pyFluent | 设置Fluent计算参数

原创 流沙CAE CFD之道 2022-07-12 08:31 发表于四川

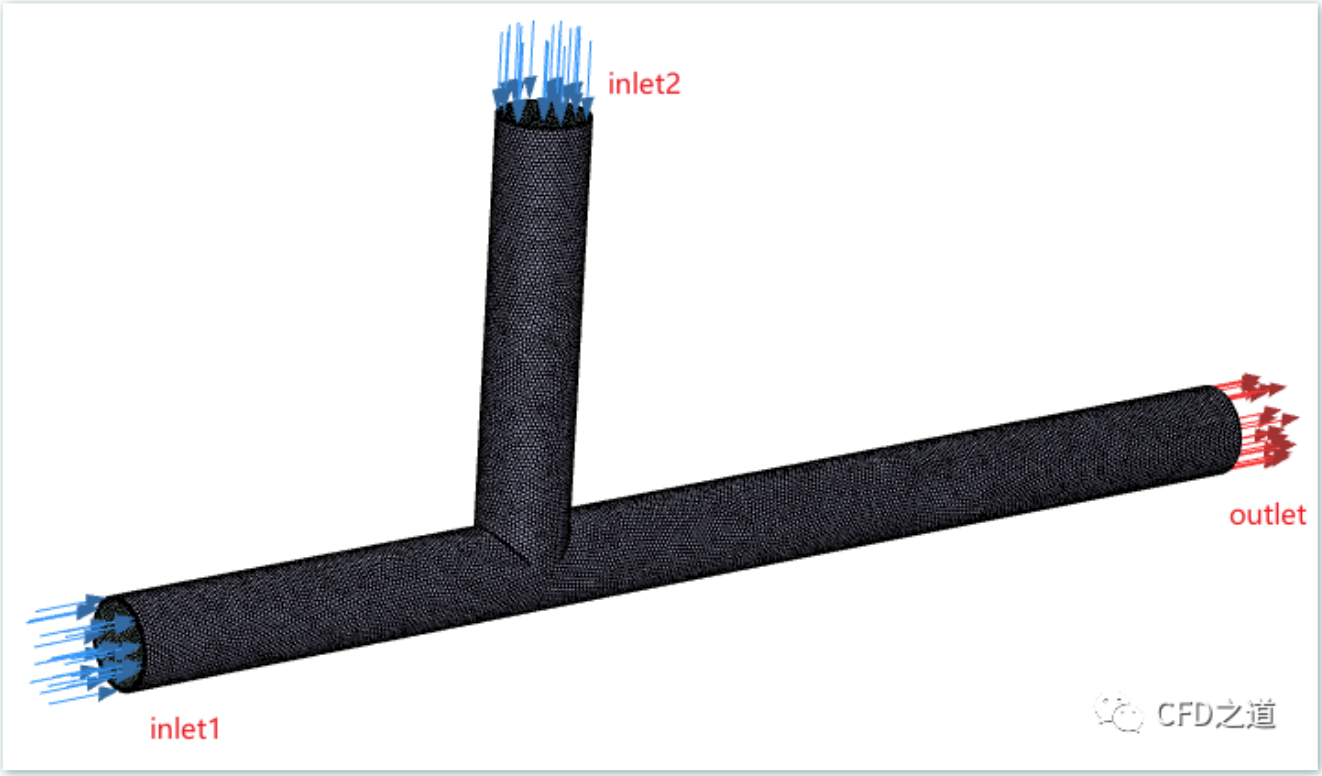
收录于合集
#pyFluent

4个

前文演示了利用pyFluent进行watertight Geometry网格生成（pyFluent | watertight网格生成）。本文接上文，采用pyFluent进行模型前处理并执行计算。

1 问题描述

计算模型与网格如下图所示。



模型中包含2个入口：inlet1及inlet2，包括1个出口：outlet。

边界条件为：

- inlet1：速度入口，速度1 m/s，温度300 K
- inlet2：速度入口，速度2 m/s，温度340 K
- outlet：压力出口，静压0 Pa

其他边界为绝热壁面。

2 pyFluent程序

2.1 读取网格并检查

利用下面的代码段启动Fluent、读取网格并对网格进行检查。

```
import ansys.fluent.core as pyFluent
session = pyFluent.launch_fluent(precision='double',processor_count = 24)
msh_filename = 'teepipe.msh.h5'
session.solver.root.file.read(file_type="case",file_name = msh_filename)
```

