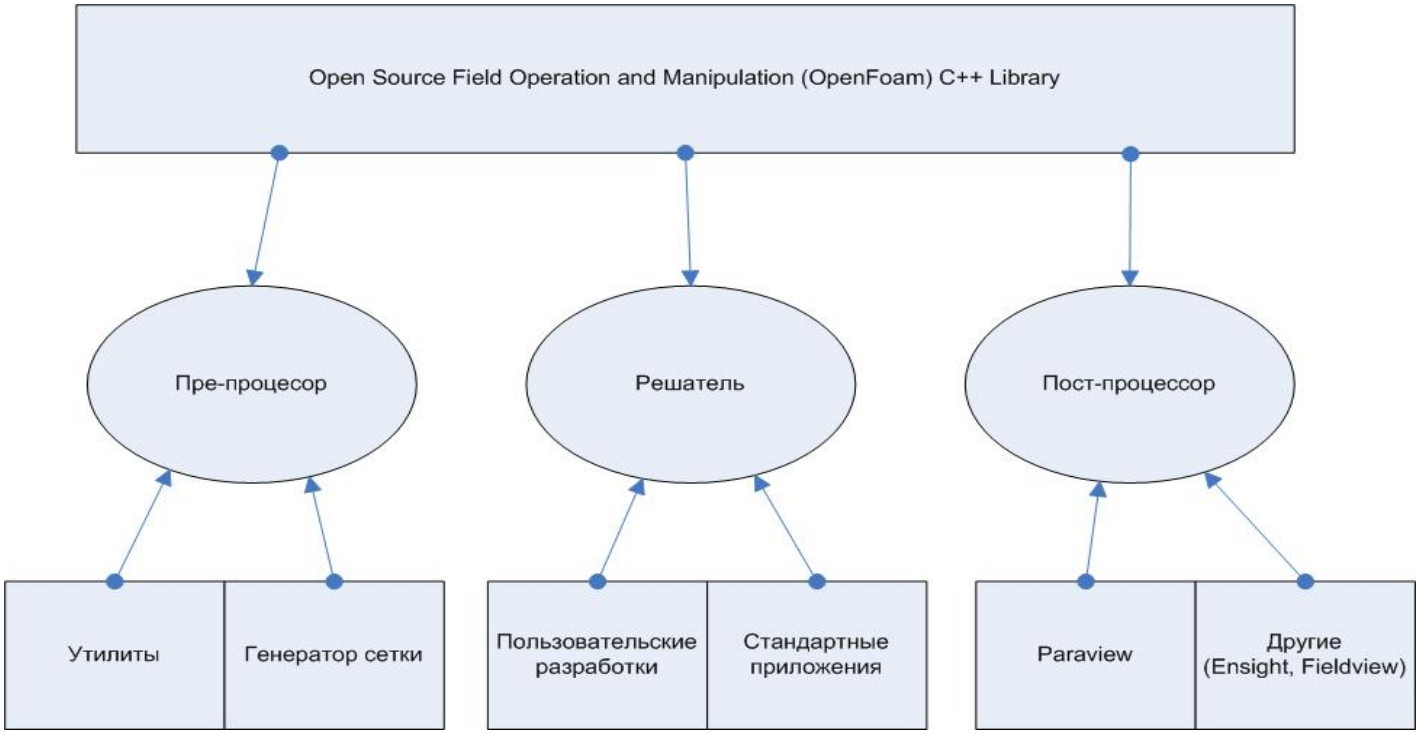
Author: Christopher J. Greenshields, CFD Direct Ltd.

Набран в LaTeX.

