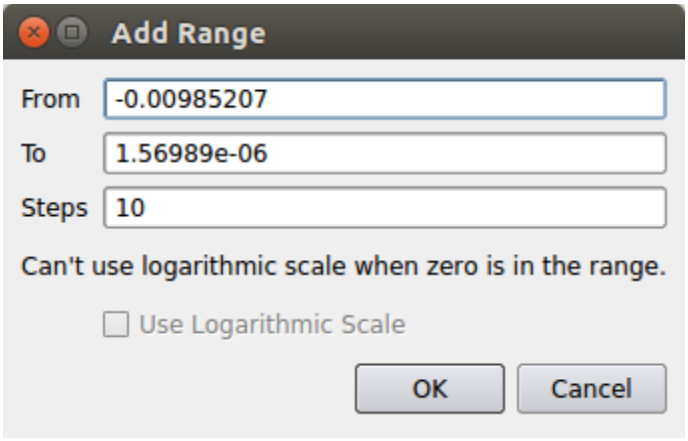


Рис. 1.16. Определение границ диапазона

Полученные линии тока окажутся очень тонкими линиями. Для регулирования их толщины можно применить еще один фильтр Tube (его можно найти в главном меню во вкладке Filters). Важно, что перед выбором фильтра Tube на панели Pipeline Browser должен быть выделен предыдущий фильтр Contour. За счет подбора параметра Radius (на панели Properties фильтра Tube) можно отрегулировать толщину линий тока. Результат должен выглядеть так, как на рис. 1.17.

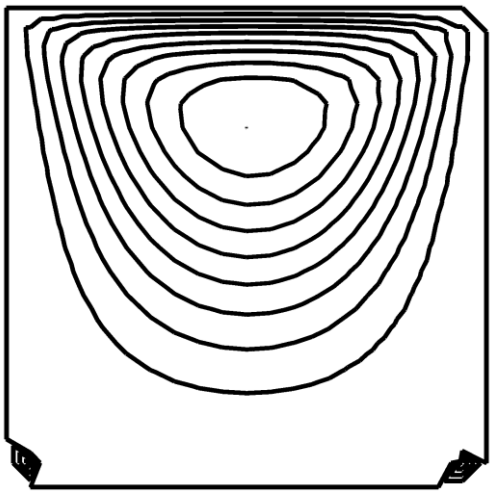


Рис. 1.17. Линии тока в момент времени t=0.5 для Re=10 на сетке 20x20 узлов

1.11. Анализ характеристик течения в квадратной каверне на основе численных расчетов

Весь изложенный выше материал был получен с использованием стандартных (грубых) настроек проекта cavity. В действительности даже для рассматриваемой тестовой задачи о течении жидкости в квадратной каверне необходимо провести детальную настройку параметров численных методов для получения точных результатов моделирования. На рис. 1.18. представлены картины течений, которые должны наблюдаться в каверне при разных значениях числа Рейнольдса после установления течения. В таблице 1.3. представлены минимальные значения функции тока при установившемся течении (по данным представленным в литературе [Егоров 2009, Sahin 2003, Nuriev2016]).

Таблица 1.3. Минимальные значения функции тока в каверне

Re	0	100	400	1000	3200	5000	7500	1000
ψ_{min}	-0.1	-0.1034	-0.1139	-0.118	-0.12	-0.122	-0.1223	-0.1224

Поставим задачу воспроизвести эти данные с помощью численной модели построенной в OpenFOAM.

Начнем со случаев Re=100, 400. Очевидно, что базовая сетка 20x20 слишком грубая для проведения расчета даже при малых числах Рейнольдса, а также что конечное время t=0.5 не достаточно для установления течения. Для корректировки расчетной схемы выполним следующие шаги.

1. В файле blockMeshDict определим новое разбиение области 100x100.

2. В файле controlDict изменим шаг по времени так, чтобы на новой сетке выполнялось условие Куранта.
3. В файле controlDict установим конечное время расчета на 3.
4. В файле controlDict изменим значение параметра writeControl на adjustableRunTime. Это позволит определять время вывода данных через заданный временной шаг, определяемый параметром writeInterval. Определим writeInterval равным 1.0.

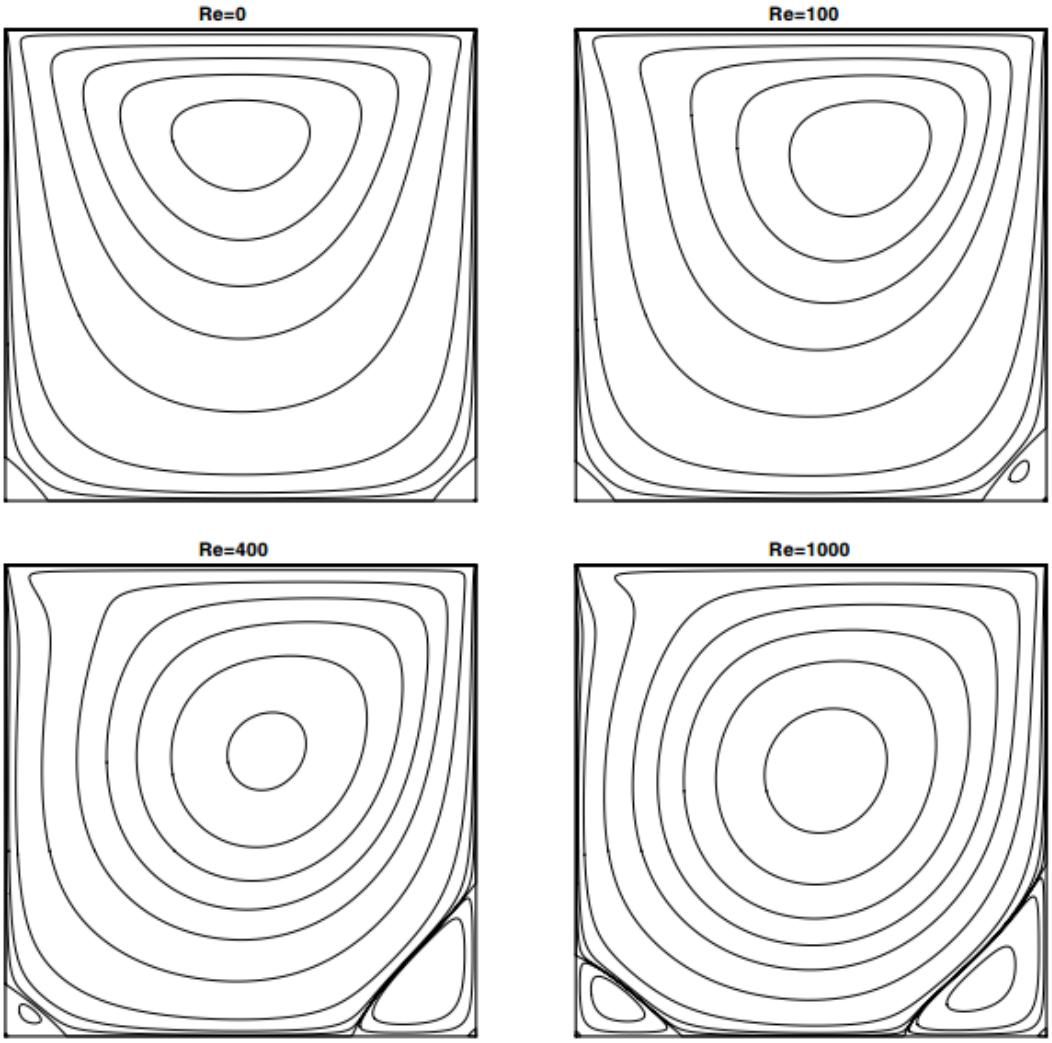


Рис. 1.18. Линии тока при разных числах Рейнольдса, полученные в работе [Nuriev 2016]

5. В файле transportProperties изменим переменную nu так, чтобы число Рейнольдса в расчете было правильно установлено.
6. Очистим директорию проекта от старых расчетных данных (удалим папки 0.1, 0.2 и т.д., полученные в ходе предыдущего расчета).

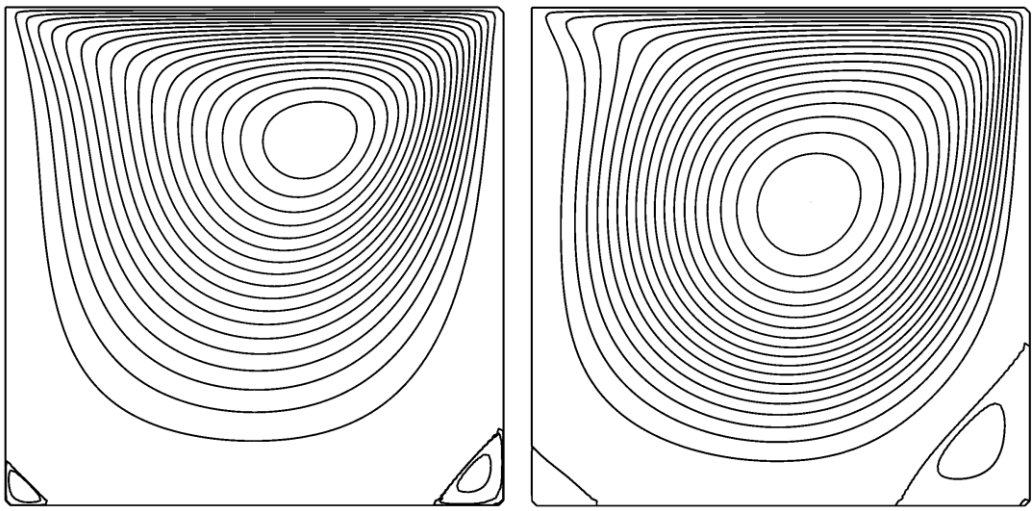


Рис. 1.19. Визуализация расчетов в OpenFOAM с помощью программы ParaView. Левое изображение получено для Re = 100, правое изображение – для Re = 400

Результаты расчетов для Re=100, 400 представлены на рис. 1.19. Как можно видеть, линии тока хорошо согласуются с известными результатами. Определение минимальных значений функции тока дает следующие результаты $Re = 100,\ \psi_{min} = -0.01033;\ Re = 400,\ \psi_{min} = -0.0113.$ Заметим, что для сравнения с результатами в таблице 1.3, полученные данные надо умножить на 10, в связи с тем, что размеры области в расчетах отличались в 10 раз. Значения ψ_{min} также хорошо согласуются. Полученные результаты свидетельствуют о хорошей точности построенной модели для расчета течений в квадратной каверне.

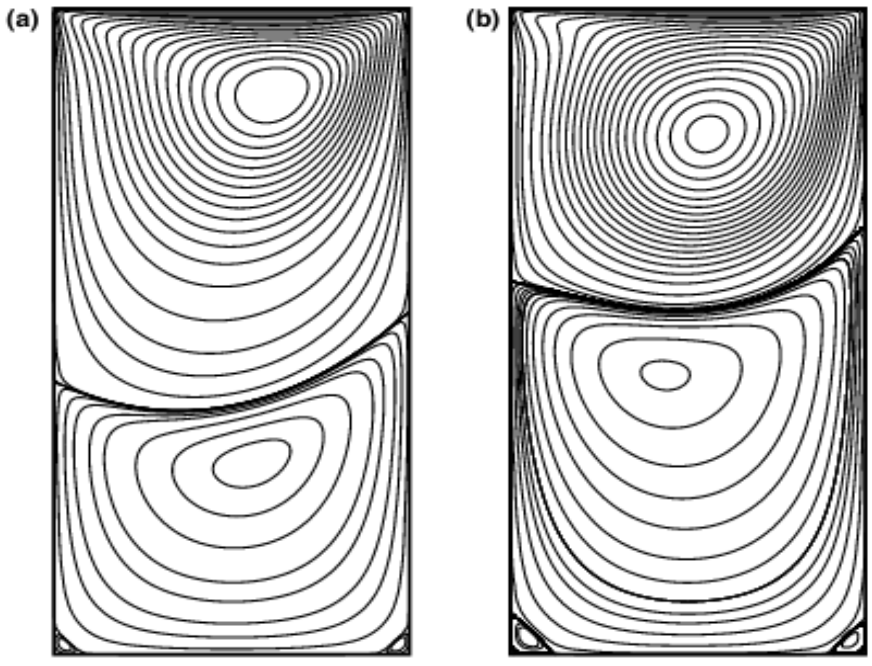


Рис. 1.20. Линии тока для Re =100 (a), Re=400 (b) в каверне с соотношением сторон 1:2.

