

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО
ОБРАЗОВАНИЯ

«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
(МОСКОВСКИЙ ПОЛИТЕХ)


09.03.01 Информатика и вычислительная техника
Образовательная программа (профиль)
«Интеграция и программирование в САПР»
Кафедра «СМАРТ-технологии»

Курсовой проект

по дисциплине:

Системы инженерного анализа

Преподаватель: _____ / _____ Лавренко И.С. /
подпись ФИО, уч. звание и
степень

Студент:  / _____ Киколиашвили. Д. З. 201-323 /
ФИО, группа, подпись(и)

Москва, 2023 г.

АННОТАЦИЯ

Киколиашвили Д. З. Создание и расчёт течения потока по каналу определённой формы.

Число страниц в отчёте 26. В работе представлено 2 приложения на 6-ти страницах, 12 иллюстраций и 11 листингов.

Данная работа посвящена разработке модели сетки канала определённой формы и расчёту течения потока по нему.

Описан процесс создания модели, создания сетки и написания кода настройки и запуска расчёта.

СОДЕРЖАНИЕ

| | |
|---|----|
| 1 ВВЕДЕНИЕ | 4 |
| 1.1 Цель работы | 4 |
| 1.2 Актуальность работы | 4 |
| 1.3 Решаемые задачи | 4 |
| 2 АНАЛИЗ И СОЗДАНИЕ МОДЕЛИ | 5 |
| 3 НАПИСАНИЕ КОДА МОДЕЛИ | 9 |
| 4 РАБОТА С КОМАНДАМИ «OpenFOAM» | 15 |
| 5 ЗАКЛЮЧЕНИЕ | 18 |
| СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ | 19 |
| ПРИЛОЖЕНИЕ 1. КОД ФАЙЛА «BLOCKMESHDict.X-CSRC» | 20 |
| ПРИЛОЖЕНИЕ 2. КОД ФАЙЛА «PHYSICALPROPERTIES.X-CSRC» | 26 |

1 ВВЕДЕНИЕ

1.1 Цель работы

Цель работы: создать сетку для расчёта течения потока по каналу заданной формы в варианте 15 курсового проекта и осуществить этот расчёт.

В данном варианте канал имеет форму как показано на Рисунок 1 - Форма канала

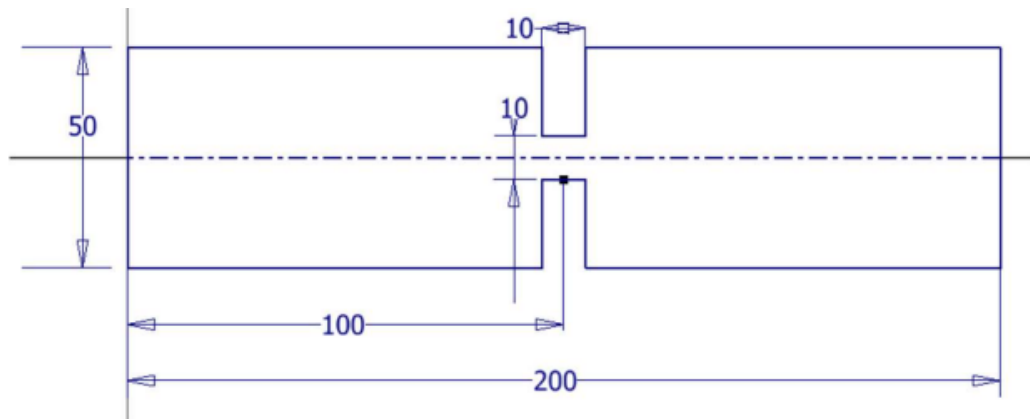


Рисунок 1 - Форма канала

1.2 Актуальность работы

Актуальность данной работы обусловлена тем, что импорт сетки из сторонних программ очень часто не даёт нужного результата при расчётах, или вообще не работает.

1.3 Решаемые задачи

Создание модели формы канала при помощи кода в программе «OpenFOAM», настройка и запуск расчёта потока воды через заданный канал.

2 АНАЛИЗ И СОЗДАНИЕ МОДЕЛИ

Для того, чтобы создать сетку для расчёта в программе «OpenFOAM» нам нужно описать нашу форму при помощи кода в файле «blockMeshDict». Как это работает: в этом файле нужно описать точки, по которым будет строиться наша сетка, затем из этих точек создать блоки, по которым программа будет понимать форму, размеры и остальные параметры сетки, такие как поверхности расчёта. Важно отметить, что этот файл делает только трёхмерные модели сеток.

Начнём строить нашу сетку. Для этого расставим точки на нашем чертеже канала (Рисунок 2).

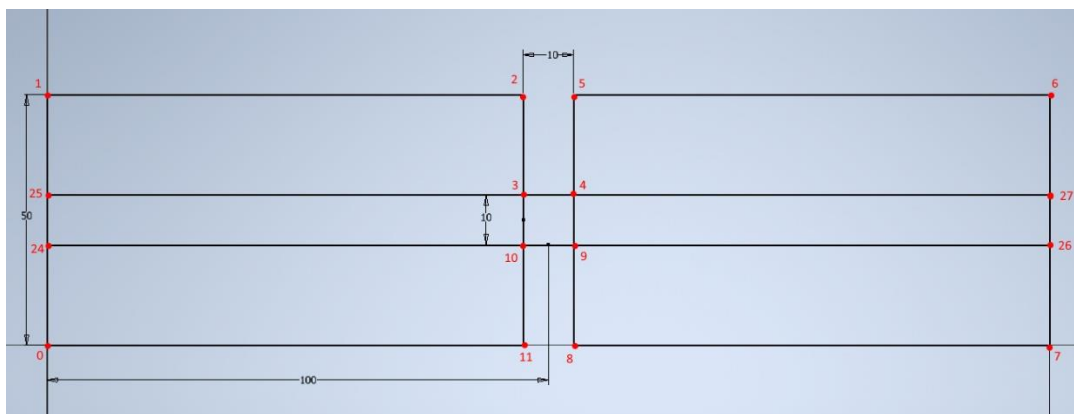


Рисунок 2 - Чертёж формы с точками

Можно заметить, что чертёж был поделен на 7 блоков, это решение будет объяснено далее. Теперь для того, чтобы наша модель сетки была трёхмерной, нам нужно создать точки, параллельные предыдущим, которые будут находиться на определённом расстоянии, в данном случае на расстоянии равном одному миллиметру. Точки и их расположение в пространстве показаны на Рисунок 3. Для наглядности была сделана трехмерная модель с увеличенным расстоянием между параллельными точками.

