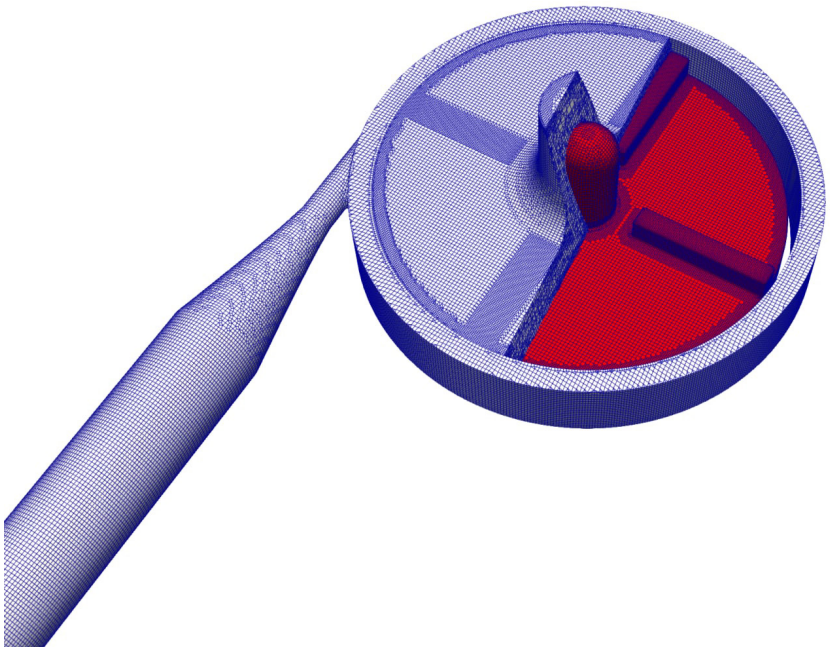
А.Н. НУРИЕВ, О.Н. ЗАЙЦЕВА, О.С. ЖУЧКОВА

МОДЕЛИРОВАНИЕ ГИДРОДИНАМИЧЕСКИХ ПРОЦЕССОВ В ПРОГРАММНОМ КОМПЛЕКСЕ OPENFOAM. СОЗДАНИЕ РАСЧЕТНЫХ СЕТОК С ПОМОЩЬЮ БИБЛИОТЕКИ CFMESH



КАЗАНСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ

А.Н. НУРИЕВ, О.Н. ЗАЙЦЕВА, О.С. ЖУЧКОВА

МОДЕЛИРОВАНИЕ ГИДРОДИНАМИЧЕСКИХ ПРОЦЕССОВ В ПРОГРАММНОМ КОМПЛЕКСЕ OPENFOAM. СОЗДАНИЕ РАСЧЕТНЫХ СЕТОК С ПОМОЩЬЮ БИБЛИОТЕКИ CFMESH

Учебное пособие

КАЗАНЬ
2022

УДК 532.5+519.6+004.9(075.8)
ББК 22.253.3:22.18я73
Н90

Печатается по рекомендации учебно-методической комиссии Института математики и механики имени Н.И. Лобачевского Казанского (Приволжского) федерального университета (протокол № 1 от 8 сентября 2022 г.), кафедры аэрогидромеханики Института математики и механики имени Н.И. Лобачевского Казанского (Приволжского) федерального университета (протокол № 1 от 31 августа 2022 г.)

Рецензенты:
доктор физико-математических наук, ведущий научный сотрудник НИЛ ИБиПС ИММ КФУ **А.Г. Егоров**;
доктор физико-математических наук, доцент кафедры прикладной математики МГТУ имени Н.Э. Баумана **И.К. Марчевский**

Нуриев А.Н.
Н90 Моделирование гидродинамических процессов в программном комплексе OpenFOAM. Создание расчетных сеток с помощью библиотеки cfMesh: учебное пособие / А.Н. Нуриев, О.Н. Зайцева, О.С. Жучкова. – Казань: Издательство Казанского университета, 2022. – 72 с.

Учебное пособие основано на курсе лекций и практических занятий, проводимых в Институте математики и механики имени Н.И. Лобачевского КФУ, Институте геологии и нефтегазовых технологий КФУ и на кафедре информатики и прикладной математики КНИТУ (КХТИ) для студентов старших курсов и магистрантов. В нем изложены методы построения и оценки качества расчетных сеток в OpenFOAM для плоских и трехмерных задач гидродинамики.

Пособие может быть полезно также аспирантам, инженерам и научным сотрудникам в качестве руководства по построению сеток в OpenFOAM.

УДК 532.5+519.6+004.9(075.8)
ББК 22.253.3:22.18я73

Содержание

Who owns the mesh, owns the solution.

H. Jasak – Wikki Ltd.

ВВЕДЕНИЕ

1. Что такое хорошая расчетная сетка?

Не существует единых общепринятых характеристик или показателей, по которым можно бы было однозначно оценить качество расчетной сетки. Фактически многое определяется условиями задачи (для которой эта сетка генерируется), особенностями расчетных схем и численных методов. О некоторых недостатках, в частности, о нехватки разрешающей способности (способности описывать все структурные особенности течения) используемой сетки, можно узнать только на этапе постпроцессинга при анализе сходимости или сравнении полученных численных результатов с экспериментальными данными или аналитическим решением (для какого-нибудь частного случая). Поэтому при построении сетки для решения конкретного класса задач нужно полагаться на накопленный по данному направлению опыт.

В то же время многие недостатки сетки можно выявить, анализируя следующие характеристики:

- неортогональность,
- скошенность,
- соотношение сторон ячеек,
- равномерность.

Их можно легко рассчитать для готовой сетки и использовать в качестве индикаторов или показателей качества сетки. Если значения по этим характеристикам сильно отклоняются от нормальных, то в большинстве случаев это указывает на плохую сетку, на которой будет невозможно выполнить качественный расчет.

2. Ячейки какой формы предпочтительнее для расчетных сеток?

Тетраэдр, гексаэдр или другие многогранники? Каждый тип ячеек имеет свои собственные преимущества и недостатки, когда речь идет о заполнении расчетной области, об аппроксимации градиентов и интерполяции потоков. Многогранные ячейки с шестью и более гранями лучше аппроксимируют градиенты по сравнению с тетраэдрами, но высокие значения скошенности на таких сетках могут свести на нет все преимущества. Чем больше граней имеют ячейки в форме многогранника, тем больше вероятность того, что из-за высокой скошенности построенное на этой сетке решение будет осцилляционным. Среди всех типов ячеек тетраэдр имеет минимальное количество граней, поэтому в такой ячейке градиент будет рассчитываться наименее точно. Однако сетку из таких ячеек можно построить практически при любой геометрии области. Во многих случаях сетки, состоящие из гексаэдров, являются золотой серединой. При том же количестве ячеек сетки из шестигранников позволяют получить более точное решение в сравнении с тетраэдрическими сетками, особенно в тех случаях, когда удается расположить грани ячеек сетки (линии сетки) параллельно потоку. В сложных областях можно использовать смешанные сетки, содержащие, помимо гексаэдров, тетраэдры и другие многогранники.

3. Общие рекомендации при построении сеток

Дадим следующий список рекомендаций для построения хорошей расчетной сетки:

- По возможности всюду используйте шестигранные ячейки, старайтесь расположить грани ячеек параллельно потоку.
- Стремитесь построить сетку с минимальными значениями неортогональности и скошенности, со сбалансированным соотношением сторон ячеек и без скачкообразных изменений размеров ячеек (с равномерным изменением длин сторон).

- Всегда проверяйте качество сетки. Помните, что одна отдельная ячейка может вызвать расхождение расчетной схемы или внести существенную ошибку в результат вычислений.
- Увеличение количества ячеек во многих случаях может повысить точность решения, но это ведет и к возрастанию вычислительных затрат. Кроме того, непродуманное измельчение сетки может испортить ее качество: повысить неортогональность, скошенность и привести к скачкообразному изменению размеров ячеек.
- Чтобы количество ячеек было минимально необходимым, используйте неоднородные сетки. Сгущайте и измельчайте ячейки только в выбранных областях. Старайтесь, чтобы переходы между ячейками разных размеров были постепенными.
- Если вы не используете пристеночные функции (для моделирования турбулентности), помните, что сетка, прилегающая к границам обтекаемых тел, должна быть достаточно подробной, чтобы разрешать течение в пограничном слое.
- Качественная сетка позволит избежать «синдрома GIGO» (garbage in, garbage out - мусор на входе, мусор на выходе).
- Можно считать построенную сетку однозначно хорошей, если результаты расчета, полученные на ней, физически реалистичны, надежны, точны и получены за минимально возможное (или близкое к нему) расчетное время.

4. Создание сеток в OpenFOAM

В данном учебном пособии мы рассмотрим работу с сетками с помощью инструментария платформы OpenFOAM. Дальнейший материал подготовлен так, что для освоения основного материала потребуются минимальные навыки по работе в данном пакете. В то же время для понимания некоторых деталей будет чрезвычайно полезным ознакомиться с устройством и средствами моделирования в OpenFOAM более подробно. Для этих целей рекомендуем воспользо-

ваться пособием для начинающих пользователей [1]. Более подробную информацию о решателях и утилитах данного пакета можно найти в следующих источниках [2,3].

Для генерации расчетных сеток в OpenFOAM имеется обширный функционал, к которому относятся утилиты **blockMesh**, **snappyhexMesh** и библиотека **cfMesh**. Кроме того, на платформе реализованы многочисленные инструменты по модификации созданных сеток, а также инструменты по импортированию сеток, созданных в сторонних генераторах. Все эти средства дают пользователю OpenFOAM огромные возможности, однако отсутствие графического интерфейса ко всем инструментам и утилитам снижает уровень доступности этих средств для начинающих пользователей.

Основное внимание в данной книге мы уделим работе с библиотекой **cfMesh**. В первой главе рассмотрим основной функционал библиотеки **cfMesh**, представим перечень ключевых настроек, опций и параметров. Во второй главе обсудим вопросы расчета показателей качества сеток, определения и визуализации плохих (с низкими показателями качества) областей средствами OpenFOAM. В третьей главе разберем примеры создания (псевдо) двумерных и трехмерных расчетных сеток в контексте практического применения материала первых двух глав.

Глава 1. БИБЛИОТЕКА cfMesh

1. Генераторы сеток в библиотеке cfMesh

cfMesh – это библиотека для автоматического построения расчетной сетки в двумерном и трехмерном пространстве [4], входящая в состав всех актуальных версий OpenFOAM и foam-extend. Она используется для построения сетки в областях с произвольной геометрией или около объектов произвольной формы, геометрия которых (в качестве входных данных) задается в виде поверхностной триангуляции. Она имеет следующие встроенные механизмы-утилиты (meshing workflows) для генерации сеток, различающиеся по типу генерируемых ячеек в шаблоне:

- **cartesianMesh** создает трехмерные сетки, состоящие преимущественно из шестигранных ячеек (с другими многогранниками в переходных областях между ячейками разного размера);
- **cartesian2DMesh** создает двумерные сетки, состоящие преимущественно из шестигранных ячеек;
- **tetMesh** создает сетки, состоящие из тетраэдрических ячеек;
- **pMesh** создает сетки, состоящие из произвольных многогранных ячеек; может применяться для «грязных геометрий», то есть входных геометрических моделей содержащих неточности и ошибки.

2. Задние геометрии расчетной области

Для задания геометрии (на входе) могут быть использованы следующие форматы файлов: fms, ftr и stl. Геометрию в других форматах можно предварительно конвертировать с помощью утилиты **surfaceConvert**, входящей в состав OpenFOAM.

Для корректного учета геометрии кромок, углов, торцов и ребер (sharp edges; далее будем все это называть ребрами) подгружаемых объектов, необходимо уделить особое внимание выделению этих эле-

ментов до запуска процедуры генерации сетки. Сохраняют свою геометрию ребра, находящиеся на стыке двух разных участков границы - патчей (patches), а также выделенные края объектов (feature edges). Выделенные края объектов могут быть созданы с помощью утилиты **surfaceFeatureEdges**. Другие границы могут не сохранить свою геометрию (например, будут сглажены) в областях с недостаточной разрешающей способностью сетки.

Базовый формат геометрии для **cfMesh** - fms. В этом формате патчи (patches), подмножества (subsets) и выделенные края объектов (feature edges) хранятся в одном файле.

Файл поверхности fms можно создать из stl файла с помощью утилиты **surfaceFeatureEdges**, которая также определяет особенности геометрии для уточнения ребер:

```
surfaceFeatureEdges <.stl file> <.fms file>
```

3. Создание проекта OpenFOAM для генерации сетки

Для запуска генераторов сетки потребуется создать проект OpenFOAM. Он представляет собой директорию, которая должна содержать следующий минимальный набор файлов и папок:

- папку *constant* с вложенной папкой *triSurface*, содержащей в свою очередь файл с геометрией в формате fms, ftr или stl;
- папку *system*, содержащую конфигурационные файлы **controlDict** и **meshDict.**

Дерево каталогов проекта можно представить следующим образом:

```
├── constant
│   └── triSurface
│       └── Geometry.stl
└── system
    ├── controlDict
    └── meshDict
```

Настройки, представленные в файле **controlDict**, для описанного далее процесса генерации сетки не имеет большого значения. Настройки можно скопировать из простейшего примера. Ниже представлено содержимое файла **controlDict** из папки *cavity* (готовый стандартный проект).

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2112                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     icoFoam;
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;

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

Другой конфигурационный файл – **meshDict** – содержит все настройки для генераторов сеток. Их описание представлено далее.

4. Настройка конфигурационного файла meshDict

4.1. Структура конфигурационного файла и основные параметры

Конфигурационный файл **cfMesh** (в терминологии OpenFOAM такие конфигурационные файлы называются словарями – dictionary) meshDict находится по адресу: *./каталог_проекта/system*. Он имеет следующую структуру:

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2112                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
*---------------------------------------------------------------------------*/

FoamFile
{
version 2;
format  ascii;
class   dictionary;
location"system";
object  meshDict;
}

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
//Выше этой строчки находится заголовок конфигурационного файла OpenFOAM
//Комментарии (как этот) в конфигурационных файлах вставляются в стиле C++

surfaceFile "..."; // обязательный параметр
maxCellSize ...; // обязательный параметр

//далее следует описание опций, используемых при необходимости
boundaryCellSize ...;
minCellSize … ;

localRefinement{
…
}
```

```
objectRefinements{
…
}
surfaceMeshRefinement{
…
}
anisotropicSources{
...
}
boundaryLayers{
…
}
renameBoundary{
…
}
```

Как и любой конфигурационный файл OpenFOAM, он начинается с заголовка. Заголовок не содержит каких-либо настроек для **cfMesh**, поэтому далее по тексту при описании meshDict его будем опускать.

Минимальные параметры для запуска утилит библиотеки **cfMesh**, которые необходимо указывать в конфигурационном файле *system/meshDict* для генерации сетки – это:

- **surfaceFile** – путь к файлу с геометрией объекта;
- **maxCellSize** – максимальный размер ячейки.

Все другие параметры указываются в случае необходимости дополнительных настроек.

4.2. Настройка разрешающей способности сетки

4.2.1. Измельчение ячеек в областях

Один из ключевых механизмов повышения разрешающей способности сетки – измельчение ячеек в заданных областях путем дробления каждой ячейки на более мелкие.

Глобальные параметры, регулирующие измельчение ячеек сетки:

- **boundaryCellSize** – определяет размер всех граничных ячеек;
- **boundaryCellSizeRefinementThickness** – указывает толщину граничной области, где ячейки измельчаются до **boundaryCellSize**;
- **minCellSize** – активирует автоматическое измельчение сетки внутри области (путем дробления ячеек) и устанавливает минимальный размер ячейки.

Можно определять также более тонкие локальные настройки измельчения сеток с помощью опций **localRefinement**, **objectRefinements**, **surfaceMeshRefinement** и **edgeMeshRefinement**.

Опция **localRefinement** переопределяет размер ячеек для указанного патча (в примере ниже патч имеет название solid). С помощью параметров **cellSize** и **additionalRefinementLevels** можно задать или размер ячеек на границе, или количество уровней измельчения ячеек на границе (учитывая, что на каждом уровне гексаэдр разбивается на 8 равных частей). С помощью параметра **refinementThickness** можно также определить толщину граничной области возле указанного патча.