Задачи для самостоятельного исследования

1. Проведите расчеты течения в каверне при Re=1000. Подберите необходимые параметры расчетной схемы для достижения заданной точности (по оценке ψ_{min}). Проведите визуализацию результатов расчета.
2. Проведите численное исследование течения в прямоугольной каверне (с соотношением сторон 1:2) с подвижной верхней крышкой для Re=100, Re=400. Сравните результаты с картинами течения, изображенными на рис. 1.20.

2. Исследование задачи об обтекании круглого цилиндра

2.1. Постановка задачи

В качестве второго примера построим численную модель обтекания круглого цилиндра потоком несжимаемой жидкости. Пусть круглый цилиндр радиуса R в начальный момент времени $t = 0$ помещен в равномерный безграничный поток вязкой несжимаемой жидкости, имеющий скорость U. В результате взаимодействия жидкости с цилиндром, поток в следе за цилиндром становится неравномерным и вихревым (см. рис. 2.1), а на цилиндр начинает действовать ненулевая гидродинамическая сила. Движение жидкости, как и в первой задаче, описывается нестационарным уравнением Навье-Стокса. Обезразмерив скорость на U, пространственные координаты на R, время на R/U, запишем это уравнение в безразмерных переменных как

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} = -\nabla \mathrm{p} + \frac{1}{Re}\Delta \boldsymbol{u}, \qquad \nabla \cdot \boldsymbol{u} = 0, \qquad Re = \frac{U\,R}{\nu}.$$

Параметром подобия здесь также является число Рейнольдса Re.

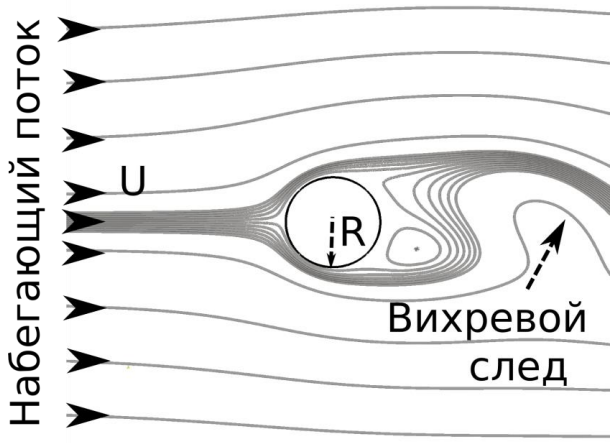


Рис. 2.1. Схема обтекания круглого цилиндра безграничным потоком жидкости

На границе цилиндра задаются условия прилипания:

$$x^2 + y^2 = 1: \; u_x = 0,\; u_y = 0.$$

На бесконечности задается равномерный невозмущенный поток:

$$x \to \infty,\; y \to \infty: \; u_x = 1,\; u_y = 0.$$

В начальный момент времени жидкость в области движется равномерно (такие условия часто называют мгновенным стартом):

$$t = 0: \quad u_x = 1\,, u_y = 0.$$

В OpenFOAM все расчеты проводятся в размерных переменных. Для дальнейшего численного решения примем, что характерная скорость движения потока равна 1 м/с, радиус цилиндра 1 м. Кинематическую вязкость ν будем изменять так, чтобы получить желаемое число Рейнольдса:

$$\nu = \frac{1}{Re}\ \text{м}^2/\text{с}.$$

2.2. Построение расчетной сетки.

Рассмотрим процедуру построения структурированной расчетной сетки. Моделирование безграничного потока будет проводиться за счет граничных условий. Расчеты будут проводиться в конечной области, ограниченной прямоугольным параллелепипедом. В используемой декартовой системе координат ребра параллелепипеда параллельны основным осям, а ось цилиндра перпендикулярна плоскости *xOy*. Считаем, что поток направлен вдоль оси *Ox*. Размеры расчетной области указаны на рис. 2.2.

Для дискретизации расчетной области будем использовать блочные структурированные сетки, построенные с помощью утилиты **blockMesh**, входящей в состав пакета OpenFOAM. Вначале вся область делится на простые геометрические части – блоки, а затем каждый блок разбивается на непересекающиеся ячейки, имеющие форму шестигранников (гексаэдров). Способ деления расчетной области на блоки в плоскости *xOy* представлен на рис. 2.3.

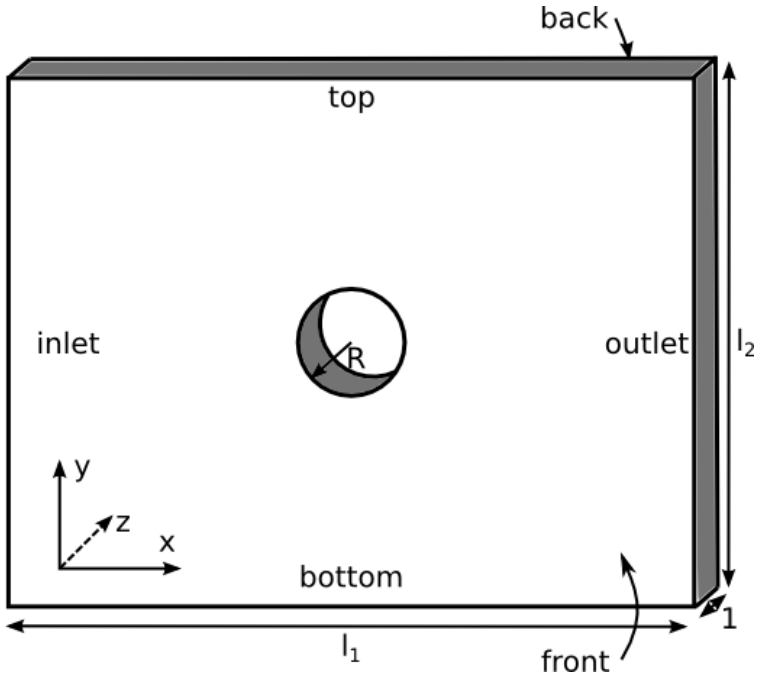
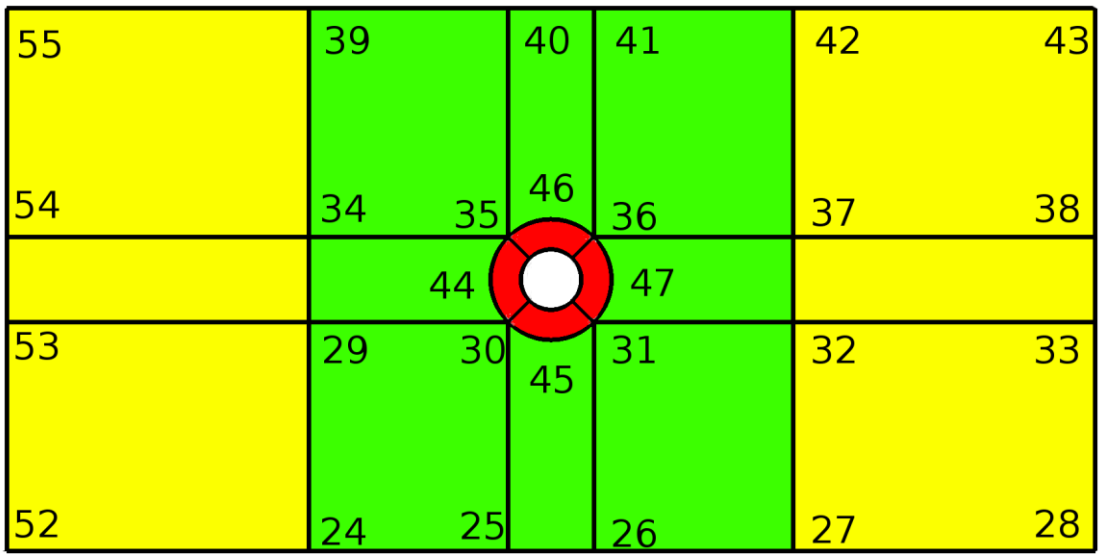


Рис 2.2. Расчетная область

Для текущей расчетной сетки предлагается использовать 18 блоков. По назначению их можно поделить на 3 разные группы (они выделены разными цветами на рисунке 2). Красная группа из 4 блоков служит для создания ортогональных к границе цилиндра слоев расчетной сетки, которые позволят минимизировать погрешности при расчете пограничных эффектов. Этой группе блоков будут использоваться ячейки минимальных

размеров для разрешения пограничного слоя на теле. Вторая группа из 8 блоков, обозначенных зеленым цветом, обеспечивает переход от круглой границы тела к прямоугольной границе внешней области. Сетка в этой группе блоков должна разрешать вихревые структуры, формирующиеся в течении в ближнем следе за телом, поэтому в ней будут использоваться ячейки средних размеров. Последняя группа из 6 блоков, обозначенная желтым цветом, служит для расчета течения в дальнем следе. Размеры левой или правой тройки могут быть уменьшены или увеличены вдоль оси *Ox* в зависимости от направления течения. Например, если течение происходит слева направо, продольный размер блоков левой тройки уменьшается, а правой увеличивается, чтобы захватить максимально возможную длину вихревого следа за телом. В данном примере рассматривается симметричная конфигурация, как для случая, когда направление течения может изменяться. Ячейки сетки в желтой группе блоков будут иметь форму параллелепипедов, что определенной конфигурации решателя может позволить ускорить расчет, кроме того их размеры могут быть достаточно крупными, если не требуется разрешать мелкомасштабные эффекты в следе.



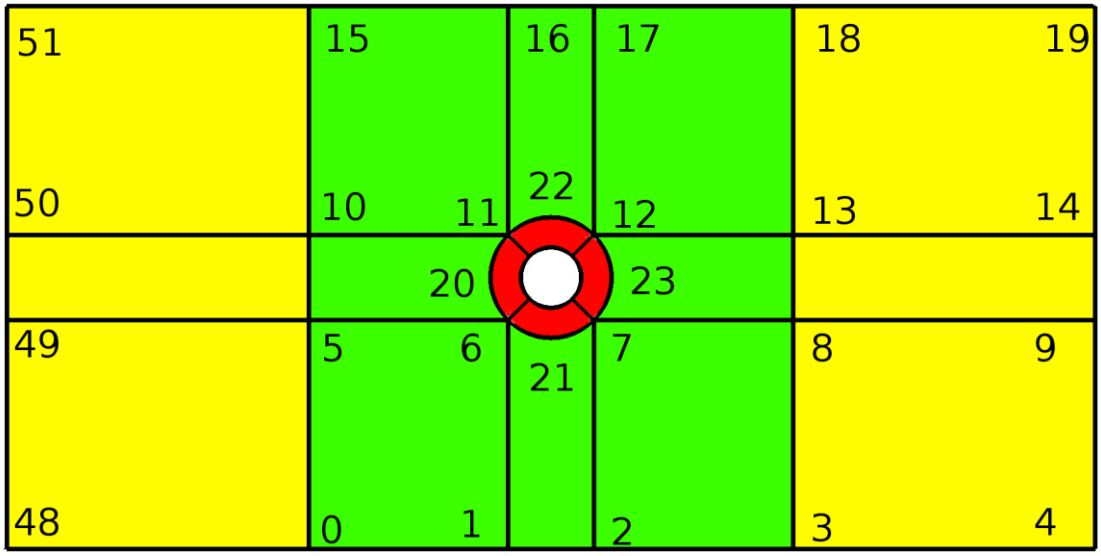


Рис 2.3. Блочная структура расчетной сетки. Числа обозначают номера вершин блоков при $z = 0$ и $z = 1$.

