Применение СПО для решения задач аэро- и газовой динамики

Семинар 2. Запуск задачи обтекания цилиндра

Преподаватель: Романова Дарья Игоревна

Институт системного программирования им. В.П. Иванникова РАН, Москва

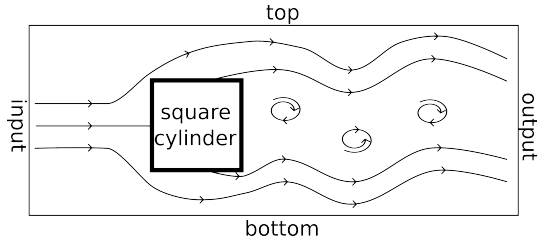
2023

Схема задачи обтекания квадратного цилиндра

Математическая модель:

$$\nabla \cdot \boldsymbol{U} = 0,$$

$$\frac{\partial \boldsymbol{U}}{\partial t} + \nabla \cdot \boldsymbol{U}\boldsymbol{U} = -\nabla p + \nabla \cdot \boldsymbol{\tau},$$



где $\boldsymbol{U}$ — скорость, $\boldsymbol{\tau} = 2\nu\boldsymbol{s}$ — тензор вязких напряжений, $\boldsymbol{s} = 0.5(\nabla \boldsymbol{U} + (\nabla \boldsymbol{U})^T)$ — тензор скоростей деформаций, ν — кинематическая вязкость, p — давление.

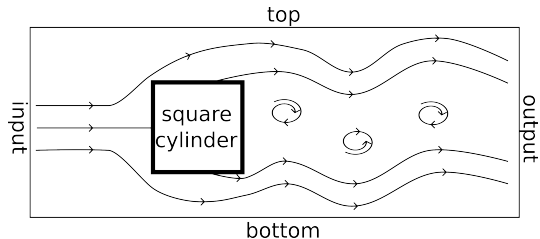
Геометрия расчётной области:

- Сторона квадрата $D = 1$
- Высота расчётной области $H = 30D$
- Длина расчётной области $L = 80D$
- Толщина $B = D$

Начальные и граничные условия

Течение ламинарное:

$$Re = \frac{UD}{\nu} = \frac{1 \cdot 1}{0.01} = 100$$



Начальные условия

$\boldsymbol{U} = 1$	$p = 0$	$\nu = 0.01$

Граничные условия

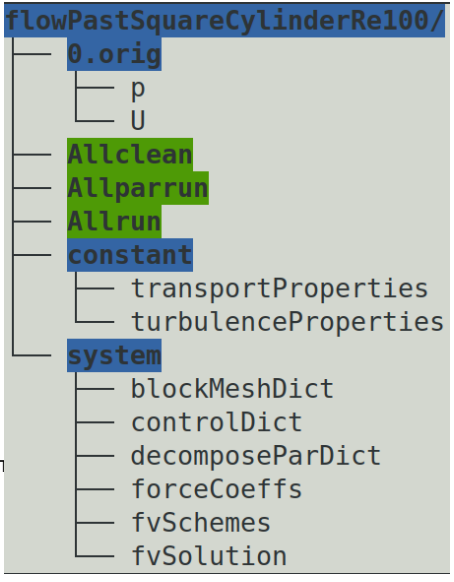
Граница	$\boldsymbol{U}$	p
inlet	Fixed value $\boldsymbol{U} = 1$	Zero gradient
outlet	Zero gradient	Fixed value $p = 0$
top/bottom/front/back	Zero gradient	Zero gradient
cylinder	Fixed value $\boldsymbol{U} = 0$	Zero gradient

Скачиваем кейс

Кейс лежит на GitHub
`https://github.com/RomanovaDI/freeSoftwareTrainingCourse`
Скачать можно по ссылке
`https://github.com/RomanovaDI/freeSoftwareTrainingCourse/archive/refs/heads/main.zip`