```
localRefinement
{
solid
{
//cellSize 1;
additionalRefinementLevels 4;
refinementThickness 1.2;
}}
```

Опция **objectRefinements** используется для повышения разрешающей способности сетки в заданных областях. Области с повышенной разрешающей способностью сетки можно определять с помощью линий (**line**), сфер (**sphere**), параллелепипедов (**box**) и конусов (**cone**, **hollowCone**).

Для задания области в форме параллелепипеда необходимо использовать параметр **type** со значением **box** и определить координаты центра (параметр **centre**) и размеры сторон параллелепипеда **lengthX**, **lengthY**, **lengthZ** в направлениях осей x, y и z соответственно.

```
objectRefinements
{
boxExample // название области (задается произвольно)
{
type box; // тип
centre (50 50 0); // координаты центра
lengthX 20;
lengthY 20;
lengthZ 1;
cellSize 0.5;
}
}
```

При необходимости измельчить сетку в осесимметричной области в форме цилиндра, усеченного или обыкновенного конуса, параметр **type** задается как **cone**, а геометрия области определяется двумя крайними точками на оси симметрии (вершиной и центром основания для конуса) и радиусом в каждой точке.

```
objectRefinements
{
coneExample // название области (задается произвольно)
{
type cone; // тип
p0 (10 10 0);// координаты первой точки
p1 (10 -10 0);// координаты второй точки
radius0 20;// радиус в первой точке
radius1 20;// радиус во второй точке
cellSize 0.5;
}
}
```

Для полых цилиндрических и конических областей (в плоском случае – колец) в настройках **objectRefinements** следует определить **type** как **hollowCone**. Геометрия в этом случае описывается двумя крайними (верхней и нижней) точками на оси симметрии и двумя радиусами для этих точек.

```
objectRefinements
{
hollowConeExample // название области (задается произвольно)
{
type hollowCone; // тип
p0 (10 10 0);// координаты первой точки
p1 (10 -10 0);// координаты второй точки
radius0_Inner 20;// внутренний радиус в первой точке
radius0_Outer 50;// внешний радиус в первой точке
radius1_Inner 20;// внутренний радиус во второй точке
radius1_Outer 50;// внешний радиус во второй точке
cellSize 0.5;
}
}
```

Для выбора сферической области измельчения сетки параметр **type** определяется как sphere и задаются координаты центра и радиуса области.

```
objectRefinements
{
sphereExample // название области (задается произвольно)
{
type sphere; // тип
centre (10 10 0) ;// координаты
radius 20;// радиус
cellSize 0.5;
refinementThickness 20;
}
}
```

Также для любой из вышеописанных областей нужно указать:

cellSize – размер ячеек внутри области, или

additionalRefinementLevels – количество уровней измельчения ячеек внутри области (по аналогии с соответствующим параметром опции **localRefinement**), начиная с ячеек размера **maxCellSize**.

С помощью параметра **refinementThickness** можно также задать глубину выхода измельченной сетки за границы обозначенной области. Еще одна опция для настройки разрешающей способности – **surfaceMeshRefinement** – позволяет задавать области для измельчения сетки как поверхности геометрических объектов, подгружаемых как stl файл.

```
surfaceMeshRefinement
{
surfaceExample // название области (задается произ-
вольно)
{
surfaceFile «surf.stl»; // тип
//cellSize 1;
additionalRefinementLevels 4;
refinementThickness 1.2;
}
}
```

Опция **edgeMeshRefinement** позволяет измельчать сетку в зонах кромок и ребер, задавая их с помощью vtk файла. Vtk файл, описывающий ребра, можно создать из fms файла геометрии при посредстве команды:

```
FMSToSurface файл.fms файл.vtk –exportFeatureEdges
```

Требуемый размер ячейки вблизи заданных объектов определяется параметром **cellSize** или с помощью **additionalRefinementLevels** – количество уровней измельчения относительно максимального размера ячейки. Данная опция применяется для улучшения качества сетки в области стыковки кромок и ребер с сеткой. Толщину зоны уточнения можно определять с помощью параметра **refinementThickness**.

```
edgeMeshRefinement
{
```

```
edgeExample // название области (задается произвольно)
{
edgeFile «edge.vtk»; // тип
//cellSize 1;
additionalRefinementLevels 4;
refinementThickness 1.2;
}
}
```

4.2.2. Сгущение сетки в области

Помимо измельчения ячеек в локальной области за счет дробления, можно улучшать локальную разрешающую способность в области путем сгущения ячеек. Для настройки сгущения или разрежения сетки используется опция **anisotropicSources**. В библиотеке реализовано два типа сгущения/разрежения **box** и **plane**.

Тип **box** определяет область в виде прямоугольного параллелепипеда, внутри которого можно провести сгущение/разрежение сетки в направлениях координатных осей. Параметры, определяющие данную область: **centre** – координаты центра, **lengthX, lengthY**, **lengthZ** – длины сторон параллелепипеда, расположенные параллельно осям x, y и z соответственно. Коэффициент сгущения/разрежения в направлении оси x задается с помощью параметра **scaleX**, аналогично в направлении осей y и z используются параметры **scaleY** и **scaleZ**. Допустимый диапазон изменения значений данных коэффициентов находится между нулем и бесконечностью. Значения меньшие 1 уменьшают размер ячеек в области пропорционально заданному коэффициенту, а значения большие 1 увеличивают размер ячеек.

```
anisotropicSources
{
box1
{
type box;
centre (0 0 0.5);
lengthX 6;
lengthY 6;
lengthZ 1;
```

```
 scaleX 0.125;
 scaleY 0.125;
}
}
```

Тип **plane** обычно используется в случаях, когда требуемое направление сгущения не совпадает с направлениями главных осей системы координат и/или область сгущения является полосой – областью, заключенной между двумя параллельными плоскостями. При этом одна из плоскостей задается точкой и нормалью, а другая строится автоматически на указанном расстоянии. Параметры, определяющие область сгущения:

- **origin** – указывает точку исходной плоскости;
- **normal** – определяет вектор нормали к плоскости;
- **scalingDistance** – указывает расстояние до второй плоскости в положительном направлении нормали;
- **scalingFactor** – коэффициент сгущения/разрежения в нормальном направлении, допустимый диапазон значений которого находится между нулем и бесконечностью. Значения коэффициента меньшие 1 обеспечивают сгущение сетки, а значения большие 1– разрежение.

```
anisotropicSources
{
    planeFWD
    {
    type plane;
    normal (1 0 0);
    origin (0 0 0.5);
    scalingDistance 6;
    scalingFactor 0.125;
    }
}
```

4.2.3. Пограничные слои сетки

Еще одним инструментом для повышения разрешающей способности сетки вблизи тела являются пограничные слои сетки. С помощью опции **boundaryLayers** можно создать такие слои около указанных патчей или на всех границах области. Они вставляются после создания основной сетки путем выдавливания (смещения) сформированной сетки от границы внутрь расчетной области и добавления в образовавшийся зазор (между сеткой и телом) новых погранслойных ячеек. Толщина слоя будет аналогична исходному размеру граничных ячеек. Пограничные слои ячеек могут охватывать несколько патчей, если они имеют общие вогнутые края или углы.

По умолчанию создание одного пограничного слоя включено в утилиты **cartesian2DMesh и cartesianMesh.**

Опция имеет следующие настраиваемые параметры, применяемые на всех границах:

- **nLayers** – количество слоев, которые будут сгенерированы в сетке (необязательный параметр, по умолчанию принят равным 1 или 0 для разных генераторов);
- **thicknessRatio** – это число большее 1, характеризующее отношение толщины двух последовательных слоев (необязательный параметр, по умолчанию принят равным 1);
- **maxFirstLayerThickness** – необязательный параметр, задающий максимально допустимую толщину первого пограничного слоя (при построении сетки она не может превысить указанного здесь значения).

Для определения параметров сеточных погранслоев на отдельных патчах нужно использовать опцию **patchBoundaryLayers**. В настройках **patchBoundaryLayers** для каждого выбранного патча можно указать локальные значения параметров **nLayers**, **thicknessRatio** и **maxFirstLayerThickness**.

По умолчанию количество слоев, генерируемых для каждого патча (если определены локальные значения), определяется общим

количеством слоев или максимальным количеством слоев, указанным в любом из сопряженных патчей, имеющих общий непрерывный погранслой с данным патчем. Активация параметра **allowDiscontinuity** (разрешение разрыва) гарантирует, что количество слоев, выставляемое для конкретного патча, не будет автоматически исправляться в соответствии с настройками сопряженных патчей.

```
boundaryLayers
{
nLayers 2;
thicknessRatio 1.2;
patchBoundaryLayers
{
square
{
nLayers 3;
thicknessRatio 1.4;
maxFirstLayerThickness 0.2;
allowDiscontinuity 0; // 0-выкл, 1-вкл
}
}
}
```

В настройках опции **boundaryLayers** можно указывать дополнительные параметры, влияющие на качество пограничных слоев. Они нужны в случаях, когда генерируется большое количество слоев с плавно изменяющейся толщиной. Параметр **optimiseLayer** включает или выключает настройки оптимизации, задаваемые в **optimisationParameters**. Опция **optimisationParameters** позволяет настроить следующие параметры:

- **nSmoothNormals** – количество итераций в процедуре сглаживания (изменения) векторов нормалей в пограничном слое. Значение по умолчанию равно 5.
- **maxNumIterations** – количество итераций в процедуре сглаживания. Значение по умолчанию равно 5.
- **featureSizeFactor** – это отношение между максимально допустимой толщиной слоя и оценочным размером элемента. Он ис-

пользуется для ограничения толщины слоя в областях с большими локальными значениями кривизны. Допустимый диапазон значений находится между 0 и 1. Значение по умолчанию – 0.3.

- **reCalculateNormals** включает (1) /выключает (0) процедуру выравнивания граней ячеек параллельно нормалям к поверхности тела. Этот параметр активен по умолчанию (равен 1).
- **relThicknessTol** управляет величиной максимального изменения толщины слоя между двумя соседними точками, вычисленной по отношению к расстоянию между точками. Допустимый диапазон значений находится между нулем и единицей. Более низкие значения уменьшают толщину слоя и обеспечивают равномерное распределение толщины.

```
boundaryLayers
{
optimiseLayer 1;// активирует сглаживание пограничного слоя, по умолчанию 0-выкл
untangleLayers 0;
optimisationParameters
{
nSmoothNormals 3;
maxNumIterations 4;
featureSizeFactor 0.6;
reCalculateNormals 1;
relThicknessTol 0.1;
}
}
```

Описанная выше процедура оптимизации не гарантирует отсутствие ячеек с отрицательным объемом в пограничном слое, и по умолчанию **cfMesh** выполняет сглаживание и «распутывание» ячеек до тех пор, пока их объем не станет положительным.

4.3. Сохранение/удаление ячеек в заданных пользователем областях

В алгоритмах генерации сетки, реализованных в **cfMesh**, процесс создания сетки начинается с создания так называемого шаблона сетки на основе заданного пользователем размера ячейки. Однако, если размер такой исходной ячейки в локальной области больше, чем размер геометрического объекта, то может возникнуть ситуация, при которой зазоры в геометрии будут заполнены сеткой. Также возможна и противоположная ситуация, когда сетка в узких (тонких) элементах геометрии может отсутствовать (например, тонкие каналы исчезнут), если указанный размер ячейки больше размера локального элемента. Для устранения таких ситуаций существуют следующие опции, позволяющие регулировать удаления ячеек в шаблонной сетке:

- **keepCellsIntersectingBoundary** – глобальная опция для сохранения всех ячеек в шаблонной сетке, которые пересекаются с границей. По умолчанию в шаблоне сетке сохраняют только те ячейки, которые полностью находятся внутри геометрии. Допустимыми значениями опции являются 1 (активно) или 0 (неактивно). Активация этой опции может привести к локальному объединению сетки поверх разрыва (образованного, например, выступающим мелкомасштабным элементом геометрии), это можно регулировать с помощью опции **checkForGluedMesh** заданием значений 1 (активно), либо 0 (неактивно). Пример настроек представлен ниже:

```
keepCellsIntersectingBoundary 1;
checkForGluedMesh 0;
```

- **keepCellsIntersectingPatches** – локальная опция, действующая аналогично **keepCellsIntersectingBoundary**, но только для указанных пользователем патчей. Она неактивна, когда включена опция **keepCellsIntersectingBoundary.** Пример настроек представлен ниже:

```
keepCellsIntersectingPatches
```

```
{
"patch0.*"
{
keepCells 1;
}
solid
{
keepCells 1;
}}
```

- **removeCellsIntersectingPatches** это опция, которая удаляет ячейки из шаблона, пересекающиеся с границей в патчах, указываемых пользователем. Она активирована, когда включена опция **keepCellsIntersectingBoundary**. Пример настроек представлен ниже:

```
removeCellsIntersectingPatches
{
"patch0.*"
{
keepCells 0;
}
solid
{
keepCells 0;
}}
```

4.4. Переименование патчей

Опция **renameBoundary** используется для изменения имен и типов патчей в процессе построения сетки. Пример ее настройки представлен ниже.

```
renameBoundary
{
defaultName walls;
defaultType wall;

newPatchNames
{
"patch0.*"
{
newName outlet;
```

```
type patch;
}
patch1
{
newName inlet;
type patch;
}
}
}
```

Параметр **defaultName** определяет новое имя для всех патчей, кроме тех, которые указаны в блоке **newPatchNames**. Настройка не является обязательной.

Параметр **defaultType** устанавливает новый тип для всех патчей, кроме тех, которые указаны в каталоге **newPatchNames**. По умолчанию всем патчам устанавливается тип empty (если использовать утилиту **cartesian2DMesh**) или patch (если использовать утилиту **cartesianMesh**). Следует учитывать, что использование типа empty на всех патчах может вызвать ошибку nonAlignedEdges при дальнейшей проверке сетки с помощью утилиты **checkMesh.**

Блок **newPatchNames** содержит названия патчей, которые необходимо переименовать. Для каждого патча можно указать новое имя или новый тип с помощью опций **newName** и **type** соответственно.

4.5. Применение геометрических ограничений

Иногда невозможно учесть все геометрические элементы в процессе построения сетки. Активация параметра **enforceGeometryConstraints** позволяет остановить процесс создания сетки, когда невозможно учесть все элементы входной геометрии, и записать подмножество точек, которые нужно удалить из геометрии, чтобы получить корректную сетку.