Чтобы сформировать описанные блоки, в **blockMeshDict** необходимо вначале задать образующие вершины. Как для плоских, так и для трехмерных расчетов, в OpenFOAM формируются трехмерные блоки. Каждый блок описывается 8 вершинами. Вершины перечисляются в произвольном порядке. Номера образующих вершин в плоскостях $z = 0$, 1 обозначены на рисунке 2.4, их координаты вносятся в список vertices в **blockMeshDict**

```
vertices
(
    (-8 -9 0) //0
    (-1.4142 -9 0) //1
    (1.4142 -9 0) //2
    (8 -9 0) //3
    (18 -9 0) //4
    …
    (-0.707 -0.707 0) //20
    (0.707 -0.707 0) //21
    (-0.707 0.707 0) //22
    (0.707 0.707 0) //23
    ...
    (-18 1.4142 0) //50
    (-18 9 0) //51
    (-18 -9 1) //52
    (-18 -1.4142 1)//53
    (-18 1.4142 1)  //54
    (-18 9 1) //55
)
```

Чтобы блоки красной группы образовали границу в форме круглого цилиндра необходимо, чтобы ребра блока были не прямыми отрезками (так они формируются по умолчанию) а дугами окружности. Их параметры задаются в секции edges.

Для того чтобы две вершины (например 20 21) соединялись дугой окружности в секции edges добавляется строка arc 20 21 (0 -1 0), где три числа в скобках обозначают x,y,z координаты одной дополнительной точки (помимо координат точек 20 и 21) дуги окружности. Для рассматриваемой геометрии блоков необходимо задать 16 таких дуг:

```
edges
(
// inside edges
    arc 20 21 (0 -1 0)
    arc 21 23 (1 0 0)
    arc 23 22 (0 1 0)
    arc 22 20 (-1 0 0)
    arc 44 45 (0 -1 1)
    arc 45 47 (1 0 1)
    arc 47 46 (0 1 1)
    arc 46 44 (-1 0 1)
    ...
);
```

Далее в секции blocks файла blockMeshDict причисляются (в произвольном порядке) все формируемые блоки и параметры расчетной сетки в них. Для описания каждого блока используется следующий синтаксис (крайний верхний левый блок на рисунке 2.3)

```
hex(50 10 15 51 54 34 39 55)(38 43 1) simpleGrading(0.5 4.0 1.0)
```

Восемь целых чисел в первой скобки это номера образующих его вершин, которые задаются в стандартном порядке. Три числа во второй скобки определяют количество ячеек по осям x,y,z. Здесь по оси *Oz* задается только она ячейка, так как сетка создается для двумерных расчетов. Опция simpleGrading(0.5 4.0 1.0) задает линейное сгущение узлов соответственно по x,y,z осям.

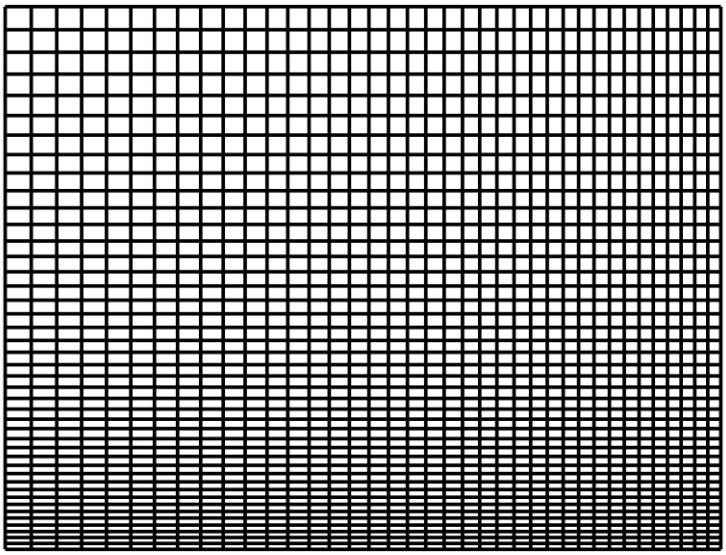


Рис. 2.4. Структура сетки в блоке hex (50 10 15 51 54 34 39 55).

Линейное сгущение узлов будем проводить вдоль осей *Ox* и *Oy*, для увеличения разрешающей способности сетки вблизи цилиндра и уменьшения общего числа ячеек расчетной сетки (см. рисунки 2.4 и 2.5).

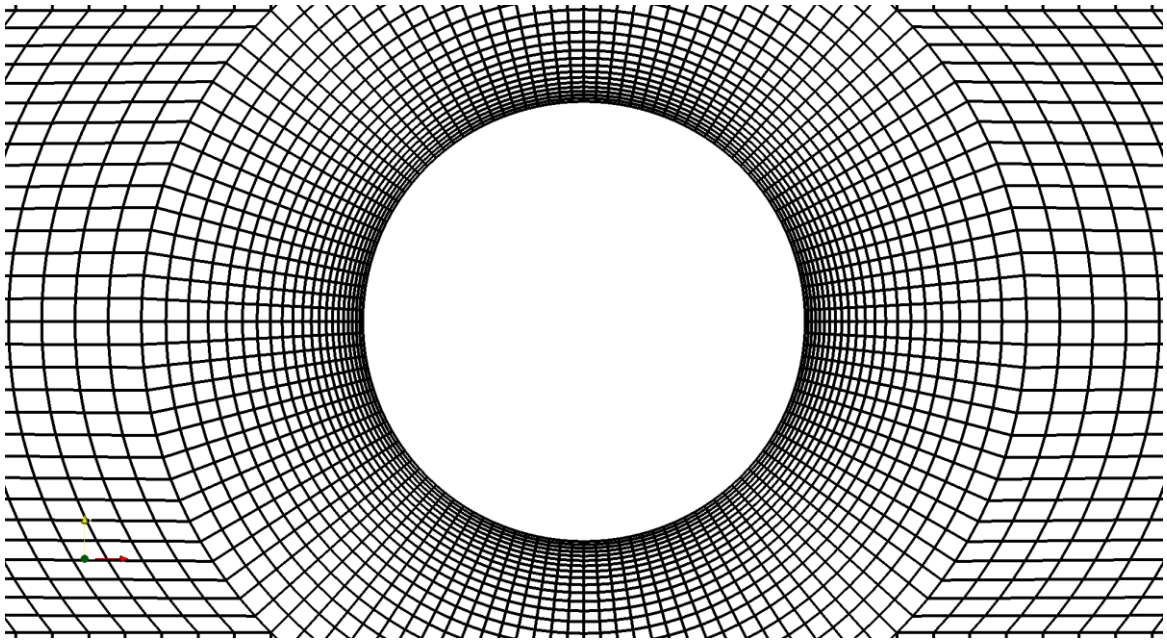


Рис. 2.5 Структура сетки в окрестности цилиндра.

На заключительном этапе описания расчетной сетки необходимо определит границы расчетной области в разделе patches. Границы формируются из внешних сторон блоков. Для примера рассмотрим описание границы inlet (см. рис. 2.2.)

```
inlet
{
    type patch;
```

```
    faces
    (
          (48 52 53 49)
          (49 53 54 50)
          (50 54 55 51)
     );
}
```

Название границы (в данном случае inlet) определяется пользователем (удобно называть границы осмысленно), после директивы type указывается один из возможных типов границы, в данном случае используется тип patch. Это наиболее универсальный вариант. После этого в разделе faces перечисляются все внешние стороны блоков, относящиеся к этой границе. В данном случае их три. Таким образом, необходимо описать все границы.

Отметим, что в представленном примере можно не выделять границы front и back (см. рис. 2.2.). В этом случае они по умолчанию будут отнесены к границе defaultFaces. Используя это название границы (defaultFaces) можно будет также поставить граничные условия на ней.

2.3 Определение качества расчетной сетки

Помимо разрешающей способности, важными характеристиками качества построенных расчетных сеток являются [14]:

- неортогональность,
- скошенность,
- равномерность.

Они оказывают влияние на точность результирующей аппроксимационной схемы и точность конечных результатов. Для определения этих характеристик рассмотрим рис. 2.6, где изображены две соседние ячейки неортогональной расчетной сетки.

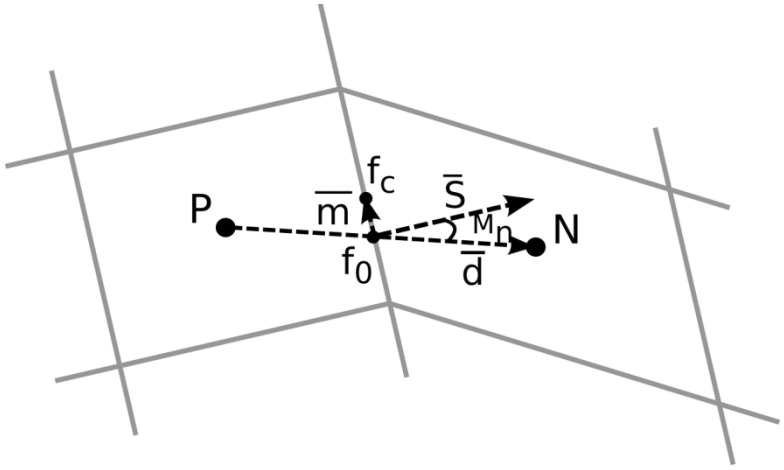


Рис 2.6: Ячейки расчетной сетки

Показатель *неортогональности* M_n измеряется как угол между вектором d, соединяющим центры двух соседних ячеек, и нормалью S к их общей грани (см. рис. 2.6)

$$M_n = \widehat{d\,S}.$$

Имеет оптимальное значение 0 градусов.

Показатель *скошенности* M_s измеряется как отношение длинны вектора m, равной расстоянию между точкой f_0 (образованной пресечением вектора d с общей гранью ячеек) и центром грани f_c, к длине вектора d (см. рис. 2.6)

$$M_s = |m|/|d|.$$

Имеет оптимальное значение 0.

Показатель *равномерности* M_u измеряется как отношение длинны вектора, соединяющего точку f_0 с центром ячейки N, к длине вектора d (см. рис. 2.6)

$$M_s = |f_0N|/|d|.$$

Определить средние и максимальные значения этих характеристик на созданной сетке можно с помощью утилиты **checkMesh**.

2.4 Создание и настройка проекта cylinder

Для того чтобы не писать все конфигурационные файлы с нуля, скопируем в новую папку проект cavity, который мы рассматривали в первой главе. Для создания проекта (папки) cylinder в домашней директории в ОС на базе Linux можно воспользоваться командой:

```
cp -r $FOAM_TUTORIALS/incompressible/icoFoam/cavity/cavity ~/cylinder
```

Далее в папку **cylinder/system** скопируем новый конфигурационный файл blockMeshDict, настройка которого обсуждалась в предыдущем разделе.