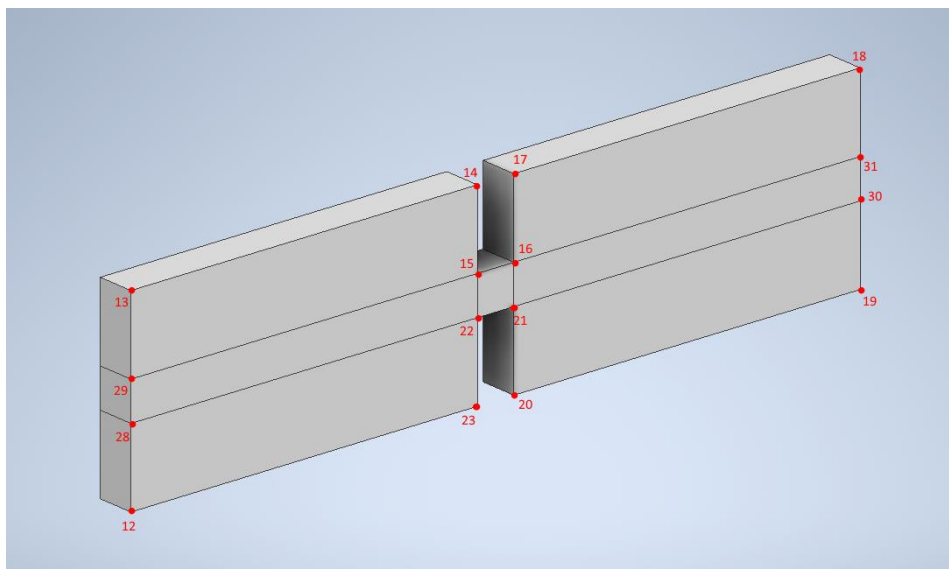


Рисунок 3 – Трёхмерная модель канала

Теперь, когда мы определились с точками, мы можем поделить модель на сегменты (Рисунок 4). Каждый сегмент выделен своим цветом.

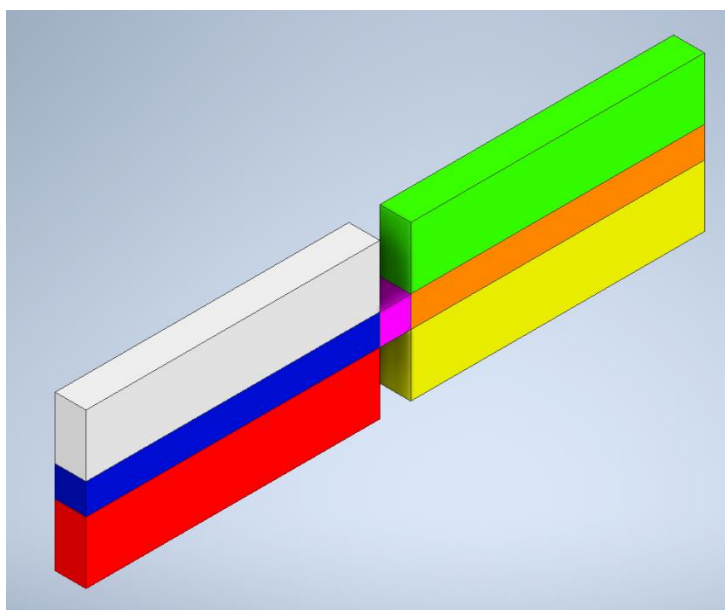


Рисунок 4 – Сегменты модели

После этого нужно определить основные поверхности сетки, для этого нужно чётко понимать какой тип расчёта мы будем проводить. В данном случае расчёт называется «pitzDailySteady». Для этого расчёта нужно определить:

- поверхность входа потока;
- поверхность выхода потока;
- верхние поверхность (потолок);

- нижние поверхность (днище);
- поверхности боковых стенок.

Эти поверхности изображены на Рисунок 5 и Рисунок 6, где:

- красным указаны поверхности боковых стенок;
- синим указана поверхность входа потока;
- зелёным указаны верхние поверхности (потолок);
- жёлтым указаны нижние поверхности (днище);
- фиолетовым указана поверхность выхода потока.