```
enforceGeometryConstraints 1;
```

4.6. Контроль процесса генерации сетки

Опция **workflowControls** позволяет выставлять точки остановки в ходе создания сетки, сохранять промежуточные результаты генерации и продолжать генерацию сетки с последнего сохраненного шага. Для настройки точки остановки процесса генерации сетки используется параметр **stopAfter**. Возможные следующие точки остановки в процессе генерации:

1) **templateGeneration** – останавливает процесс создания сетки после того, как был сгенерирован исходный шаблон из стандартных ячеек одного вида (например, гексаэдров). Эту точку можно использовать для проверки разрешающей способности сетки шаблона (см. рис. 1).

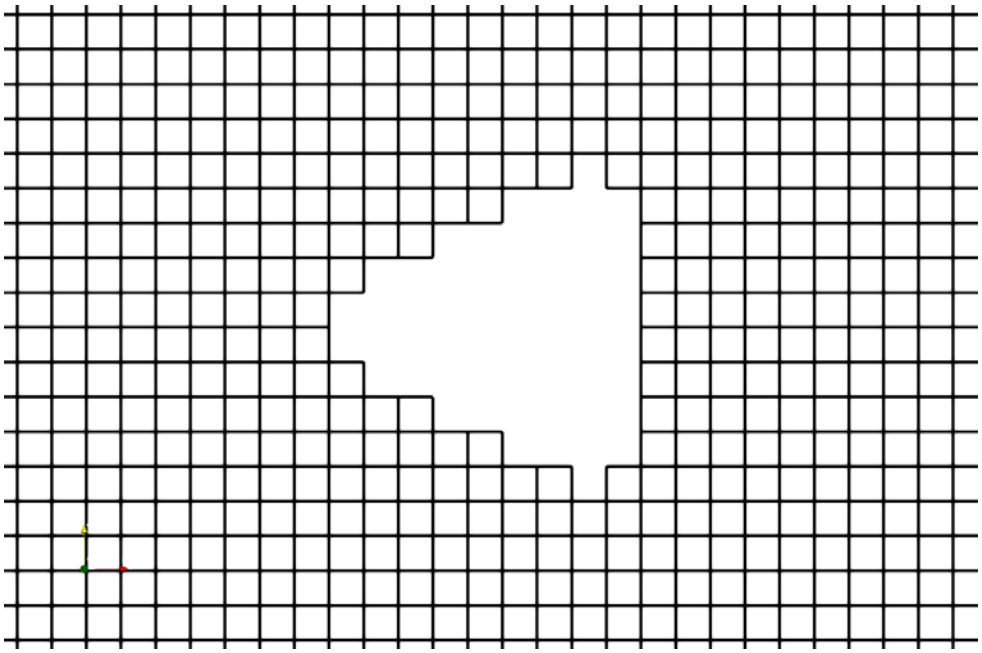


Рис. 1. Пример сетки шаблона (точка остановки **templateGeneration**)

2) **surfaceTopology** – останавливает процесс создания сетки после очистки сетки от недопустимых соединений граней на поверхности патчей.

3) **surfaceProjection** – останавливает процесс создания сетки после проецирования пограничных узлов сетки на заданную геометрию.

Эту точку можно использовать для проверки корректности проецирования сетки на заданную геометрию (см. рис. 2).

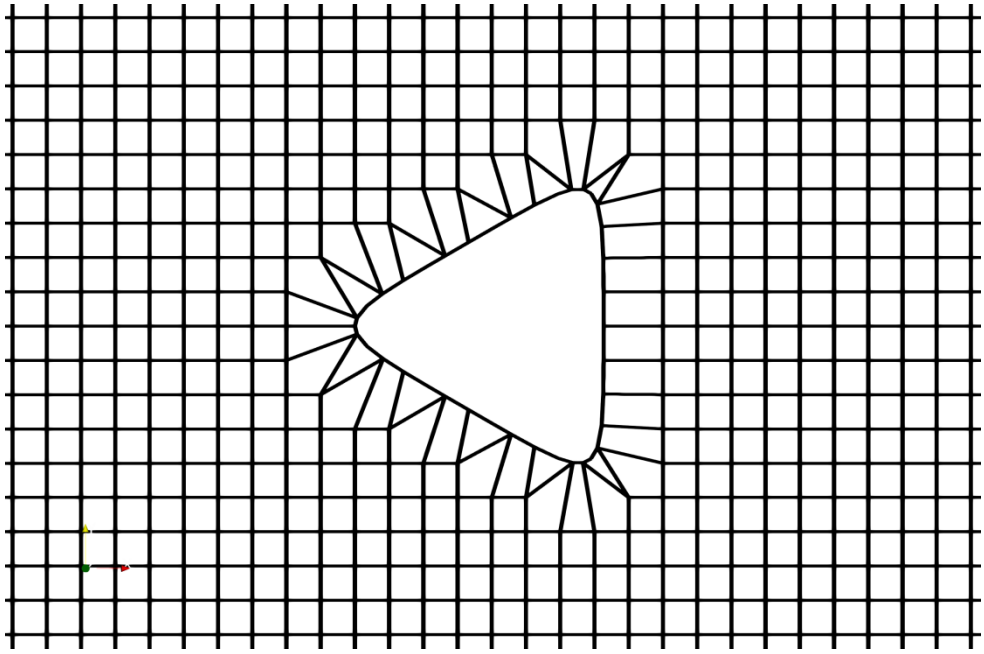


Рис. 2. Пример сетки, полученной после проецирования пограничных узлов шаблонной сетки на заданную геометрию
(точка остановки **surfaceProjection**)

4) **patchAssignment** – останавливает процесс построения сетки после того, как грани ячеек, прилегающие к патчам, присваиваются этим патчам. Эту точку можно использовать для проверки разрешающей способности границ.

5) **edgeExtraction** – останавливает процесс генерации после окончания процесса формирования сетки на поверхности тела. На этом этапе проявятся дефекты деформации геометрии. Как видно из рисунка 3, на данном этапе произошла деформация поверхности из-за смещения пограничных узлов на поверхности тела по причине недостаточно подробного шага сетки.

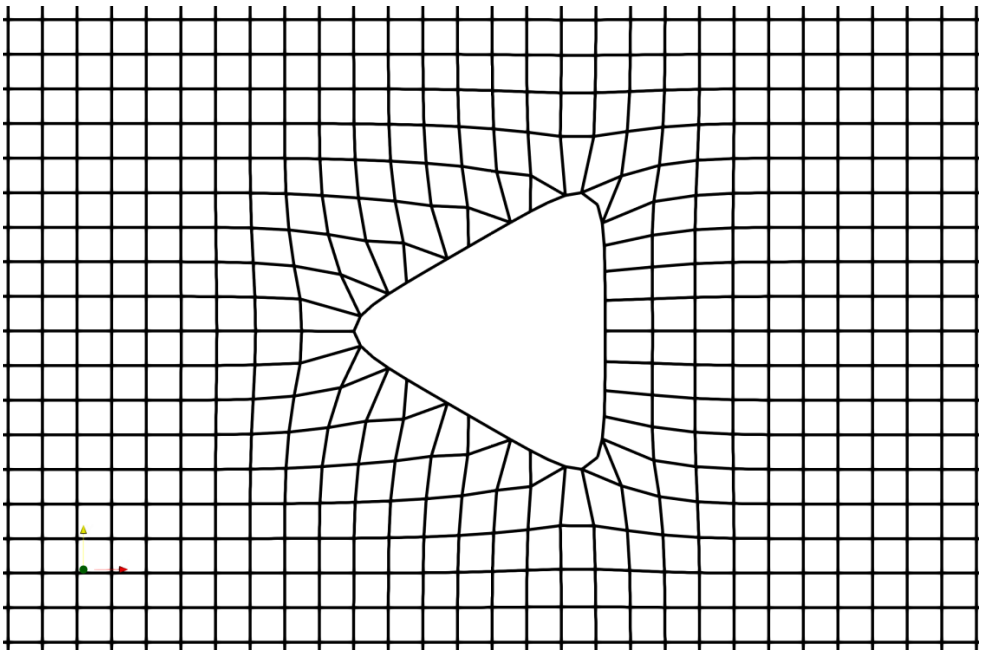


Рис. 3. Пример сетки, полученной после окончания процесса формирования сетки на поверхности тела (точка остановки **edgeExtraction**)

6) **boundaryLayerGeneration** – останавливает процесс построения сетки после создания первичного пограничного слоя (см. рис. 4).

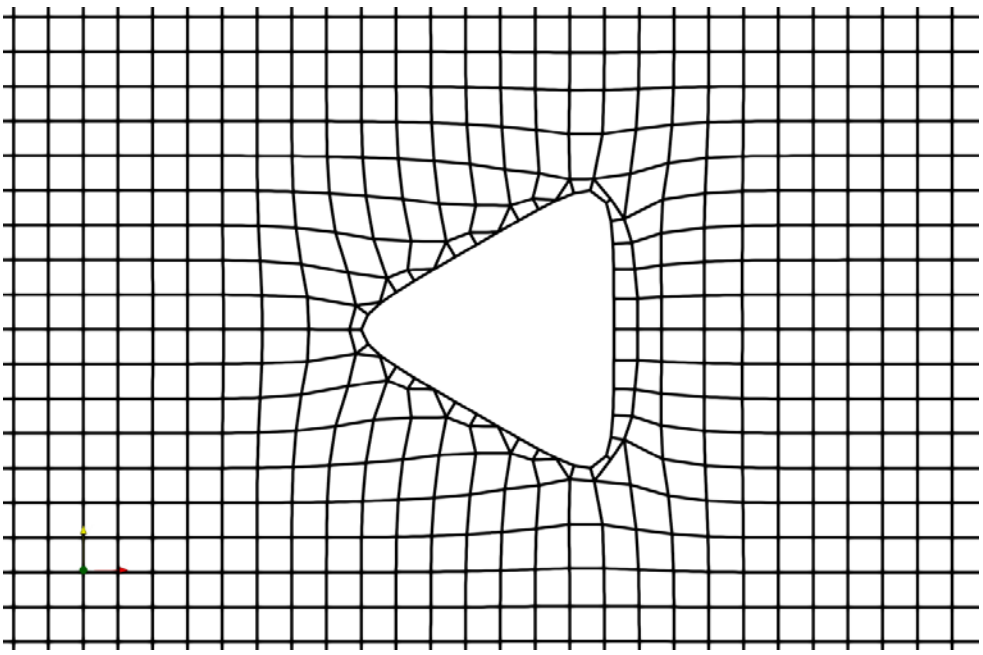


Рис. 4. Пример сетки, полученной после создания первичного пограничного слоя (точка остановки **boundaryLayerGeneration**)

7) **meshOptimisation** – останавливает генерацию после полной оптимизации (выравнивание толщины погранслоев и др.). Результат оптимизации показан на рис. 5.

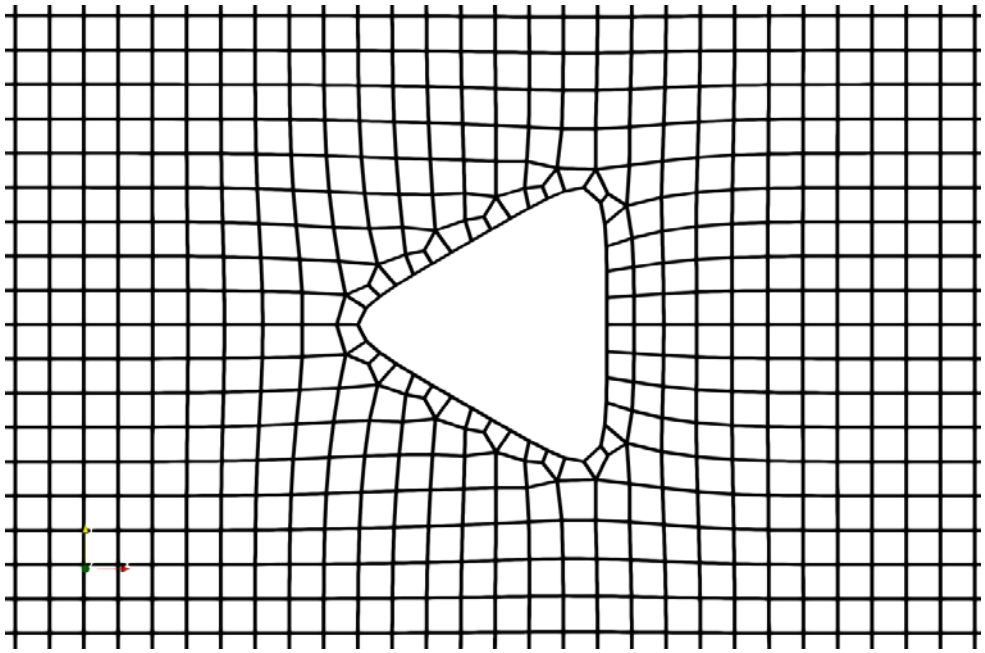


Рис. 5. Пример сетки, полученной после оптимизации
(точка остановки **meshOptimisation**)

8) **boundaryLayerRefinement** – останавливает генерацию после завершения формирования пограничных слоев (после добавления указанного пользователем количества погранслоев и др.). Результат остановки в этой точке показан на рис. 6.

С помощью параметра **restartFromLatestStep** можно задействовать (1) механизм продолжения создания сетки с последнего сохраненного шага.

```
workflowControls
{
stopAfter templateGeneration;
restartFromLatestStep 1;
}
```

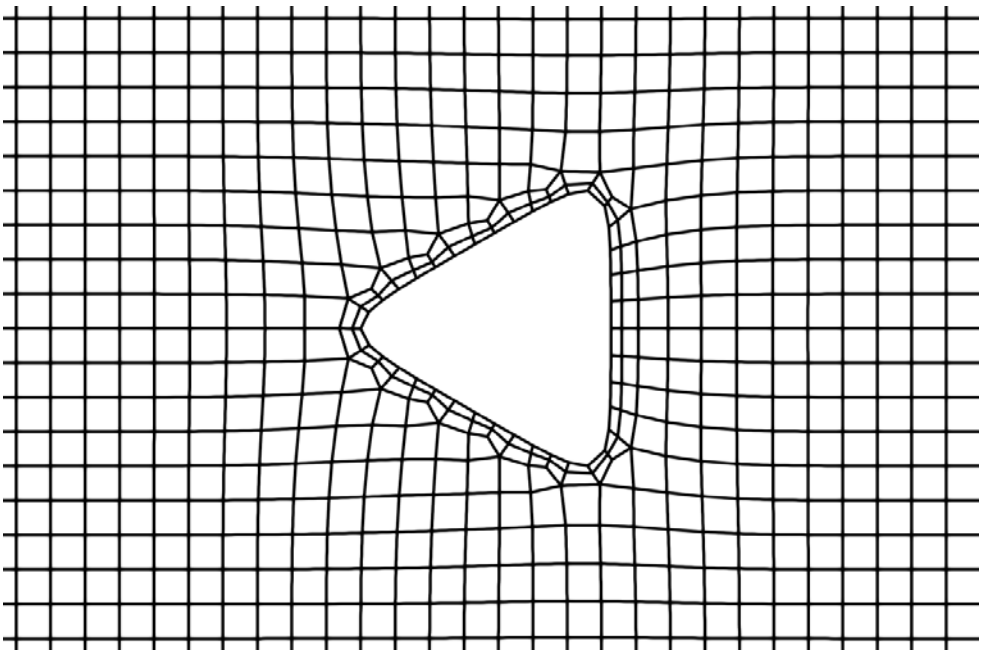


Рис. 6. Пример сетки, полученной после завершения формирования пограничных слоев (точка остановки **boundaryLayerRefinement**)

ГЛАВА 2. ОПРЕДЕЛЕНИЕ КАЧЕСТВА РАСЧЕТНОЙ СЕТКИ

1. Индикаторы качества расчетных сеток

Как отмечалось во введении, важными характеристиками построенных расчетных сеток являются [2]:

- неортогональность (nonorthogonality),
- скошенность (skewness),
- равномерность (smoothness),
- соотношение сторон ячеек (aspect ratio)

Они оказывают влияние на точность результирующей аппроксимационной схемы и точность конечных результатов (более подробную информацию можно найти в [1, 5]). Для определения этих характеристик рассмотрим рис. 7, где изображены две соседние ячейки неортогональной расчетной сетки.

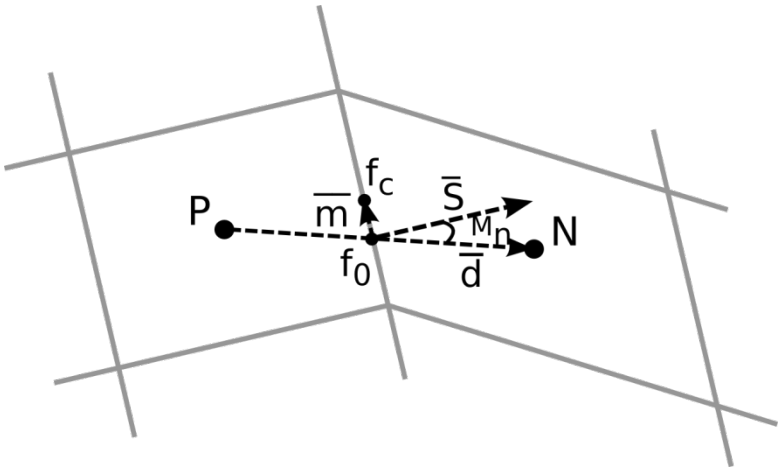


Рис. 7. Ячейки расчетной сетки

Показатель *неортогональности* M_n измеряется как угол между вектором d , соединяющим центры двух соседних ячеек, и нормалью S к их общей грани (см. рис. 7)

$$M_n = d\hat{}S.$$

Имеет оптимальное значение 0 градусов.

Показатель *скошенности* M_s измеряется как отношение длинны вектора m, равной расстоянию между точкой f_0 (образованной пересечением вектора d с общей гранью ячеек) и центром грани f_c, к длине вектора d (см. рис. 7)

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

Имеет оптимальное значение 0.

Показатель *равномерности* M_u будем определять как отношение объемов меньшей из рассматриваемых ячеек к большей. Имеет оптимальное значение 1.

Средние и максимальные значения этих характеристик являются важными индикаторами качества сетки. Их можно рассчитать в OpenFOAM с помощью утилиты **checkMesh**.

2. Пороговые значения индикаторов

В файле primaryMeshCheck.C, расположенном в каталоге *$WM_PROJECT_DIR/src/OpenFOAM/meshes/primitiveMesh/primitiveMeshCheck/*, можно найти все ключевые индикаторы качества (сетки), используемые в OpenFOAM. Их пороговые значения определяется следующим образом:

```
    Foam::scalar Foam::primitiveMesh::closedThreshold_ =
1.0e-6;
    //Порог предупреждения о незамкнутости ячейки, уста-
новленный как доля незакрытой области в закрытой области
    Foam::scalar Foam::primitiveMesh::aspectThreshold_ =
1000;
    //Порог предупреждения о некорректном соотношении
сторон.
    Foam::scalar Foam::primitiveMesh::nonOrthThreshold_ =
70; // deg
    // Порог предупреждения о высокой неортогональности
    Foam::scalar Foam::primitiveMesh::skewThreshold_ = 4;
    // Порог предупреждения о высокой скошенности
    Foam::scalar  Foam::primitiveMesh::planarCosAngle_  =
1.0e-6;
    // Порог, при котором грани считаются компланарными.
```

Как видно, максимальное значение скошенности равно 4, предельное значение неортогональности составляет 70 градусов, а максимальное соотношение сторон равно 1000.
Алгоритм вычисления скошенности и неортогональности в OpenFOAM описан в файле cellQuality.C.

3. Использование утилиты checkMesh для анализа качества сетки

Для проверки качества и валидности сетки в OpenFOAM можно воспользоваться утилитой **checkMesh**. Вызов утилиты с обычными настройками осуществляется запущенной в директории проекта командой:

```
checkMesh
```

Директория проекта должна содержать следующий минимальный набор конфигурационных файлов и файлов сетки:

.