В качестве следующего шага необходимо правильно описать граничные и начальные условия в файлах p и U в папке **0**. Рассмотрим описание условий для скорости. Листинг настроек (без заголовочной части) в файле **0/U** представлен ниже.

```
dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (1 0 0);

boundaryField
{
    inlet
    {
        type            fixedValue;
        value           uniform (1 0 0);
    }
    outlet
    {
```

```
        type            zeroGradient;
    }
    bottom
    {
        type            slip;
    }
    top
    {
        type            slip;
    }
    cylinder
    {
       type            fixedValue;
       value           uniform (0 0 0);
    }
    defaultFaces
    {
        type            empty;
    }
}
```

Обозначим соответствие между граничными и начальными условиями математической модели, описанными в разделе 2.1, и условиями численной модели. Начальные условия, определяющие невозмущенный поток со скоростью 1 м/с, направленный параллельно оси Ox, задаются в строке

```
internalField uniform (1 0 0);
```

На границе outlet, через которую жидкость вытекает из области, задаются «мягкие» условия вида

$$\frac{\partial \mathrm{u_x}}{\partial \mathrm{x}} = 0, \qquad \frac{\partial \mathrm{u_y}}{\partial \mathrm{x}} = 0.$$

В OpenFOAM они описываются как

```
    type            zeroGradient;
```

Такие условия используются для корректного описания вихревого следа за телом, где поток около границы престает быть равномерным.

На границе defaultFaces ставятся пустые граничные условия

```
    type            empty;
```

так как мы описываем двумерную модель течения.

На границе цилиндра ставятся условия прилипания:

```
    type            fixedValue;
    value           uniform (0 0 0);
```

Далее представим листинг настроек (без заголовочной части) в файле 0/p, где задаются начальные и граничные условия для давления.

```
dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 0;
```

```
boundaryField
{
    inlet
    {
        type            zeroGradient;
    }
    outlet
    {
        type            fixedValue;
        value           uniform 0;
    }
    bottom
    {
        type            zeroGradient;
    }
    top
    {
        type            zeroGradient;
    }
    cylinder
    {
        type            zeroGradient;
    }
    defaultFaces
    {
        type            empty;
    }
}
```

В начальный момент времени давление во всей области равно нулю:

```
internalField   uniform 0;
```

На границах inlet bottom, top, cylinder нормальная производная от давления задается равной нулю:

```
type            zeroGradient;
```

На границе defaultFaces ставятся пустые граничные условия:

```
type            empty;
```

На выходной границе outlet давление задается равным нулю:

```
type            fixedValue;
value           uniform 0;
```

После того как все граничные условия заданы, можно сгенерировать расчетную сетку с помощью утилиты blockMesh. Далее можно попробовать установить подходящий шаг по времени в файле controlDict, определить значение кинематической вязкости nu в фале transportProperties и запустить решатель icoFoam. Формально даже если расчет будет выполнен (решение вообще может разойтись), его результаты будут очень не точными.

Чтобы научиться правильно конфигурировать расчетную схему для произвольного случая, рассмотрим далее какие расчетные алгоритмы используются в решателе icoFoam.

2.5 Дискретизация уравнений в OpenFOAM

Дискретизация системы уравнений движения в пакете OpenFOAM проводится по методу конечных объемов (FVM) в декартовой системе координат. Для этого дискретные значения составляющих скорости и дискретные давления локализуются в центрах ячеек построенной расчетной сетки.

Для произвольной ячейки сетки с объемом V система уравнений записывается в следующей интегральной форме:

$$\int_V \frac{\partial U}{\partial t} dV + \int_V \nabla \cdot (UU) dV = -\int_V \nabla p dV + \int_V \Delta U dV , \qquad (2.1)$$

$$\int_V \nabla \cdot U dV = 0 .$$

Первое слагаемое системы аппроксимируется в центре ячейки как произведение среднего значения подынтегральной функции на объем ячейки V. Для вычисления остальных объемных интегралов по контрольному объему V системы уравнений (2.1) используется общая процедура Гаусса, согласно которой осуществлялся переход от объемного интеграла к поверхностному. Далее поверхностные интегралы представляются в виде суммы интегралов по граням ячейки и приближенно вычисляются по формуле средних прямоугольников. После этого, полудискретная система уравнений для произвольной ячейки представляется в виде:

$$\left(\frac{\partial U}{\partial t}\right)_P V + \sum_f S_f \cdot (UU)_f = -\sum_f S_f p_f + \nu \sum_f S_f \cdot (\nabla U)_f , \qquad (2.2)$$

$$\sum_f S_f \cdot (U)_f = 0 .$$

Здесь индекс f указывает на то, что переменная или градиент определены на грани ячейки, P – в центре ячейки, а S_f определяется как вектор ортогональный к грани ячейки и по модулю равный площади этой грани.

Для линеаризации системы (2.2), конвективные слагаемые представляются в следующей форме:

$$\sum_f S_f \cdot (UU)_f = \sum_f F (U)_f ,$$

где массовый поток F через грань с индексом f считается известным. Обновление F связано с итерационной процедурой метода решения задачи.

Значения функции и нормальных градиентов на поверхности ячеек в системе (2.2) для внутренних ячеек расчетной области интерполируются из значений функции в

центрах соседних ячеек. Для интерполяции переменных в OepnFOAM можно использовать разные схемы. Схемы интерполяции указываются в файле **fvSchemes.**

Для аппроксимации градиента давления традиционно применяется линейная интерполяция. Значение p_f на грани f между двумя ячейками с центрами P и N (рис. 1.) находится по формуле:

$$p_f = p_P f_x + p_N(1 - f_x),$$

где f_x – интерполяционный фактор. Порядок точности используемой аппроксимации обусловлен локальными характеристиками сетки. Он понижается до первого в случае локальной скошенности сетки, т. к. интерполируемое значение определяется не в центре грани. В остальных случаях аппроксимация имеет второй порядок точности. В файле **fvSchemes** эта аппроксимационная схема для градиента давления задается как

```
gradSchemes
{
    grad(p)         Gauss linear;
}
```

Для интерполяции переменных в конвективных слагаемых в примере cavity также использовалась линейная интерполяция (Gauss linear). Однако, как хорошо известно, такой выбор приведет к неустойчивости всей расчетной схемы уже при умеренных числах Рейнольдса. В расчетах рекомендуется использовать нелинейные TVD и NVD схемы. Далее рассмотрим использование нелинейной NVD (normalised variable diagram) схемы «Gamma». Применение этой схемы позволяет обеспечить устойчивость численной задачи, внося минимальную численную диффузию. В качестве схемы высокого порядка точности в «Gamma» используются линейная интерполяция, в качестве безусловно устойчивой схемы низкого порядка – противопоточная схема upwind. Вычисление дискретных составляющих скорости $(u_f, v_f) = U_f$ на грани с индексом f для $F > 0$ производится по формулам:

$$\begin{cases} u_f = u_P, & \tilde{u}_P \le 0, \tilde{u}_P \ge 1 \\ u_f = u_P f_x + u_N(1 - f_x), & \beta_m \le \tilde{u}_P < 1 \\ u_f = (1 - \gamma_u(1 - f_x))u_P + \gamma_u(1 - f_x)u_N, & 0 < \tilde{u}_P < \beta_m \end{cases}$$

$$\begin{cases} v_f = v_P, & \tilde{v}_P \le 0, \tilde{v}_P \ge 1 \\ v_f = v_P f_x + v_N(1 - f_x), & \beta_m \le \tilde{v}_P < 1 \\ v_f = (1 - \gamma_v(1 - f_x))v_P + \gamma_v(1 - f_x)v_N, & 0 < \tilde{v}_P < \beta_m \end{cases}$$

Здесь $\tilde{u}_P = 1 - ((\nabla u)_f \cdot d)/(2(\nabla u)_P \cdot d)$, $\tilde{v}_P = 1 - ((\nabla v)_f \cdot d)/(2(\nabla v)_P \cdot d)$ – нормализованные переменные, d – вектор, направленный из точки P в точку N, $\gamma_u = \tilde{u}_P / \beta_m$, $\gamma_v = \tilde{v}_P / \beta_m$ –

факторы смешивания, $1/10 < \beta_m < 1/2$ – предопределенная константа метода. Выбор больших значений β_m из указанного диапазона обеспечивает наилучшую устойчивость схемы, меньших – увеличивает точность интерполяции. В данной работе использовалось значение $\beta_m = 0.25$. В случае противоположного направления потока *F (F*<0*)* формулы изменяются соответствующим образом. В файле **fvSchemes** эта аппроксимационная схема для интерполяции переменных в конвективных слагаемых задается как

```
divSchemes
{
    div(phi,U)      Gauss GammaV 0.5;
}
```

В диффузионных слагаемых при дискретизации оператора Лапласа необходимо вычислять нормальные градиенты скорости на поверхности ячейки. На ортогональных участках сетки, где вектор *S* параллелен вектору *d*, они вычисляются из значений скорости в центрах соседних ячеек по симметричной схеме второго порядка (такая схема используется практически во всех расчетах):

$$S \cdot (\nabla u)_f = |S| \frac{u_N - u_P}{|d|}. \quad (2.3)$$

На неортогональных участках сетки скалярное произведение $S \cdot (\nabla u)_f$ представляется в виде суммы двух слагаемых:

$$S \cdot (\nabla u)_f = l \cdot (\nabla u)_f + k \cdot (\nabla u)_f. \quad (2.4)$$

Первое из слагаемых отвечает за ортогональный вклад, второе – за неортогональную поправку, при этом для векторов *k* и *l* выполняется соотношение

$$S = k + l. \quad (2.5)$$

Ортогональный вклад вычисляется по формуле (2.3), где вместо вектора *S* используется вектор *l*, параллельный вектору *d*, длинна которого определяется по формуле:

$$l = \frac{d}{d \cdot S} |S|^2.$$

Неортогональная поправка вычисляется следующим образом: вектор *k* находится из соотношения (2.5), а значение градиента $(\nabla u)_f$ на грани ячейки интерполируется из значений градиентов в центрах соседних ячеек:

$$(\nabla u)_f = (\nabla u)_P f_x + (\nabla u)_N (1 - f_x).$$

Для интерполяции слагаемого $(\nabla v)_f$ применяется аналогичный подход.

В файле **fvSchemes** в примере **cavity** выбрана схема для ортогональной сети

```
laplacianSchemes
{
    default         Gauss linear orthogonal;
```

```
}
```

В этом случае нормальные градиенты скорости вычисляются по формуле (2.3). Для сетки, сгенерированной в примере **cylinder**, необходимо использовать схему с коррекцией на неортогональность в форме (2.4). Такая схема задается как

```
laplacianSchemes
{
    default         Gauss linear corrected;
}
```

Аппроксимация диффузионных слагаемых имеет второй порядок точности для равномерных участков сеток, на неравномерных участках порядок понижается до первого.

