pyFluent | 常用的代码片段

胡坤 CFD之道 2022-07-14 08:38 发表于四川

收录于合集
#pyFluent

4个

下面列举了PyFluent常用的一些代码片段。虽然PyFluent代码可以很容易地通过TUI命令转化，不过熟悉一些使用频繁的代码也是有用的。

下面的内容取自PyFluent的文档。

1 启动Fluent

启动Fluent可以通过 `launch_fluent()` 函数来实现。该函数可以包含众多的参数。

```
ansys.fluent.core.launcher.launcher.launch_fluent(version=None, precision=None, processor_count=N
```

该函数返回一个 `Session` 对象。

函数参数包括：

- `version` ：可选参数为 `'2d'` 或 `'3d'` ，不设置则默认为 `'3d'`
- `precision` ：可选参数为 `'single'`或`'double'` ，不设置则默认为 `'double'`
- `processor_count` ：指定处理器数量，默认值为 `1`
- `journal_filename` ：指定journal文件的路径
- `meshing_mode` ：可选参数为 `True`或`False` ，指定为 `True` 表示启动Fluent Meshing
- `start_timeout` ：指定连接Fluent的最大时间，默认为100 s
- `additional_arguments` ：指定启动Fluent时可以添加的额外参数，参数类型为字符串形式
- `env` ：在Fluent中修改环境变量的映射，参数为字典形式
- `start_instance` ：此参数如果为False，则连接到ip和端口上的现有Fluent实例。否则启动Fluent的本地实例。此参数默认为True，也可以由环境变量设置
- `ip` ：连接到现有Fluent实例的IP地址。仅当start_instance为False时使用。默认值为 `'127.0.0.1'` ，也可以由环境变量设置
- `port` ：连接到现有Fluent实例的端口。仅当start_instance为False时使用。默认值可以由环境变量 `PYFLUENT_FLUENT_PORT=<PORT>` 设置。

- **cleanup_on_exit** ：如果为True，则当PyFluent退出或在会话实例上调用 **exit()** 函数时，连接的Fluent会话将关闭，默认情况下为True。
- **start_transcript** ：当参数指定为True时，客户端中会启动Fluent transcript。它可以通过对会话对象的方法调用来启动和停止。
- **show_gui** ：当此参数为True时，且START_INSTANCE也为True时，会显示Fluent的图形用户界面，这也可以通过环境变量 **PYFLUENT_SHOW_SERVER_GUI=0或1** 来设置。show-gui参数的作用是覆盖 **PYFLUENT_SHOW_SERVER_GUI** 变量。例如当PYFLUENT_SHOW_SERVER_GUI设置为1，若show_gui设置为False，则隐藏图形用户界面。默认设置为None，以便可以检测到显式False设置。

在使用时，可以使用下面的代码片段：

```
import ansys.fluent.core as pyfluent
session = pyfluent.launch_fluent(meshing_mode = False,version='3d',precision='double')
```

可以以3d、double precision方式启动Fluent Solution模式。其他参数按上面的参数说明进行添加。

2 导入及操纵网格

利用下面的diamante启动Fluent：

```
import ansys.fluent.core as pyFluent
session = pyFluent.launch_fluent(precision='double',processor_count = 24)
```

假设网格文件为teepipe.msh.h5。

- 网格导入有两种方式，这两种方式是等效的

```
session.solver.tui.file.read_case(case_file_name = 'teepipe.msh.h5')
session.solver.root.file.read(file_type='case',file_name = 'teepipe.msh.h5')
```

- 检查网格可以使用下面的代码：

```
session.solver.tui.mesh.check()  # 检查网格信息
session.solver.tui.mesh.quality() # 输出网格质量信息
```

- 缩放网格

```
session.solver.tui.mesh.scale(1,1,1)  # 指定xyz三方向的缩放因子对网格进行缩放
```

- 指定单位

```
session.solver.tui.define.units('length','in') # 将长度单位设置为in
```

3 General设置

- 读取msh后，可以通过下面的代码设置稳态模拟或瞬态模拟

```
session.solver.tui.define.models.steady('yes') # 设置稳态计算
session.solver.tui.define.models.unsteady_1st_order('yes') # 采用时间1阶瞬态计算
session.solver.tui.define.models.unsteady_2nd_order('yes') # 采用时间2阶瞬态计算
# 还有其他瞬态格式
```