```
├── constant
│   └── polyMesh
│       ├── boundary
│       ├── faces
│       ├── meshMetaDict
│       ├── neighbour
│       ├── owner
│       └── points
└── system
    ├── controlDict
    ├── fvSchemes
    └── fvSolution
```

По результатам работы программа выдаст следующий набор данных:

```
    // общее количество точек, внутренних точек, граней ячеек и т.д.
    Mesh stats
        points:           920667
```

```
        internal points:  654393
        faces:            2398836
        internal faces:   2131924
        cells:            738420
        faces per cell:   6.13575
        boundary patches: 4
        point zones:      0
        face zones:       0
        cell zones:       0


    // типы ячеек и их количество
    Overall number of cells of each type:
        hexahedra:     700768  //шестигранные ячейки
        prisms:        1280
        wedges:        0
        pyramids:      3200
        tet wedges:    0
        tetrahedra:    1280
        polyhedra:     31892 //другие многоранники
        Breakdown of polyhedra by number of faces:
            faces   number of cells
                6   2296
                9   24612
               12   3892
               15   1020
               18   56
               21   16

    // информацию о корректности топологии сетки
    Checking topology...
        Boundary definition OK.
        Cell to face addressing OK.
        Point usage OK.
        Upper triangular ordering OK.
        Face vertices OK.
        Number of regions: 1 (OK).

    // названия патчей
    Checking patch topology for multiply connected sur-
faces...
     Patch    Faces   Points            Surface topology
    rotor   121136   121169  ok (non-closed singly con-
nected)
    rotor_wall   104344   104856  ok (non-closed singly
connected)
```

```
    interface_rotor_inlet       2400       2441   ok (non-
closed singly connected)
    interface_rotor_out      39032      39984   ok (non-
closed singly connected)

    // наличие областей faceZone
    Checking faceZone topology for multiply connected
surfaces...
        No faceZones found.

    // наличие областей cellZone
    Checking basic cellZone addressing...
        No cellZones found.

    // количественные параметры сетки, в том числе значе-
ния показателей качества
    Checking geometry...
        Overall domain bounding box (-0.0270001 -0.027 -
0.0145) (0.027 0.027 0.00900006)
        Mesh has 3 geometric (non-empty/wedge) directions
(1 1 1)
        Mesh has 3 solution (non-empty) directions (1 1 1)
        Boundary  openness  (3.19506e-18  -3.43876e-16  -
9.43393e-16) OK.
        Max cell openness = 3.25685e-16 OK.
        Max aspect ratio = 6.50906 OK.
        Minimum face area = 3.71945e-09. Maximum face ar-
ea = 8.16254e-07.  Face area magnitudes OK.
        Min volume = 1.45482e-13. Max volume = 8.19086e-
10.  Total volume = 1.26937e-05.  Cell volumes OK.
        Mesh  non-orthogonality  Max:  35.0161  average:
3.97115
        Non-orthogonality check OK.
        Face pyramids OK.
        Max skewness = 1.54867 OK.
        Coupled point location match (average 0) OK.

    Mesh OK. //говорит о том, что все параметры находятся
в допустимых пределах
    // Если у сетки есть структурные проблемы, то по ре-
зультатам проверок будет выдано сообщение: Failed n mesh
checks (где n – количество ошибок).
    End
```

В разделе «Checking geometry» можно найти (максимальные и средние) значения трех из описанных выше показателей качества:

Max aspect ratio – максимальное соотношение сторон ячеек.

Mesh non-orthogonality Max: average: – максимальное и среднее значение неортогональности

Max skewness – максимальная скошенность.

4. Визуализация проблемных участков расчетной сетки

Визуализировать рассчитанные с помощью **checkMesh**, локальные значения показателей качества расчетных сеток можно с помощью программы **ParaView**. Для этого сначала необходимо записать их в соответствующие файлы. Это можно сделать с помощью следующей команды, запустив ее в директории проекта:

```
checkMesh –writeAllFields
```

Далее откроем проект в графическом интерфейсе **ParaView**. Для этого выполним команду

```
paraFoam
```

В группе настроек **Coloring** панели свойств объекта **Properties** или в соответствующем списке на панели инструментов (см. рис. 8) **ParaView** для визуализации цветом станут доступны следующие поля:

```
"nonOrthoAngle",
"faceWeight",
"skewness",
"cellDeterminant",
"aspectRatio",
"cellShapes",
"cellVolume",
"cellVolumeRatio",
"cellAspectRatio",
"minTetVolume",
"minPyrVolume",
"cellRegion",
```

```
"wallDistance",
"cellZone",
```

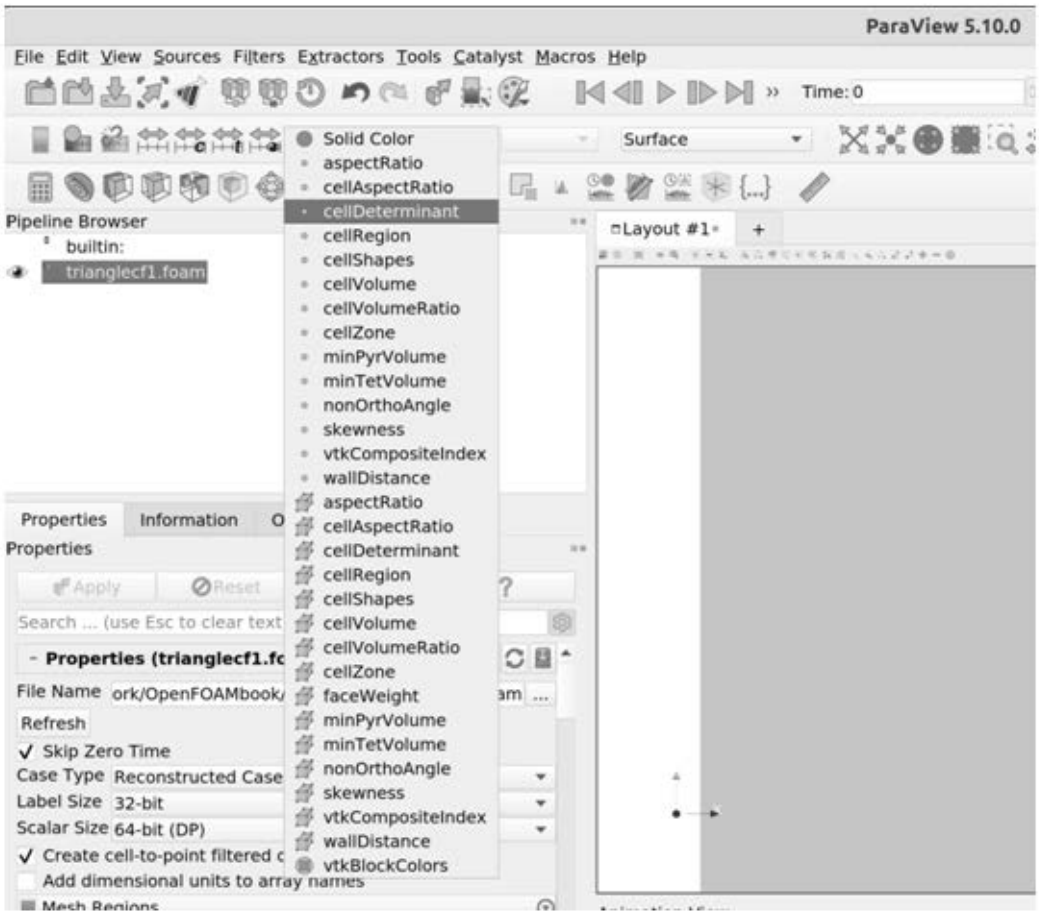


Рис. 8. Выбор поля для визуализации в группе настроек **Coloring**

Все они были записаны утилитой **checkMesh**. Обратим особое внимание на следующие из них:

- **aspectRatio** – поле, которое содержит данные по соотношению сторон ячеек;
- **nonOrthoAngle** – поле, которое содержит данные по неортогональности ячеек;
- **skewness** – поле, которое содержит данные по скошенности ячеек;
- **cellVolumeRatio** – поле, которое содержит данные по соотношению между объемами соседних ячеек (индикатор равномерности).

Отображая их, мы можем увидеть проблемные участки сетки.

В двумерном случае для этого достаточно выбрать требуемое поле в группе настроек **Coloring** и правильно задать палитру (диапазон colormap) с помощью опции **Set Range** (см. рис. 9). Пример визуализации поля **skewness** представлен на рис. 10, на котором хорошо видно расположение ячеек с максимальной скошенностью поскольку они выделены серым цветом (более темные тона соответствуют более высоким значениям скошенности).

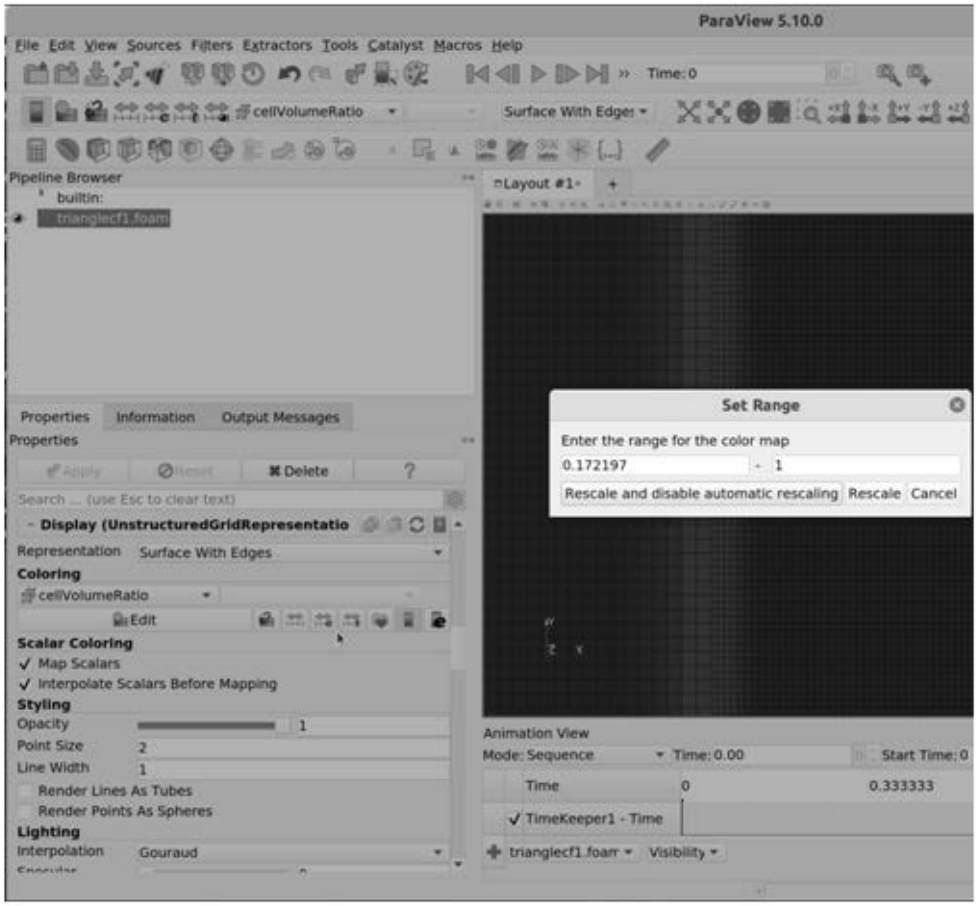


Рис. 9. Настройка палитры

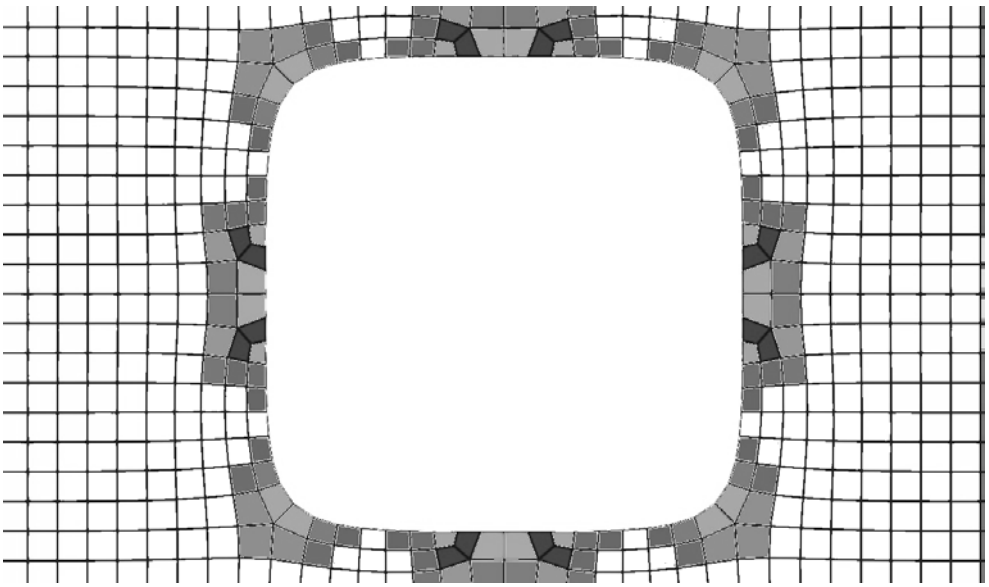


Рис. 10. Расположение ячеек с высоким значением скошенности в двумерной области

В трехмерной области визуализацию проблемных ячеек нужно проводить несколько сложнее. Один из возможных вариантов – это воспользоваться фильтром **Threshold** (его можно найти в меню **Filters** главного меню или на панели инструментов), фактически визуализируя только ячейки, в которых значения выбранной характеристики находятся в некотором диапазоне значений (см. рис. 11).

Этот диапазон можно установить с помощью параметров **Lower Threshold** (нижний порог) и **Upper Threshold** (верхний порог) во вкладке **Properties**. При этом бывает удобно одновременно отобразить и весь объем расчетной области, чтобы хорошо видеть место расположения проблемных ячеек. Для этого нужно активировать соответствующий чекбокс – нажать на значок в виде глаза на главном объекте (на рис. 11 имеет название «1.foam») на панели обзора. Чтобы расчетная область не перекрывала выбранные ячейки, нужно изменить параметр **Opacity** (например, установить значение 0.3) в свойствах объекта **Properties.**

Пример визуализации зон повышенной скошенности для трехмерной сетки показан на рис. 12, где маркеры темно-серых оттенков показывают ячейки с наибольшей скошенностью.

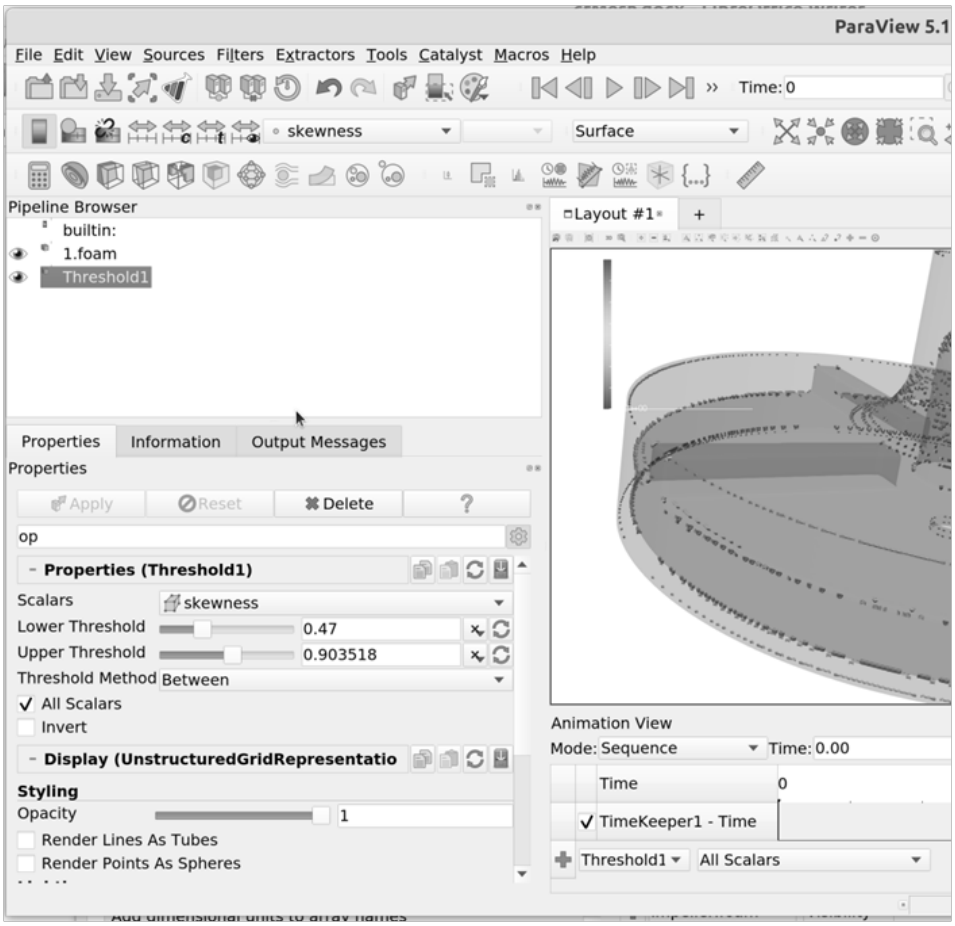


Рис. 11. Настройка фильтра Threshold

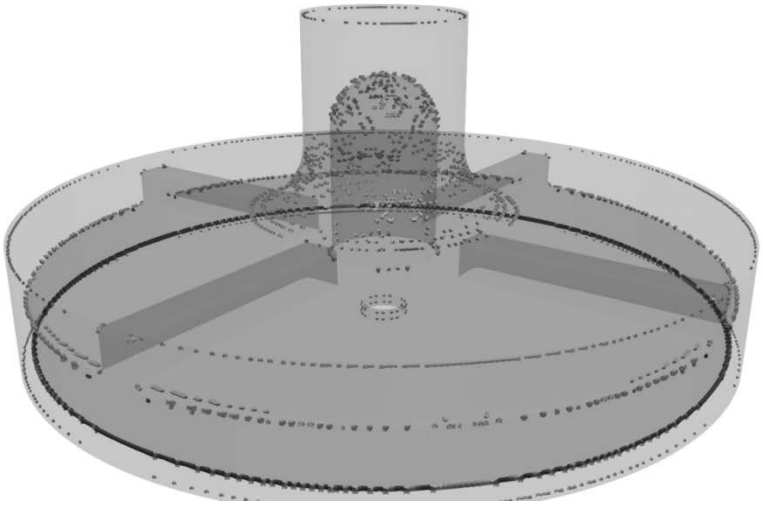


Рис. 12. Расположение ячеек с высокой скошенностью в трехмерной области

ГЛАВА 3. ПРИМЕРЫ СОЗДАНИЯ РАСЧЕТНЫХ СЕТОК

1. Создание 2D сетки с помощью cartesian2DMesh

Рассмотрим построение расчетной сетки для решения плоской задачи обтекания цилиндрического тела разнонаправленным потоком (жидкость может двигаться как слева направо, так и справа налево вдоль оси Ox) в «безграничной» области. Представим геометрию в виде 5 частей: четырех частей внешней границы (1-4) и поверхности цилиндра (5). Они выделены разными оттенками серого и пронумерованы на рисунке 13. Все элементы геометрии строго ортогональны плоскости xOy и имеют форму лент (ширина по оси Oz фиксирована), что необходимо для генерации двумерной сетки с помощью **cartesian2DMesh**. Каждая часть границы хранится в отдельном stl файле: Square_round.stl, right.stl, left.stl, bottom.stl, top.stl.

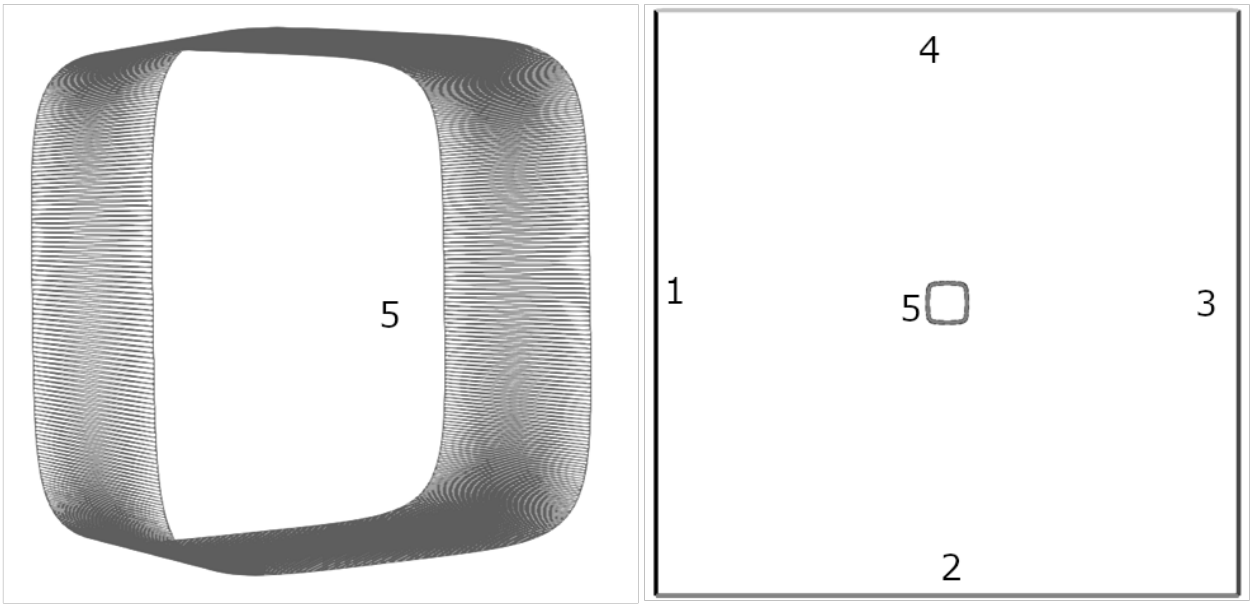


Рис. 13. Геометрия расчетной области. Слева – загружаемая поверхностная триангуляция скругленного квадратного цилиндра, справа – общий вид геометрии в плоскости xOy

Для дальнейшей работы удобны stl файлы в текстовом формате, для конвертации бинарных stl в текстовые файлы можно воспользоваться командами (запускаем их в директории, где расположены stl файлы):