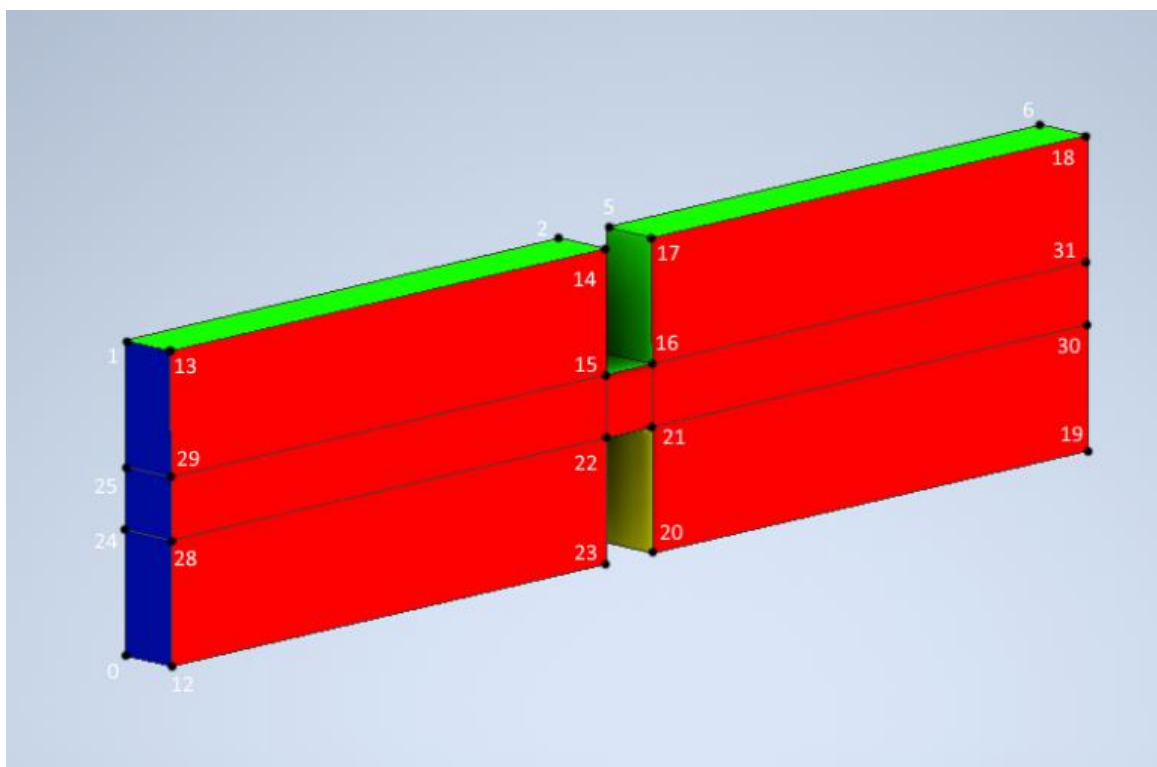


Рисунок 5 – Передняя сторона модели

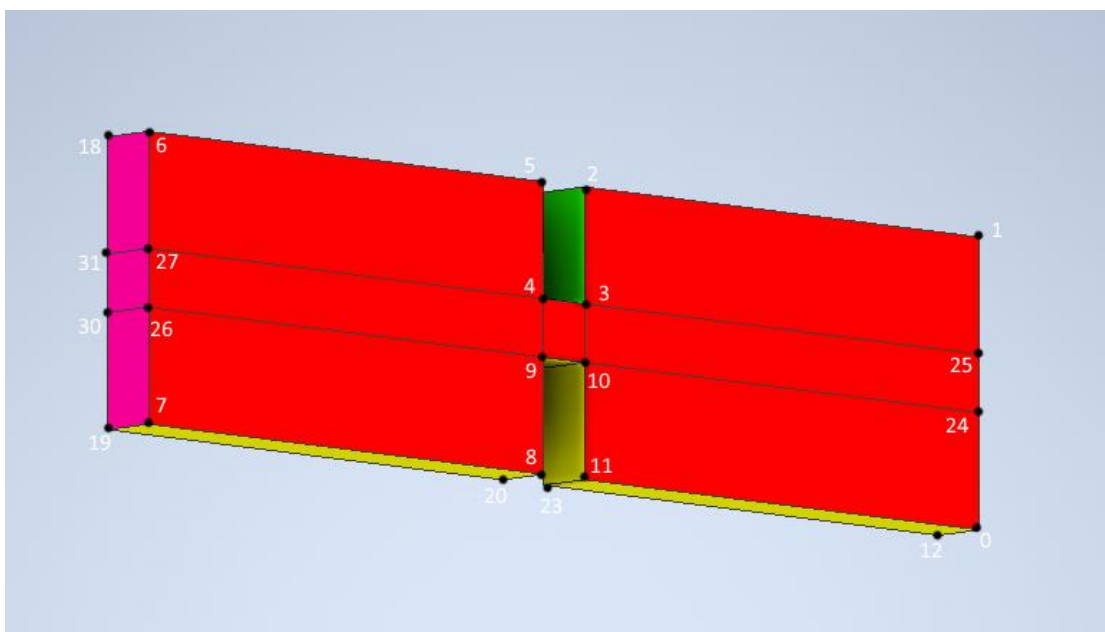


Рисунок 6 – Задняя сторона модели

Теперь, имея полное представление о том, из чего состоит наш канал, какие точки, блоки и поверхности нужно определить в коде, можно приступить к его написанию.

3 НАПИСАНИЕ КОДА МОДЕЛИ

Для начала нужно понять, как определить точку. Чтобы это сделать, нужно в «vertices» описать нашу точку по определённому шаблону: (X Y Z), где X, Y, Z – положения точки по соответствующим осям. Нумерация точек начинается с нуля. Как пример, можно взять нулевую точку с координатами (0 0 -0.5). Можно заметить, что по оси Z точка находится на расстоянии -0.5, сделано это для того, чтобы при описывании параллельной ей точки, которая будет находиться на расстоянии 0.5 по оси Z, центр ширины нашей модели находился на расстоянии 0 по оси Z.

Теперь можно перейти к определению первых основных точек 0-11 (номер точки указан после двойных косых линий), (Листинг 1).