部分输出如下图所示。

```
Reading from XTZJ-20220704JV:"d:\TeePipe\teepipe.msh.h5" in NODE0 mode ...
  Reading mesh ...
     178374 cells,     1 cell zone  ...
        178374 polyhedra cells,  zone id: 98
     966694 faces,     5 face zones ...
        937582 polygonal interior faces,  zone id: 97
         27456 polygonal wall faces,  zone id: 20
           552 polygonal pressure-outlet faces,  zone id: 19
           552 polygonal velocity-inlet faces,  zone id: 18
           552 polygonal velocity-inlet faces,  zone id: 17
     738297 nodes,     4 node zones ...

Building...
     mesh
        distributing mesh
                parts.....................,
                faces.....................,
                nodes.....................,
...
        fluid
     parallel,
Done.
Mesh is now scaled to meters.

False
```


利用下面的代码进行网格检查。

```
session.solver.tui.mesh.check()
```

程序输出如下图所示。

```
  Domain Extents:
    x-coordinate: min (m) = -9.997412e-03, max (m) = 1.000000e-01
    y-coordinate: min (m) = 0.000000e+00, max (m) = 2.900000e-01
    z-coordinate: min (m) = -9.996252e-03, max (m) = 9.997924e-03
  Volume statistics:
    minimum volume (m3): 3.776785e-11
    maximum volume (m3): 2.864450e-09
      total volume (m3): 1.195508e-04
  Face area statistics:
    minimum face area (m2): 3.148268e-09
    maximum face area (m2): 1.980027e-06
  Checking mesh.....................................
Done.

Note: Settings to improve the robustness of pathline and
      particle tracking have been automatically enabled.
```


2.2 激活能量方程

利用下面的代码激活能量方程。

```
session.solver.root.setup.models.energy.enabled = True
```

2.3 创建材料

采用液态水作为案例中使用的材料介质。可以从材料库中复制材料。

```
session.solver.root.setup.materials.copy_database_material_by_name(
    type='fluid',name='water-liquid'
)
```

2.4 设置边界条件

案例中有三个边界需要设置。2个速度入口边界需要指定入口速度及湍流物理量，出口边界采用默认设置。

```
session.solver.root.setup.boundary_conditions.velocity_inlet['inlet1'].vmag={
    "option": "constant or expression",
    "constant": 1,
}
```

```
session.solver.root.setup.boundary_conditions.velocity_inlet['inlet1'].ke_spec= 'Intensity and Hy
session.solver.root.setup.boundary_conditions.velocity_inlet['inlet1'].turb_intensity = 5
session.solver.root.setup.boundary_conditions.velocity_inlet['inlet1'].turb_hydraulic_diam = '0.0
session.solver.root.setup.boundary_conditions.velocity_inlet['inlet1'].t={
    'option':'constant or expression',
    'constant':300,
}


session.solver.root.setup.boundary_conditions.velocity_inlet['inlet2'].vmag={
    "option": "constant or expression",
    "constant": 2,
}
session.solver.root.setup.boundary_conditions.velocity_inlet['inlet2'].ke_spec= 'Intensity and Hy
session.solver.root.setup.boundary_conditions.velocity_inlet['inlet2'].turb_intensity = 5
session.solver.root.setup.boundary_conditions.velocity_inlet['inlet2'].turb_hydraulic_diam = '0.0
session.solver.root.setup.boundary_conditions.velocity_inlet['inlet2'].t={
    'option':'constant or expression',
    'constant':340,
}
```

2.5 初始化并计算

利用下面的代码进行初始化及计算。

```
session.solver.root.solution.initialization.hybrid_initialize()
session.solver.tui.solve.monitors.residual.plot("no")

session.solver.root.solution.run_calculation.iterate.get_attr('arguments')
session.solver.root.solution.run_calculation.iterate(number_of_iterations=150)
```