Глава 1

Введение

Это руководство сопровождает выпуск версии 2.1.0 Open Source Field Operation And Manipulation (OpenFOAM) (операции и манипуляции с полями с открытым исходным кодом) C++ библиотек. Оно предоставляет описание основных операций с OpenFOAM, сначала посредством ряда примеров руководства в гл. 2 и далее с использованием более детальных описаний индивидуальных компонент, которые составляют пакет OpenFOAM. OpenFOAM это во-первых и прежде всего *библиотеки на C++*, использованные сначала для создания исполняемых модулей (executables), называемых как *приложения* (*applications*). Приложения разделяются на две категории: решатели или *солверы* (*solvers*), каждый из которых создан чтобы решить частную задачу в механике сплошной среды; и сервисные программы или *утилиты* (*utilities*), которые разрабатывались для выполнения задач включающих обработку данных. Дистрибутив OpenFOAM содержит многочисленные решатели и утилиты обеспечивающие решение широкого диапазона задач, как описано в гл. 3. Одна из наиболее сильных возможностей OpenFOAM заключается в том, что новые решатели и утилиты могут быть созданы его пользователями с некоторыми предварительными, минимально необходимыми знаниями основных использованных методов физики и методов программирования. OpenFOAM поставляется с вычислительной средой включающей пре- и постпроцессор. Интерфейс (связь с) пре- и постпроцессором обеспечивается утилитами OpenFOAM, таким образом гарантируя совместимость обращения с данными для всей вычислительной среды. Общая структура OpenFOAM показана на Рис 1.1. (Figure 1.1.) Обработка предпроцессором и запуск кейса OpenFOAM представлен в **??**.



UGFigure 1-1.PNG: 973x557 pixel, 96dpi, 25.74x14.74 cm, bb=0 0 730 418

Рис. 1.1: Рисунок 1.1: Условная структура пакета

В **??** мы представили как генерацию (создание) сеток, используя утилиты генерации сеток поставляемых с OpenFOAM, так и преобразование сеточных данных, созданных с использованием сторонних продуктов. Постпроцессорная обработка решения описана в **??**.

Глава 2

Учебные примеры

В этой главе мы опишем подробно процесс установки, моделирования и пост- процесорной обработки для некоторых тестовых примеров OpenFOAM, с основной целью: познакомить пользователя с основными процедурами работы и управления OpenFOAM. *$FOAM_ TUTORIALS* директория содержит много тестовых случаев, которые демонстрируют использование всех решателей и многих утилит поставляемых с OpenFOAM. Прежде чем пытаться использовать обучающие программы, пользователь должен предварительно удостовериться в том, что он установил OpenFOAM правильно.

Обучающие примеры описывают использование утилиты *blockMesh* - инструмента предпроцессорной обработки, установки примера и запуска на счет решателя OpenFOAM и постпроцессорной обработки с использованием *paraFoam*. Те пользователи, которые имеют доступ к постпроцессорам, разработанным другими фирмами (из инструментария посторонних разработчиков), также поддерживаемых в OpenFOAM, имеют выбор: либо они могут следовать за примерами обучающих программ, используя *paraFoam*; либо обратиться к описанию использования продукта постороннего разработчика, как описано в **??**, когда требуется постпроцессорная обработка.

Копии всех обучающих программ доступны в директории *tutorials* из поставки OpenFOAM. Обучающие программы организованы в виде ряда директорий согласно типу течения в которых расположены поддиректории согласно решателю. Например, все *icoFoam* примеры хранятся в пределах подкаталога *incompressible/icoFoam*, где *incompressible* (несжимаемое) указывает на тип рассматривваемого течения. Если пользователь желает запустить на расчет несколькотестовых примеров, рекомендуется, чтобы пользователь сначала скопировал директорию *tutorials* в свою локальноую директорию *run*. Они могут быть легко скопированы набором:

```
mkdir -p $FOAM_RUN
cp -r $FOAM_TUTORIALS $FOAM_RUN
```

2.1 Течение в прямоугольной полости, инициированное верхней плитой

Этот раздел руководства описывает как выполнить этапы пре-процессор, расчет и постпроцессорную обработку для тестового примера, рассматривающего изотермическое течение несжимаемой среды в двумерной квадратной области. Геометрия течения изображена на Рис. 2.1, в которой все границы квадрата являются стенками. Верхняя стенка перемещается в x-направлении со скоростью 1 м/с, в то время как другие 3 неподвижны. Первоначально принято, что будет рассматриваться ламинарное течение и задача будет решаться на однородной сетке, используя решатель *icoFoam* для ламинарного, изотермического, несжимаемого потока. В течение курса обучающей программы, будут исследованы влияние увеличения числа узлов сетки на решение и влияние измельчения шага сетки по направлению к стенкам. Наконец, число Рейнольдса будет увеличено, и решатель *pisoFoam* будет использоваться для турбулентного, изотермического, несжимаемого течения.

