МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«ТЮМЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ НАУК
Кафедра фундаментальной математики и механики

ОТЧЕТ ПО НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

Выполнил работу
Студент 1 курса,
аспирантуры
очной формы обучения

Кусайко
Георгий
Николаевич

Руководитель работы
к.ф.-м.н., ?

Игошин
Дмитрий
Евгеньевич

Тюмень, 2021

Содержание

1 Введение

На момент написания отчета были сформулированы следующие цели и задачи:

- подбор необходимой литературы и документации;
- изучение следующих вычислительных программных пакетов и утилит:
  - SALOME;
  - OpenFOAM;
  - Paraview.
- автоматизация вычислительного процесса, а также использование параллельных вычислений.

2 Процесс вычисления

Обычно, моделирование и процесс вычисления происходит последовательно по следующим этапам:

1. **Подготовительный этап**. На данном этапе формируется геометрия модели, формулируются необходимые физические условия, геометрия дискретизируется, задаются начальные и граничные условия дифференциальных уравнений;

2. **Расчёт**. На данном этапе машина, подчиняясь заданному алгоритму, численно решает основные уравнения с точки зрения базовых физических параметров (скорость, давление, плотность, температура, энтальпия и т. д.), а также записывает результаты решения в память;

3. **Постобработка**. Выполняется конвертация результатов расчёта в форматы, пригодные для отображения и визуализации.

4. **Анализ**. Результаты решения отображаются в виде графиков, таблиц, а также контурных или векторных схем, привязанных к исходной геометрии.

На этих этапах чаще всего используется не один, а сразу ряд инструментов. Такой подход и будет рассматриваться в дальнейшем.

3 Подготовительный этап

В качестве модели возьмем модифицированную объемно-центрированную кубическую решетку. Радиус зерен (сфер) будет

$$R = \frac{1}{1-\alpha},$$

где α - параметр, определяющий пересечение зерен. Т.е. при $\alpha = 0$ они не будут пересекаться, но будут соприкасаться. Размер самой структуры

$$\Delta x = 2\sqrt{2}, \Delta y = 2\sqrt{2}, \Delta z = 2.$$

Построение геометрии, а также дискретизацию будем проводить в программном пакете SALOME. Данный пакет обладает обширным функционалом, а рабочий про-

цесс обычно производится через графический интерфейс. Однако почти все подключаемые модули написаны на языке программирования Python. Это дает возможность для написания исполняемых скриптов, которые можно запускать из командной строки. Для этого требуется запустить сессию сервера SALOME и передать туда скрипт.

Недостатком же SALOME, является то, что запущенная сессия может работать только на одном ядре процессора, т.е. если потребуется построить несколько структур с отличающими параметрами и дискретизировать их за один раз, это будет исполнено последовательно, что сильно скажется на затраченном времени. Таким образом, приходим к тому, что необходимо распараллелить последовательность вычислений для разных параметров. В данном случае это будет параметр α.

Для решения данного момента используем стандартную библиотеку Python, которая называется multiprocessing. Также, чтобы инициализировать сессию SALOME и передать туда исполняемый скрипт, потребуется библиотека subprocess.

```
1  import os
2  import subprocess
3  import multiprocessing
4
5  src = os.getcwd()
6  build = os.path.join(src, "../build")
7
8  if not os.path.exists(build):
9      os.makedirs(build)
10
11 ###
12 alpha = [0.1, 0.15, 0.2]
13 simpleCubic = os.path.join(src, "simple-cubic/main.py")
14
15 ###
16 processes = []
17 structure = ["simple-cubic"]
18
19 def salome(src_path, build_path, arg):
20     subprocess.run(["salome", "start", "-t", src_path, "args:{},{}".
       format(build_path, arg)])
21
```

```
22 for s in structure:
23     s_path = os.path.join(build, s)
24 
25     for c in alpha:
26         src_path = os.path.join(src, "%s/main.py" % s)
27         build_path = os.path.join(s_path, str(c))
28 
29         if not os.path.exists(build_path):
30             os.makedirs(build_path)
31 
32         print("starting process")
33         p = multiprocessing.Process(target = salome, args = (src_path,
    build_path, c))
34         processes.append(p)
35         p.start()
36 
37     for process in processes:
38         process.join()
```

Листинг 1: genmesh.py

Как мы можем видеть на Листинге 1, происходят следующее важные шаги:

1. Подключаем необходимые библиотеки;
2. Задаем параметр α, а также структуры, которые необходимо просчитать (в данном случае одну);
3. Создаем функцию, через которую будем запускать сессиию SALOME с необходимыми аргументами.
4. Инициализируем процесс для каждой структуры, а также для каждого параметра α, которые будут выполняться параллельно, в зависимости от количества ядер процессора, а также их занятости.

Также можно заметить, что был использован скрипт <simple-cubic/main.py>, в котором как раз и содержится построение геометрии модели и ее дискретизация. Это файл тут не будет прикладываться, так как он объемный и разбит на несколько логических модулей.