Для дискретизации системы (2.2) по времени будем использовать неявную схему Эйлера:

$$\left(\frac{U_P^n - U_P^0}{\tau}\right)V + \sum_f FU_f^n = -\sum_f S_f p_f^n + \nu \sum_f S_f \cdot (\nabla U)_f^n,$$

$$\sum_f S_f \cdot U_f^n = 0, \ \sum_f S_f \cdot U_f^0 = 0.$$

Здесь верхние индексы «*o*» и «*n*» указывают на использование переменной со старого или нового временного слоя соответственно, τ – шаг по времени. В файле **fvSchemes** эта схема задается как

```
ddtSchemes
{
    default         Euler;
}
```

Хотя сама схема является (безусловно) устойчивой, но для минимизации эффектов связанных с аппроксимацией первого порядка точности, шаг по времени в расчетах рекомендуется выбирать из условия $C_o^{\max} < 0.1$ – максимальное число Куранта не превышает значения 0.1. Напомним, что число Куранта определяется по формуле:

$$C_o = \frac{|U_P|\tau}{\delta},$$

где $|U_P|$ – модуль скорости в ячейке, δ – размер ячейки в направлении вектора скорости.

2.6 Аппроксимация граничных условий в OpenFOAM

Используемые в данной задаче граничные условия делятся на два вида: условия Дирихле и условия Неймана. Дискретизация уравнений движения во всех граничных ячейках сводится к рассмотрению этих двух различных случаев.

При аппроксимации конвективных слагаемых на границе с условиями Дирихле определены значения скоростей u_b, v_b, что без дополнительной интерполяции позволяет вычислить соответствующие граничные компоненты:

$$Fu_b,\ Fv_b.$$

Аналогично для аппроксимации градиента давления в случае граничных условий Дирихле используется значение p_b на грани ячейки.

Для аппроксимации диффузионных слагаемых на границах с условиями Дерихле необходимо вычислять нормальные градиенты скорости. Если используемые сетки всюду ортогональны границам расчетной области (т.е. вектор *d*, направленный из центра любой граничной ячейки в центр ее грани, принадлежащей границе области, параллелен вектору нормали к этой грани), соответствующие нормальные градиенты могут быть вычислены по формулам:

$$S\cdot(\nabla u)_b = |S|\frac{u_b - u_P}{|d|},\ S\cdot(\nabla v)_b = |S|\frac{v_b - v_P}{|d|}.$$

В файле **fvSchemes** для ортогонального случая задается следующая конфигурация

```
snGradSchemes
{
    default         orthogonal;
}
```

В других случаях надо проводить коррекцию на неортогональность, для этого в **fvSchemes** блок snGradSchemes определяется следующим образом

```
snGradSchemes
{
    default         corrected;
}
```

Для случая граничных условий Неймана значения скоростей u_b, v_b в центре грани ячейки, необходимые для аппроксимации конвективных слагаемых, могут быть вычислены по следующим формулам:

$$u_b = u_P + d\cdot(\nabla u)_b,\ v_b = v_P + d\cdot(\nabla v)_b,$$

где $d\cdot(\nabla u)_b$, $d\cdot(\nabla v)_b$ – известные нормальные градиенты.

По аналогичной формуле, в случае граничных условий Неймана, вычисляется значение p_b на грани ячейки для аппроксимации градиента давления:

$$p_b = p_P + d\cdot(\nabla p)_b.$$

При аппроксимации диффузионных слагаемых на границах с условиями Неймана значения нормальных градиентов скорости известны:

$$S\cdot(\nabla u)_b,\ S\cdot(\nabla v)_b.$$

2.7. Итерационный алгоритм решения

Дальнейшее решение дискретной задачи в программе icoFoam основано на подходе «segregated approach» раздельного решения уравнений для скорости и давления. Для соблюдения согласованности аппроксимационных схем, уравнение для давления выражается из дискретизованных уравнений движения и неразрывности.

Представим уравнение движения в следующем полудискретном виде:

$$a_P U_P^n = -\sum_N a_N U_N^n + \frac{U_P^o}{\tau} - \nabla p \,. \quad (2.6)$$

Недискретизованным здесь остается градиент давления, который определен как произведение среднего значения подынтегральной функции на объем ячейки и локализован в центре ячейки. Дискретизация других слагаемых проведена согласно представленным выше схемам, коэффициенты при соответствующих компонентах скоростей U_P^n и U_N^n содержатся в диагональных матрицах a_P и a_N размерности 2x2. В приведенной форме (2.6), все слагаемые уравнения разделены на объем ячейки V.

Выразим из уравнения (2.6) вектор скорости в центре расчетной ячейки U_P^n:

$$U_P^n = a_P^{-1} H(U) - a_P^{-1} \nabla p \,, \quad (2.7)$$

где за *H(U)* обозначен оператор вида:

$$H(U) = -\sum_N a_N U_N^n + \frac{U_P^o}{\tau} \,.$$

Полученный вектор скорости можно интерполировать на грань ячейки следующим образом:

$$U_f = \left(a_P^{-1} H(U)\right)_f - \left(a_P^{-1}\right)_f (\nabla p) f \quad , (2.8)$$

где $\left(a_P^{-1}\right)_f$ – обозначает интерполяцию соответствующих коэффициентов матрицы.

Подставляя (2.8) в уравнение неразрывности, получим дискретное уравнение для давления:

$$\sum_f S_f \cdot \left(a_P^{-1}\right)_f (\nabla p)_f = \sum_f S_f \cdot \left(a_P^{-1} H(U)\right)_f \,.$$

Аппроксимация оператора Лапласа здесь проводится так же, как и в диффузионном члене.

Результирующую систему уравнений с выделенным уравнением для давления запишем в виде:

$$a_P U_P = H(U) - \sum_f S_f p_f \quad (2.9)$$

$$\sum_f S_f \cdot \left(\frac{1}{a_P} \nabla p\right)_f = \sum_f S_f \cdot \left(\frac{H(U)}{a_P}\right)_f \,. \quad (2.10)$$

Задача (2.9)-(2.10) решается с помощью алгоритма PISO (Pressure Implicit Splitt Operator) (см. подробнее [Issa 1986, Ferziger 2002]). Итерационная процедура алгоритма основана на последовательном решении уравнений для скоростей и давления. Согласно используемой реализации вычисление неизвестных полей на новом временном слое проводится по следующей схеме:

0. По полю скоростей *U*, известному с предыдущего временного слоя, вычисляются потоки *F*.

1. Проводится неявное вычисление предиктора нового поля скоростей. Для этого решается система, образованная из уравнений (2.9), где поле давления берется со старого временного слоя:

$$a_P U_P^n + \sum_N a_N U_N^n = \frac{U_P^o}{\tau} - \sum_f S_f p_f^o$$

(В общем случае найденный предиктор не удовлетворяет уравнению неразрывности).

2. Проводится k-циклов коррекции:

2.1. По найденному приближению поля скоростей вычисляются значения оператора *H(U)*:

$$H(U) = -\sum_N a_N U_N^n + \frac{U_P^o}{\tau},$$

и рассчитываются значения U_P^*:

$$U_P^* = a_P^{-1} H(U),$$

представляющие собой новые значения скоростей без учета давления (см. (2.7)).

2.2. Найденные скорости интерполируются на границы $U_P^* \to U_f^*$, после чего вычисляются соответствующие значения потоков:

$$F^* = S_f \cdot U_f^* + \frac{\left(F^o - S_f \cdot U_P^o\right)}{\tau}\left(a_P^{-1}\right)_f .$$

Второе слагаемое в правой части выражения служит для замены слагаемого $\frac{S_f \cdot U_f^o}{\tau}\left(a_P^{-1}\right)_f$ оператора $\left(a_P^{-1} H(U)\right)_f$, полученного в результате интерполяции, слагаемым $\frac{F^o}{\tau}\left(a_P^{-1}\right)_f$ – известным с предыдущего временного слоя, для которого в точности выполняется уравнение неразрывности.

2.3. Рассчитывается новое поле давления. Для этого решается система уравнений вида:

$$\sum_f S_f \cdot \left(a_P^{-1}\right)_f (\nabla p)_f = \sum_f F_f^* .$$

2.4. Найденные давления используются для коррекции потоков:

$$F = F^* + S_f \cdot \left(a_P^{-1}\right)_f (\nabla p)_f$$

и поля скоростей:

$$U_P^n = U_P^* + a_P^{-1}(\nabla p)$$

2.5. Выполняется коррекция граничных условий.

В представленном алгоритме следует отметить несколько важных моментов:

- При малых числах Куранта связи между скоростью и давлением в уравнениях движения считаются более сильными, чем нелинейные связи на фиксированном временном слое. В связи с этим матрицы коэффициентов a_P, a_N, зависящие от значений потоков *F*, обновляются на каждой итерации по времени только на этапе предиктора.
- В алгоритме решаются две линейные задачи: первая для вычисления предиктора поля скоростей (шаг 1), вторая для расчета нового поля давления в цикле коррекций (шаг 2.3). Все остальные операции выполняются по явным формулам.
- При решении линейных задач для обеспечения диагонального преобладания матриц соответствующих систем используется отложенная коррекция [15]. Согласно этому методу конвективные слагаемые представляются в виде суммы:

$$\sum_f F(U)_f = \sum_f F(U)_f^{up} + \left(\sum_f F(U)_f - \sum_f F(U)_f^{up} \right).$$

Первое слагаемое в правой части, полученное в результате дискретизации с использованием противопоточной схемы «upwind», не нарушает диагонального преобладания матрицы и поэтому используется для определения коэффициентов системы. Оставшийся комплекс, представляющий поправку более высокого прядка, определяется явным образом и добавляется в столбец свободных членов.

Аналогичный подход используется для дискретизации диффузионных членов на неортогональных участках сетки. Как было отмечено ранее, дискретизация этих слагаемых представляется в виде:

$$\sum_f S_f \cdot (\nabla u)_f = \sum_f l(\nabla u)_f + \sum_f k \cdot (\nabla u)_f \ .$$

Ортогональный вклад используется для определения коэффициентов системы, неортогональная поправка задается явным образом и добавляется в столбец свободных членов.

Для решения линейных систем с необходимой точностью, с учетом вышеописанного представления слагаемых, требуется несколько итераций: решение, полученное на каждой итерации, используется для обновления добавочных слагаемых в столбце свободных членов. В случае системы уравнений для давления такой цикл коррекций реализуется на шаге 2.3. Точное решение задачи здесь обеспечивает консервативность результирующих потоков.

Основные параметры алгоритма PISO определяются в файле **fvSolution** в модуле PISO

```
PISO
{
    nCorrectors     3;
    nNonOrthogonalCorrectors 3;
    pRefCell        0;
    pRefValue       0;
}
```

Параметр nCorrectors задает число коррекций k (шаг 2), параметр nNonOrthogonalCorrectors задает число неортогональных коррекций (шаг 2.3).

Методы решения системы уравнений для давления (шаг 2.3) системы уравнений для скоростей (шаг 1) так же задаются файле **fvSolution**. Стандартные решатели СЛАУ, используемые в примере cavity, являются неэффективными для сеток с большим числом узлов. Для решения системы уравнений для давления рекомендуется применять метод сопряженных градиентов PCG с геометро-алгебраическим многосеточным предобуславливателем GAMG (Geometric agglomerated algebraic multigrid solver). В реализации GAMG для сглаживания можно использовать метод Гауса-Зейделя, для агломерации ячеек сетки применять faceAreaPair алгоритм. Для решения системы уравнений для скоростей можно использовать метод би-сопряженных градиентов PBiCG с предобуславливателем основанным на неполной LU факторизации (Diagonal incomplite-LU). Соответствующую конфигурацию можно задать следующим образом

```
    p
    {
            solver           PCG;
            preconditioner
            {
               preconditioner GAMG;
               tolerance       1e-05;
               relTol          0;
               smoother         GaussSeidel;
               cacheAgglomeration true;
               nCellsInCoarsestLevel 50;
               agglomerator    faceAreaPair;
               mergeLevels     1;
            }
            minIter          0;
            maxIter          100;
            tolerance        1e-5;
            relTol           0.0;
    };

    U
```

```
{
     solver          PBiCG;
     preconditioner  DILU;
     tolerance       1e-08;
     relTol          0.1;
}
```

2.8. Запуск расчета в параллельном режиме

Расчеты на сетках с большим количеством узлов могут выполняться очень продолжительное время (дни, недели или даже месяцы) при запуске в обычном (последовательном) режиме. Для сокращения времени расчета можно запускать решатели OpenFOAM в параллельном режиме.