```
    surfaceFeatureEdges ./Square_round.stl ./Square_round
1.stl
    surfaceFeatureEdges ./right.stl ./right1.stl
    surfaceFeatureEdges ./left.stl ./left1.stl
    surfaceFeatureEdges ./bottom.stl ./bottom1.stl
    surfaceFeatureEdges ./top.stl ./top1.stl
```

Далее нужно, чтобы каждый патч имел уникальное название. Это потребуется для задания граничных условий. В данном примере все патчи имеют названия patch0_0. Для их переименования в linux можно воспользоваться командами (либо вручную отредактировать текстовые stl файлы):

```
    sed -i 's/patch0_0/left/g' left1.stl
    sed -i 's/patch0_0/right/g' right1.stl
    sed -i 's/patch0_0/top/g' top1.stl
    sed -i 's/patch0_0/bottom/g' bottom1.stl
    sed -i 's/patch0_0/square/g' Square_round1.stl
```

После этого соединим все файлы геометрии в один. Можно просто скопировать содержимое всех файлов в один файл или воспользоваться командой:

```
    cat left1.stl right1.stl top1.stl bottom1.stl
Square_round1.stl >> Geometry.stl
```

Результирующий файл Geometry.stl скопируем в папку проекта по адресу: *путь_к_директории_проекта/constant/triSurface/ (в этой папке принято хранить файлы геометрии проекта).*

Перейдем к этапу генерации сетки. Для начала рассмотрим простейшие настройки. Для этого вставим в конфигурационный файл meshDict (расположен по адресу *путь_к_директории_проекта/system/meshDict*) следующие строчки:

```
    surfaceFile "./constant/triSurface/Geometry.stl";
//путь к stl файлу  Geometry.stl из текущей директории
    maxCellSize 0.5; // размер ячеек в основной области
    minCellSize 0.02; // размер ячеек в областях выбран-
ных программой для автоматического измельчения
```

и, предварительно перейдя в директорию проекта, наберем в консоли команду **cartesian2DMesh.** В результате мы должны получит сетку с размером основных ячеек равным 0.5 (при характерном размере цилиндра равным 2) и 0.02 в областях, где реализуется автоматическое измельчение.

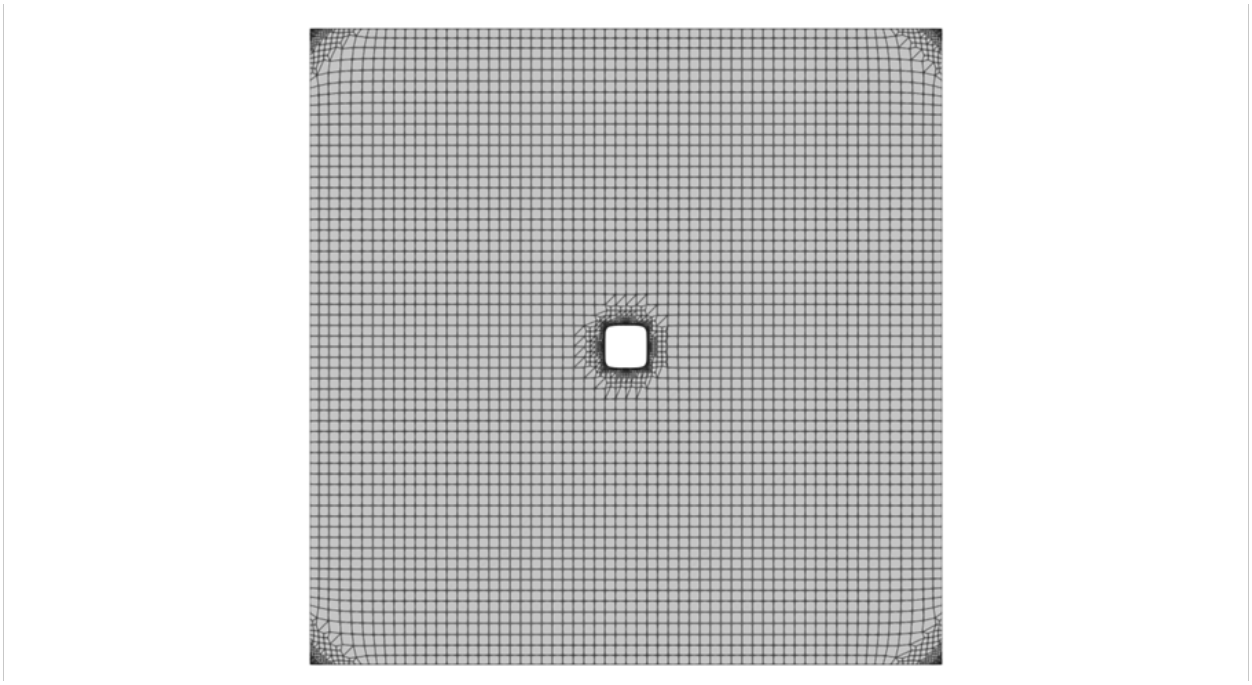


Рис. 14. Расчетная сетка, сгенерированная **cartesian2DMesh** с использованием процедуры автоматического измельчения ячеек

Результирующая сетка, визуализированная средствами **ParaView**, представлена на рис. 14. Главный недостаток полученной сетки состоит в дроблении (измельчении) ячеек вблизи углов внешней границы расчетной области: необходимость в этом, очевидно, полностью отсутствует и приводит к негативным эффектам для обозначенной задачи. Течение в этих зонах должно иметь очень простую струк-

туру (фактически там реализуется равномерный поток), и лишние ячейки необоснованно увеличивают расчетную сложность задачи. Более того, измельчение в этих зонах повышает локальную скошенность, что в итоге приводит к возникновению дополнительных численных ошибок в расчете. Таким образом, в контексте обозначенной задачи полученный результат генерации – это пример неправильного измельчения сетки.

Для данного случая лучше отключить глобальные механизмы измельчения сетки и использовать более тонкие локальные настройки.

Рассмотрим возможность сгущения ячеек в области, где расположен цилиндр. Для этого воспользуемся следующими настройками:

```
surfaceFile "./constant/triSurface/Geometry.stl";
maxCellSize 0.5;

anisotropicSources
{
box1
{
type box;
centre (0 0 0.5);
lengthX 6;
lengthY 6;
lengthZ 1;
 scaleX 0.25;
 scaleY 0.25;
}
}
```

Согласно этим настройкам в области с размерами 6 на 6, расположенной в центре расчетной области, посредством сгущения должна быть сформирована сетка с размером ячеек в 4 раза мельче (за это отвечают параметры **scaleX** 0.25; **scaleY** 0.25) по каждой оси по сравнению с исходным размером ячейки (**maxCellSize** 0.5). Полученный результат представлен на рис. 15.

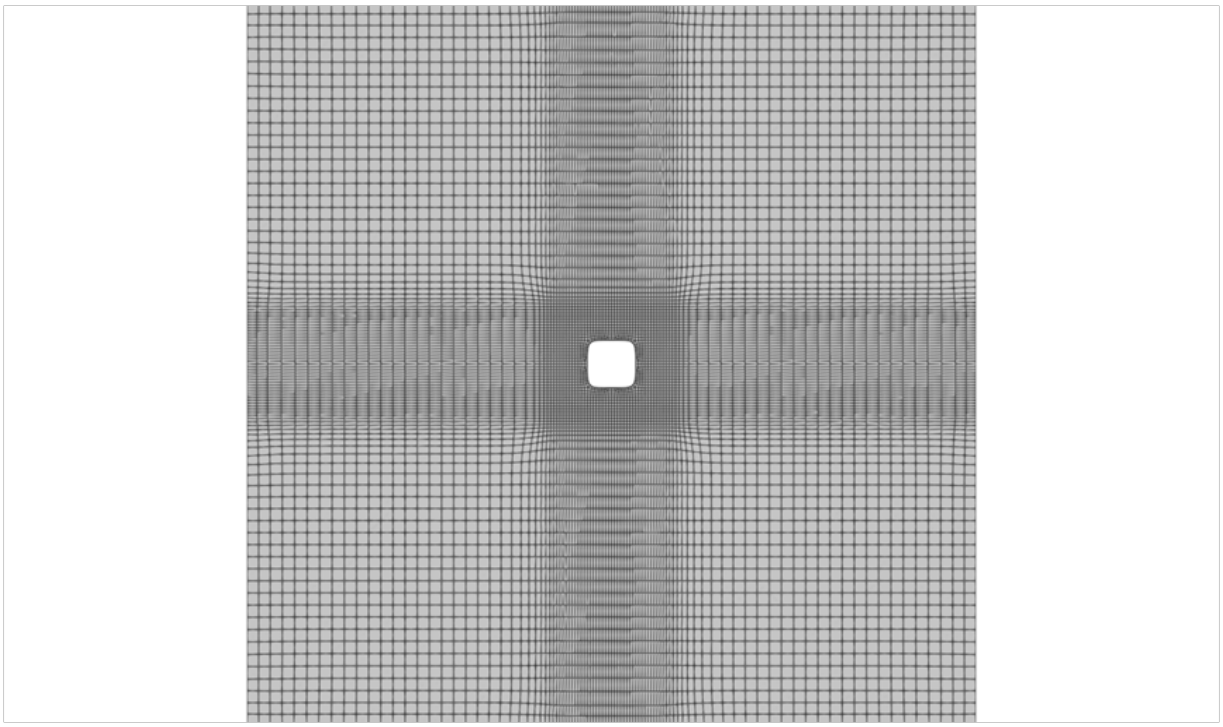


Рис. 15. Расчетная сетка, сгенерированная **cartesian2DMesh,** со сгущением узлов в центре области

Сгенерированная сетка состоит только из шестигранных ячеек и имеет достаточно хорошую структуру с плавным изменением размеров ячеек и относительно хорошими показателями качества (по данным **checkMesh**):

```
        Mesh non-orthogonality Max: 16.090017 average:
1.4453102
        Max skewness = 0.45653694 OK.
```

Однако непосредственно в малой окрестности границы тела сетка еще не достаточно подробна для точного описания граничных эффектов. Рассмотрим несколько вариантов по улучшению сетки вдоль границы.

В качестве первого варианта добавим пограничные сеточные слои, пользуясь опцией **boundaryLayers**. Для этого запишем конфигурационный файл следующим образом.

