Глава 1

Лабораторные работы

1.1. Лабораторная работа №1: Знакомство со структурой проекта. Течение в ударной трубе.

В данной лабораторной работе будет рассмотрена классическая задача течения в одномерной ударной трубе, широко используемая для валидации сжимаемых решателей [1]. На данном примере будет изучен типовой проект в OpenFOAM и основные шаги моделирования в нем.

Математическая модель течения в ударной трубе

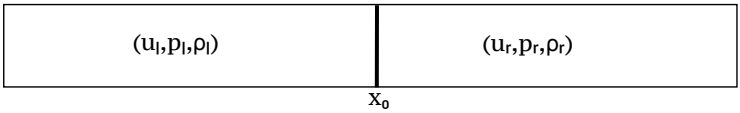


Рис. 1.1: Одномерная ударная труба с диафрагмой в точке x_0

Рассмотрим отрезок $(0, 1)$, соответствующий длине расчетной области, разделенный на две части диафрагмой в точке $x_0 = 0.5$. Положим следующее начальное распределение величин в момент времени $t = 0$:

$$\begin{pmatrix} u_l \\ p_l \\ \rho_l \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} u_r \\ p_r \\ \rho_r \end{pmatrix} = \begin{pmatrix} 0 \\ 0.125 \\ 0.1 \end{pmatrix},$$

где $(u_l, p_l, \rho_l)^T, (u_r, p_r, \rho_r)^T$ – начальные условия в левой и правой области, аналогично статье [1]. Тем самым задается идеализированная ударная труба с диафрагмой, разделяющей две порции газа, где область высокого давления расположена слева. При мгновенном разрыве диафрагмы происходит возникновение волны разрежения, контактного разрыва и ударной волны.

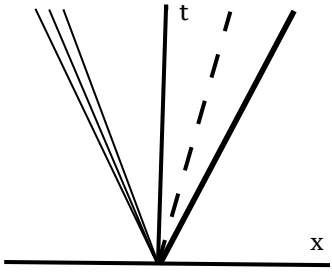


Рис. 1.2: Диаграмма ударно-волнового процесса, волны разрежения слева, ударная волна справа, пунктиром обозначена зона контакта

Такая задача может быть описана одномерной системой уравнений газовой динамики:

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}(\mathbf{U})}{\partial x} = 0, \tag{1.1}$$

где

$$\mathbf{U} = \begin{pmatrix} \rho \\ \rho u \\ \rho E \end{pmatrix}, \qquad \mathbf{F}(\mathbf{U}) = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ (\rho E + p)u \end{pmatrix}$$

— вектора консервативных переменных и потока. Для замыкания используется уравнение состояния идеального газа:

$$\rho = \frac{p}{RT},$$

здесь R — индивидуальная газовая постоянная.

Решение также может быть найдено аналитически из соотношений:

$$u_* = \frac{1}{2}(u_l + u_r + f_r(p_*) - f_l(p_*)), \tag{1.2}$$

где u_*, p_* – скорость и давление между ударной волной и волнами разряжения, давление p_* находится численно (например, методом секущих) из уравнения:

$$f_l + f_r + u_r - u_l = 0, \tag{1.3}$$

где нижний индекс $x \in \{r, l\}$ обозначает откуда берутся соответствующие характеристики (самая левая или правая область газа):

$$f_x(p) = \begin{cases} (p - p_x)\sqrt{\dfrac{A_x}{p + B_x}}, & p > p_x, \\ \dfrac{2a_x}{\gamma - 1}\left(\left(\dfrac{p}{p_x}\right)^{(\gamma-1)/2\gamma} - 1\right), & p \le p_x, \end{cases} \tag{1.4}$$

$$A_x = \frac{2}{(\gamma + 1)\rho_x}, \qquad B_x = \frac{(\gamma - 1)}{(\gamma + 1)} p_x. \tag{1.5}$$

Здесь a_x – скорость звука, γ – отношение теплоемкости газа при постоянном давлении к теплоемкости при постоянном объеме, которую для воздуха можно принять равной $7/5$. Более подробно данная схема решения описана в монографии [2].