В OpenFOAM для распараллеливания вычислений используется метод декомпозиции расчетной области (domain decomposition), в рамках которого геометрия и связанные ней поля разбиваются на подобласти, численное решение уравнений в которых проводится на разных ядрах процессора и/или на разных узлах вычислительной системы. Между такими подобластями, безусловно, остаются связи, и проводится обмен данными. Взаимодействие между разными ядрами/процессорами при этом реализуется с помощью программного интерфейса для передачи информации MPI.

Первый шаг, для запуска решателя в параллельном режиме, состоит в декомпозиции расчетной области. Эту задачу решает утилита **decomposePar**. Для работы с этой утилитой надо создать конфигурационный файл decomposeParDict, который необходимо поместить в директорию **system**. Пример такого конфигурационного файла со всеми возможными опциями можно найти в директории исходного кода OpenFOAM, по адресу $FOAM_UTILITIES/parallelProcessing/decomposePar/decomposeParDict. Для данной задачи воспользуемся простейшими настройками. После стандартного заголовка, определим в файле decomposeParDict следующие параметры

```
numberOfSubdomains 4;

method          simple;

simpleCoeffs
{
    n               (4 0 0);
    delta           0.001;
}

  distributed     no;

 roots           ( );
```

Параметр numberOfSubdomains задает число подобластей, на которые будет разбита исходная расчетная область. Далее с помощью определения параметра method задается

метод декомпозиции, в данном примере используется метод simple. Его параметры задаются в модуле simpleCoeffs.

Параметр n в модуле simpleCoeffs задается тремя числами (*rx ry rz*), они определяют количество разбиений в направлении осей *Ox*, *Oy* и *Oz* соответственно. При этом должно выполняться равенство $rx \cdot ry \cdot rz =$ numberOfSubdomains. В данном примере параметр n задается как (4 0 0). Это означает, что все разбиения проводятся по оси *Ox*.

После того как проект полностью настроен, запустим (в директории проекта) утилиту **decomposePar.** В результате в директории проекта появятся новые папки processor0, processor1, processor2 …, которые содержат всю необходимую информацию о подобластях.

Для проведения параллельных расчетов можно использовать разные версии MPI. Информацию о том, как настроить нужную для работы в OpenFOAM можно найти в руководстве для пользователя пакета. Обычно (при использовании Open MPI в ОС на базе Linux) параллельный расчет можно запустить командой (в директории проекта)

```
mpirun -np 4 icoFoam –parallel 2>&1 >output.log &
```

В этом случае расчет будет запущен параллельно (опция –parallel) на 4 ядрах (опция -np 4) в фоновом режиме (опция &), все выходные данные будут записываться в output.log файл (опция 2>&1 >output.log).

После завершения параллельного расчета необходимо соединить данные, полученные в разных подобластях, это можно сделать с помощью команды

```
reconstructPar
```

2.9 Определение гидродинамических сил, действующих на цилиндр

Одной из важнейших интегральных характеристик в задачах внешнего обтекания, является гидродинамическая сила, действующая на тело со стороны жидкости. Для расчета гидродинамических сил в проект необходимо добавить дополнительные настройки. Настройка функций пост-обработки (фактически силы нужно определять на каждом шаге после расчета всех полей) проводится в файле controlDict. В данном примере будут вычисляться не сами силы, а безразмерные коэффициенты сил:

$$C_D = \frac{F_x}{U^2 2RL\rho/2}, \qquad C_L = \frac{F_y}{U^2 2RL\rho/2}.$$

Здесь C_D – коэффициент сопротивления, C_L – коэффициент подъемной силы.

Для их расчета необходимо дописать в конец файла controlDict новый модуль

```
functions
{
  forces
  {
```

```
    type         forceCoeffs;
    functionObjectLibs ( "libforces.so" );
    outputControl timeStep;
    outputInterval 1;
    patches    (cylinder);
    pName        p;
    UName        U;
    log          true;
    rho      rhoInf;
    rhoInf       1;
    CofR         ( 0 0 0 );
    liftDir      ( 0. 1.0 0 );
    dragDir      ( 1.0 0. 0 );
    pitchAxis    ( 0 0 1 );
    magUInf      1.0;
    lRef         1;
    Aref         2;
}
```

Параметр type задает тип вызываемой функции пост-обработки, в данном примере это forceCoeffs. Далее указывается библиотека (параметр functionObjectLibs), в которой эта функция определена. Параметры outputControl и outputInterval (в новых версиях OpenFOAM параметры называются writeControl и writeInterval, но оставлена совместимость со старыми версиями) определяют временной интервал, через который необходимо проводить сохранение данных. Используемые здесь значения указывают на то, что запись надо проводить на каждом шаге по времени. Параметр patches задает границы, на которых необходимо рассчитывать значения коэффициентов сил. Параметр CofR задает точку (ее координаты), относительно которой необходимо рассчитывать момент сил. Параметр pitchAxis задает ось вращения (с помощью вектора), относительно которой необходимо рассчитывать момент сил. Параметры liftDir и dragDir задают направления (с помощью векторов) силы сопротивления (для расчета коэффициента сопротивления C_D) и подъемной силы (для расчета коэффициента подъемной силы C_L). Параметр magUInf задает характерную скорость (в нашем случае она равна единице), параметр Aref – максимальную площадь сечения тела в направлении liftDir или dragDir (для цилиндра они равны $2RL$, где L длина цилиндра в направлении оси Oz).

После запуска расчета файл с коэффициентами сил появится в директории проекта по адресу ./postProcessing/forces/0/forceCoeffs.dat.

2.10 Моделирование течения и анализ результатов

После успешного выполнения всех пунктов настройки можно перейти к решению задачи. Из экспериментальных исследований известно, что режимы течения около цилиндра изменяются в зависимости от диапазона чисел Рейнольдса.

- При Re < 2 (число Рейнольдса определено по радиусу цилиндра) около цилиндра наблюдается стационарный безотрывной режим обтекания.
- При 2 < Re < 20 за цилиндром образуются присоединенные вихри, режим обтекания (после фазы установления) остается стационарным.
- При Re > 20 течение становится периодическим.
- При Re > 45 за цилиндром возникает вихревая дорожка Кармана.
- При Re ~ 100 в следе за цилиндром развивается трехмерная неустойчивость.
- При Re > 200 трехмерные эффекты начинают оказывать видимое влияние на основные гидродинамические характеристики.
- С дальнейшим ростом числа Рейнольдса течение становится турбулентным.

Таким образом, построенную двумерную численную модель можно использовать для описания течения при Re<200. При Re>200 2D модель остается валидной на начальных отрезках времени (после начала движения), когда трехмерные течения в следе еще не успевают развиться.

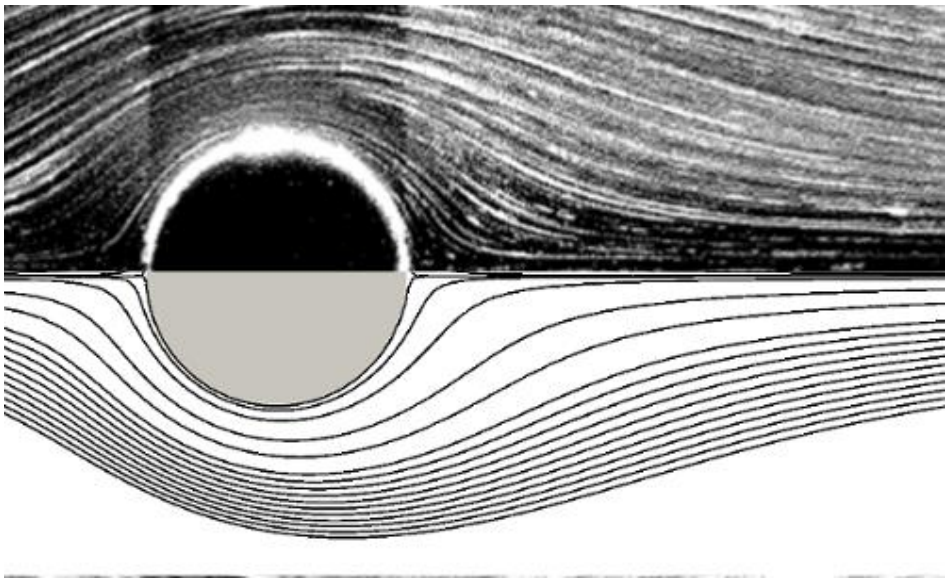


Рис. 2.7. Стационарный безотрывной режим обтекания круглого цилиндра при Re=0.8. Эксперимент и численное моделирование в OpenFOAM.

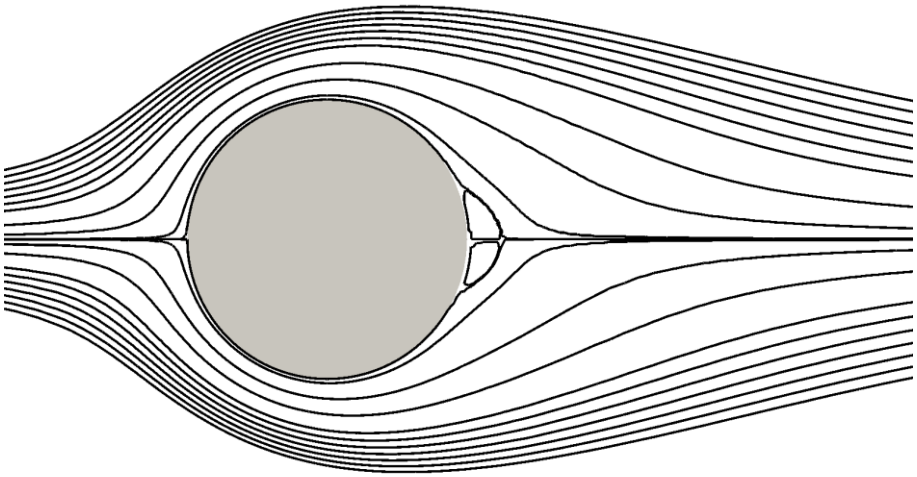


Рис. 2.8. Стационарный режим обтекания круглого цилиндра с формированием присоединенных вихрей при Re=4. Численное моделирование в OpenFOAM.

Проведем моделирование движения жидкости в разных режимах течения. На рис. 2.7, 2.8 представлены линии тока установившихся течений при Re=0.8, 4, 13, полученные в ходе моделирования, и экспериментально наблюдаемые картины течения при Re=0.8, 13. Как можно видеть при Re=0.8 линии тока плавно огибают цилиндр без отрыва от его поверхности. При Re=4 за цилиндром образуются два малых присоединенных вихря. Эти вихри существенно увеличиваются в размерах при Re=13. Численная модель во всех представленных случаях полностью описывает размер и положение вихрей, наблюдаемые в экспериментах.