Структура кейса

- 0.orig — каталог с начальными условиями
  - p — начальные условия на давление
  - U — начальные условия на скорость
- Allclean — скрипт очистки директории кейса
- Allrun — скрипт запуска расчёта
- Allparrun — скрипт запуска расчёта в параллели
- constant — каталог с константными параметрами
  - transportProperties — реологические свойства
  - turbulenceProperties — параметры турбулентности
- system — каталог с параметрами расчёта задачи
  - blockMeshDict — описание сетки
  - controlDict — параметры управления расчётом
  - decomposeParDict — описание декомпозиции област
  - forceCoeffs — описание расчета сил на цилиндре
  - fvSchemes — схемы аппроксимации
  - fvSolution — параметры решателя



Скрипты выполнения

Allclean

```
0 #!/bin/sh
1 cd "${0%/*}" || exit                                # Run from this directory
2 . ${WM_PROJECT_DIR:?}/bin/tools/CleanFunctions # Tutorial clean functions
3 #------------------------------------
4
5 cleanCase0 # Чистим директорую кейса от старых расчётов
6
7 #------------------------------------
```

Скрипты выполнения

Allrun

```
0  #!/bin/sh
1  cd "${0%/*}" || exit                                # Run from this directory
2  . ${WM_PROJECT_DIR:?}/bin/tools/RunFunctions # Tutorial run functions
3  #------------------------------------
4
5  touch flowPastSquareCylinderRe100.foam # Создаем файл для визуализации
6  runApplication blockMesh               # Создаем сетку
7  restore0dir                            # Создаем нулевой момент
8  runApplication $(getApplication)       # Запускаем расчёт
9
10 #------------------------------------
```

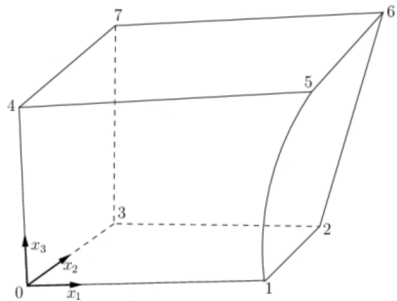
Скрипты выполнения

Allparrun

```
0  #!/bin/sh
1  cd "${0%/*}" || exit                                # Run from this directory
2  . ${WM_PROJECT_DIR:?}/bin/tools/RunFunctions # Tutorial run functions
3  #------------------------------------
4
5  touch flowPastSquareCylinderRe100.foam # Создаем файл для визуализации
6  runApplication blockMesh                  # Создаем сетку
7  restore0Dir                               # Создаем нулевой момент
8  runApplication decomposePar               # Производим декомпозицию расчётной
       области
9  runParallel $(getApplication)             # Запускаем расчёт
10
11 #------------------------------------
```

Сетка. Куб. blockMeshDict

- Система координат должна быть правосторонней.
- Вершины объявляются в соответствии с выбранной системой координат.
- При объявлении блока порядок перечисления вершин блока строго определён и показан на рисунке. Он зависит от системы координат.
- Объявляются внешние грани расчетной области, нормаль должна смотреть наружу (определяется по принципу правой руки).