2.2 Пре- процессорная подготовка (pre-processing)

В составе дистрибутива OpenFOAM поставляются примеры. Для редактирования файлов примера пользователь должен выбрать xeditor или другой редактор , типа **emacs**, **vi**, **gedit**, **kate**, **nedit**, и т.п. Редактирование файлов возможно в OpenFOAM, потому что ввод / вывод использует формат словаря с ключевыми словами, которые представляют смысловое значение, чтобы быть понятными даже наименее опытным пользователям.

Моделируемый случай включает данные для сетки, полей, свойств, параметров контроля решения, и т.д. Как описано в секции 4.1, в OpenFOAM эти данные хранятся в ряде файлов в пределах директории примера, а не в единственном файле примера, как во многих других пакетах вычислительной гидродинамики (CFD). Директории примера дают соответственно описательное название, например первый случай для этого обу-

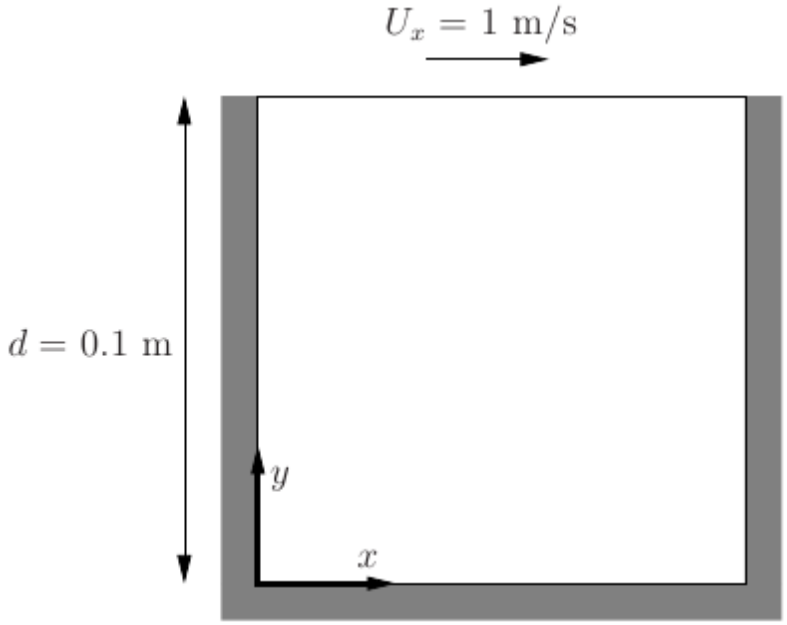


Рис. 2.1: Геометрия течения в каверне, инициированного движением верхней плиты.

чающего примера просто назван *cavity* (каверна). В подготовке редактирования файлов примера и запуска первого примера *cavity*, пользователь должен перейти в каталог примера

```
cd $FOAM_RUN/tutorials/incompressible/icoFoam/cavity
```

2.2.1 Генерация сетки

OpenFOAM всегда работает в 3 мерной декартовой системе координат и все геометрические конфигурации производятся в 3 измерениях. OpenFOAM решает примеры в 3 измерениях по умолчанию, но может быть проинструктирован решать в 2 измерениях, определяя специальное *empty* (пустое) граничное условие на границах, нормальных к (3-ему) z измерению (для данного примера), для которого никакое решение проводить не требуется. Область примера *cavity* состоит из квадрата с длинами сторон $d = 0.1$ м. в x-y плоскости. Первоначально будет использоваться равномерная сетка 20 на 20 ячеек. В блочной конструкции показанной на рис. 2.2. генератор сеток, поставляемый с OpenFOAM, *blockMesh*, создает сетки, используя команды из описания определенного в словаре ввода, *blockMeshDict* расположенного в каталоге *constant/polyMesh* данного примера. Чтобы перейти к *blockMeshDict* для этого примера возьмем следующий файл:

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.1.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;


    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

convertToMeters 0.1;

vertices
(
    (0 0 0)
    (1 0 0)
```

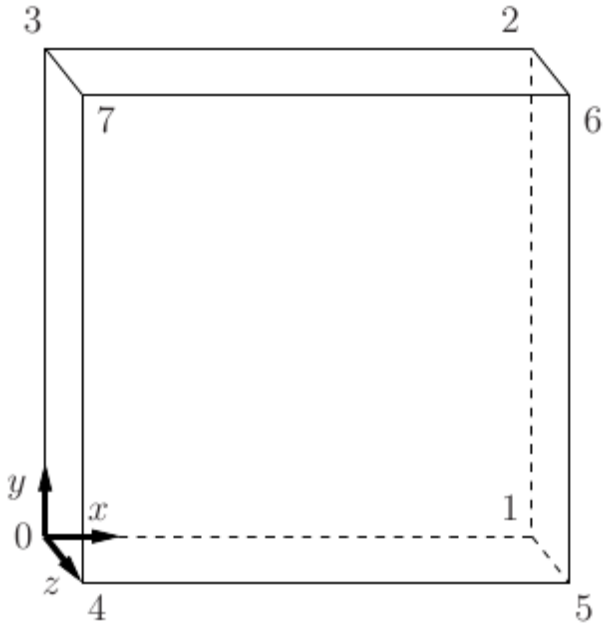


Рис. 2.2: Блочная конструкция сетки для каверны.

```
    (1 1 0)
    (0 1 0)
    (0 0 0.1)
    (1 0 0.1)
    (1 1 0.1)
    (0 1 0.1)
);

blocks
(
    hex (0 1 2 3 4 5 6 7) (20 20 1) simpleGrading (1 1 1)
);

edges
(
);

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)
        );
    }
);

mergePatchPairs
(
);

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

В начале файла содержится заголовок в форме баннера (строки 1-7), далее информация файла содержится в подсловаре *FoamFile*, заключенная в фигурных скобках ({...}).

В остальной части руководства: Ради сохранения места и для ясности, шапка или заголовок файла, включая баннер и *FoamFile* подсловарь, и обрамление в скобках не будут включаться в файлы примеров.

Файл в начале содержит блок вершин `vertices` (x,y,z координаты точек); затем в нем определяются сами блоки (`blocks`) (в данном примере имеется только 1) с помощью описанных выше меток вершин и номеров ячеек внутри них; и наконец, в файле определяются поверхности для задания граничных патчей. Пользователю предлагается обратиться к **??** чтобы понять значение вводимых величин в *blockMeshDict* файле.

Сетка генерируется при запуске утилиты *blockMesh*, использующей файл *blockMeshDict*. Запуск производится из директории задачи, просто набором команды в терминале:

```
blockMesh
```

Состояние выполнения *blockMesh* выводится в окне терминала. Любые ошибки в исходном файле сетки *blockMeshDict* обрабатываются textslblockMesh и итоговые сообщения об ошибках показывают пользователю номер строки в которой возникла проблема. На этой стадии не должно быть никаких сообщений об ошибках.