Структура проекта OpenFOAM

Проект в OpenFOAM организован как набор текстовых конфигурационных файлов, собранных в структуре каталогов, подобной рис. 1.3. Минимально необходимый набор каталогов состоит из:

- **0**, содержащего начальные и краевые условия в файлах, названия которых соответствуют задаваемой величине. В дальнейшем, решатель будет создавать аналогичные каталоги, содержащие сведения о значении полей на текущем временном шаге.
- **constant**, содержащего значения, не изменяющиеся во время вычисления, например, коэффициенты переноса *transportProperties* или каталог с конфигурацией вычислительной сетки **polyMesh**.
- **system**, содержащего настройки численных методов и вспомогательных утилит.

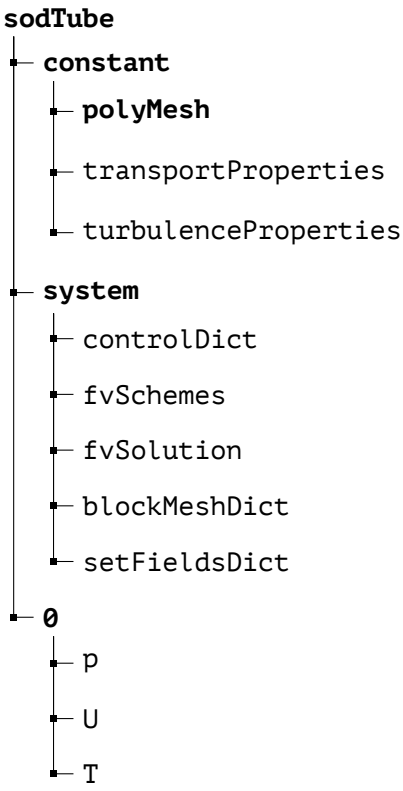


Рис. 1.3: Пример структуры проекта в OpenFOAM

Построение расчетной сетки

Для дискретизации области методом конечных объемов необходимо построить вычислительную сетку. Одним из способов её построения является утилита *blockMesh*, позволяющая строить блочные структурированные сетки на основании содержания файла *blockMeshDict* (листинг 1.5). Все конфигурационные файлы в OpenFOAM имеют один и тот же синтаксис, напоминающий C++, например, строка заканчивается на точку с запятой, а комментарии обозначаются при помощи // (однострочные) и //*, *// (многострочные).

В шапке каждого конфигурационного файла расположен сло-

варь *FoamFile*, характеризующий текущий файл:

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
```

Словарь содержит ключевые слова (ключи) и их значения, в данном случае: версию формата файлов (*version*) — 2.0, тип содержимого (*format*) — текстовый (*ascii*, также возможен бинарный, *binary*), тип создаваемого в коде OpenFOAM класса (*class*) — словарь (*dictionary*), а также имя файла (*object*). В большинстве случаев нет необходимости менять эти значения, они позволяют понять, что в данный момент редактируется. Отметим, что порядок значений в словаре значения не имеет, однако, рекомендуется его сохранять для сохранения привычного вида конфигурационных файлов.

Далее *blockMeshDict* содержит ключ *scale*, чье значение соответствует выбранному масштабу:

```
scale 1;
```

К примеру, если измерения были бы заданы в миллиметрах, то необходимо было бы использовать значение 0.001.

Следом расположены последовательно ключи, определяющие непосредственно геометрию сетки: вершины (*vertices*), ребра (линии, соединяющие вершины) (*edges*), блоки (*block*), границы (*boundary*) и соединенные поверхности (*mergedPatchPairs*).

Вершины определяются списком, который обособляют круглыми скобками. Он, в свою очередь, содержит вложенные списки из трех элементов для координат вершин, как показано ниже на примере блока (рис. 1.4:

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

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

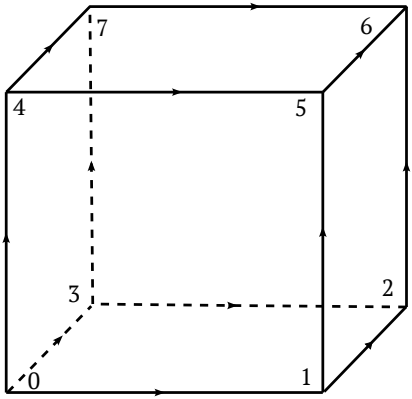


Рис. 1.4: Пример блока

Отметим, что вершины в нем не должны повторяться. Порядок вершин имеет значение, получить корректный порядок возможно следующим образом:

- Выбрать четыре вершины с минимальными значением третьей координаты и записать их, начиная от вершины с минимальными значениями остальных координат, обходя против часовой стрелки.
- Перейти к следующему значению координаты по третьему измерению.

Теперь вершины нужно собрать в блоки. Для этого используется следующий формат описания:

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

Здесь *hex* обозначает шестигранник — основную форму ячейки в OpenFOAM, далее следует список номеров вершин в том порядке, в котором они заданы в своём словаре (*vertices*), начиная от нуля. Следующий список обозначает число клеток по каждой из осей координат. Отметим, что текущая задача по своей сути одномерная, а значит по двум осям требуется лишь одна клетка. Ключевое слово *simpleGrading* обозначает сгущение сетки согласно списку параметров сгущения, представляющих собой отношений длин самой правой ячейки к самой левой для каждой из осей. Так как нам одинаково интересна детализация результата по всей оси, то сгущение в данном случае не требуется.

После, необходимо определить словарь ребер блоков (*edges*). По умолчанию, ребра считаются прямыми линиями и нет необходимости специально указывать это, поэтому данный словарь можно оставить пустым. Случай криволинейных ребер будет рассмотрен в следующей лабораторной работе.

Затем, задаются границы сетки, состоящие из отдельных участков (*patch*), на которых потом будут заданы граничные условия. Каждому участку необходимо определить грани при помощи следующего синтаксиса:

```
boundary
(
    patchname
    {
        type patch;
        faces
        (
        (1 2 6 5)
        (0 4 7 3)
        );
    }
);
```

Здесь *patchname* — название участка, *patch* — тип границы, *faces* определяет список граней (грань — список из четырех вершин), содержащийся в данном участке. Порядок номеров вершин в нем должен соответствовать связным ребрам. Наиболее распространенные типы границ:

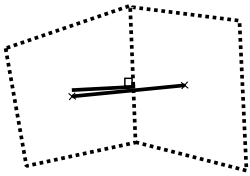
- patch — стандартная граница без дополнительной информации о сетке,

- empty — пустая (например, для граней по третьей оси для плоских двумерных задач),
- wall — стенка (например, для пристеночных функций в моделях турбулентности),
- wedge — осисимметричная, соответствующая боковым граням клина (*wedge*),
- symmetryPlane — плоскость симметрии,
- symmetry — условие симметрии, применяется к любым граням,
- cyclic — периодическая граница, для повторяющейся геометрии.

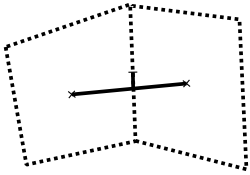
Список соединений поверхностей (*mergePatchPairs*) тоже оставим пустым, так как используется лишь один блок. В случае нескольких блоков с совпадающими гранями, они будут автоматически объединены. Необходимость в данном списке возникает, если их необходимо объединять по пересечению граней.

После этого для генерации сетки необходимо запустить в терминале команду *blockMesh* и, глядя на вывод этой команды, убедиться, что генерация прошла без ошибок. Для проверки получившейся сетки на качество можно использовать консольную утилиту *checkMesh*. Её вывод в том числе содержит статистику по числу точек, ячеек, граней, информацию о топологии и метрики качества, такие как соотношение стороны, скошенность и неортогональность. Также полезно проверить минимальный объем ячейки (*minVolume*) на неотрицательность.

Неортогональность ячеек сетки определяется углом между вектором, соединяющим центры соседних ячеек и нормалью к смежной грани. Чем меньше среднее значение неортогональности, тем более точной получается дискретизация градиента диффузионного члена уравнения. Хорошей можно считать сетку, имеющую среднее значение неортогональности менее 30 градусов, а приемлемой менее 70.



Скошенность представляет собой отношение расстояния между центром смежной стороны и вектором между центрами ячеек к длине последнего. Чем меньше значение скошенности, тем более точной получается интерполяция на грань ячейки, что улучшает точность дискретизации конвективного и диффузионного членов.



Соотношение сторон — это отношение между самой длинной и самой короткой сторонами ячейки, значения сильно отличные от единицы приводят к смазанности градиентов и замедлению сходимости.

После выполнения следующих команд:

```
1    blockMesh # создаем сетку
2    checkMesh # проверяем
```

в каталоге **constant** должен появиться подкаталог **polyMesh**, содержащий сгенерированную сетку. Необходимо отметить, что команды здесь и далее запускаются в каталоге с задачей (**sodTube**).

Задание начальных и граничных условий

OpenFOAM позволяет работать только в трехмерной правосторонней прямоугольной системе координат, для двумерных задач необходимо задавать специальные граничные условия на гранях.

Граничные условия задаются в файлах в каталоге, соответствующей начальному времени (в данном случае **0**), название файлов

соответствует определяемым величинам. Хорошим тоном является использование каталога **0.orig**, где хранится оригинальная копия граничных условий задачи. В данной лабораторной работе используется решатель для сжимаемого потока и необходимо задать начальные распределения скоростей (*U*), давлений (*p*) и температур (*T*). Для примера, рассмотрим файл *p* для начальных и граничных условий по давлению (листинг 1.6).

В шапке заголовка в этот раз задается класс *volScalarField*, соответствующий заданию поля скаляров. Для примера задания векторного поля, рекомендуем изучить содержимое файла *U* (листинг 1.7). Далее следует ключевое слово *dimensions* и вектор размерности поля, соответствующий степеням при размерностях системы СИ:

$$x = \text{кг}^{a_1}\,\text{м}^{a_2}\,\text{с}^{a_3}\,\text{К}^{a_4}\,\text{моль}^{a_5}\,\text{ампер}^{a_6}\,\text{кандела}^{a_7}. \tag{1.6}$$

Для давления в паскалях необходимо взять $a_1 = 1, a_2 = -1, a_3 = -2$, что соответствует вектору [1 -1 -2 0 0 0 0].

Далее определяются значения внутри расчетной области (*internalField*) и на границах (*boundaryField*). На текущий момент файл содержит равномерное нулевое поле давлений:

```
    internalField uniform 0;
```

Граничное условие на правой и левой границе расчетной области трубы соответствует нулевому градиенту (условие Неймана), а условие *empty* необходимо для игнорирования двух лишних пространственных направлений:

```
boundaryField
{
    sides
    {
        type            zeroGradient;
    }
    empty
    {
        type            empty;
    }
}
```

Для того, чтобы узнать возможные типы граничных условий для данного параметра, можно воспользоваться выводом консольной утилиты *foamHelp*:

```
foamHelp boundary -field p
```

Равномерные начальные и граничные условия можно задавать прямо в файле, однако, для более сложных конфигураций удобно воспользоваться утилитой setFields. Для её использования в каталоге **system** необходимо создать файл *setFieldsDict* со следующим содержанием (листинг 1.8).

Здесь основными настройками являются значения по умолчанию (*defaultFieldValues*), которые задают значения во всем поле параметра, и определение областей (*regions*), содержащие разные физические и геометрические опции для задания начальных условий. Для примера введем начальное условие в правой части трубы, для этого создадим словарь *boxToCell*, где ключевое слово *box* соответствует прямоугольнику, определяемого двумя противоположными точками, а *fieldsValues* содержит значения скаляров внутри этого прямоугольника:

```
boxToCell
{
    box (0 0 0) (0.5 1 1);
    fieldValues
        (
            volScalarFieldValue T 0.00278746
            volScalarFieldValue p 0.1
        );
}
```

Для задания новых начальных условий теперь необходимо запустить утилиту *setFields*, которая изменит содержимое файлов начальных и граничных условий в каталоге **0**. Для восстановления исходных значений необходимо скопировать файлы из резервного каталога **0.orig** в каталог **0**:

```
rm -r 0 # удаляем старый каталог
cp 0.orig 0 -r # копируем резервный на место старого
setFields # задаем начальные условия
```

Подробнее о командах *rm* и *cp* возможно узнать в приложении 1.2.

Свойства сплошной среды

Параметры вычисления свойств среды расположены в каталоге **constant**. Например, файл *turbulenceProperties* (листинг 1.9) содержит параметры турбулентности потока и будет подробнее рассмотрен в четвертой лабораторной работе. В данной работе рассматривается случай ламинарного течения. В файле *thermophysicalProperties* (листинг 1.10) происходит выбор физико-химических моделей и соответствующих констант, в словаре *thermoType* содержатся следующие ключевые слова:

- *type* — определяет тип термодинамических моделей в зависимости от сжимаемости течения,
- *mixture* — необходимо для выбора типа реагирующей смеси, при отсутствии изменения состава смеси используется *pureMixture*,
- *transport* — задает модели переноса, значение *const* соответствует постоянной вязкости,
- *thermo* — отвечает за расчет удельной теплоемкости при постоянном давлении c_p, выбранное значение $hConst$ соответствует постоянному значению,
- *equationOfState* — определяет уравнение состояние, *perfectGas* — это модель совершенного газа, для которого $p = \rho RT$,
- *energy* — позволяет выбрать вид энергии, необходимо ли использовать внутреннюю энергию, энтальпию и включать ли энтальпию образования. Значение *sensibleInternalEnergy* приводит к расчету через обыкновенную удельную энтальпию h. Выбор данного показателя может повлиять на стабильность численных методов.

Далее необходимо задать константы для выбранных моделей, также как и везде, они записываются в системе СИ. Ключевое слово *species* содержит число молей (по умолчанию 1) и молекулярный вес компоненты смеси. В словаре *thermodynamics* определяются термодинамические константы, такие как удельная теплоемкость c_p, в словаре *transport* определены параметры модели переноса, например, вязкость μ и число Прандтля Pr, через которое

может быть выражена теплопроводность: $\kappa = c_p\mu/Pr$. В данной лабораторной работе используется невязкий нетеплопропроводный газ, поэтому $\mu = 0$.

Выбор и настройка решателя

Наконец, необходимо настроить численные методы. Для данной задачи применим решатель *rhoCentralFoam*, который будет описан более подробно в третьей лабораторной работе, выбрав соответствующее значение для ключевого слова *application* (листинг 1.11). Следом идут параметры:

- Физического времени расчета. Ключевое слово *startFrom* — соответствует выбору начального времени, если используется *startTime*, то расчет будет происходить с времени указанного в *startTime*, чтобы продолжать расчет с последнего записанного времени, необходимо выбрать значение *latestTime*. Ключевые слова *stopAt* и *endTime* определяют конечное время, которое чаще выбирается из физических соображений, также может быть задан критерий остановки.

- Шаг по времени. Сам шаг определяется значением в *deltaT*, а также параметрами *adjustTimeStep*, *maxCo*, *maxDeltaT*. Если *adjustTimeStep* выключен (*no*), то вычисления происходят по выбранному в *deltaT* шагу. Однако, часто возможно ускорить вычисления, если ограничить шаг по времени оптимальным максимальным числом Куранта (чаще всего это $Co = \Delta t|U|/\Delta x < 1$), при этом полезно также ограничить максимальный шаг по временем, значение *maxDeltaT*.

- Запись в файл контролируется параметром *writeControl*, который изменяет период записи результатов в файлы. Значение *adjustable* соответствует записи на каждом интервале, определенном ключевым словом *writeInterval*. Параметры *writeFormat, writePrecision, writeCompression* определяют формат записи (текстовый/бинарный), точность результатов в знаках после запятой и включение сжатия выходных данных. При помощи *purgeWrite* можно ограничить количество хранимых результатов, например, установленное значение 5 приводит к сохранению данных для последних 5 времен. Название каталогов с результата-

ми можно изменить при помощи параметров *timeFormat*, *timePrecision*.

- Наконец, значение ключа *runTimeModifiable* отвечает за считывание настроек для каждого временного шага, тем самым регулируя возможность изменять настройки во время вычислений.

Файл *fvSchemes* (листинг 1.12) посвящен настройке схем дискретизации и ограничителей потока. Например, *fluxScheme* определяет римановский решатель для вычисления потоковых членов, параметры *ddtSchemes*, *gradSchemes*, *divSchemes*, *laplacianSchemes* — методы дискретизации дифференциальных операторов, *interpolationSchemes* — метод интерполяции из центра ячейки на ребро, *snGradSchemes* — задает аппроксимацию нормальных производных на границе.

После конечно-объемной дискретизации система уравнений в частных производных приводится к системе линейных уравнений. Методы и параметры решателей линейных уравнений описаны в файле *fvSolution* (листинг 1.13) в словаре *solvers* для каждой неизвестной переменной. Можно видеть, что для плотности задан метод прогонки (*diagonal*), для скорости и энергии — метод релаксации (*smoothSolver*). Параметры *tolerance* и *relTol* определяют абсолютную и относительную точность для итерационного процесса. Отметим, что в поставляемых с OpenFOAM примерах обычно заданы оптимальные значения по умолчанию, править их необходимо при возникновении численных артефактов и проблем со сходимостью.

Решение

Наконец, построив расчетную сетку, можно приступить к решению задачи. Для этого достаточно вызвать команду:

```
1 rhoCentralFoam
```

И увидеть текстовый вывод данной команды, содержащий сначала заданные параметры вычислений и следующую информацию:

```
deltaT = 1.20048e-06
Mean and max Courant Numbers = 0.00638479 0.00673935
Time = 1.20048e-06

diagonal:  Solving for rho, Initial residual = 0,
Final residual = 0, No Iterations 0
diagonal:  Solving for rhoUx, Initial residual = 0,
Final residual = 0, No Iterations 0
diagonal:  Solving for rhoE, Initial residual = 0,
Final residual = 0, No Iterations 0
ExecutionTime = 0.02 s  ClockTime = 0 s
```

Здесь указаны три времени: физическое время симуляции (*Time*), процессорное время (*ExecutionTime*) и реальное время (*ClockTime*). Также дана информация о числах Куранта, невязках и итерациях, но так как вычисления для невязкого потока происходят в данном случае явно, все невязки равны нулю и коррекция не производится. Для диагностики полезно журналировать вывод, перенаправив его в текстовый файл, для этого необходимо запустить *rhoCentralFoam* с командой перенаправления *tee*:

```
rhoCentralFoam | tee rhoCentralFoam.log # запишем результат
```

В процессе работы алгоритма будут создаваться папки, соответствующие настроенным временным отрезкам записи результатов на диск. Если работа команды завершилась без ошибок, то вычисление прошло успешно, однако, это еще не означает отсутствие численных ошибок и сходимости решения. Самым простым способом анализа служит визуализация решения, где многообещающим признаком получившейся симуляции является наблюдение ясной картины получившегося течения, не меняющейся со временем.

Отметим, что итоговая последовательность команд может быть записана в виде сценария (листинг 1.15), который традиционно сохраняют в файл с именем *Allrun*. Это упрощает повторное решение задачи, например, при изменении параметров. Для запуска сценария достаточно ввести:

```
./Allrun
```

Анализ полученных результатов

В комплекте с OpenFOAM поставляется мощный инструмент для визуализации — ParaView. Для обзора текущей задачи достаточно ввести скрипт запуска ParaView в каталоге решения командой:

```
paraFoam
```

После ввода откроется графический интерфейс, где можно видеть трехмерную визуализацию задачи, и в котором, среди прочего, можно строить графики, изолинии, амплитудно-частотные характеристики, совершать алгебраические преобразования над полями, создавать анимации и многое другое. Более подробно о ParaView можно узнать в приложении **??**.

Однако, в рамках данной работы мы рассмотрим более простой инструмент для создании выборки данных (*sampling*). Для этого создадим словарь *sample* в каталоге **system** (листинг 1.14). Его синтаксис описывает взятие равномерных (*uniform*) проб из 100 точек (*nPoints*) на отрезке по горизонтальной оси с использованием линейной интерполяции между центром и гранями (*cellPoint*, и запись в текстовый файл в формате (*setFormat*) значений, разделенных запятыми *csv — comma-separated values*. Запуск команды