Листинг 1 – Определение точек 0-11

```
vertices
(  
  (0 0 -0.5) // 0  
  (0 50 -0.5) // 1  
  (95 50 -0.5) // 2  
  (95 30 -0.5) // 3  
  (105 30 -0.5) // 4  
  (105 50 -0.5) // 5  
  (200 50 -0.5) // 6  
  (200 0 -0.5) // 7  
  (105 0 -0.5) // 8  
  (105 20 -0.5) // 9  
  (95 20 -0.5) // 10  
  (95 0 -0.5) // 11
```

Теперь остановимся по подробнее на некоторых точках. Если с точкой «0» мы разобрались, а положение точки «1» по оси Y видно на чертеже, то нужно расписать, как были определены координаты для второй, третьей, четвёртой и девятой точек.

Вторая точка по оси X находится на расстоянии 95, потому что её положение высчитывается из чертежа, где видно, что она находится на левом краю центрального квадрата, размеры которого равны 10-ти миллиметрам. Так как центр квадрата является центром нашей модели, то можно посчитать, что

наша точка находится от центра нашей модели левее на половину ширины квадрата. Из этого можно посчитать, что наша точка находится левее центра на 5 миллиметров, а так как центр нашей модели находится на расстоянии 100, то можно сделать вывод, что точка находится на расстоянии 95 миллиметров по оси X.

Третья точка находится на том же расстоянии по оси X, что и вторая, но на другом расстоянии по оси Y, которое высчитывается суммой половины всей высоты модели с половиной высоты квадрата, что равно 30 миллиметрам.

Расстояние по оси X четвертой точки считается аналогично второй, но при этом она находится правее центра модели.

Расстояние по оси Y девятой точки считается аналогично третьей, но мы не суммируем половину всей высоты с половиной высоты квадрата, а вычитаем из половины всей высоты половину высоты квадрата и получаем 20 миллиметров.

Теперь опишем точки, зеркальные предыдущим, основным отличием будет только расстояние по оси Z, которое будет равно -0.5 миллиметров (Листинг 2 – Определение точек 12-23).

Листинг 2 – Определение точек 12-23

```
(0 0 0.5) // 12
(0 50 0.5) // 13
(95 50 0.5) // 14
(95 30 0.5) // 15
(105 30 0.5) // 16
(105 50 0.5) // 17
(200 50 0.5) // 18
(200 0 0.5) // 19
(105 0 0.5) // 20
(105 20 0.5) // 21
(95 20 0.5) // 22
(95 0 0.5) // 23
```

И наконец, опишем промежуточные точки, которые будут делить левую и правую часть на дополнительные блоки. Из особенностей: они находятся на высотах точек вершин центрального квадрата (Листинг 3).

Листинг 3 – Определение точек 28-31

```
(0 20 0.5) // 28  
(0 30 0.5) // 29  
(200 20 0.5) // 30  
(200 30 0.5) // 31
```

Имея все точки, мы можем определить блоки, на которые мы поделим нашу модель (Рисунок 7). Точка «9» находится позади точки «15», поэтому выделена в скобки.

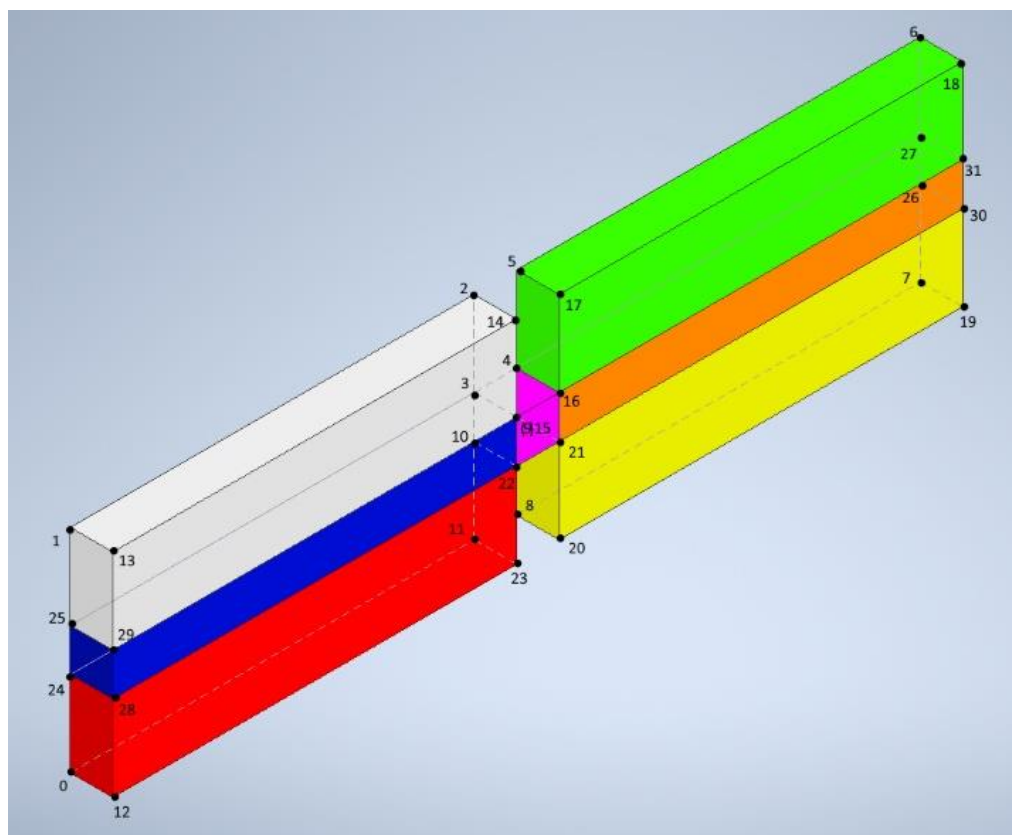


Рисунок 7 – Сегментированная модель с точками

Для того, чтобы определить блок, нам нужно записать в определённой последовательности точки боковых поверхностей блока. Как пример возьмём красный блок. У него есть 2 поверхности, начнём с задней. Чтобы её определить в блоке, нужно записать её точки начиная с левого нижнего угла, идя против часовой стрелки, тогда у нас будет последовательность 0, 11, 10, 24. Затем таким же образом нам нужно записать точки второй поверхности, тогда у нас получится 12, 23, 22, 28. И теперь, имея эти последовательности,