2.2.2 Граничные и начальные условия

Как только генерация сетки окончена, пользователь может посмотреть на заданные начальные поля установленные для этого расчетного случая. Пример установлен на начальное время t = 0 сек, так что данные о начальных полях хранятся в поддиректории *0* каталога примера *cavity*. Поддиректория *0* содержит 2 файла, *p* и *U*, по одному для полей давления (p) и скорости (U), чьи начальные значения и граничные условия должны быть заданы. Давайте проверим файл давления *p*:

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

internalField   uniform 0;

boundaryField
{
    movingWall
    {
        type            zeroGradient;
    }

    fixedWalls
    {
        type            zeroGradient;
    }

    frontAndBack
    {
        type            empty;
    }
}

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

Существуют 3 главных раздела ввода полей данных файлов:

`dimensions` (размерности) определяют величину размерности поля, здесь *кинематическое* давление, m^2s^{-2} (баротропия или отношение давления к плотности => (N/m^2) /(кг/m^3)=m^2/s^2) (смотрите **??** для более детальной информацией);

`internalField` данные поля внутри расчетной области, которое может быть задано однородным (`uniform`), описываемым одной цифровой величиной; или неоднородным (`nonuniform`), где все значения поля должны быть заданы (смотрите **??** для более подробной информацией);

`boundaryField` данные о граничных полях, которые включают граничные условия, и данные обо всех граничных патчах (boundary patches) (смотрите **??** для более подробной информации).

Для данного примера каверны (*cavity*), границы состоят только из стенок, разбитые на 2 патча названных: (1) *fixedWall* для неподвижных фиксированных боковых стенок и основания каверны; (2) *movingWall* для подвижного верха каверны. На стенках, и тех и других, граничное условие для давления p есть *zeroGradient*, что означает "нормальный градиент давления равен нулю". Патчи *frontAndBack* представляют переднюю и заднюю плоскости и поэтому для двумерного случая должны быть заданы *empty* (пустой).

В этом примере, как и в большинстве с которыми мы встретились, начальные поля заданы однородными (uniform). Здесь давление является кинематическим, и как для несжимаемой среды, абсолютное значение его не имеет значения (во всех уравнениях нет самого давления а только его производные), поэтому оно задается как `uniform 0` для удобства.

Пользователь может аналогично проверить поле скорости в файле *0/U*. Размерности *dimensions* необходимы для скорости, внутреннее поле инициализируется как одногродное нулевое *uniform zero*, что в случае скорости должно быть представлено тремя компонентами, т.е. *uniform (0 0 0)* (смотри **??** для более подробной информации).

Граничные условия для скорости требуют аналогичных граничных условий (empty) для переднего и заднего патчей *frontAndBack*. Другими патчи являются стенками (walls): условие непроскальзывания (no-slip) задаётся на поверхностях *fixedWalls*, следовательно условие *[U+FB01]xedValue* со значением `uniform (0 0 0)`. Верхняя поверхность движется со скоростью 1 м/с в направлении оси x, что тоже требует условия *[U+FB01]xedValue*, но со значением `uniform (1 0 0)`.

2.2.3 Физические свойства

Физические свойства для кейса хранятся в словарях (dictionaries), к чьим именам добавляются суффиксы *...Properties* (Свойства), расположены в дереве каталогов *Dictionaries*. Для кейса *icoFoam*, должно быть задано единственное свойство это кинематическая вязкость, которая хранится в словаре *transportProperties*. Пользователь может проверить, что кинематическая вязкость задана правильно открыв словарь *transportProperties* а также просмотреть/отредактировать его содержимое. Ключевое слово для кинематической вязкости *nu*, фонетическое обозначение греческой буквы ν которой вязкость обозначена в уравнениях. Первоначально этот кейс будет запущен с числом Рейнольдса равным 10, где число Рейнольдса определяется как:

$$Re = \frac{d|\mathbf{U}|}{\nu} \tag{2.1}$$

где d и $\mathbf{U}$ характеристики размер и модуль скорости соответственно, а ν кинематическая вязкость. В примере $d = 0.1m$, $|\mathbf{U}| = \mathbf{1ms^{-1}}$, так что для $Re = 10$, $\nu = 0.01m^2s^{-1}$.