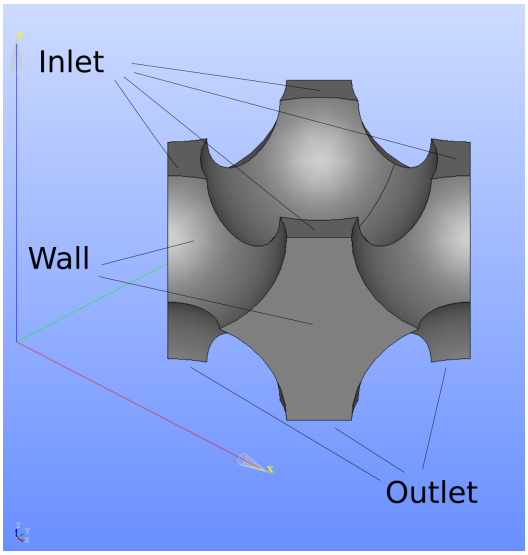

Рис. 1: ASD

Стоит упомянуть, что полученные поверхности геометрии структуры были сгруппированы как inlet, outlet и wall, т.е. поверхности для входящего потока, выходящего и стенок соответственно. Это шаг необходим для задания начальных и граничных условий в OpenFOAM. Так как мы задаем однофазный поток, направленный вертикально вниз, то получаем группы, которые изображены на Рис. 1. Граничные и начальные условия:

$$wall : \vec{v} = 0, \frac{dp}{dn} = 0;$$

$$p_{inlet} = 1\text{Па};$$

$$p_{outlet} = 0;$$

$$\vec{v}(x, y, z) = 0.$$

Численное решение уравнений

$$\nabla \cdot \vec{v} = 0,$$

$$\nabla \cdot (\vec{v}\vec{v}) - \nu\Delta\vec{v} = -\frac{1}{\rho}\nabla p,$$

будет осуществляться с помощью решателя SIMPLE.

4 Расчёт

После дискретизации геометрии получаем на выходе файл формата UNV, который содержит информацию по сетке, а также по группам, необходимым для начальных и граничных условий. В силу небольшой специфики OpenFOAM, нужно создать директорию, где и будут находиться файлы, необходимые для расчёта.

```
simple-cubic/0.1
|-- 0
|   |-- U
|   |-- p
|-- constant
|   |-- transportProperties
|   |-- turbulenceProperties
|-- mesh.unv
|-- system
    |-- controlDict
    |-- decomposeParDict
    |-- fvSchemes
    |-- fvSolution
```

Листинг 2: tree simple-cubic/0

Для импортирования сетки из формата UNV в формат, понятный для OpenFOAM, используется утилита ideasUnvFoam. После ее применения появляется директория constant/polyMesh, в которой необходимо изменить в файле boundary значение для группы wall на wall, так как стандартно после импортирования для всех групп задается значение patch.

Таким образом, мы подготовили все к расчету, остался только момент с параллельными вычислениями. OpenFOAM изначально их поддерживает, а реализация осуществляется через OpenMPI. Однако чтобы запустить, требуется сначала выполнить декомпозицию текущей директории с помощью decomposePar. Настройки для данной утилиты задаются в system/decomposeParDict соответственно. Допустим, у нас есть 4 ядра процессора, которые можем использовать для расчета, тогда после выполнения получим на выходе 4 связанные директории processor0 - processor3 (индексация с нуля), в которых будут содержаться измененные файлы с начальными и

граничными условиями 0/U и 0/p, а также constant/.

Далее остается только запустить расчёт, попутно записывая стандартный вывод утилиты в файл simpleFoam.log (см. Листинг 3).

```
$ mpirun -np 4 simpleFoam -parallel | tee simpleFoam.log

```

Листинг 3: sh

Результируя, получаем следующий bash-скрипт, который автоматизирует процесс запуска вычислений для каждого параметра α (см. Листинг 4).

```
#!/usr/bin/env bash

cd src
# Generate mesh
python src/genmesh.py
python prefoam.py
cd ../

#
for alpha in $( find build/simple-cubic -maxdepth 1 -type d); do
    path="$alpha/system/controlDict"

    if [ -f $path ]; then
        ideasUnvFoam -case $path mesh.unv
        checkMesh -case $path
        foamDictionary -case $path constant/polyMesh/boundary -entry
    entry0.wall.type -set wall
        decomposePar -case $path
        mpirun -np 4 --oversubscribe simpleFoam -parallel -case $path >
    $path/simpleFoam.log
    fi
done
```

Листинг 4: compute.bash

5 Анализ результатов

Визуализация будет проводиться с помощью программного пакета ParaView. Данный пакет также имеет широкий функционал, позволяющий отображать не только объемные фигуры и строить графики, но и проводить дополнительные манипуляции с объектами, например, срезы, отображение полей для полученных физический величин. Однако этот этап процесса вычисления не будет приведен в данном отчете по следующим причинам:

1. Требуется доработка, связанная со сходимостью решения. Необходима автоматизация корректировки параметров влияющих на это, например, коэффициентов релаксации и др.

2. Расчёт ведется еще для нескольких структур. Финальные результаты всей работы будут оформлены в статье, поэтому данные не раскрыты, а параметры упомянутые выше выбраны условно.

6 Выводы

7 Список литературы