```
surfaceFile "./constant/triSurface/Geometry.stl";
```

```
maxCellSize 0.5;
anisotropicSources
{
box1
{
type box;
centre (0 0 0.5);
lengthX 6;
lengthY 6;
lengthZ 1;
 scaleX 0.25;
 scaleY 0.25;
}}

boundaryLayers
{
patchBoundaryLayers
{
square
{
nLayers 2;
thicknessRatio 1.4;
maxFirstLayerThickness 0.2;
allowDiscontinuity 0;
}}
}
```

Заметим, что один пограничный сеточный слой уже был сформирован на границе тела (эта опция обязательна для утилиты **cartesian2DMesh**). Задавая параметр **nLayers** 2, мы фактически разделяем его на два. Это хорошо видно на рис. 16. Также на рисунке можно видеть распределение по ячейкам значений параметра скошенности.

В случае задания трех пограничных слоев базовый слой поделится на 3 части и т. д. Это позволяет получить очень тонкие сеточные слои около тела, равномерность которых можно регулировать с помощью параметра **thicknessRatio**. При этом все ячейки остаются шестигранными. Однако здесь следует обратить внимание на два момента:

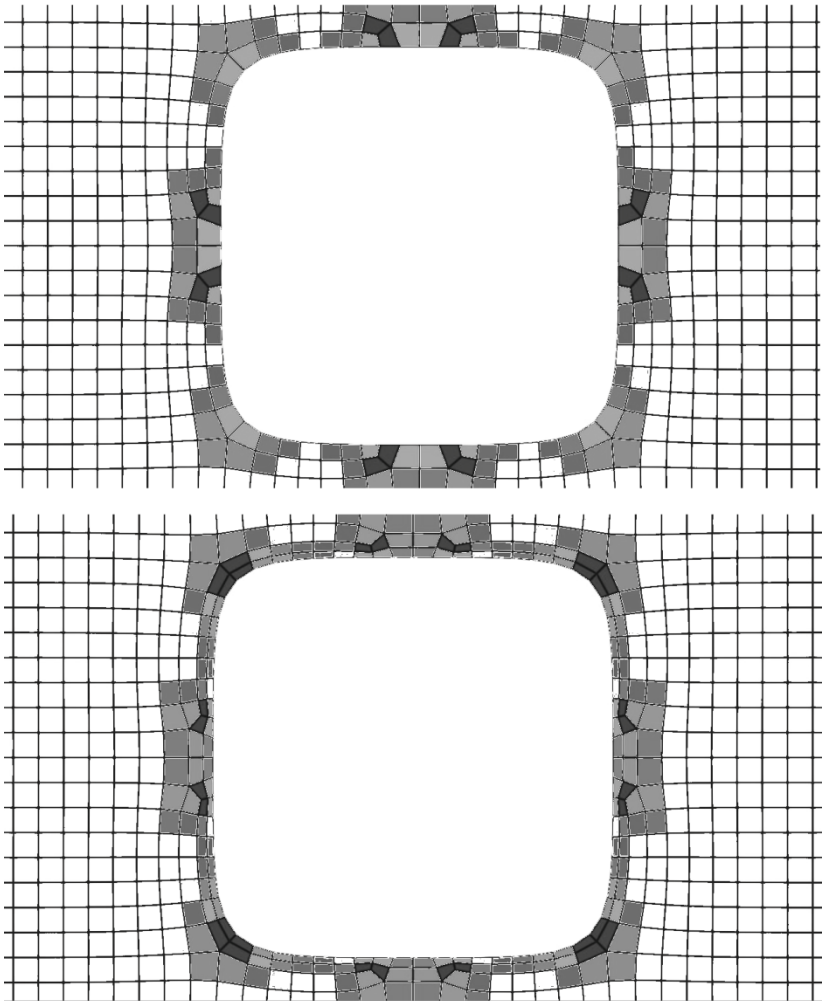


Рис. 16. Сетки с одним (по умолчанию) и двумя пограничными слоями. Цветом показано распределение значений скошенности. Темно-серый цвет указывает значения скошенности большие или равные 0.45

1. Увеличение количества слоев приводит к резкому росту максимального значения скошенности с 0.45 до 0.72 для двух слоев, до 0.85 для трех слоев (nLayers 3). При этом скошенность возрастает у ячеек, расположенных в непосредственной близости от тела, что может увеличивать ошибки расчета. Здесь нужно определить баланс между допустимой скошенностью и разрешающей способностью, определив какой из этих двух факторов будет оказывать более негативное влияние на результат.

2. Суммарная толщина пограничных слоев (в направлении ортогональном поверхности тела) фактически остается неизменной вне зависимости от их количества.

В качестве второго варианта для улучшения разрешающей способности сетки в пограничном слое рассмотрим измельчение ячеек около тела. Для этого можно воспользоваться опциями **localRefinement** или **objectRefinements**. Рассмотрим применение опции **localRefinement** со следующими параметрами:

```
surfaceFile "./constant/triSurface/Geometry.stl";
maxCellSize 0.5;

localRefinement
{
    square//triangle
    {
        refinementThickness 2;
        additionalRefinementLevels 1;
    }
}


anisotropicSources
{
box1
{
type box;
centre (0 0 0.5);
lengthX 6;
lengthY 6;
lengthZ 1;
 scaleX 0.25;
 scaleY 0.25;
}
}
```

Сгенерированная сетка представлена на рис. 17. Некоторые параметры сетки, полученные с помощью утилиты **checkMesh**, представлены ниже:

```
Overall number of cells of each type:
    hexahedra:     10092
```

```
        prisms:         0
        wedges:         0
        pyramids:       0
        tet wedges:     0
        tetrahedra:     0
        polyhedra:      96
        Breakdown of polyhedra by number of faces:
            faces   number of cells
                7   88
                8   8
    Mesh non-orthogonality Max: 16.090017 average:
1.621033
        Max skewness = 0.45170465 OK.
```

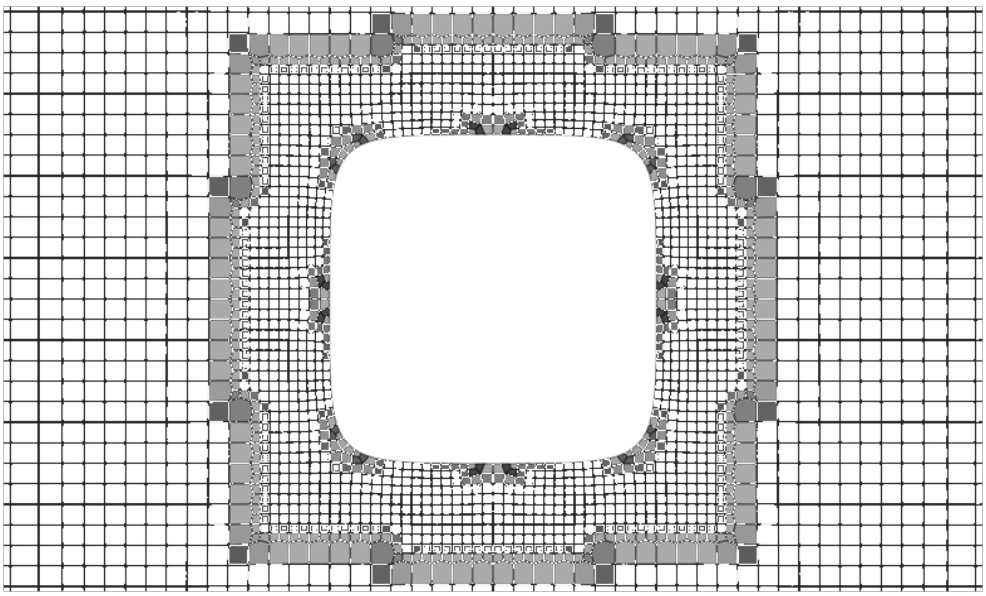


Рис. 17. Расчетная сетка, сгенерированная с использованием опции **localRefinement**. Цветом показано распределение значений скошенности. Темно-серый цвет указывает значения скошенности большие или равные 0.45

Аналогичных результатов можно добиться и с помощью опции **objectRefinements** со следующим настройками:

```
    surfaceFile "./constant/triSurface/Geometry.stl";
    maxCellSize 0.5;

    objectRefinements
    {
```

```
box2
{
type box;
centre (0 0 0.5);
lengthX 3;
lengthY 3;
lengthZ 1;
additionalRefinementLevels 1;
}
}

anisotropicSources
{
box1
{
type box;
centre (0 0 0.5);
lengthX 6;
lengthY 6;
lengthZ 1;
 scaleX 0.25;
 scaleY 0.25;
}
}
```

Сгенерированная сетка и распределение параметра скошенности (темно-серый цвет ячейки соответствует локальному значению равному 0.45) представлены на рис. 18. Максимальные значения скошенности и неортогональности для этой сетки:

```
        Mesh non-orthogonality Max: 16.090017 average:
1.5574694
        Max skewness = 0.45170454 OK.
```

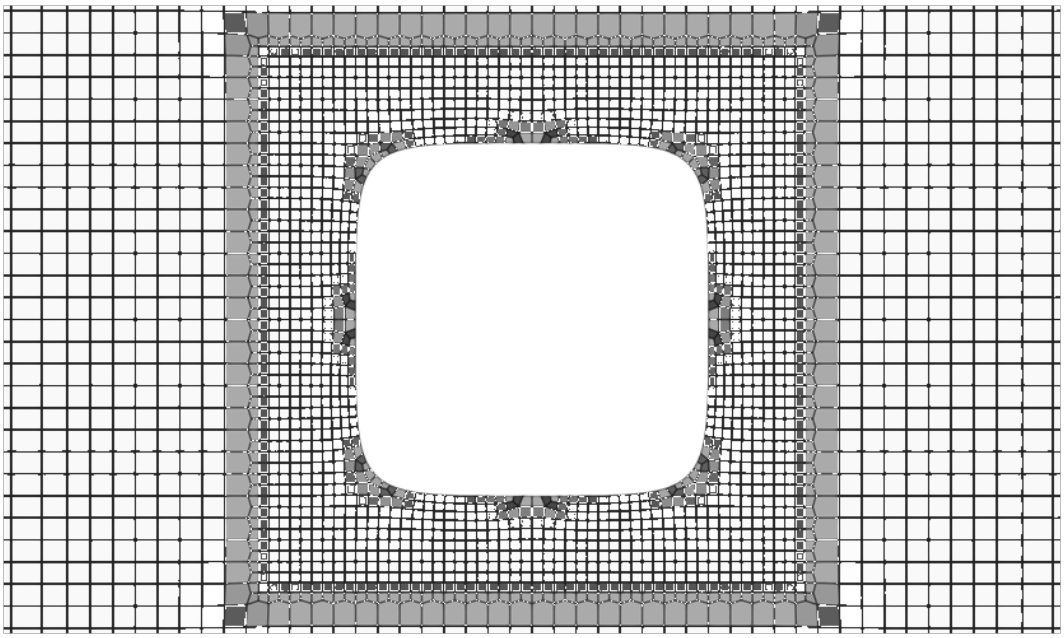


Рис. 18. Расчетная сетка, сгенерированная с использованием опции **objectRefinements**

В обоих случаях измельчение ячеек не приводит к росту максимальных значений скошенности и неортогональности (по сравнению с начальной сгущенной к центру области сеткой), но ведет к формированию еще одной области с повышенной скошенностью ячеек. Как можно видеть на рисунке, эта область расположена по контуру заданной зоны измельчения и формируется при дроблении ячеек начальной сетки до требуемого размера, в результате чего в ней образуются семи и восьмиграные ячейки.

С помощью **objectRefinements** можно создать каскадно измельчающиеся сетки:

```
objectRefinements
{
box2
{
type box;
centre (0 0 0.5);
lengthX 3;
lengthY 3;
lengthZ 1;
additionalRefinementLevels 1;
```

```
}
box3
{
type box;
centre (0 0 0.5);
lengthX 2.5;
lengthY 2.5;
lengthZ 1;
additionalRefinementLevels 2;
}
box4
{
type box;
centre (0 0 0.5);
lengthX 2.1;
lengthY 2.1;
lengthZ 1;
additionalRefinementLevels 3;
}
}
```

Результат генерации сетки для приведенных параметров представлен на рис. 19.

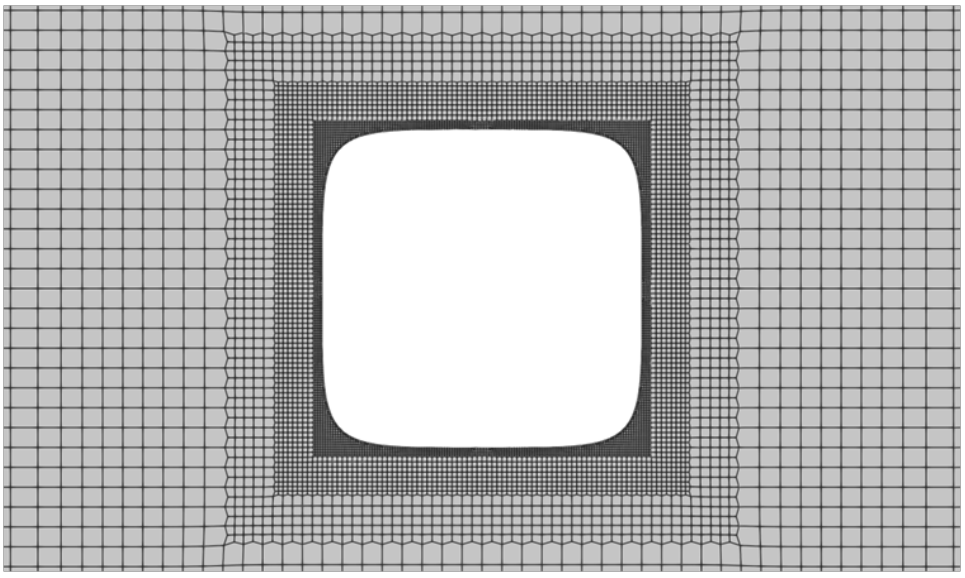


Рис. 19. Расчетная сетка, сгенерированная с использованием опции **objectRefinements**

В заключение данного раздела обратим внимание на еще одну особенность. После успешной генерации сетки утилита **checkMesh** может возвращать следующую ошибку:

```
  ***Number of edges not aligned with or perpendicular
to non-empty directions: 54488
   <<Writing 31274 points on non-aligned edges to set
nonAlignedEdges
…
Failed 1 mesh checks.\
```

Эта ошибка связанна с неправильным определением типов границ. Как уже отмечалось выше, по умолчанию всем границам присваивается тип empty. В этом можно убедиться, открыв файл *./катоалог_проекта/constant/polyMesh/boundary* с границами сгенерированной сетки. В рассматриваемом случае он имеет следующее содержимое:

```
7
(
left
{
    type empty;
    nFaces 96;
    startFace 29968;
}
right
{
    type empty;
    nFaces 96;
    startFace 30064;
}
top
{
    type empty;
    nFaces 96;
    startFace 30160;
}
bottom
{
    type empty;
    nFaces 96;
```

```
        startFace 30256;
    }
    square
    {
        type empty;
        nFaces 520;
        startFace 30352;
    }
    bottomEmptyFaces
    {
        type empty;
        nFaces 15074;
        startFace 30872;
    }
    topEmptyFaces
    {
        type empty;
        nFaces 15074;
        startFace 45946;
    }
    )
```

Исправить ошибку можно, изменив параметр **type** для границ **left**, **right**, **top**, **bottom** на **patch**, а для square на **wall** в данном файле. Однако правильнее будет настроить опцию **renameBoundary**, поскольку файл с границами перезаписывается в случае повторной генерации сетки и вновь требует исправления. В рассматриваемом случае запуск опции осуществляется путем добавления в конфигурационный файл следующего кода:

```
    renameBoundary
    {
    defaultName frontback; //для всех необозначенных в
newPatchNames границ
    defaultType empty;

    newPatchNames
    {
    left
    {
    newName left;
    type patch;
    }
```

```
right
{
newName right;
type patch;
}
top
{
newName top;
type patch;
}
bottom
{
newName bottom;
type patch;
}
square
{
newName square;
type wall;
}}
}
```

2. Создание 3D сетки с помощью cartesianMesh

2.1. Геометрия расчетной области

В качестве второго примера использования **cfMesh** рассмотрим процесс построения сетки для расчета течения жидкости в центробежном насосе (рассматривается тестовая модель FDA [6, 7] для перекачки крови). Геометрия насоса представлена на рис. 20. На изображении слева представлен общий вид геометрии с входным и выходным (напорным) патрубками, справа – корпус в разрезе с рабочим колесом (ротором) внутри.

Важнейшими условиями построения качественной сетки в области сложной геометрии (с помощью **cartesianMesh**) являются хорошо подготовленная поверхностная триангуляционная сетка, иными сло-

Рис. 20. Геометрия расчетной области

вами, чистая незашумленная геометрия, и правильное выделение всех ребер, краев и углов.

В рассматриваемом примере в качестве входных данных мы имеем 3 группы stl файлов:

- ротор и границы области, присоединенной к нему подвижной части сетки, (подробности о подвижной и подвижной частях сетки представлены ниже) описываются файлами interface-rotor-inlet-extenstion.stl, interface-rotor-out.stl, rotor.stl, rotor-wall.stl;
- входной патрубок описывается файлами inlet.stl, inlet-extension-wall.stl, interface-rotor-inlet-extenstion.stl;
- корпус (улитка) и выходной патрубки описываются файлами interface-rotor-out.stl outlet.stl outlet-extension-wall.stl

Поверхности в первой группе файлов (см. рис. 21) ограничивают область, в которой должна быть построена подвижная часть сетки, жестко связанная с вращающимся ротором. Во всей остальной части расчетной области будет строиться неподвижная сетка. Взаимодействие между подвижной и неподвижной частями сетки (интерполяция данных) будут реализовываться через границы-интерфейсы

(cyclicAMI), которые есть и у неподвижных частей сетки. Таким образом, будет моделироваться динамика жидкости внутри насоса с вращающимся ротором.

Генерировать сетку в каждой из 3-х частей геометрии нужно отдельно (в трех различных папках проекта), так как сетка на границах-интерфейсах в любом случае будет не совпадать при разных углах поворота ротора. Подготовим три папки проекта с названиями rotor, inlet, outlet. Скопируем туда все необходимые файлы. Соответствующие файлы геометрии (stl файлы) разместим по адресу *путь_к_директории_проекта/constant/triSurface/*.

Используемая поверхностная триангуляция не содержит каких-либо артефактов, поэтому сразу перейдем на этап выделения ребер. Это можно сделать с помощью дробления поверхности на патчи (при работе с форматом stl) или выделения их как feature edges (при работе с форматом fms).

2.2. Построение подвижной части сетки

Рассмотрим более подробно первую группу файлов. Открыть каждый файл можно с помощью команды

```
paraview /путь/название_файла.stl
```

или с помощью меню графического интерфейса **ParaView** (меню File->Open). Их визуализация представлена на рис.21, где четырьмя оттенками серого (пронумерованы на рисунке) отмечены данные из разных файлов (открыты все 4 файла): номером 1 выделен ротор (rotor.stl), номером 2 – части корпуса без «улитки» (rotor-wall.stl), номером 3 – граница-интерфейс между корпусом и входным патрубком (interface-rotor-inlet-extenstion.stl) и номером 4 – граница-интерфейс между объемом жидкости в непосредственной окрестности ротора и жидкости в «улитке» (interface-rotor-out.stl). Как уже отмечалось выше, эти поверхности ограничивают область, в которой должна быть построена подвижная часть сетки.

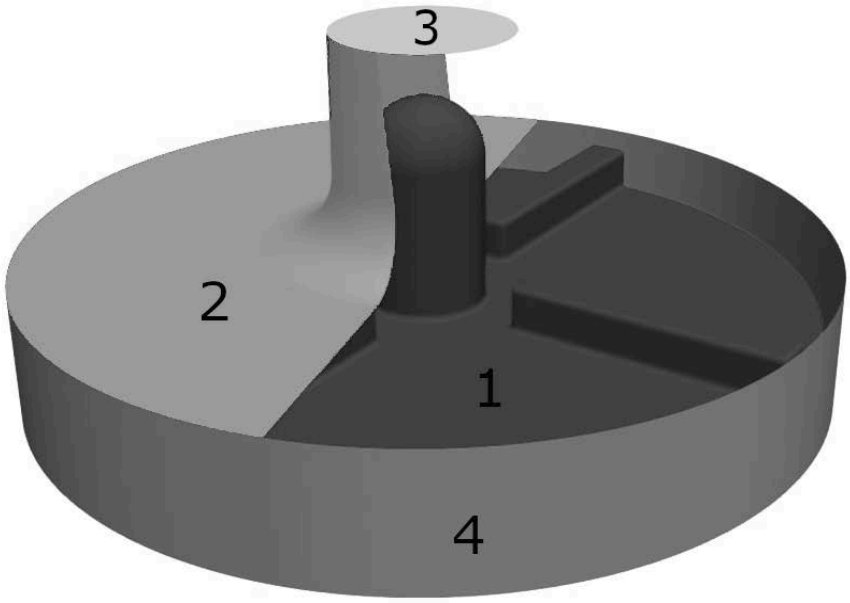


Рис. 21. Ротор и границы области присоединенной
к нему подвижной части сетки

Для подготовки геометрии проведем начальную серию шагов аналогичную той, что выполнялась для двумерного случая: используем утилиту **surfaceFeatureEdges** для конвертации файлов в текстовой формат и выделения ребер, используя утилиту **surfaceFeatureEdges**, переименуем патчи и соединим все в единый stl файл. Для этого можно воспользоваться командами (перейдя предварительно в папку *путь/rotor/constant/triSurface/*):

```
    surfaceFeatureEdges ./rotor.stl ./rotor1.stl
    surfaceFeatureEdges     ./rotor-wall.stl     ./rotor-
wall1.stl
    surfaceFeatureEdges  ./interface-rotor-inlet-extensti
on.stl ./interface-rotor-inlet-extenstion1.stl
    surfaceFeatureEdges ./interface-rotor-out.stl ./inter
face-rotor-out1.stl

    sed -i 's/patch0/rotor/g'  rotor1.stl
    sed -i 's/patch0/rotor_wall/g' ./rotor-wall1.stl
    sed  -i  's/patch0/interface_rotor_inlet/g'  ./inter
face-rotor-inlet-extenstion1.stl
```

```
    sed -i 's/patch0/interface_rotor_out/g' ./interface-
rotor-out1.stl

    cat ./rotor1.stl ./rotor-wall1.stl ./interface-rotor-
inlet-extenstion1.stl ./interface-rotor-out1.stl>> Geome-
try.stl
```

Заметим, что команда **surfaceFeatureEdges** при выводе результата в stl файл раздробит поверхность на несколько патчей, если угол между различными частями поверхности превышает 45 градусов (если сохранять результат как stl файл). Изменить настройки угла можно, используя опцию **angle**. Например, в случае применения следующей команды

```
    surfaceFeatureEdges ./rotor.stl ./rotor1.stl -angle 100
```

разделение поверхности на разные патчи будет происходить только при превышении угла в 100 градусов. Напоминаем, что в ребрах, образованных стыковкой различных патчей, сетка, сгенерированная **cartesianMesh**, должна в точности повторять геометрию поверхности.

Для примера далее приведем результаты генерации сетки для следующих случаев:

а) когда для обработки геометрии (с помощью **surfaceFeatureEdges**) использовалась опция -angle 45 (т. е. значение по умолчанию);

б) когда для обработки геометрии (с помощью **surfaceFeatureEdges**) использовалась опция -angle 180, т. е. дробление на патчи было полностью отключено. Соответствующие команды для этого случая представлены ниже.

```
    surfaceFeatureEdges ./rotor.stl ./rotor1.stl -angle
180
    surfaceFeatureEdges ./rotor-wall.stl ./rotor-wall1.stl
-angle 180
    surfaceFeatureEdges ./interface-rotor-inlet-extenstion.
stl ./interface-rotor-inlet-extenstion1.stl -angle 180
    surfaceFeatureEdges ./interface-rotor-
out.stl ./interface-rotor-out1.stl -angle 180
```

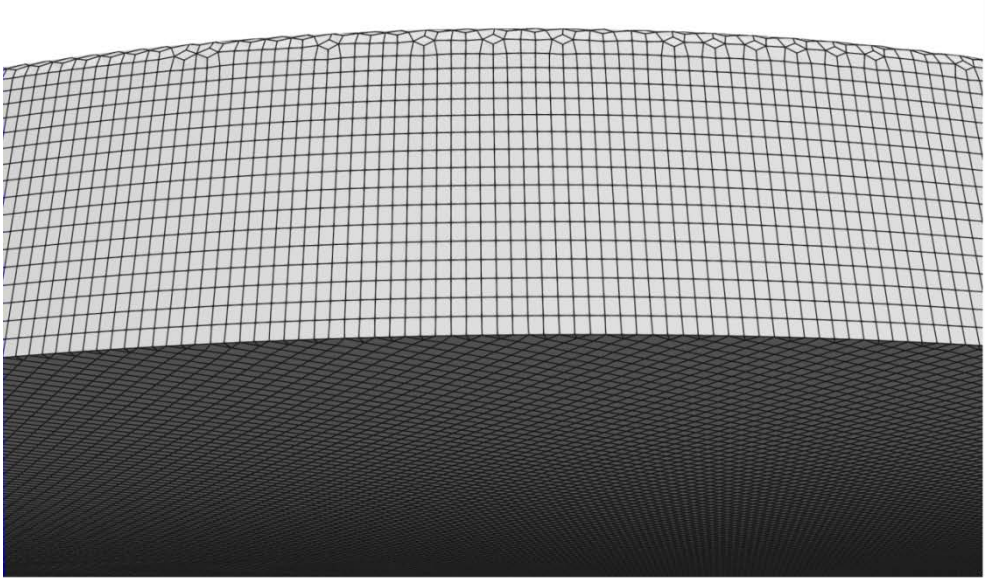
```
    sed -i 's/patch0_./rotor/g'  rotor1.stl
    sed -i 's/patch0_./rotor_wall/g' ./rotor-wall1.stl
    sed  -i  's/patch0_./interface_rotor_inlet/g'  ./inter
face-rotor-inlet-extenstion1.stl
    sed -i 's/patch0_./interface_rotor_out/g' ./interface-
rotor-out1.stl

    cat  ./rotor1.stl  ./rotor-wall1.stl  ./interface-rotor-
inlet-extenstion1.stl  ./interface-rotor-out1.stl>>  Geome-
try2.stl
```

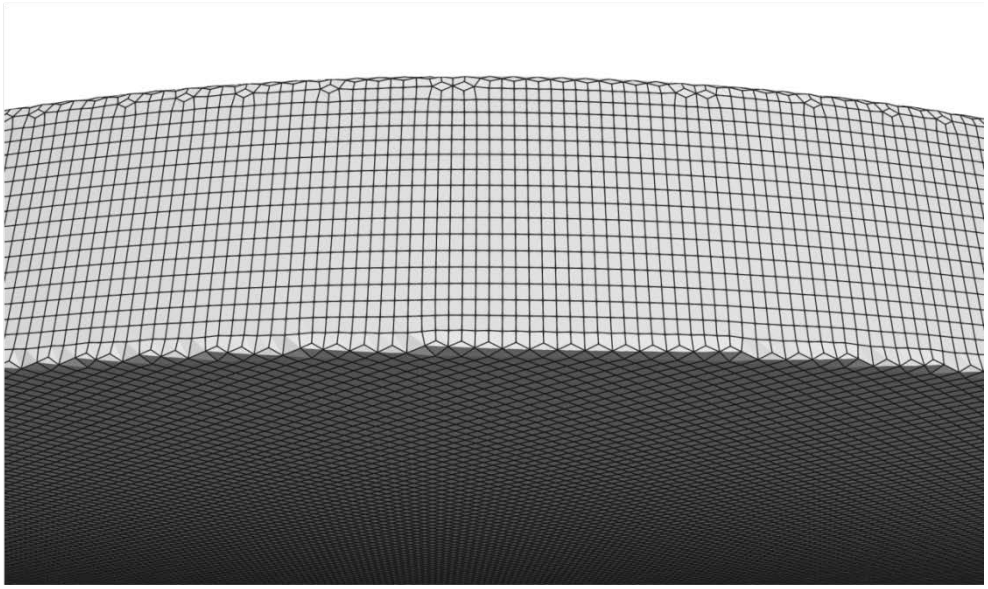
В обоих случаях сетка генерируется в подвижной части с помощью утилиты **cartesianMesh** (для запуска команды надо перейти в папку *rotor*) со следующими настройками конфигурационного файл meshDict:

```
    surfaceFile "./constant/triSurface/Geometry.stl"; //
или  Geometry2.stl
    maxCellSize 0.0009;
    boundaryCellSize 0.0004;
```

Как можно видеть, кромка ротора, образованная стыковкой нижнего диска с боковой поверхностью, в случае б) получается сглаженной (т. е. не соответствует исходной геометрии поверхности), что может снижать точность расчета. В случае а) она в точности соответствует исходной геометрии.



а)



б)

Рис. 22. Расчетные сетки на поверхности ротора. Верхнее изображение соответствует случаю а) геометрии с выделенными кромками, нижнее - случаю б) геометрии без выделенных кромок

При дальнейшей обработке следует учесть, что в случае а) сетка будет содержать следующие патчи:

```
rotor_0      384      448
rotor_1    78568    79033
rotor_2    42600    42808
rotor_3        0        0
```

```
        rotor_wall_0    45856    46068
        rotor_wall_1    58488    58788
interface_rotor_inlet_0     2400     2441
interface_rotor_out_0    39032    39984
```

В случае б) количество и название патчей будут другими:

```
                Patch    Faces   Points
                rotor   121136   121169
          rotor_wall   104344   104856
interface_rotor_inlet     2400     2441
 interface_rotor_out    39032    39984
```

Несколько отличными будут и максимальные значения показателей качества для сеток. Для случая а):

```
Max aspect ratio = 6.50906 OK.
Mesh non-orthogonality Max: 38.7473 average: 4.06468
Max skewness = 1.64276 OK.
```

Для случая б):

```
Max aspect ratio = 6.50906 OK.
Mesh non-orthogonality Max: 35.0161 average: 3.97115
Max skewness = 1.54867 OK.
```

Выделить ребра и кромки можно и не разделяя сетку на патчи. Для этого воспользуемся все той же утилитой **surfaceFeatureEdges**. Переконветрируем результирующий файл геометрии Geometry2.stl (для случая b) в формат fms с помощью следующей команды:

```
surfaceFeatureEdges ./Geometry2.stl ./Geometry2.fms -angle 45
```

В новом файле Geometry2.fms ребра будут перечислены в разделе feature edges, при этом количество патчей и их названия останутся неизменными, а результирующая сетка (после запуска **cartesianMesh**) будет выглядеть также, как и в случае а).

Заметим, что во всех представленных выше случаях, сгенерированная с помощью простейших настроек, сетка (см. рис. 23) имеет

достаточно хорошую структуру и относительно низкие показатели скошенности и неортогональности. В данном примере, подбирая только значения параметров **maxCellSize** и **boundaryCellSize**, можно добиться необходимой разрешающей способности и качества сетки.

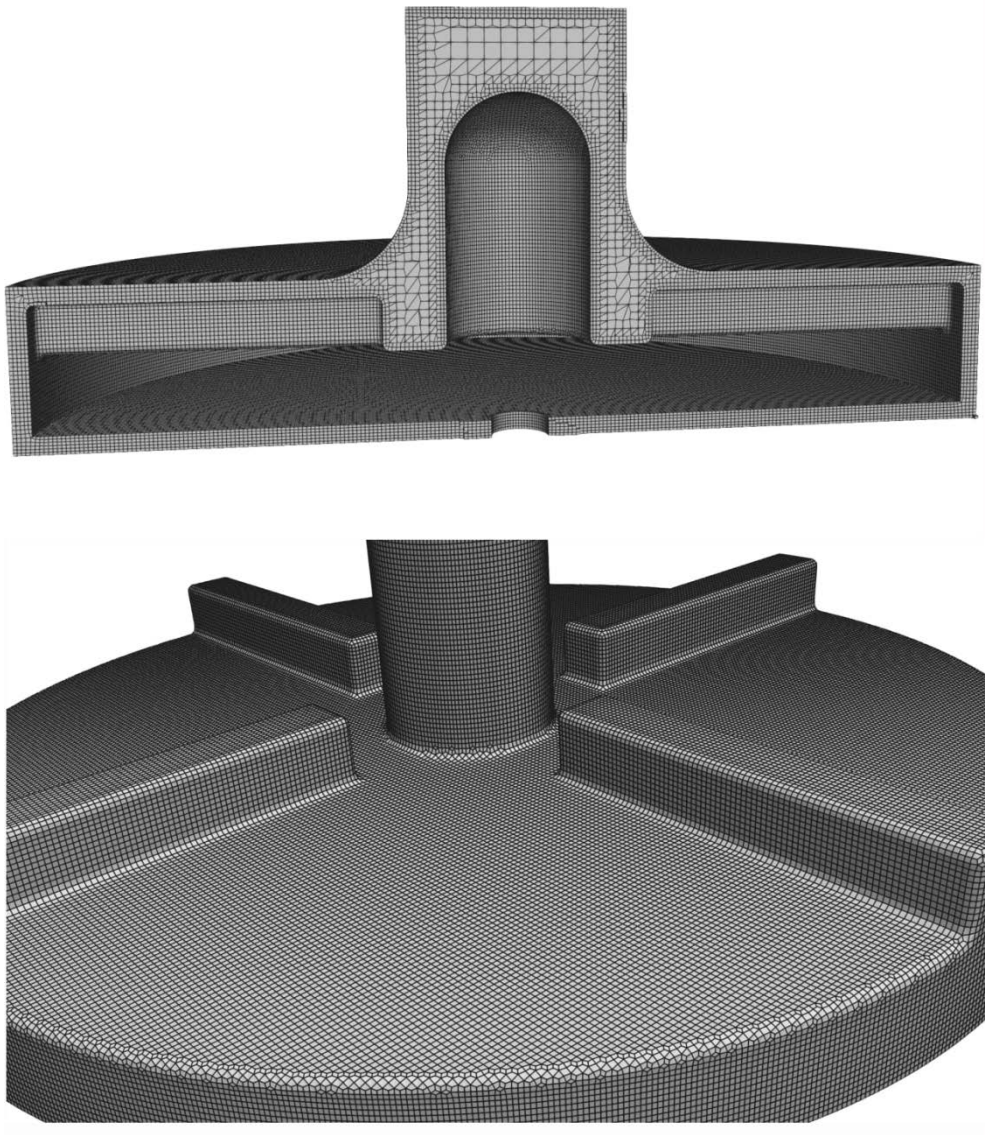


Рис. 23. Структура расчетной сетки

В заключение работы над подвижной частью сетки настроим опцию **renameBoundary**. Для случая, когда имеется 4 патча, с помощью этой опции можно задать их типы. Для этого добавим в конфигурационный файл meshDict следующие строчки:

```
    renameBoundary
    {
    newPatchNames
    {
    rotor
    {
    newName rotor;
    type wall;
    }
    rotor_wall
    {
    newName rotor_wall;
    type wall;
    }
    "interface_rotor_inlet"
    {
    newName interface_rotor_inlet;
    type patch; // далее будет изменен на cyclicAMI
    }
    interface_rotor_out
    {
    newName interface_rotor_out;
    type patch;// далее будет изменен на cyclicAMI
    }
    }
    }
```

Если патчи раздроблены, то опцию **renameBoundary** можно также использовать для их объединения. Для этого добавим в конфигурационный файл meshDict следующие строчки:

```
    renameBoundary
    {
    newPatchNames
    {
    "rotor.*" // соединим rotor_1, rotor_2, rotor_3, ro-
tor_4
    {
    newName rotor;
    type wall;
    }
    "rotor_wall.*"
    {
    newName rotor_wall;
```

```
type wall;
}
"interface_rotor_inlet.*"
{
newName interface_rotor_inlet;
type patch;
}
"interface_rotor_out.*"
{
newName interface_rotor_out;
type patch;
}
}
}
```

Нужно помнить, что для перестроения сетки необходимо вновь запустить утилиту **cartesianMesh**.

2.3. Построение неподвижных частей сетки

Аналогичным образом сгенерируем неподвижные части сетки в директориях *intel* и *outlet*. Сначала подготовим геометрию и создадим сетку в *inlet*. Перейдем в папку *путь/inlet/constant/triSurface/* и выполним там следующие команды:

```
surfaceFeatureEdges ./inlet.stl ./inlet1.stl
surfaceFeatureEdges ./inlet-extension-wall.stl ./inlet-extension-wall1.stl
surfaceFeatureEdges ./interface-rotor-inlet-extenstion.stl ./interface-rotor-inlet-extenstion1.stl

sed -i 's/patch0_./inlet/g' inlet1.stl
sed -i 's/patch0_./inlet_extension_wall/g' ./inlet-extension-wall1.stl
sed -i 's/patch0_./interface_rotor_inlet1/g' ./interface-rotor-inlet-extenstion1.stl
cat ./inlet1.stl ./inlet-extension-wall1.stl ./interface-rotor-inlet-extenstion1.stl >> GeometryInlet.stl
```

В файл *путь/inlet/system/meshDict* вставим следующий код:

```
    surfaceFile "./constant/triSurface/GeometryInlet.stl";
    maxCellSize 0.0009;
    boundaryCellSize 0.0004;

    renameBoundary
    {
    newPatchNames
    {
    inlet
    {
    newName inlet;
    type patch;
    }
    inlet_extension_wall
    {
    newName inlet_extension_wall;
    type wall;
    }
    interface_rotor_inlet1
    {
    newName interface_rotor_inlet1;
    type patch;
    }
    }
    }
```

Перейдем в папку *путь/inlet/* и запустим утилиту **cartesianMesh**. Таким образом, сетка в директории *inlet* будет создана.

Далее подготовим геометрию и создадим сетку в *outlet*. Перейдем в папку *путь/outlet/constant/triSurface/* и выполним там следующие команды:

```
    surfaceFeatureEdges ./outlet.stl ./outlet1.stl
    surfaceFeatureEdges ./outlet-extension-wall.stl ./out
let-extension-wall1.stl
    surfaceFeatureEdges ./interface-rotor-out.stl ./inter
face-rotor-out1.stl
    sed -i 's/patch0_./outlet/g' outlet1.stl
    sed  -i  's/patch0/outlet_extension_wall/g'  ./outlet-
extension-wall1.stl
    sed  -i  's/patch0_./interface_rotor_out1/g'  ./inter
face-rotor-out1.stl
```

```
    cat ./outlet1.stl ./outlet-extension-wall1.stl ./inter
face-rotor-out1.stl >> GeometryOut.stl
```

В файл *путь/outlet/system/meshDict* вставим следующий код:

```
    surfaceFile "./constant/triSurface/GeometryOut.stl";
    maxCellSize 0.0009;
    boundaryCellSize 0.0004;

    renameBoundary
    {
    newPatchNames
    {
    outlet
    {
    newName outlet;
    type patch;
    }
    "outlet_extension_wall.*"
    {
    newName outlet_extension_wall;
    type wall;
    }
    interface_rotor_out1
    {
    newName interface_rotor_out1;
    type patch;
    }
    }
    }
```

Перейдем в папку *путь/outlet/* и запустим утилиту **cartesianMesh**. Результат генерации сетки представлен на рис. 24. Таким образом, мы закончим создание трех различных частей сетки.

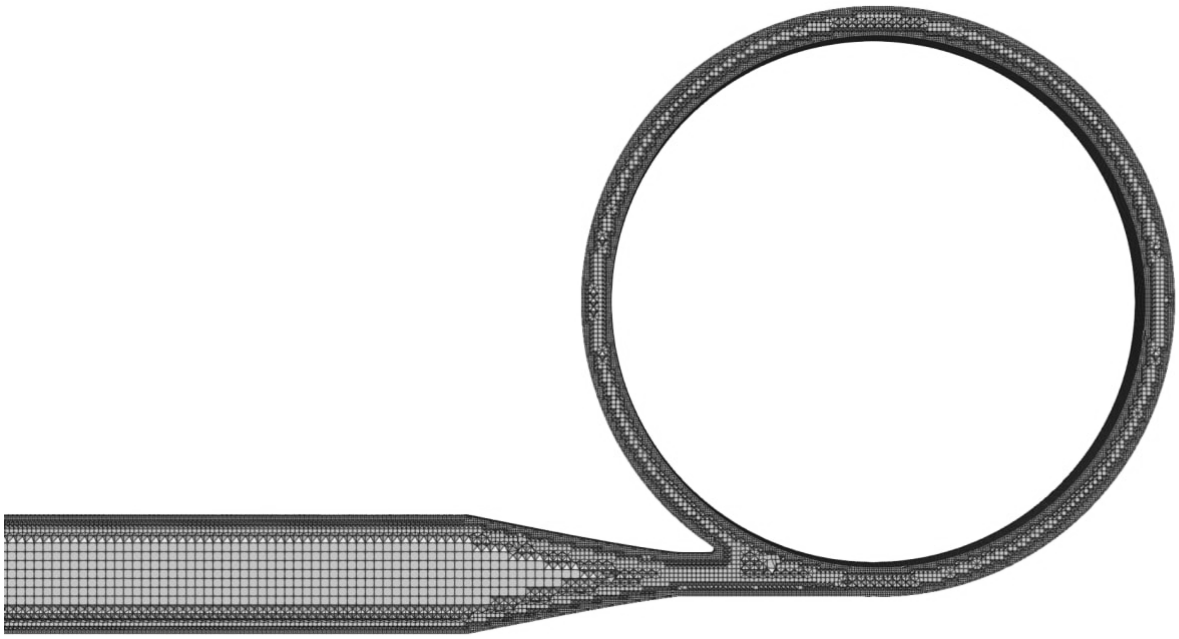


Рис. 24. Расчетная сетка (в разрезе) в области диффузора
на выходе из улитки насоса

2.4. Объединение сеток

Для проведения расчетов все части сетки необходимо соединить воедино. Перед этим выделим все ячейки подвижной части сетки в отдельную группу **cellZoneSet**, чтобы далее можно было проводить операции с этими ячейками. Создадим файл topoSetDict по адресу *путь/rotor/system/* со следующим содержимым:

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2112                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
    FoamFile
    {
        version     2.0;
        format      ascii;
        class       dictionary;
        object      topoSetDict;
    }
    // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

```
actions
(
    {
        name rotorSet;
        type cellSet;
        action clear;
    }
    {
        name rotorSet;
        type cellSet;
        action invert;
    }
    {
        name rotor;
        type cellZoneSet;
        action new;
        source setToCellZone;
        sourceInfo
        {
            set rotorSet;
        }
    }
);

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

Далее запустим утилиту **topoSet**, в результате будет образована группа **cellZoneSet** с названием rotor. После этого перейдем к соединению сеток, которое выполним в два шага. На первом шаге соединим сетку rotor с сеткой inlet. Для этого, находясь в папке *rotor*, выполним команду (предполагая, что *rotor* и *inlet* находятся в одной директории):

```
mergeMeshes ./ ../inlet/ -overwrite
```

На втором шаге соединим результирующую сетку с сеткой outlet. Для этого, находясь в папке *rotor*, выполним команду:

```
mergeMeshes ./ ../outlet/ -overwrite
```

После этого, в папке *rotor* появится соединенная сетка, которая, однако, содержит три отдельных несвязанных блока, разделенных друг от друга границами interface_rotor_inlet и interface_rotor_inlet1,

interface_rotor_out и interface_rotor_out1 (это границы-интерфейсы, которых нет в реальном насосе). Чтобы иметь возможность рассчитывать течение жидкости во всем объеме необходимо либо срастить ячейки сеток по данным границам (для этого нужно использовать утилиту **stitchMesh**), либо задать интерполяцию между ячейками на этих границах. Для решаемой динамической задачи будем использовать второй вариант.

Для настройки интерполяции нужно специальным образом переопределить типы границ. Создадим файл createPatchDict по адресу *путь/rotor/system/* со следующим содержимым:

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  v2112                                 |
|   \\  /    A nd           | Website:  www.openfoam.com                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
    FoamFile
    {
        version     2.0;
        format      ascii;
        class       dictionary;
        object      createPatchDict;
    }

    // * * * * * * * * * * * * * * ** * * * * * * * * * * * * * * //
    pointSync false;

    patches
    (
      {
          // Название нового патча
          name interface_rotor_inlet;
          // Тип нового патча
          patchInfo
          {
              type cyclicAMI; //указываем, что это
                              //граница-интерфейс
              //далее указываем соседний патч,
              //с которым будет проводится интерполяция
              neighbourPatch interface_rotor_inlet1;
          }
    // Указываем, что новый патч создается из другого патча
```

```
        constructFrom patches;
      //Название патча из которого конструируется новый
        patches (interface_rotor_inlet);
    }
    {
        name interface_rotor_inlet1;
        patchInfo
        {
            type cyclicAMI;
            neighbourPatch interface_rotor_inlet;
        }
        constructFrom patches;
        patches (interface_rotor_inlet1);
    }
    {
        name interface_rotor_out;
        patchInfo
        {
            type cyclicAMI;
            neighbourPatch interface_rotor_out1;
        }
        constructFrom patches;
        patches (interface_rotor_out);
    }
    {
        name interface_rotor_out1;
        patchInfo
        {
            type cyclicAMI;
            neighbourPatch interface_rotor_out;
        }
        constructFrom patches;
        patches (interface_rotor_out1);
    }
);
//********************************************** //
```

Для внесения этих исправлений в сетку запустим команду:

```
createPatch -overwrite
```

После этого сетка будет окончательно готова для проведения расчетов.

ЛИТЕРАТУРА

1. Нуриев А.Н. Введение в компьютерное моделирование гидродинамических процессов в программном комплексе OpenFOAM: Учебное пособие / А.Н. Нуриев, О.Н. Зайцева, А.М. Камалутдинов, О.С. Жучкова. – Казань: Казан. ун-т, 2022. – 84 с.

2. OpenFOAM v2006 User Guide. – 2022. – URL: https://www. openfoam.com/ documentation/user-guide

3. OpenFOAM Wiki. – 2021. – URL: https://wiki.openfoam. com/ Main_Page

4. Juretić F. cfMesh v1.1 User Guide / F. Juretić. – Zagreb, 2015. – URL: https://cfmesh.com/wp-content/uploads//2015/09/User_Guide-cfMesh_ v1.1.pdf

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

6. Hariharan P. Inter-Laboratory Characterization of the Velocity Field in the FDA Blood Pump Model Using Particle Image Velocimetry (PIV) / P. Hariharan, K.I. Aycock, M. Buesen, S.W. Day, B.C. Good, L. H. Herbertson, U. Steinseifer, K.B. Manning, B.A Craven, R. A. Malinauskas // CVET Journal. – 2018. – Vol. 9(4). – Pp. 623–640. – URL: https://doi.org/10.1007/s13239-018-00378-y

7. Malinauskas R.A. FDA Benchmark Medical Device Flow Models for CFD Validation/ R.A. Malinauskas, P. Hariharan, S.W. Day, L.H. Herbertson, M. Buesen, U. Steinseifer, …, B.A. Craven // ASAIO Journal. – 2017. – Vol. 63(2). – Pp. 150–160. – URL: https://doi.org/ 10.1097/MAT.0000000000000499

Учебное издание

Нуриев Артем Наилевич
Зайцева Ольга Николаевна
Жучкова Ольга Сергеевна

МОДЕЛИРОВАНИЕ ГИДРОДИНАМИЧЕСКИХ ПРОЦЕССОВ В ПРОГРАММНОМ КОМПЛЕКСЕ OPENFOAM. СОЗДАНИЕ РАСЧЕТНЫХ СЕТОК С ПОМОЩЬЮ БИБЛИОТЕКИ CFMESH

Учебное пособие

Подписано в печать 10.10.2022.
Бумага офсетная. Печать цифровая.
Формат 60x84 1/16. Гарнитура «Times New Roman».
Усл. печ. л. 4,1. Тираж 100 экз. Заказ 44/10.

Отпечатано с готового оригинал-макета
в типографии Издательства Казанского университета

420008, г. Казань, ул. Профессора Нужина, 1/37
тел. (843) 206-52-14 (1705), 206-52-14 (1704)