их можно записать понятным для программы кодом: `hex(0 11 10 24 12 23 22 28)`. Но это ещё не всё, так же нужно описать то, как у нас будет строиться сетка. Для этого под кодом `hex` нужно написать `(X Y Z)`, где в `X`, `Y`, `Z` будет написано количество блоков сетки по соответствующим осям, после этого написать `simpleGrading(1 1 1)`, это нужно для того, чтобы программа понимала, как распределить блоки сетки, в данном случае оно распределит их на равном друг от друга расстоянии по осям `X`, `Y`, `Z`. Тогда по итогу у нас получится код как на Листинг 4 – Определение первого блока.

Листинг 4 – Определение первого блока

```
hex (0 11 10 24 12 23 22 28)
(25 10 1)
simpleGrading (1 1 1)
```

Аналогично поступаем с остальными блоками и получаем такой код (Листинг 5).

Листинг 5 – Определение блоков модели

```
blocks
(
  hex (0 11 10 24 12 23 22 28)
  (25 10 1)
  simpleGrading (1 1 1)

  hex (24 10 3 25 28 22 15 29)
  (25 10 1)
  simpleGrading (1 1 1)

  hex (25 3 2 1 29 15 14 13)
  (25 10 1)
  simpleGrading (1 1 1)

  hex (10 9 4 3 22 21 16 15)
  (25 10 1)
  simpleGrading (1 1 1)

  hex (8 7 26 9 20 19 30 21)
  (25 10 1)
  simpleGrading (1 1 1)

  hex (9 26 27 4 21 30 31 16)
  (25 10 1)
  simpleGrading (1 1 1)

  hex (4 27 6 5 16 31 18 17)
  (25 10 1)
  simpleGrading (1 1 1)

);
```

Осталось только определить поверхности. Поверхности определяются похожим на определение блоков способом. Нужно взять определённую поверхность и описать её точки против часовой стрелки. Как пример возьмём поверхность входа потока Рисунок 5. Можно заметить, что в ней присутствуют поверхности с каждого блока модели, это значит, что чтобы описать эту поверхность нужно описать каждую из поверхностей путём записи точек против часовой стрелки. По итогу мы получим три набора точек:

- 0, 12, 28, 24;
- 24, 28, 29, 25;
- 25, 29, 13, 1.

Имея эти точки, мы можем определить поверхность входа потока. Для этого в поле «boundary» зададим поверхность «inlet» тип поверхности «patch», а точки запишем тремя отдельными группами в скобках (Листинг 6).

Листинг 6 – Код поверхности входа потока

```
boundary
(
  inlet
  {
    type patch;
    faces
    (
      (0 12 28 24)
      (24 28 29 25)
      (25 29 13 1)
    );
  }
)
```

Аналогично поступаем с поверхностью выхода потока, назвав её «outlet» (Листинг 7).

Листинг 7 – Код поверхности выхода потока

```
outlet
{
  type patch;
  faces
  (
    (19 7 26 30)
    (30 26 27 31)
    (31 27 6 18)
  );
}
```

Теперь остановимся подробнее на верхней стенке. В частности, ранее я упоминал, что мы делим модель на 7 частей, как раз нужно это для того, чтобы правильно сделать верхнюю стенку.

Как можно видеть из Рисунок 7 и Рисунок 5 верхняя стенка проходит через 3 блока модели, а точнее белого, фиолетового и зелёного. Если бы мы сделали левую часть модели одним блоком и аналогично поступили бы с правой частью, то программа бы не смогла построить поверхности верхней стенки. Почему? Можно заметить, что точка «4» и точка «16» относятся как к фиолетовому блоку, так и к зелёному. В случае, если бы правая часть модели была одним блоком, то тогда у её левой поверхности были бы только точки 8, 20, 17, 5, из-за этого программа бы не понимала, что точки 4 и 16 на самом деле находятся не только на поверхности фиолетового блока, но и на поверхности правой части модели. Задаём ей тип «wall» и из этого следует описание верхней стенки в Листинг 8.

Листинг 8 – Определение поверхности верхней стенки

```
upperWall
{
    type wall;
    faces
    (
        (1 13 14 2)
        (15 3 2 14)
        (3 15 16 4)
        (4 16 17 5)
        (5 17 6 18)
    );
}
```

Теперь аналогично с верхней стенкой определяем нижнюю (Листинг 9).

Листинг 9 – Определение поверхности нижней стенки

```
lowerWall
{
    type wall;
    faces
    (
        (0 12 23 11)
        (23 11 10 22)
        (22 10 9 21)
        (8 20 21 9)
        (8 20 19 7)
    );
}
```

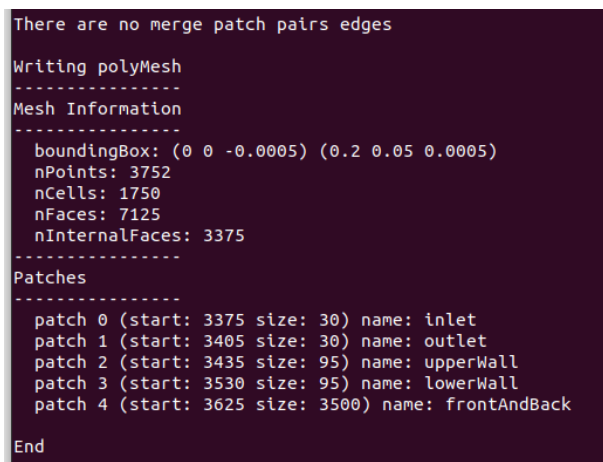
В случае с боковыми стенками нужно просто описать все боковые поверхности нашей модели и задать им тип «empty», потому что наш расчёт является по своей сути двухмерным и эти стенки не будут на него влиять. Код указан в Листинг 10.