Верными входными величинами для кинематической вязкости являются представленные ниже значения:

```
[file, linenum=17]

nu              [0 2 -1 0 0 0 0] 0.01;


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

2.2.4 Управление

Входные данные, относящиеся к управлению временем чтения и записи данных решения находятся в *controlDict*. Пользователь должен просмотреть данный файл, так как это основной файл управления, находится в папке *system*.

Для запуска необходимо задать время начала и конца решения, а также шаг по времени. OpenFOAM предоставляет широкие возможности по управлению временем. Подробное описание вы найдете в **??**. В этом руководстве мы хотим начать запуск в момент времени $t = 0$, это значит что OpenFOAM необходимо считать данные из каталога 0 (см. **??**). Поэтому мы задаем значения для `startFrom` и `startTime`, а затем назначаем `startTime` значение 0.

Для конечного шага по времени мы хотим получить стационарное решение для потока, циркулирующего внутри полости каверны. Как правило жидкость должна пройти через объем 10 раз прежде чем достичь

состояния устойчивого ламинарного течения. В данном случае поток не проходит сквозь объем, так как здесь нет ни входа ни выхода, поэтому в качестве конечного момента времени расчета может быть задано время, требующееся на то, чтобы крышка проехала через каверну десять раз, т.е. 1 с; по предыдущим расчетам видим, что достаточно и 0.5 с, поэтому будем использовать это значение. Чтобы задать это конечное время дадим параметру `stopAt` значение `EndTime`, а затем установим для параметра `EndTime` значение 0.5.

Теперь необходимо установить шаг по времени, который задется параметром `deltaT`. Для достижения точности по времени и численной стабильности при запуске icoFoam, необходимо число Куранта меньше 1. Число Куранта для одной ячейки определяется как:

$$C_o = \frac{\delta t|\mathbf{U}|}{\delta x} \tag{2.2}$$

где δt это шаг по времени, $|\mathbf{U}|$ это величина скорости через эту ячейку и δx размер ячейки по направлению скорости. Скорость потока непостоянна в пределах области, но условие $C_o < 1$ должно выполняться во всей области. Поэтому мы выберем δt на основе наихудшего варианта: *максимальное* значение $C_o < 1$ соответствует совокупности большой величины скорости течения и малого размера ячейки. В нашем случае размер ячейки в области постоянный, поэтому максимальное значение $C_o < 1$ будет вблизи двигающейся крышки, где скорость около 1 м/с. Соответственно размер ячейки определяется по формуле:

$$\delta x = \frac{d}{n} = \frac{0.1}{20} = 0.005m \tag{2.3}$$

Следовательно чтобы получить число Куранта меньше или равное 1 во всей области, шаг по времени `deltaT` должен быть меньше или равен:

$$\delta x = \frac{C_o \delta x}{|\mathbf{U}|} = \frac{1 \times 0.005}{1} = 0.005s \tag{2.4}$$

В процессе моделирования мы решили записывать результаты через определенные промежутки времени, что мы можем позднее просмотреть с помощью постпроцессора. Оператор `writeControl` предоставляет несколько опций для настройке моментов времени, в которые будут записываться результаты; здесь мы выбираем опцию `timeStep`, которая задает, что результаты будут записываться каждый n-ый шаг по времени, где значение n задается параметром `writeInterval`. Допустим мы хотим записывать результаты в моменты 0.1, 0.2, . . . , 0.5 с - с шагом 0.005 с, для этого необходимо выводить результаты на каждом 20-ом шаге, поэтому мы установили значение `writeInterval` равное 20.

OpenFOAM создает новую папку с *именем соответсвующего момента времени*, например, 0.1 с. И каждый раз, записывается набор данных, что более подробно описано в разделе 4.1. В решателе *icoFoam* во временные каталоги записываются значения полей `U` и `p`. Для нашего примера, содержание *controlDict* приведено ниже:

```
application     pisoFoam;

startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         0.5;

deltaT          0.005;

writeControl    timeStep;

writeInterval   20;

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression off;

timeFormat      general;

timePrecision   6;
```

```
runTimeModifiable true;