如下图所示。

```
    30  2.7893e-02  5.4804e-06  1.5290e-05  2.8399e-06  2.0182e-05  5.7229e-03  6.1087e-03  0:00:33  120
    31  2.6023e-02  5.0949e-06  1.4130e-05  2.6934e-06  1.8738e-05  6.3095e-03  1.0401e-02  0:00:26  119
    32  2.4443e-02  4.7803e-06  1.3087e-05  2.5765e-06  1.7319e-05  5.7175e-03  5.9707e-03  0:00:21  118
    33  2.3092e-02  4.4564e-06  1.2104e-05  2.4857e-06  1.5916e-05  6.3045e-03  1.0274e-02  0:00:40  117

  iter  continuity  x-velocity  y-velocity  z-velocity      energy           k       omega     time/iter
    34  2.1894e-02  4.1929e-06  1.1211e-05  2.4029e-06  1.4627e-05  5.7133e-03  5.8605e-03  0:00:32  116
    35  2.0787e-02  3.9156e-06  1.0365e-05  2.3362e-06  1.3444e-05  6.3006e-03  1.0176e-02  0:00:25  115
    36  1.9686e-02  3.6848e-06  9.5925e-06  2.2648e-06  1.2377e-05  5.7100e-03  5.7750e-03  0:00:20  114
    37  1.8645e-02  3.4354e-06  8.8710e-06  2.2053e-06  1.1372e-05  6.2973e-03  1.0097e-02  0:00:38  113
...
```

```
148  2.1508e-04  4.8462e-07  1.8294e-07  1.6539e-07  5.7682e-09  5.6883e-03  5.3403e-03  0:00:00  2
149  2.1354e-04  4.8036e-07  1.9064e-07  1.7371e-07  5.3069e-09  6.2768e-03  9.6991e-03  0:00:00  1
150  2.0829e-04  4.8393e-07  1.8009e-07  1.6431e-07  5.0859e-09  5.6883e-03  5.3402e-03  0:00:00  0
```

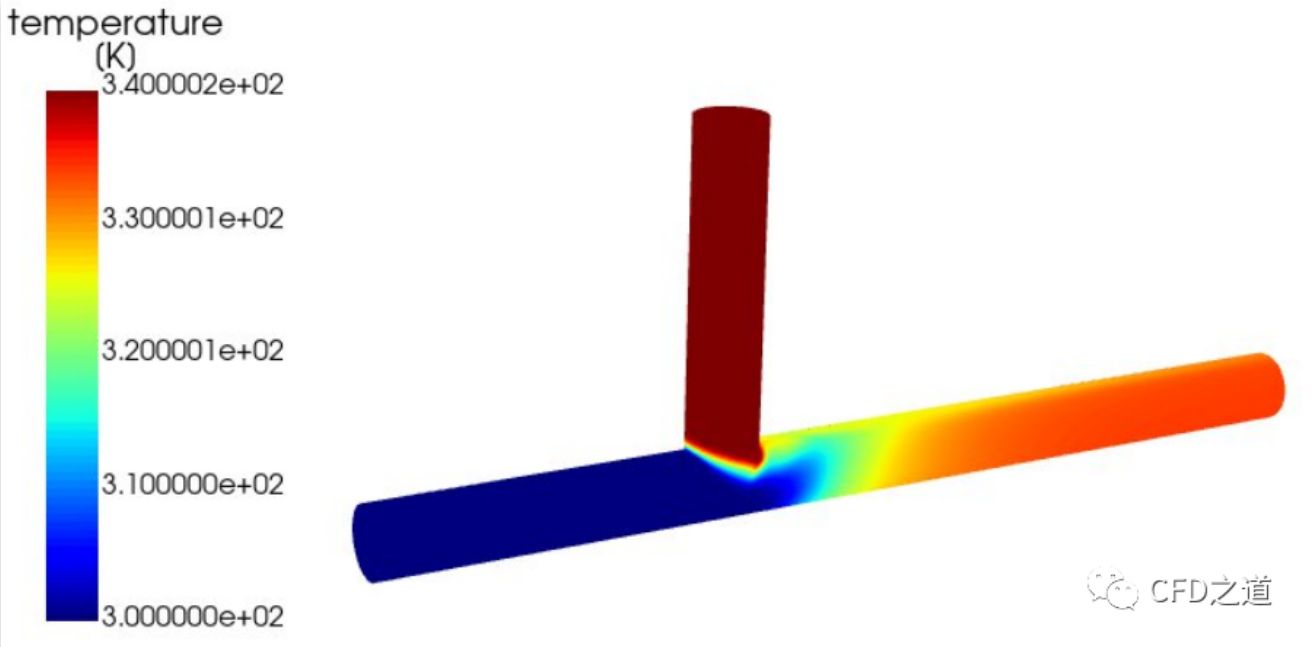

2.6 后处理

后处理可以使用 **ansys-fluent-visualization** 模块。

如下面的代码可以显示walls壁面上的温度分布。

```
from ansys.fluent.visualization.pyvista import Graphics
graphics = Graphics(session=session)
temperature_contour = graphics.Contours['contour-temperature']
temperature_contour.field = 'temperature'
temperature_contour.surfaces_list = ['walls']
temperature_contour.display("window-1")
```