```
postProcess -func sample
```

создаст каталог **postProcessing**, содержащий результаты проб для каждой из записанных временных точек решения. По получившимся точкам в дальнейшем возможно построить график.

Задание для лабораторной работы

- Провести численное моделирование задачи в OpenFOAM с начальными данными согласно варианту.
- Построить графики T, p, ρ в моменты времени $t = 0$, $t = 0.1$, $t = 0.2$ с.
- Проверить сеточную сходимость: сравнить результаты на сетках разной густоты, с числом ячеек: 10^n, $n = 1, 2, 3, 4, 5$ ячеек.

- Написать на любом языке программирования расчетный код для аналитического решения и сравнить с полученными результатами*.

Вариант	Отрезок	T_0/p_0 слева	T_0/p_0 справа	Положение диафрагмы
1	$(0, 1)$	$0.003458/1$	$0.00278/0.1$	0.5
2	$(0, 1)$	$0.00278/0.1$	$0.003458/1$	0.5
3	$(-1, 1)$	$0.003458/1$	$0.00278/0.1$	0
4	$(-1, 1)$	$0.00278/0.1$	$0.003458/1$	0.5
5	$(-5, 5)$	$0.03458/2$	$0.00278/0.2$	0
6	$(-5, 5)$	$0.00278/2$	$0.003458/0.2$	1

Таблица 1.1: Варианты для первой лабораторной работы

1.2. Краткое описание команд терминала

Исторически самым экономичным способом общения с системой была системная консоль (терминал), позволяющая посылать текстовые команды и получать текстовый вывод в ответ. Этот способ взаимодействия до сих пор незаменим для администрирования систем при ограниченном пропускной способности канала или если использование графического интерфейса было бы избыточно. Удобства использование консоли предоставляет командная оболочка (например, bash), сочетающая обработку текстовых команд и скриптовый язык программирования, позволяющий потенциально неограниченные возможности использования. В операционных системах на основе GNU/Linux используются программы из проекта *GNU* Далее кратко будут описаны некоторые из них и показаны примеры использования.

ls — просмотр содержимого папки (**lis**t).

```
ls # просмотр файлов в текущей папке
```

```
ls -l /home # просмотр файлов с детализированной информацией в
            # папке /home
```

pwd — напечатать текущую папку (**p**rint**w**orking**dir**ectory)

cd — смена рабочей папки (**c**hange **d**irectory), поддерживает абсолютные и относительные пути.

```
cd /home/ # выбрать рабочей корневую папку /home
```

```
cd video # выбрать рабочей папку video в текущей (спуститься
         # на уровень ниже)
```

```
cd .. # выбрать рабочей родительскую папку для текущей
      # (подняться на уровень выше)
```

mv — перемещение содержимого (**m**ove)

```
mv file /home # переместить файл в папку /home
```

```
mv file folder /home # переместить файл и папку в папку /home
```

```
mv * /home # переместить все содержимое текущей папки в папку
           # /home
```

cp — копирование файлов. Для копирования папок с содержимым необходимо использовать флаг -r.

```
cp file /home # скопировать файл в текущей папке в /home
```

```
cp -r /mnt/folder /home # скопировать папку вместе с содержимым
                        # в папку /home
```

```
cp *txt /home # скопировать все файлы, чье имя кончается на txt,
              # в папку /home
```

rm — удалить файл (**rem**ove). Для удаления папок с содержимым необходимо использовать флаг -r. Использовать с большой осторожностью.

```
rm file # удалить файл
```

```
rm -r /home/folder # удалить папку folder
```

```
rm -rf / # удалить всю систему и остаться ни с чем
```

mkdir — создание папки (**ma**k**e** **dir**ectory)

```
mkdir /home/folder # создать папку folder
```

man — чтение встроенных мануалов (man-page от англ. **man**ual). Команда позволяет разобраться в любой программе, не выходя из консоли. Также краткую справку можно получить, набрав команду с ключом help.

```
man man # получить подробный мануал для команды man
```

```
man --help # получить краткую справку для команды man
```

cat — вывод содержимого файла (con**cat**enate).

less — просмотр содержимого файла. Листать содержимое можно при помощи стрелок, для выхода необходимо нажать клавишу q.

nano — простейший текстовый редактор. Для управления используются стрелки и сочетания клавиш с Ctrl (^), указанные на экране (например, Ctrl+q — выход).

Полезные команды, которые можно изучить далее самостоятельно: **tee**, **head**, **tail**, **tree**, **touch**, **echo**, **uname**, **history**, **grep**, **wc**, **find**, **bash**, **vi** ...

1.3. Листинги кода

Для лабораторной работы №1