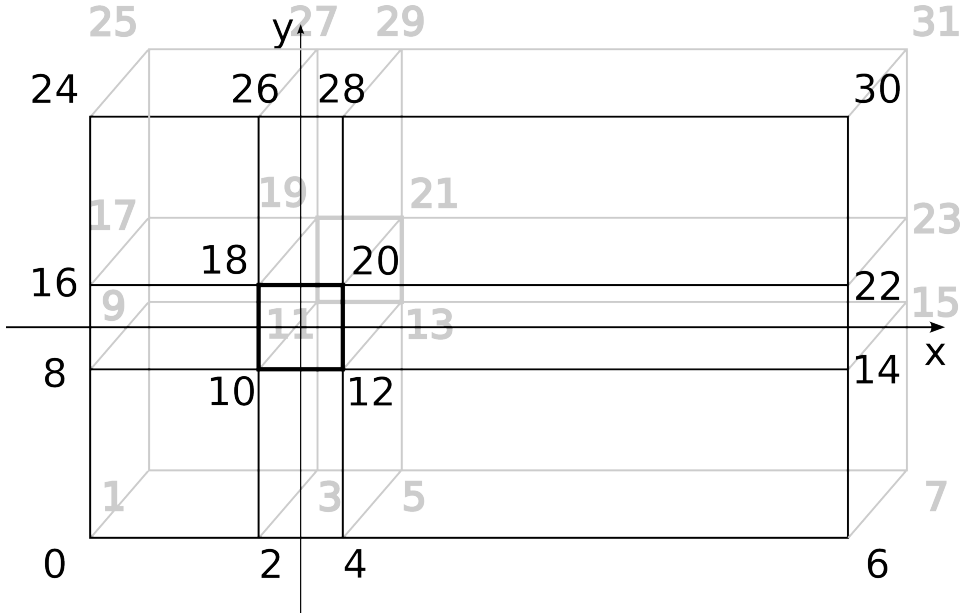


Сетка. Куб. blockMeshDict

```
0  scale   0.1;
1  vertices
2  (
3      (0 0 0)
4      (1 0 0)
5      (1 1 0)
6      (0 1 0)
7      (0 0 0.1)
8      (1 0 0.1)
9      (1 1 0.1)
10     (0 1 0.1)
11 );
12 blocks
13 (
14     hex (0 1 2 3 4 5 6 7) (20 20 1) simpleGrading (1 1 1)
15 );
16 edges
17 ();
```

```
boundary
(
    walls {
        type wall;
        faces
        (
            (4 5 6 7) // top
            (0 3 2 1) // bottom
            (0 4 7 3) // left
            (2 6 5 1) // right
        ); }
    frontAndBack {
        type empty;
        faces
        (
            (1 5 4 0) // front
            (3 7 6 2) // back
        );}
);
```

Сетка. Обтекание квадратного цилиндра. blockMeshDict



Сетка. blockMeshDict

```
0  scale   1.0;
1
2  DHalf   0.5;
3  HHalf   15.5;
4  LLHalf  15.5;
5  LRHalf  65.5;
6  BHalf   0.5;
7  nB      1;
8  nD      5;
9  nHHalf  75;
10 nLLHalf 75;
11 nLRHalf 325;
12
13 vertices
14 (
15     (-$LLHalf -$HHalf  $BHalf) // 0
16     (-$LLHalf -$HHalf -$BHalf) // 1
17     (-$DHalf  -$HHalf  $BHalf) // 2
```

```
18      (-$DHalf  -$HHalf -$BHalf) // 3
19      ( $DHalf  -$HHalf  $BHalf) // 4
20      ( $DHalf  -$HHalf -$BHalf) // 5
21      ( $LRHalf -$HHalf  $BHalf) // 6
22      ( $LRHalf -$HHalf -$BHalf) // 7
23      (-$LLHalf -$DHalf  $BHalf) // 8
24      (-$LLHalf -$DHalf -$BHalf) // 9
25      (-$DHalf  -$DHalf  $BHalf) // 10
26      (-$DHalf  -$DHalf -$BHalf) // 11
27      ( $DHalf  -$DHalf  $BHalf) // 12
28      ( $DHalf  -$DHalf -$BHalf) // 13
29      ( $LRHalf -$DHalf  $BHalf) // 14
30      ( $LRHalf -$DHalf -$BHalf) // 15
31      (-$LLHalf  $DHalf  $BHalf) // 16
32      (-$LLHalf  $DHalf -$BHalf) // 17
33      (-$DHalf   $DHalf  $BHalf) // 18
34      (-$DHalf   $DHalf -$BHalf) // 19
35      ( $DHalf   $DHalf  $BHalf) // 20
36      ( $DHalf   $DHalf -$BHalf) // 21
37      ( $LRHalf  $DHalf  $BHalf) // 22
```

```
    ( $LRHalf   $DHalf -$BHalf) // 23
    (-$LLHalf   $HHalf  $BHalf) // 24
    (-$LLHalf   $HHalf -$BHalf) // 25
    (-$DHalf    $HHalf  $BHalf) // 26
    (-$DHalf    $HHalf -$BHalf) // 27
    ( $DHalf    $HHalf  $BHalf) // 28
    ( $DHalf    $HHalf -$BHalf) // 29
    ( $LRHalf   $HHalf  $BHalf) // 30
    ( $LRHalf   $HHalf -$BHalf) // 31
);

blocks
(
    hex (1 3 11 9 0 2 10 8) ($nLLHalf $nHHalf $nB) simpleGrading (1 1 1)
    hex (3 5 13 11 2 4 12 10) ($nD $nHHalf $nB) simpleGrading (1 1 1)
    hex (5 7 15 13 4 6 14 12) ($nLRHalf $nHHalf $nB) simpleGrading (1 1 1)
    hex (9 11 19 17 8 10 18 16) ($nLLHalf $nD $nB) simpleGrading (1 1 1)
```

```
    hex (13 15 23 21 12 14 22 20) ($nLRHalf $nD $nB) simpleGrading (1
    1 1)
    hex (17 19 27 25 16 18 26 24) ($nLLHalf $nHHalf $nB) simpleGrading
     (1 1 1)
    hex (19 21 29 27 18 20 28 26) ($nD $nHHalf $nB) simpleGrading (1 1
     1)
    hex (21 23 31 29 20 22 30 28) ($nLRHalf $nHHalf $nB) simpleGrading
     (1 1 1)
);

edges
(
);

boundary
(
    inlet
    {
        type patch;
        faces
```

```
        (
            (0 8 9 1)
            (8 16 17 9)
      (16 24 25 17)
        );
    }
    outlet
    {
        type patch;
        faces
        (
            (6 7 15 14)
            (14 15 23 22)
      (22 23 31 30)
        );
    }
  hole
    {
        type wall;
        faces
```

```
91          (
92              (10 11 19 18)
93              (18 19 21 20)
94              (12 20 21 13)
95              (10 12 13 11)
96          );
97      }
98    topAndBottom
99      {
100         type wall;
101         faces
102         (
103             (24 26 27 25)
104             (26 28 29 27)
105             (28 30 31 29)
106             (0 1 3 2)
107       (2 3 5 4)
108       (4 5 7 6)
109         );
110     }
```

```
111 );
112
113 mergePatchPairs
114 (
115 );
116
117
118 // ************************************************************************* //
```