Листинг 10 – Определение боковых стенок

```
frontAndBack
{
    type empty;
    faces
    (
        (0 11 10 24)
        (24 10 3 25)
        (25 3 2 1)
        (10 9 4 3)
        (8 7 26 9)
        (9 26 27 4)
        (4 27 6 5)
        (12 23 22 28)
        (28 22 15 29)
        (29 15 14 13)
        (22 21 16 15)
        (20 19 30 21)
        (21 30 31 16)
        (16 31 18 17)
    );
}
```

4 РАБОТА С КОМАНДАМИ «OpenFOAM»

Теперь протестируем создание нашей модели. Для этого при помощи консоли в папке с расчётом нужно запустить команду «blockMesh», после чего мы получим результат как на Рисунок 8.



```
There are no merge patch pairs edges
Writing polyMesh
-----
Mesh Information
-----
boundingBox: (0 0 -0.0005) (0.2 0.05 0.0005)
nPoints: 3752
nCells: 1750
nFaces: 7125
nInternalFaces: 3375
-----
Patches
-----
patch 0 (start: 3375 size: 30) name: inlet
patch 1 (start: 3405 size: 30) name: outlet
patch 2 (start: 3435 size: 95) name: upperWall
patch 3 (start: 3530 size: 95) name: lowerWall
patch 4 (start: 3625 size: 3500) name: frontAndBack
End
```

Рисунок 8 – Результат команды «blockMesh»

Теперь в файле «physicalProperties» поменяем параметр « ν » на значение $1e-10$, чтобы у нас был поток воды, а не воздуха (Листинг 11).

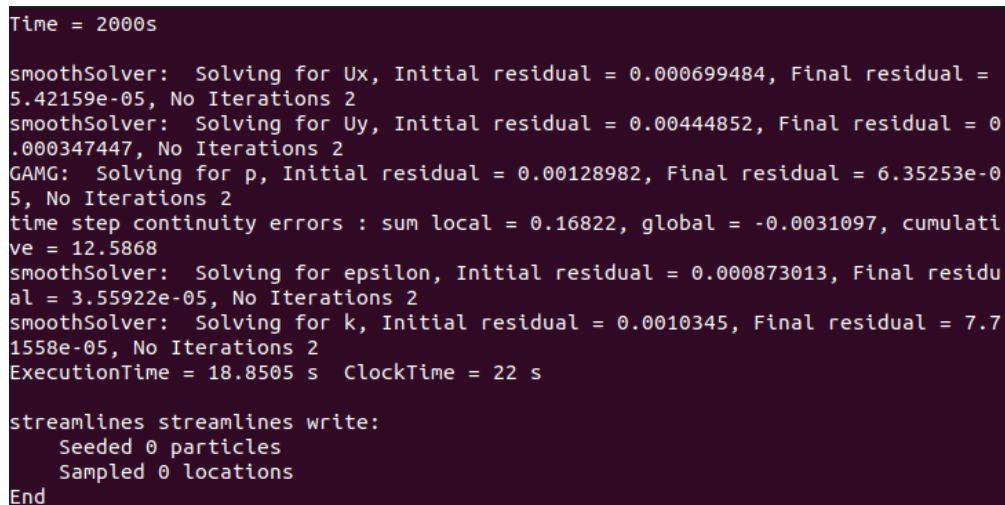
Листинг 11 – Файл «physicalProperties»

```
FoamFile
{
    format      ascii;
    class       dictionary;
    location    "constant";
    object      physicalProperties;
}
// * * * * *

viscosityModel constant;

nu              1e-10;
```

По итогу мы можем запустить наш расчёт при помощи команды «foamRun» (Рисунок 9 – Результат команды «foamRun») и увидеть его результат при помощи команды «paraFoam» (Рисунок 10, Рисунок 11, Рисунок 12).



```
Time = 2000s
smoothSolver: Solving for Ux, Initial residual = 0.000699484, Final residual = 5.42159e-05, No Iterations 2
smoothSolver: Solving for Uy, Initial residual = 0.00444852, Final residual = 0.000347447, No Iterations 2
GAMG: Solving for p, Initial residual = 0.00128982, Final residual = 6.35253e-05, No Iterations 2
time step continuity errors : sum local = 0.16822, global = -0.0031097, cumulative = 12.5868
smoothSolver: Solving for epsilon, Initial residual = 0.000873013, Final residual = 3.55922e-05, No Iterations 2
smoothSolver: Solving for k, Initial residual = 0.0010345, Final residual = 7.71558e-05, No Iterations 2
ExecutionTime = 18.8505 s  ClockTime = 22 s

streamlines streamlines write:
    Seeded 0 particles
    Sampled 0 locations
End
```

Рисунок 9 – Результат команды «foamRun»