```
/*---------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2206                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
```

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

scale   1;

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

);

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

edges
(
);

boundary
(
    sides
    {
        type patch;
        faces
        (
            (1 2 6 5)
            (0 4 7 3)
        );
    }
    empty
    {
        type empty;
        faces
        (
            (0 1 5 4)
            (5 6 7 4)
            (3 7 6 2)
            (0 3 2 1)
        );
    }
);

mergePatchPairs
(
```

```
);


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

Рис. 1.5: Файл blockMeshDict для задачи Сода

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2206                                  |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    arch        "LSB;label=32;scalar=64";
    class       volScalarField;
    location    "0";
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [1 -1 -2 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    sides
    {
        type            zeroGradient;
    }
    empty
    {
        type            empty;
    }
}


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

Рис. 1.6: Файл p для задачи Сода

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2206                                  |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    arch        "LSB;label=32;scalar=64";
    class       volVectorField;
    location    "0";
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

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

internalField   uniform (0 0 0);

boundaryField
{
    sides
    {
        type            zeroGradient;
    }
    empty
    {
        type            empty;
    }
}


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

Рис. 1.7: Файл p для задачи Сода

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2206                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    arch        "LSB;label=32;scalar=64";
    class       dictionary;
```

```
    location    "system";
    object      setFieldsDict;
}

defaultFieldValues
(
    volVectorFieldValue U (0 0 0)
    volScalarFieldValue T 0.00348
    volScalarFieldValue p 1
);

regions
(
    boxToCell
    {
        box (0 0 0) (0.5 1 1) ;
        fieldValues
        (
            volScalarFieldValue T 0.00278
            volScalarFieldValue p 0.1
        );
     }
);
```

Рис. 1.8: Файл setFieldsDict для задачи Сода

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2206                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      turbulenceProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

simulationType  laminar;


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

Рис. 1.9: Файл turbulenceProperties для задачи Сода

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2206                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      thermophysicalProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

thermoType
{
    type            hePsiThermo;
    mixture         pureMixture;
    transport       const;
    thermo          hConst;
    equationOfState perfectGas;
    specie          specie;
    energy          sensibleInternalEnergy;
}

mixture
{
    specie
    {
        nMoles          1;
        molWeight       28.96;
    }
    thermodynamics
    {
        Cp              1004.5;
        Hf              0;
    }
    transport
    {
        mu              0;
        Pr              0.705;
    }
}


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

Рис. 1.10: Файл thermophysicalProperties для задачи Сода

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2206                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    class       dictionary;
    object      controlDict;
    version     2.0;
    format      ascii;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     rhoCentralFoam;

startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         0.2;

deltaT          1e-06;

writeControl    adjustable;

writeInterval   0.1;

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression off;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;

adjustTimeStep  yes;

maxCo           0.2;

maxDeltaT       1;


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

Рис. 1.11: Файл controlDict для задачи Сода

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2206                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

fluxScheme          Kurganov;

ddtSchemes
{
    default         Euler;
}

gradSchemes
{
    default         Gauss linear;
}

divSchemes
{
    default         none;

    div(tauMC)      Gauss linear;
}

laplacianSchemes
{
    default         Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;

    reconstruct(rho) vanLeer;
    reconstruct(U)  vanLeerV;
    reconstruct(T)  vanLeer;
}

snGradSchemes
{
    default         corrected;
```

```
}


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

Рис. 1.12: Файл fvSchemes для задачи Сода

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2206                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    "(rho|rhoU|rhoE)"
    {
        solver          diagonal;
    }

    U
    {
        solver          smoothSolver;
        smoother        GaussSeidel;
        nSweeps         2;
        tolerance       1e-09;
        relTol          0.01;
    }

    h
    {
        $U;
        tolerance       1e-10;
        relTol          0;
    }
}


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

Рис. 1.13: Файл fvSolution для задачи Сода

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2206                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      sample;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

type            sets;
libs            (sampling);
interpolationScheme cellPoint;
setFormat       csv;

fields          (T rho p);

sets
{
    data
    {
        type    uniform;
        axis    x;
        start   (0 0 0);
        end     (1 0 0);
        nPoints 100;
    }
}


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

Рис. 1.14: Файл sample для задачи Сода

```
#!/bin/bash

# Командный сценарий для лабораторной работы №1

blockMesh
rm -rf 0
cp 0.orig 0 -r
setFields
rhoCentralFoam | tee rhoCentralFoam.log
postProcess -func sample
```

Рис. 1.15: Сценарий для запуска лабораторной работы

Список литературы

1. Sod G. A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws // Journal of Computational Physics. — 1978. — Т. 27, № 1. — С. 1—31.
2. Toro E. F. Riemann Solvers and Numerical Methods for Fluid Dynamics. — Springer Berlin Heidelberg, 2009.