Начальные условия. p

```
0  FoamFile
1  {
2      version     2.0;
3      format      ascii;
4      class       volScalarField;
5      object      p;
6  }
7  dimensions      [0 2 -2 0 0 0 0];
8  internalField   uniform 0;
9  boundaryField
10 {
11     hole
12     {
13         type            zeroGradient;
14     }
15     inlet
16     {
17         type            zeroGradient;
```

```
    }
    outlet
    {
        type            fixedValue;
        value           uniform 0;
    }
    topAndBottom
    {
        type            zeroGradient;
    }
    defaultFaces
    {
        type            empty;
    }
}
```

Начальные условия. U

```
0  FoamFile
1  {
2      version     2.0;
3      format      ascii;
4      class       volVectorField;
5      object      U;
6  }
7  dimensions      [0 1 -1 0 0 0 0];
8  internalField   uniform (1 0 0);
9  boundaryField
10 {
11     hole
12     {
13         type            fixedValue;
14         value           uniform (0 0 0);
15     }
16     inlet
17     {
```

```
        type            fixedValue;
        value           uniform (1 0 0);
    }
    outlet
    {
        type            zeroGradient;
    }
    topAndBottom
    {
        type            zeroGradient;
    }
    defaultFaces
    {
        type            empty;
    }
}
```

Константные параметры. transportProperties

```
0  FoamFile
1  {
2      version     2.0;
3      format      ascii;
4      class       dictionary;
5      location    "constant";
6      object      transportProperties;
7  }
8  // * * * * * * * * * * * * * * * * * * //
9
10 transportModel  Newtonian;
11
12 nu              0.01;
13
14 // ************************************ //
```

Константные параметры. turbulenceProperties

```
0  FoamFile
1  {
2      version     2.0;
3      format      ascii;
4      class       dictionary;
5      location    "constant";
6      object      turbulenceProperties;
7  }
8  // * * * * * * * * * * * * * * * * * * * //
9
10 simulationType laminar;
11
12 // ********************************** //
```