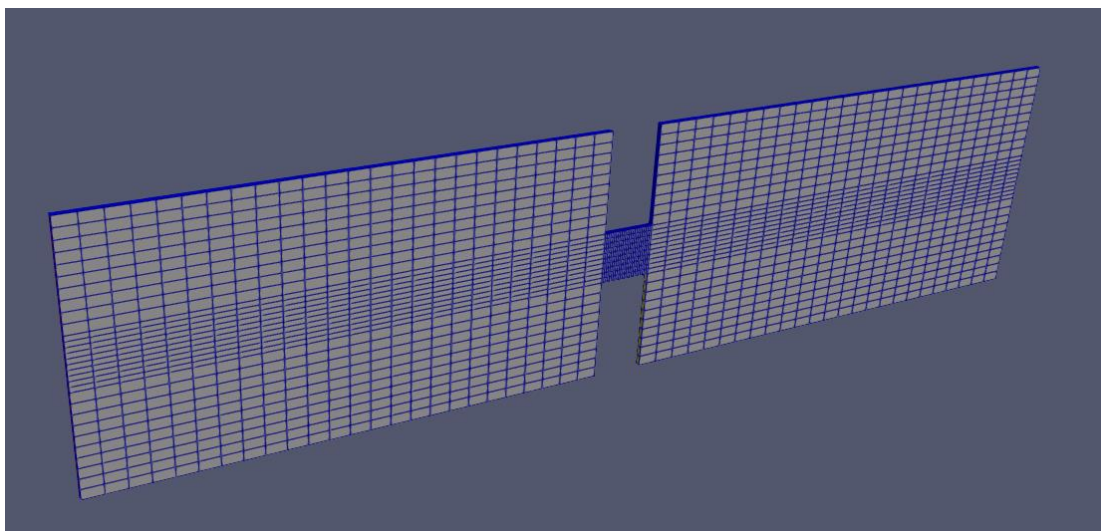


Рисунок 10 – Сетка в ParaFoam

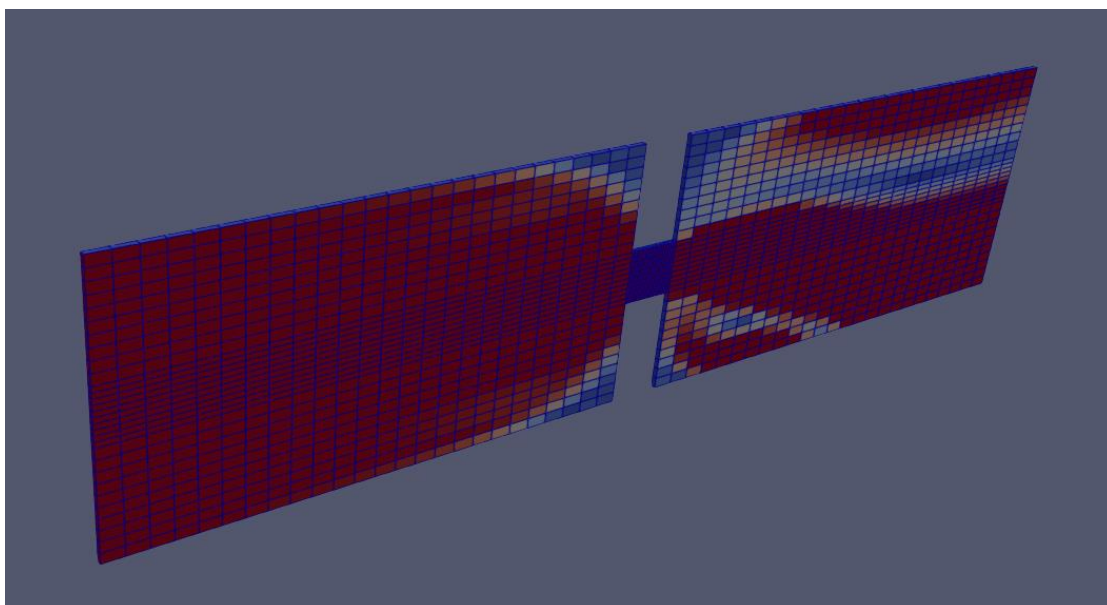


Рисунок 11- Расчёт «U» в ParaFoam

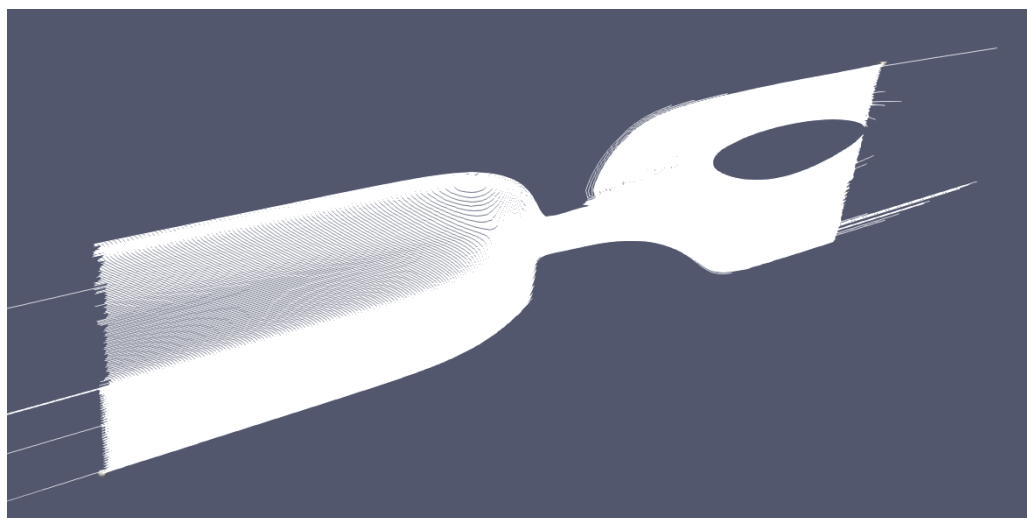


Рисунок 12 – Линии потоков в ParaFoam

5 ЗАКЛЮЧЕНИЕ

По итогу данной курсовой работы были получены знания работы с «OpenFOAM», созданию и настройке расчётов в данной программе и навыки написания сетки при помощи кода для «OpenFOAM».