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

2.2.5 Дискретизация и параметры решателя

Пользователь указывает выбранные схемы конечно-элементной дискретизации в файле *fvSchemes* директории *system*. Описание решателей систем линейных уравнений, задание точности и настройка алгоритмов происходит в файле *fvSolution* той же директории *system*. Пользователь может просмотреть все эти файлы, но мы пока не будем подробно описывать их содержимое, за исключением параметров `pRefCell` и `pRefValue` в *PISO* - подсловаре *fvSolution*. В закрытой несжимаемой системе, такой как каверна, давление относительное: в диапазоне давлений значения не абсолютные. В таких случаях в решателе задается исходный уровень с помощью опции `pRefValue` в ячейке `pRefCell`. В нашем примере оба значения равны 0. Изменение любого из этих значений повлияет на абсолютное значение давления, но не на относительные значения скорости или давления.

2.3 Просмотр сетки

Проверить сетку на наличие ошибок перед запуском - хорошая идея. Сетку можно просмотреть в *paraFoam*, постпроцессоре, распространяемом вместе с OpenFOAM. Постпроцессор *paraFoam* можно запустить набрав в терминале следующую команду из каталога проекта

```
paraFoam
```

Также он может быть запущен из любой другой директории если добавлена опция `-case` аргументом которой является путь к проекту, например

```
paraFoam -case $FOAM_RUN/tutorials/incompressible/icoFoam/cavity
```

Эта команда запускает окно *ParaView* как показано на рисунке 6.1. В *Pipeline Browser*, пользователь может видеть, что в *ParaView* открыт файл *cavity.OpenFOAM*, для *cavity*. Перед нажатием на кнопку *Apply* (Применить), пользователю нужно выбрать геометрию в панели *Mesh Parts*. Так как проект небольшой, то самое простое - выбрать все компоненты в заголовке панели *Mesh Parts*. Пользователю необходимо нажать кнопку *Apply* (Применить) для начала загрузки геометрии в *ParaView*.

Откройте панель *Display*, которая отвечает за визуализацию выбранного модуля. В панели *Display* выполните следующие действия, как показано на рисунке 2.3: (1) установите Цвет по *Solid Color*, (2) нажмите *Set Solid Color* и выберите подходящий цвет, например черный (для белого фона); (3) в панели *Style* выберите *Wireframe* в меню Representation. Цвет фона можно задать, выбрав *View Settings* в меню *Edit* на строке меню вверху.

Особенно если вы впервые пользуетесь ParaView, мы рекомендуем вам работать с изображением, как описано в разделе 6.1.5. В частности, поскольку это двумерная модель, рекомендуем вам выбрать Use Parallel Projection в панели General настройки окна выбранного в меню Edit. Ориентацию осей можно включить или выключить в окне Annotation, также положение переключателя можно изменять при помощи мыши.

2.4 Запуск приложения

Как и любые исполняемые Unix/Linux приложения, OpenFOAM можно запустить двумя способами: как главный процесс, т.е. в котором оболочка ждет завершения выполнения команды, прежде чем запустить командную строку; и как фоновый процесс, который в течение процесса выполнения позволяет вызвавшей оболочке получать дополнительные команды. В данном случае мы будем запускать icoFoam в качестве главного процесса. Солвер icoFoam выполняется либо путем ввода в каталоге, набрав

```
icoFoam
```

в командной строке, либо с помощью -case с выборочным параметром,зависящим от каталога, например:

```
icoFoam -case $FOAM RUN/tutorials/incompressible/icoFoam/cavity
```

Ход решения записывается в окне терминала. Он сообщает пользователю текущее время, максимальное значение числа Куранта, начальные и конечные невязки решения по всем направлениям течения.

2.5 Последующая обработка (Post-processing)

В то время как результаты записываются во временные каталоги, их можно просмотреть с помощью paraFoam. Вернитесь к окну paraFoam и выберите панель Properties для cavity.OpenFOAM модуля. Если нужное окно

панелей в случае модуля, не отображается в любой момент времени, пожалуйста, убедитесь, что: cavity.OpenFOAM подсвечивается синим цветом, а кнопка (глаза) eye рядом с ним нажата и показывает, что графика включена; Чтобы подготовить paraFoam для отображения данных, представляющих интерес, мы должны сначала загрузить данные в соответствующий момент времени 0,5 сек. Если процесс был запущен в то время как ParaView был открыт, выходные данные во временных директориях не будут автоматически загружаться в среду ParaView. Для загрузки данных пользователь должен выбрать Update GUI в окне Properties, а затем нажать зеленую кнопку Apply. Временные данные будут загружены в ParaView.