- 利用下面的代码选择使用压力基或密度基求解器

```
session.solver.tui.define.models.solver.density_based_explicit('yes') # 密度基显式
session.solver.tui.define.models.solver.density_based_implicit('yes') # 密度基隐式
session.solver.tui.define.models.solver.pressure_based('yes') # 压力基求解器
```

- 可以设置重力加速度

```
session.solver.tui.define.operating_conditions.gravity('yes','0','-9.81','0')
```

3 Models定义

- 下面两种方式都可以激活能量方程

```
session.solver.tui.define.models.energy('yes', 'no', 'no', 'no', 'yes')
session.solver.root.setup.models.energy.enabled = True
```

- 激活湍流模型

```
session.solver.tui.define.models.viscous.laminar('yes')  # 采用层流模型
```

```
session.solver.tui.define.models.viscous.kw_sst('yes')   # 使用SST K-Omega模型
session.solver.tui.define.models.viscous.ke_standard('yes') # 使用标准k-epsilon模型
# 相同方式启用其它湍流模型
# 湍流模型也可以使用下面的形式启用
session.solver.root.setup.models.viscous.k_epsilon_model.enabled = True
session.solver.root.setup.models.viscous.k_omega_model.enabled = True
```

- 激活辐射模型

```
session.solver.tui.define.models.radiation.s2s('yes') # 激活使用s2s模型
session.solver.tui.define.models.radiation.p1('yes')    # 激活使用p1模型
session.solver.tui.define.models.radiation.discrete_ordinates('yes') # 使用DO模型

session.solver.root.setup.models.radiation.discrete_ordinates = True# 另一种方式激活do模型
```

- 激活多相流模型

```
session.solver.tui.define.models.multiphase.model('vof')
session.solver.tui.define.models.multiphase.model('eulerian')
session.solver.tui.define.models.multiphase.model('mixture')
session.solver.tui.define.models.multiphase.model('wetsteam')
```

4 定义材料

两种方式指定材料。

- 使用TUI方式进行指定

```
session.solver.tui.define.materials.copy('fluid', 'water-liquid')
session.solver.tui.define.boundary_conditions.fluid(
    'elbow-fluid',
    'yes',
    'water-liquid',
    'no',
    'no',
    'no',
    'no',
    '0',
```

```
    'no',
    '0',
    'no',
    '0',
    'no',
    '0',
    'no',
    '0',
    'no',
    '1',
    'no',
    'no',
    'no',
    'no',
    'no',
)
```

- 利用对象赋值的方式

```
session.solver.root.setup.materials.copy_database_material_by_name(type='fluid', name='water-liqu
session.solver.root.setup.cell_zone_conditions.fluid['elbow-fluid'].material = 'water-liquid'
```

5 指定边界条件

边界条件的指定也可以使用TUI或设置对象两种方式来实现。

- 使用TUI方式指定边界条件

```
session.solver.tui.define.boundary_conditions.set.velocity_inlet(
    'cold-inlet',
    [],
    'vmag',
    'no',
    0.4,
    'quit'
)
session.solver.tui.define.boundary_conditions.set.velocity_inlet(
    'cold-inlet',
    [],
```

```
    'ke-spec',
    'no',
    'no',
    'no',
    'yes',
    'quit'
)
session.solver.tui.define.boundary_conditions.set.velocity_inlet(
    'cold-inlet',
    [],
    'turb-intensity',
    5,
    'quit'
)
session.solver.tui.define.boundary_conditions.set.velocity_inlet(
    'cold-inlet',
    [],
    'turb-hydraulic-diam',
    4,
    'quit'
)
session.solver.tui.define.boundary_conditions.set.velocity_inlet(
    'cold-inlet',
    [],
    'temperature',
    'no',
    293.15,
    'quit'
)
```

- 复制边界条件

```
session.solver.tui.define.boundary_conditions.copy_bc('cold-inlet','hot-inlet','()')
```

- 列出所有区域

```
session.solver.tui.define.boundary_conditions.list_zones()
```

- 采用设置对象的方式指定边界条件

```
session.solver.root.setup.boundary_conditions.velocity_inlet['cold-inlet'].vmag = {
    'option': 'constant or expression',
    'constant': 0.4,
}
session.solver.root.setup.boundary_conditions.velocity_inlet[
    'cold-inlet'
].ke_spec = 'Intensity and Hydraulic Diameter'
session.solver.root.setup.boundary_conditions.velocity_inlet[
    'cold-inlet'
].turb_intensity = 5
session.solver.root.setup.boundary_conditions.velocity_inlet[
    'cold-inlet'
].turb_hydraulic_diam = '4 [in]'
session.solver.root.setup.boundary_conditions.velocity_inlet['cold-inlet'].t = {
    'option': 'constant or expression',
    'constant': 293.15,
}
```