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Никифоров А. И. Термодинамика и теплопередача: методические указания по выполнению курсовой работы «Расчёт сопла Лаваля» / Никифоров А. И.; рецензент: канд. техн. наук, доц. Т.В Петрова. – Санкт-Петербург: Университет гражданской авиации, 2019. – 29 с.
2. Изотов Б.А. Расчет и построение профиля сопла Лаваля: методические указания к лабораторной работе по курсу «Термодинамика и теплопередача» / Б.А. Изотов.; рецензент: канд. техн. наук, доц. А.Д.Припадчев. – Выпуск 1. Серия научно-методических документов, содействующих профессиональному самоопределению студентов в учебном процессе «Я – специалист». – Оренбург: ГОУ ОГУ, 2009. – 20 с.
3. OpenFOAM.org: сайт. – Лондон, 2011 – 2021. – URL: <https://openfoam.org/> (дата обращения: 10.10.2023).
4. Salome: сайт. – Франция, 2005 – 2022. – URL: <https://www.salome-platform.org/> (дата обращения: 9.10.2023).
5. ParaView.org: сайт. – США, 2002 – . – URL: <https://www.paraview.org/> (дата обращения: 15.10.2023).
6. Введение в компьютерное моделирование в программном комплексе OpenFOAM: учебное пособие / А.Н. Нуриев, О.Н. Зайцева, А.М. Камалутдинов. – Казань: Казан. ун-т, 2021. – 65 с.
7. Обучение OpenFOAM: [József Nagy] // Youtube : [видеохостинг]. – URL: <https://www.youtube.com/c/JózsefNagyOpenFOAMGuru> (дата обращения: 9.10.2023).

ПРИЛОЖЕНИЕ 1. КОД ФАЙЛА «BLOCKMESHDict.X-CSRC»

```
/*-----*- C++ -*-----*\
===== |
\\ / Field | OpenFOAM: The Open Source CFD Toolbox
\\ / Operation | Website: https://openfoam.org
\\ / And | Version: dev
\\ \ Manipulation |
\*-----*/

FoamFile
{
    format    ascii;
    class     dictionary;
    object    blockMeshDict;
}

// * * * * *

// Note: this file is a Copy of
$FOAM_TUTORIALS/resources/blockMesh/pitzDaily

convertToMeters 0.001;

vertices
(
    (0 0 -0.5) // 0
    (0 50 -0.5) // 1
    (95 50 -0.5) // 2
    (95 30 -0.5) // 3
    (105 30 -0.5) // 4
```

(105 50 -0.5) // 5

(200 50 -0.5) // 6

(200 0 -0.5) // 7

(105 0 -0.5) // 8

(105 20 -0.5) // 9

(95 20 -0.5) // 10

(95 0 -0.5) // 11

(0 0 0.5) // 12

(0 50 0.5) // 13

(95 50 0.5) // 14

(95 30 0.5) // 15

(105 30 0.5) // 16

(105 50 0.5) // 17

(200 50 0.5) // 18

(200 0 0.5) // 19

(105 0 0.5) // 20

(105 20 0.5) // 21

(95 20 0.5) // 22

(95 0 0.5) // 23

(0 20 -0.5) // 24

(0 30 -0.5) // 25

(200 20 -0.5) // 26

(200 30 -0.5) // 27

(0 20 0.5) // 28

(0 30 0.5) // 29

(200 20 0.5) // 30

```
(200 30 0.5) // 31

);

blocks
(
  hex (0 11 10 24 12 23 22 28)
  (25 10 1)
  simpleGrading (1 1 1)

  hex (24 10 3 25 28 22 15 29)
  (25 10 1)
  simpleGrading (1 1 1)

  hex (25 3 2 1 29 15 14 13)
  (25 10 1)
  simpleGrading (1 1 1)

  hex (10 9 4 3 22 21 16 15)
  (25 10 1)
  simpleGrading (1 1 1)

  hex (8 7 26 9 20 19 30 21)
  (25 10 1)
  simpleGrading (1 1 1)
```

```

hex (9 26 27 4 21 30 31 16)
(25 10 1)
simpleGrading (1 1 1)

hex (4 27 6 5 16 31 18 17)
(25 10 1)
simpleGrading (1 1 1)

);

boundary
(
  inlet
  {
    type patch;
    faces
    (
      (0 12 28 24)
      (24 28 29 25)
      (25 29 13 1)
    );
  }
  outlet
  {
    type patch;
    faces
    (

```

```
(19 7 26 30)
(30 26 27 31)
(31 27 6 18)
```

```
);
}
```

```
upperWall
```

```
{
    type wall;
    faces
    (
        (1 13 14 2)
        (15 3 2 14)
        (3 15 16 4)
        (4 16 17 5)
        (5 17 6 18)
    );
}
```

```
lowerWall
```

```
{
    type wall;
    faces
    (
        (0 12 23 11)
        (23 11 10 22)
        (22 10 9 21)
        (8 20 21 9)
        (8 20 19 7)
    );
}
```



```

}
frontAndBack
{
    type empty;
    faces
    (
        (0 11 10 24)
        (24 10 3 25)
        (25 3 2 1)
        (10 9 4 3)
        (8 7 26 9)
        (9 26 27 4)
        (4 27 6 5)
        (12 23 22 28)
        (28 22 15 29)
        (29 15 14 13)
        (22 21 16 15)
        (20 19 30 21)
        (21 30 31 16)
        (16 31 18 17)
    );
}
);

//
*****
***** //

```

ПРИЛОЖЕНИЕ 2. КОД ФАЙЛА «PHYSICALPROPERTIES.X-CSRC»

```
/*-----*- C++ -*-----*\
===== |
\\ / F i e l d | OpenFOAM: The Open Source CFD Toolbox
\\ / O p e r a t i o n | Website: https://openfoam.org
\\ / A n d | Version: dev
\\ \ M a n i p u l a t i o n |
\*-----*/

FoamFile
{
    format    ascii;
    class     dictionary;
    location  "constant";
    object    physicalProperties;
}

// * * * * *

viscosityModel constant;

nu          1e-10;

//
*****

***** //
```