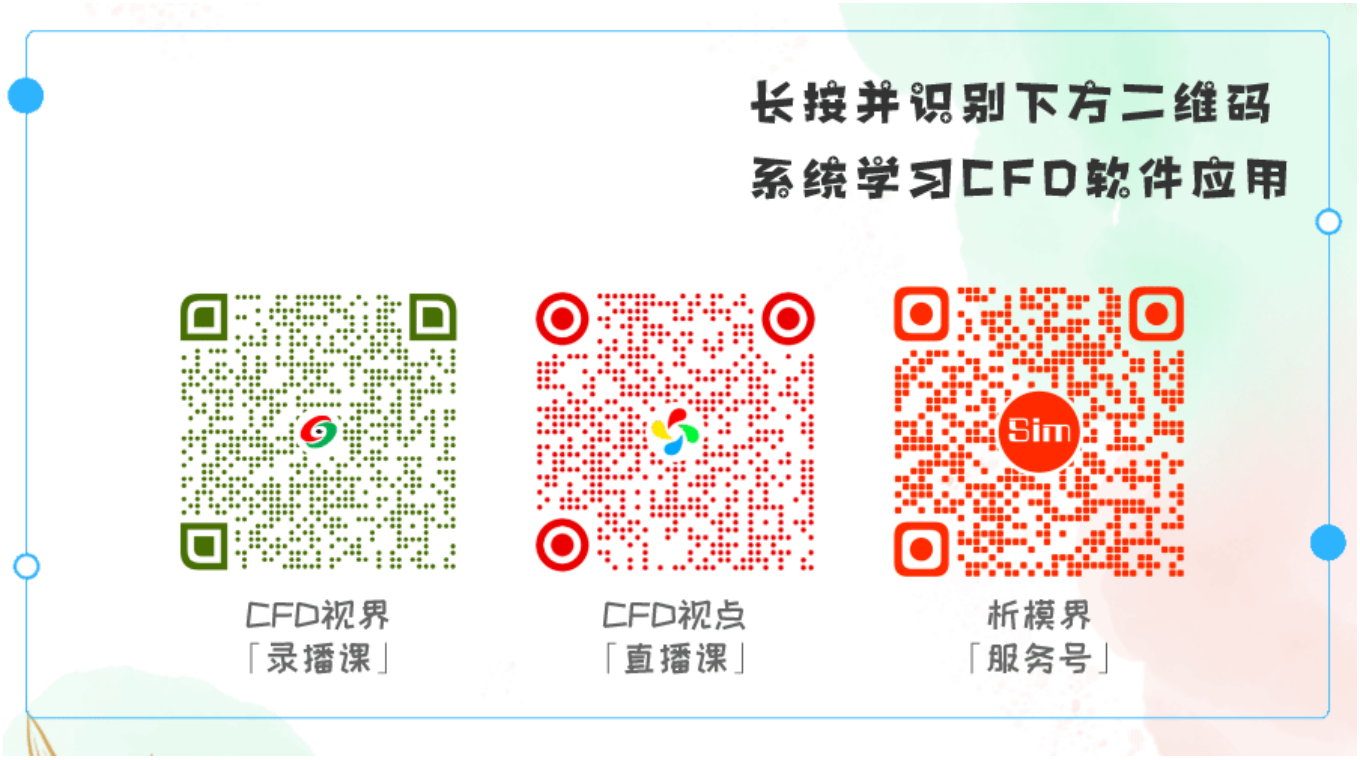
运行结果如下图所示。



可以使用下面的代码在当前目录下保存case与data文件。

```
session.solver.tui.file.write_case_data()
```

（完毕）



收录于合集 #pyFluent 4

上一篇
pyFluent | 一点使用体验

下一篇
pyFluent | watertight网格生成

喜欢此内容的人还喜欢

从read开始分析系统调用的上下文切换
CodeTrap

VLOOKUP函数搭配OFFSET和MATCH函数近似匹配，经典函数嵌套案例！
老徐的Excel

如何在PowerQuery中实现DAX中的IN运算？
PowerBI星球