2.6 Поверхности постоянных значений и профильные диаграммы (Isosurface and contour plots)

Глава 3

N

Глава 4

N

Глава 5

Создание и конвертация сеток

Эта глава описывает все вопросы, связанные с созданием сеток в пакете OpenFOAM: Раздел 5.1 даёт обзор путей, которыми может быть описана сетка в OpenFOAM; Раздел 5.3 охватывает подпрограмму blockMesh для генерации простых сеток из блоков гексаэдрических элементов; Раздел 5.4 охватывает подпрограмму snappyHexMesh для автоматического создания сложных сеток из гексаэдрических и расщеплённых гексаэдрических элементов из триангулированных геометрических поверхностей; раздел 5.5 описывает имеющиеся опции для конвертации сетки, созданной в других пакетах, в формат, доступный для чтения OpenFOAM.

5.1 Описание сетки

Этот раздел раскрывает описание классов OpenFOAM C++, работающих с сеткой. Сетка является составной частью численного расчёта и должна удовлетворять определённому критерию для получения качественного и точного результата. В процессе запуска, OpenFOAM проверяет соответствие сетки набору строгих ограничений и завершит работу, если ограничения не будут соблюдены. Это важно, потому что пользователь перед запуском OpenFOAM может некорректно исправить большую сетку в программах сторонних разработчиков. Это является некоторым недостатком, но мы (разработчики) не оправдываем OpenFOAM просто усваивая хорошие примеры из практики. Убедитесь в том, что сетка соответствует требованиям расчёта; в ином случае решение будет некорректным уже перед запуском программы. По умолчанию OpenFOAM определяет 3D сетку из случайных многогранников, ограниченных случайными полигональными гранями, т.е. элементы могут иметь бесконечное количество поверхностей, в которых, для каждой поверхности, имеется неограниченное количество рёбер и нет ограничений на их положение. Сетка с такой общей структурой известна в OpenFOAM как polyMesh. Детально она описывается в разделе 2.1 Руководства программиста, но важно запомнить, что этот тип сетки позволяет получить большую свободу при создании и манипуляции с сетками, в частности когда геометрия домена сложна, либо изменяется во времени. Ценой такой универсальности сетки является сложность её генерации с помощью программ-преобразователей. Поэтому в OpenFOAM предусмотрена утилита cellShape для управления общепризнанными форматами сеток, основанных на множестве предопределённых форм элементов.

5.1.1 Описание сетки и её применимость

Перед описанием OpenFOAM-формата сетки, polyMesh, и утилит cellShape, мы, в первую очередь, установим критерии применимости, используемые в OpenFOAM. Условия, которым должна удовлетворять сетка даны ниже:

Точки

Точка является областью 3D пространства, определяемой вектором в метрах. Точки компилируются в список и каждая точка связывается с указателем, который обозначает её позицию в списке, начинающемся с нуля. Список точек не может содержать две точки, обладающие совершенно идентичным расположением, так же как точку, не принадлежащую по крайней мере одной грани.

Грани

Грань - это упорядоченный список точек, в котором каждая точка имеет свою метку. Упорядочивание меток точек происходит из условия того, чтобы две соседние точки были соединены ребром, т.е. следуя нумерации вы перебираете точки, обходя по кругу грани. Грани компилируются в список и каждая грань имеет свою метку, отображающую её положение в списке. Направление вектора нормали грани выбирается по правилу правой руки, т.е. при взгляде по направлению к грани, если нумерация точек идёт против часовой стрелки, нормаль направлена к Вам, как показано на рисунке 5.1.