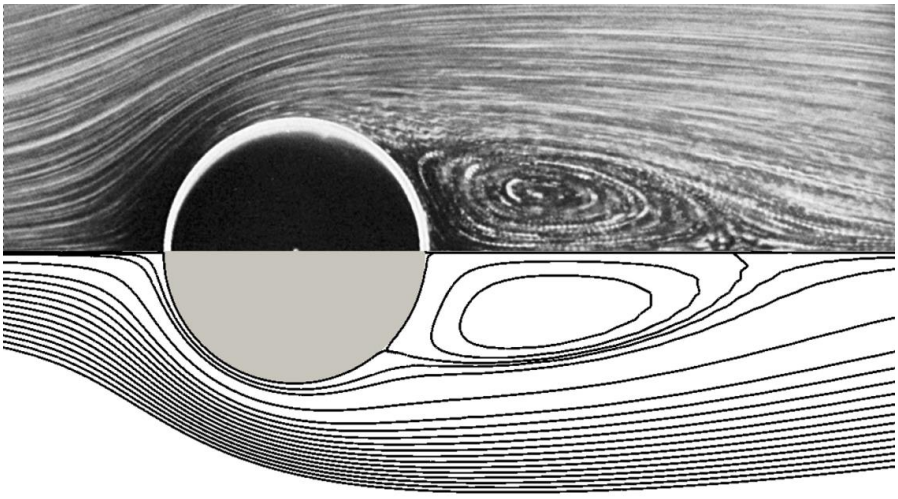


Рис. 2.9. Стационарный режим обтекания круглого цилиндра с формированием присоединенных вихрей при Re=13. Эксперимент и численное моделирование в OpenFOAM.

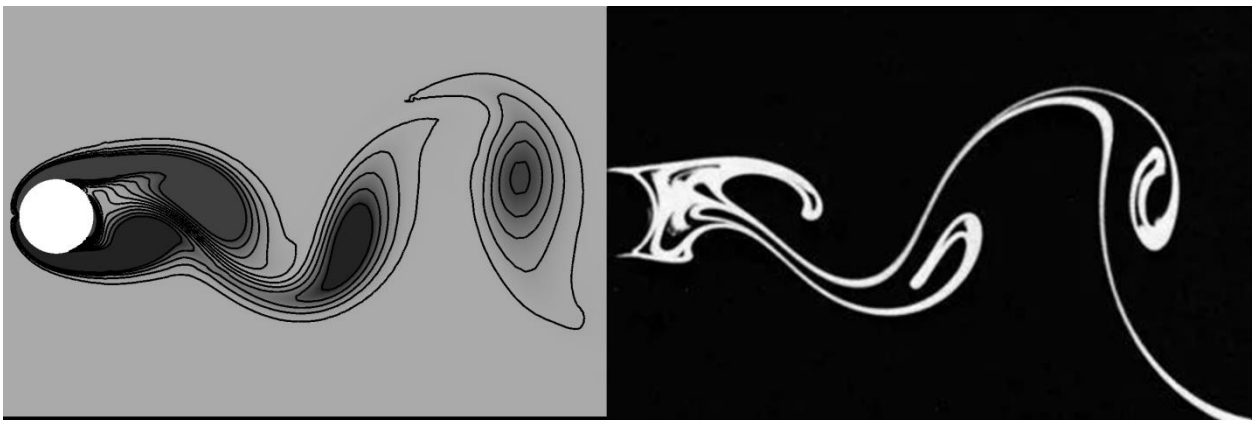


Рис. 2.10. Установившийся периодический режим обтекания круглого цилиндра с формированием дорожки Кармана при Re=70. На левом изображении показаны изолинии завихренности (момент времени t=294), полученные с помощью численного моделирования. На правом представлена экспериментальная визуализация течения с помощью краски.

На рис. 2.10 изображен нестационарный режим обтекания при Re=70 с формированием дорожки Кармана. Визуализация картины течения, полученной в численном расчете, проведена с помощью изолиний завихренности. Визуализация эксперимента проведена с помощью добавления в течение краски (streakline pattern). Волнообразный характер следа хорошо виден на обоих изображениях, кроме того области высокой концентрации краски хорошо согласуются с границами областей с максимальной (по модулю) завихренностью.

На рис. 2.11 представлена визуализация течения (с помощью линий тока) в окрестности цилиндра при относительно большом числе Рейнольдса Re=1500 после мгновенного старта. Как можно видеть, даже в этом случае построенная численная модель достаточно хорошо описывают гидродинамику (включая структуру и расположение присоединенных вихрей) в начальные моменты времени. Однако область согласования (по времени) при таких числах Рейнольдса будет весьма ограниченной.

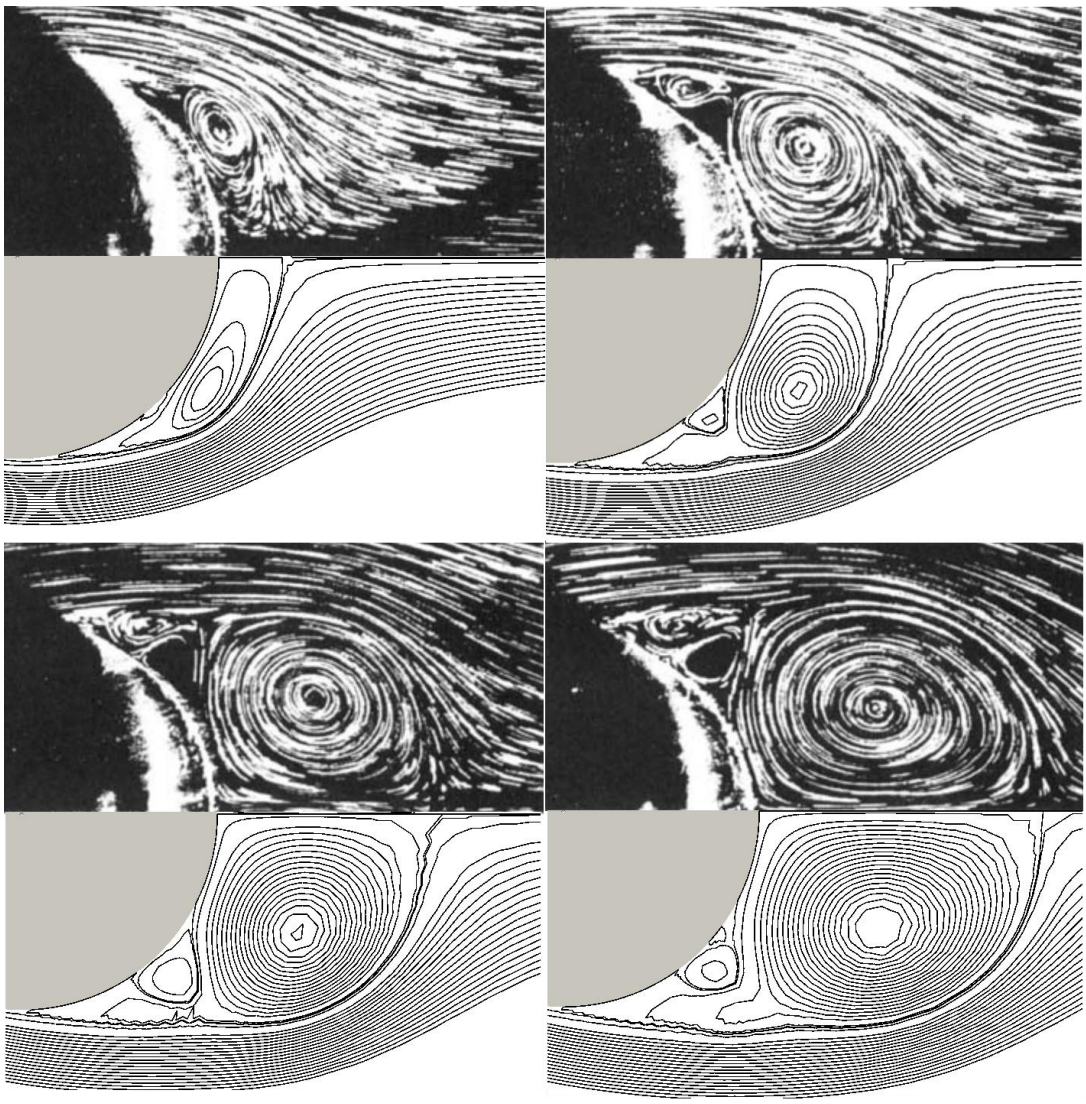


Рис. 2.11. Формирование вихрей за цилиндром после мгновенного старта при Re=1500 в моменты времени t=2,3,4,5. Сравнение результатов эксперимента [Bouard & Coutanceau 1980] и результатов численного моделирования в OpenFOAM.

Продемонстрированное временное окно, в котором поток сохраняет двумерный характер течения, обусловлено тем, что развитие трехмерной неустойчивости происходит медленнее, чем формирования плоских вихревых структур.

В заключение данного раздела проведем анализ гидродинамического воздействия на цилиндр. На рис. 2.12 изображено изменение коэффициента C_D в зависимости от времени (сплошная линия) при Re=13. В начальный момент (после мгновенного старта) сила сопротивления принимает большие значения, далее снижается более чем в 10 раз и выходит на постоянное значение. Это значение и принимается за коэффициент сопротивления установившегося течения. Для Re = 13 в расчете получено значение $C_D = 2.14$. На рисунке пунктиром также показана средняя экспериментальная оценка коэффициента сопротивления в установившемся течении, для Re = 13 $C_D \approx 2 \pm 5\%$. Расчетные данные немного выходят за верхнюю границу экспериментального диапазона.

Такая ошибка связана, прежде всего, с недостаточно большим размером области (при малых числах Рейнольдса течение особо чувствительно к влиянию внешних границ). В данном расчете использовалась область 36x18, для уточнения оценки коэффициента сопротивления расчетную область необходимо увеличить.

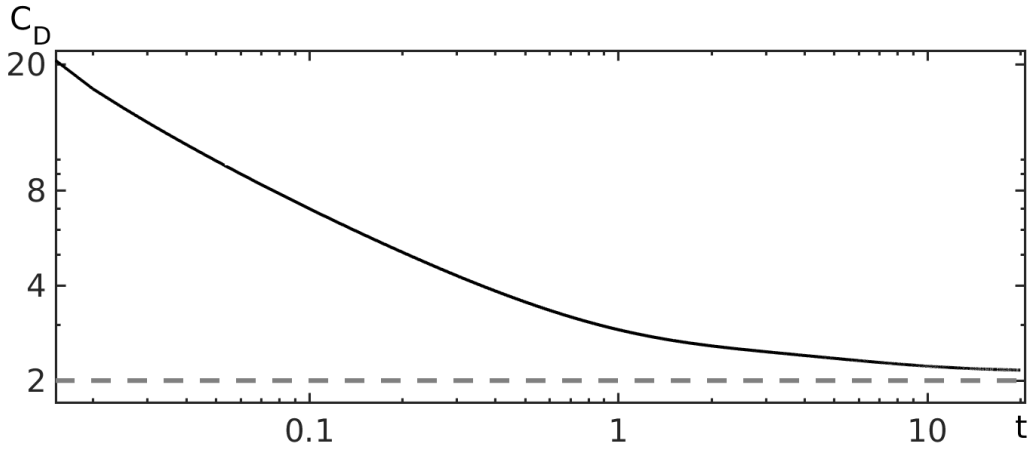


Рис. 2.12 Значения коэффициента сопротивления при Re=13.