Декомпозиция расчётной области. decomposeParDict

```
0  FoamFile
1  {
2      version     2.0;
3      format      ascii;
4      class       dictionary;
5      object      decomposeParDict;
6  }
7  // * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
8
9  numberOfSubdomains 2;
10 method          simple;
11 coeffs
12 {
13     n           (2 1 1);
14 }
15
16 // ************************************************* //
```

Управление расчетом. controlDict

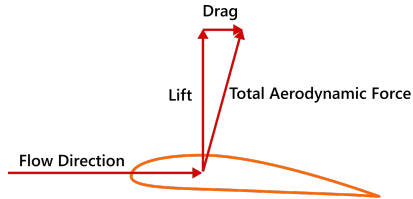
```
0  application     pimpleFoam;
1  startFrom       latestTime;
2  startTime       0;
3  stopAt          endTime;
4  endTime         10;
5  deltaT          0.004;
6  writeControl    timeStep;
7  writeInterval   25;
8  writeFormat     binary;
9  writePrecision  8;
10 writeCompression off;
11 timeFormat      general;
12 timePrecision   6;
13 runTimeModifiable true;
14
15
16
17
```

```
functions
{
    forces
    {
        type            forces;
        libs            (forces);
        writeControl    writeTime;

        patches         (hole);
        // Centre of rotation for moment calculations
        CofR            (0 0 0);
        writeFields     yes;

        rho             rhoInf;
        rhoInf          1.0;
    }

    #include "forceCoeffs"
}
```

Аэродинамическая сила.

Полная аэродинамическая сила делится на подъёмную силу и силу лобового сопротивления:

$$\boldsymbol{F} = \boldsymbol{F_L} + \boldsymbol{F_D}.$$



Расчёт полной аэродинамической силы происходит через интеграл по поверхности тела от давления и вязких напряжений на поверхности тела:

$$\boldsymbol{F} = \boldsymbol{F_p} + \boldsymbol{F_v},$$

$$\boldsymbol{F_p} = \oint_S \rho(p - p_{ref})dS, \quad \boldsymbol{F_v} = \oint_S \boldsymbol{\tau}^{(d)} \cdot dS,$$

где p — давление, p_{ref} — давление на бесконечности, $\boldsymbol{\tau}^{(d)}$ — девиатор тензора вязких напряжений.

Коэффициент аэродинамического сопротивления и коэффициент подъёмной силы.

Коэффициент подъёмной силы — безразмерная величина, характеризующая подъёмную силу крыла определённого профиля при известном угле атаки.

$$C_L = \frac{F_L}{0.5U^2S},$$

где S — характерная площадь, чаще всего площадь миделевого сечения (наибольшее по площади поперечное сечение тела, движущегося в воде или воздухе), U — скорость набегающего потока в случае, если используется связанная система координат.

Коэффициент аэродинамического сопротивления — коэффициент, характеризующий способность тела преодолевать аэродинамическое сопротивление воздуха.

$$C_D = \frac{F_D}{0.5U^2S}.$$

C_D и C_L. forceCoeffs

```
0  forceCoeffs1
1  {
2      type            forceCoeffs;
3      libs            (forces);
4      writeControl    timeStep;
5      timeInterval    1;
6      log             yes;
7      patches         (hole);
8      rho             rhoInf;       // Indicates incompressible
9      rhoInf          1.0;            // Required when rho = rhoInf
10     liftDir         (0 1 0);
11     dragDir         (1 0 0);
12     CofR            (0 0 0);   // Axle midpoint on ground
13     pitchAxis       (0 0 1);
14     magUInf         1;
15     lRef            1;         // Wheelbase length
16     Aref            1;         // Estimated
17 }
```

Задачи

1. Посчитать кейс flowPastSquareCylinder и произвести визуализацию
2. Посчитать кейс в параллельном режиме, изучить зависимость времени счета от количества процессоров (время расчёта есть в файле `log.pimpleFoam`)
3. Изучить зависимость аэродинамических сил и коэффициентов (лежат в папке `postProcess`) от формы цилиндра (вытянутой вдоль или поперёк потока). При изменении формы сохранить квадратную форму ячеек.
4. Посчитать 150 секунд и посмотреть как изменится картина течения

СПАСИБО ЗА ВНИМАНИЕ!

Список литературы