6 求解控制参数

- 选择Methods

```
session.solver.tui.solve.set.p_v_coupling(24) # 使用coupled算法
session.solver.tui.solve.set.gradient_scheme('yes')    # 使用Green-Gauss Node Based
session.solver.tui.solve.set.gradient_scheme('no','yes') # 使用Least Squares Cell Base
```

- 设置求解控制参数

```
session.solver.tui.solve.set.p_v_controls(0.3,0.4) # 指定Momentum 及 Pressure方程的亚松弛因子
```

- 创建Report

```
session.solver.tui.solve.report_definitions.add(
    'outlet-temp-avg',
    'surface-massavg',
    'field',
    'temperature',
    'surface-names',
    'outlet',
```

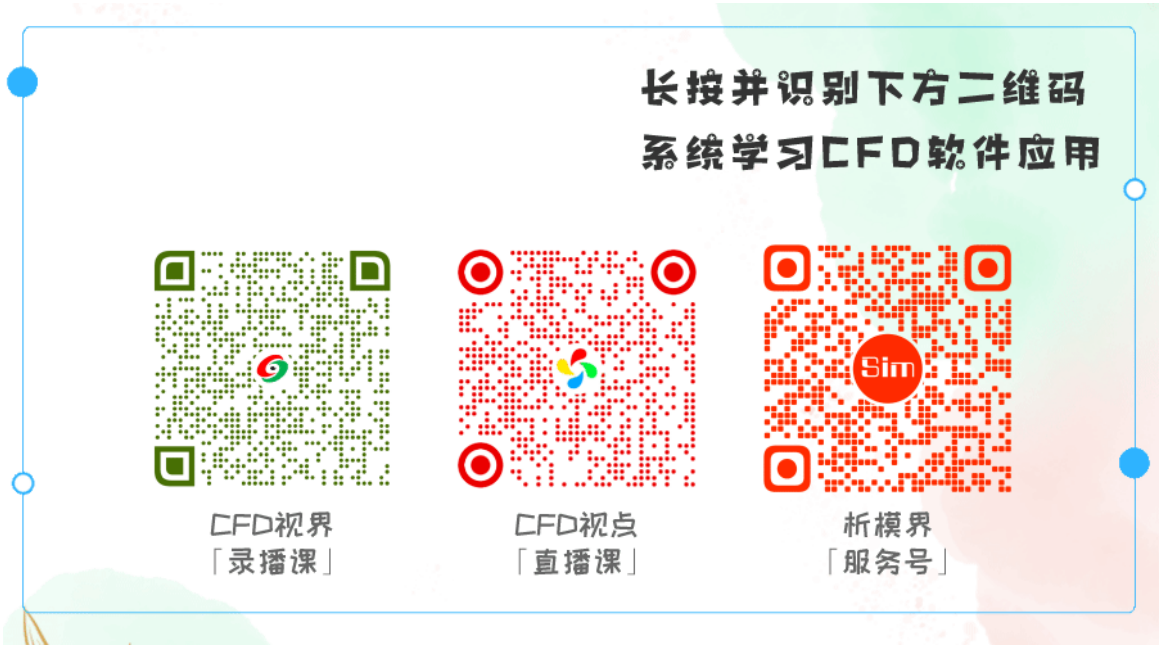
```
    '()',
    'quit',
)
```

- 初始化及求解

```
session.solver.tui.solve.initialize.hyb_initialization()
session.solver.tui.solve.iterate(100)

# 也可以使用对象设置方式
session.solver.root.solution.initialization.hybrid_initialize()
session.solver.root.solution.run_calculation.iterate(number_of_iterations=150)
```

（完毕）



收录于合集 #pyFluent 4

下一篇 · pyFluent | 一点使用体验

喜欢此内容的人还喜欢

一行 Python 代码实现并行

yangyidba

Linux Shell 脚本入门到实战详解

杰哥的IT之旅

几行代码就能实现复杂的 Excel 导入导出，这个工具类真心强大！

马士兵