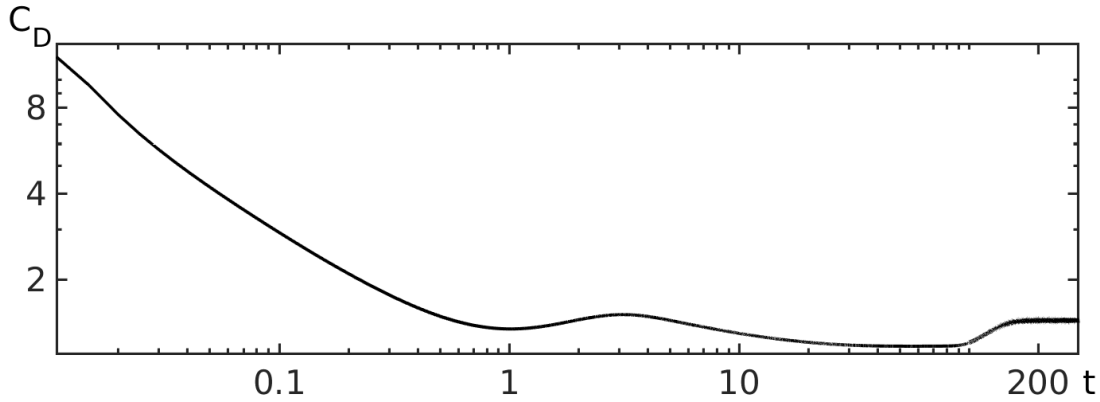


Рис. 2.13 Значения коэффициента сопротивления при Re=70.

На рис. 2.13-2.14 изображено изменение коэффициента C_D в зависимости от времени при Re=70. Как видно, коэффициент сопротивления здесь имеет более сложное поведение. Это связано с переходом течения в периодический режим. Течение полностью устанавливается при t>200 (см. рис. 2.14). Для оценки сопротивления в установившемся режиме используют осредненное по времени значение. Полученная численная оценка $C_D = 1.437$ при Re=70 очень хорошо согласуется с экспериментальными данными $C_D \approx 1.43 \pm 5\%$ (среднее значение показано на рис. 2.14 пунктирной линией).

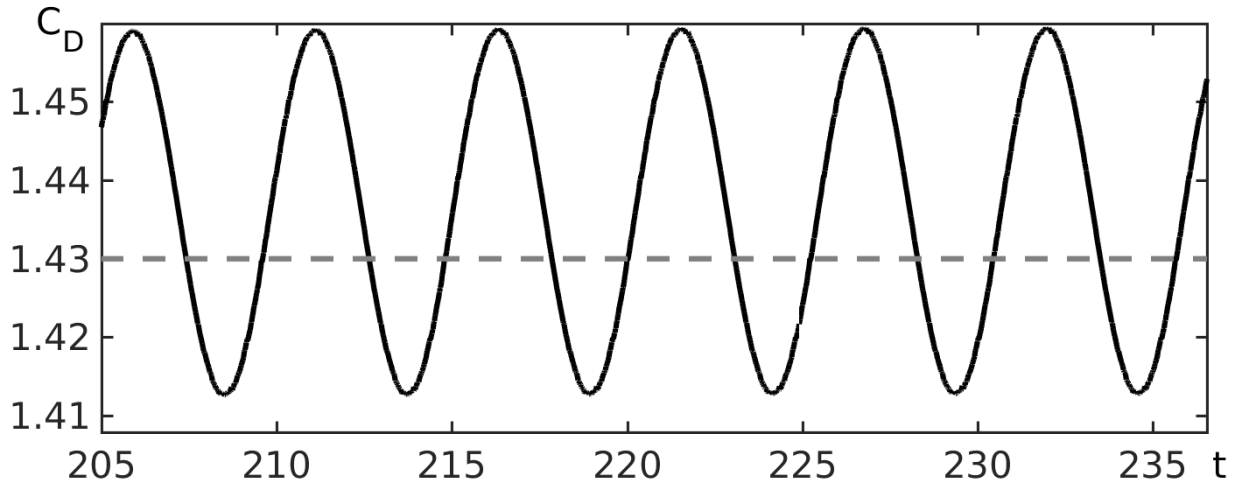


Рис. 2.14 Значение коэффициента сопротивления при Re=70 в установившемся режиме.

Задачи для самостоятельного исследования.

1. Проведите расчет гидродинамических сил, действующих на цилиндр, при разных числах Рейнольдса. Сравните полученные значения коэффициента сопротивления с экспериментальными данными, представленными в таблице 2.1 и на рис. 2.15.
2. Опишите поведение подъемной силы (постройте графики), действующей на цилиндр, в разных режимах обтекания.
3. Постройте численную модель для расчета обтекания квадратного цилиндра.
4. В рамках моделирования определите, какие режимы течения наблюдаются около квадратного цилиндра. Сравните наблюдаемые режимы течения с картинами течения, представленными на рис. 2.16.
5. Рассчитайте гидродинамические силы, действующие на квадратный цилиндр. Сравните полученные значения коэффициентов сопротивления и подъемной силы с аналогичными значениями для круглого цилиндра.

Таблица. 2.1 Значения коэффициента сопротивления при разных числах Рейнольдса, полученные по экспериментальным данным. В зависимости от источника значения могут варьироваться на $\pm 5\%$.

Re	0.05	0.5	5	20	50	100	150	300
C_D	63	10	2.2	1.6	1.44	1.35	1.2	1.1

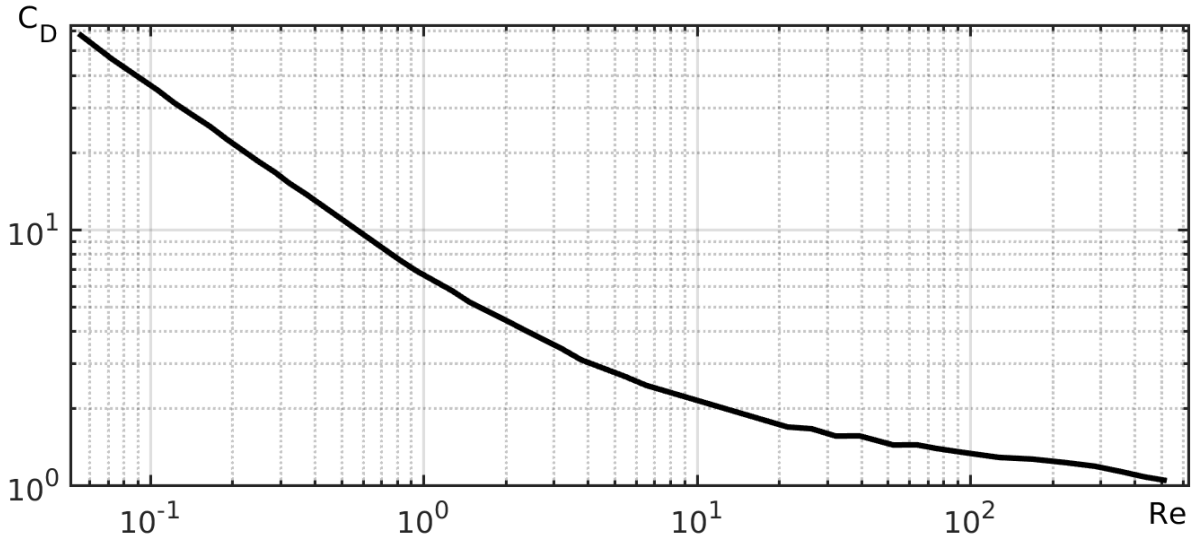


Рис. 2.15. Значения коэффициента сопротивления при разных числах Рейнольдса, полученные по экспериментальным данным Визельсбергера.

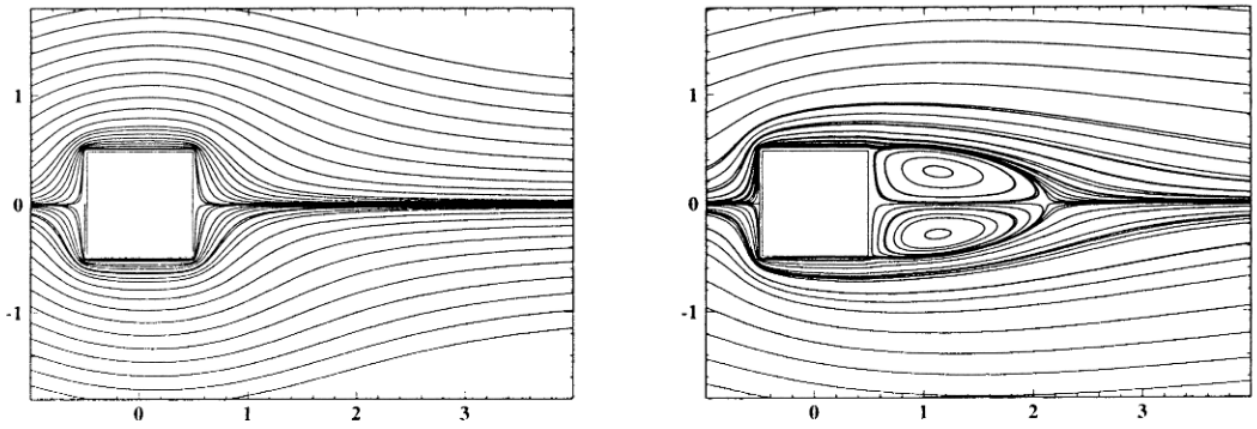


Рис. 2.16. Линии тока в установившихся режимах течения при Re =1 и Re=30 (число Рейнольдса определено по длине стороны).

Список литературы.

1. Greenshields, C. 2019 OpenFOAM v7 User Guide.

2. H. G. Weller, G. Tabor, H. Jasak, C. Fureby, A tensorial approach to computational continuum mechanics using object-oriented techniques, COMPUTERS IN PHYSICS, VOL. 12, NO. 6, NOV/DEC 1998.

3. Jasak, H. Error analysis and estimation for the Finite Volume method with applications to fluid flows: Ph.D. thesis / Imperial College, University of London.— 1996

4. Ferziger, J. H. Computational methods for fluid dynamics / J. H. Ferziger, M. Peric. — 3rd rev. edition. — Berlin: Springer, 2002. — P. 424.

5. R. I. Issa, Solution of Implicitly Discretized Fluid Flow Equations by Operator Splitting, / R. I. Issa // J. Comput. Phys. — 1986. — Vol. 62. — Pp. 40–65

6. Sahin, M. and Owens, R.G. (2003), A novel fully implicit finite volume method applied to the lid-driven cavity problem—Part I: High Reynolds number flow calculations. Int. J. Numer. Meth. Fluids, 42: 57-77. https://doi.org/10.1002/fld.442

7. A. Nuriev, A. Egorov, O. Zaitseva Bifurcation analysis of steady state flows in the lid-driven cavity. Fluid Dynamics Research 2016 Volume 48, Number 6, № 061405 p. 16, https://doi.org/10.1088/0169-5983/48/6/061405

8. Егоров, А. Г. Неединственность стационарного течения вязкой жидкости в квадратной каверне / А. Г. Егоров, А. Н. Нуриев // Учен. зап. Каз. гос. ун-та. Сер. Физ.-матем. Науки. — 2009. — Т. 151, № 3. — С. 130–143.

9. Нуриев, А. Н. Решение задачи об осциллирующем движении цилиндра в вязкой жидкости в пакете OpenFOAM / А. Н. Нуриев, О. Н. Зайцева // Вестник Казанского технологического университета. — 2013. — № 8. — С. 116–123

10. Coutanceau, M., & Bouard, R. (1977). Experimental determination of the main features of the viscous flow in the wake of a circular cylinder in uniform translation. Part 1. Steady flow. Journal of Fluid Mechanics, 79(02), 231. doi:10.1017/s0022112077000135

11. Bouard, R., & Coutanceau, M. (1980). The early stage of development of the wake behind an impulsively started cylinder for 40 < Re < 10 4. Journal of Fluid Mechanics, 101(03), 583. doi:10.1017/s0022112080001814