知乎 写文章 ···

基于icoFoam求解器学习OpenFOAM中的PISO算法

阿牛
有动力、不机械
关注他

9 人赞同了该文章

最近学习OpenFOAM的程序设计，发现在很多求解器中采用PISO来进行NS方程的求解。花了些时间，基于icoFoam求解器研究了OpenFOAM中PISO的实现。在知乎上记录自己的学习过程，希望能跟大家多交流。

1. 数学模型

首先，讨论如下形式的连续性方程和动量方程

$$\nabla \cdot \vec{u} = 0$$

$$\frac{\partial \vec{u}}{\partial t} + \frac{\partial \vec{u}\vec{u}}{\partial x} - \nabla \cdot (\nu \nabla \cdot \vec{u}) = -\nabla p$$

方程组（1）
定义求解的问题

1）求解方程推导

PISO算法将上述方程离散并求解出流场中的速度和压力

/***\
Jasak, H. Error Analysis and Estimation for the Finite Volume Method with Applications to Fluid Flows, Ph.D Thesis, Imperial College of Science, Technology and Medicine, London, 1996.
***/

根据Prof. Jasak的论文，求解连续性方程和动量方程的PISO算法，可以构造一个矩阵M，由A、B、C、D四个矩阵构成，分别为$p \times p, p \times q, q \times p, q \times q$矩阵、其中$A$是可逆的

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

该矩阵对应如下方程组

$$Ax + By = a$$

$$Cx + Dy = b$$

上面讨论的方程组(1)即可写为如下形式

$$\begin{bmatrix} [A_u] & [\nabla(.)] \\ [\nabla \cdot (.)] & [0] \end{bmatrix} \begin{bmatrix} \vec{u} \\ p \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$[A_u]$是描述动量方程的矩阵，为了便于求解，进行如下转换

$$[A_u] = [D_u] + [LU_u]$$

上述矩阵则可以变换为如下形式

$$\begin{bmatrix} [D_u] & [\nabla(.)] \\ [\nabla \cdot (.)] & [0] \end{bmatrix} \begin{bmatrix} \vec{u} \\ p \end{bmatrix} = \begin{bmatrix} -[LU_u][\vec{u}] \\ 0 \end{bmatrix}$$

下面，考虑动量方程的离散形式如下

$$a_P^{\vec{u}} \vec{u}_P + \sum_N a_N^{\vec{u}} \vec{u}_N = \vec{r} - \nabla p$$

简化起见，OpenFOAM引入算子$H(\vec{u})$

$$H(\vec{u}) = \vec{r} - \sum_N a_N^{\vec{u}} \vec{u}_N$$

这样就有

$$a_P^{\vec{u}} \vec{u}_P = H(\vec{u}) - \nabla p$$

即

$$\vec{u}_P = \left(a_P^{\vec{u}}\right)^{-1} (H(\vec{u}) - \nabla p)$$

方程（2） 速度修正方程

将上式代入连续性方程$\nabla\cdot\vec{u}=0$可得

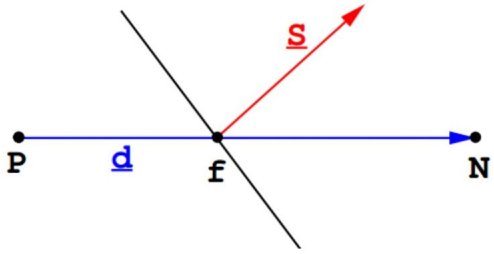
$$\nabla\cdot\left[\left(a_P^{\vec{u}}\right)^{-1}\nabla p\right]=\nabla\cdot\left[\left(a_P^{\vec{u}}\right)^{-1}H(\vec{u})\right]$$

方程（3） 压力方程

2) 非正交校正和连续性方程校正

在扩散项的数值计算中，如下图所示的网格面向量与ø梯度非正交，需要进行非正交校正

$$\int_{V_P}\nabla\cdot(\rho\Gamma_\phi\nabla\phi)\,dV=\sum_f\vec{S}\cdot(\rho\Gamma_\phi\nabla\phi)_f=\sum_f(\rho\Gamma_\phi)_f\vec{S}\cdot(\nabla\phi)_f$$



对于正交网格

$$\vec{S}\cdot(\nabla\phi)_f=|\vec{S}|\frac{\phi_N-\phi_P}{|\vec{d}|}$$

对于非正交网格存在

$$\mathbf{S}\cdot\left(\nabla\phi\right)_f=\underbrace{\mathbf{\Delta}\cdot\left(\nabla\phi\right)_f}_{\text{orthogonal contribution}}+\underbrace{\mathbf{k}\cdot\left(\nabla\phi\right)_f}_{\text{non-orthogonal correction}}$$

$$\vec{S}\cdot(\nabla\phi)_f=|\vec{\Delta}|\frac{\phi_N-\phi_P}{|\vec{d}|}+\vec{k}\cdot(\nabla\phi)_f$$

$\vec{\Delta}$ 和 $\vec{k}$有几种不同的求解方法，在这里不作详述

在PISO算法中，根据连续方程来校验计算误差

$$\nabla\cdot\vec{u}=\sum_f\vec{s}_f\cdot\vec{u}=\sum_f F$$

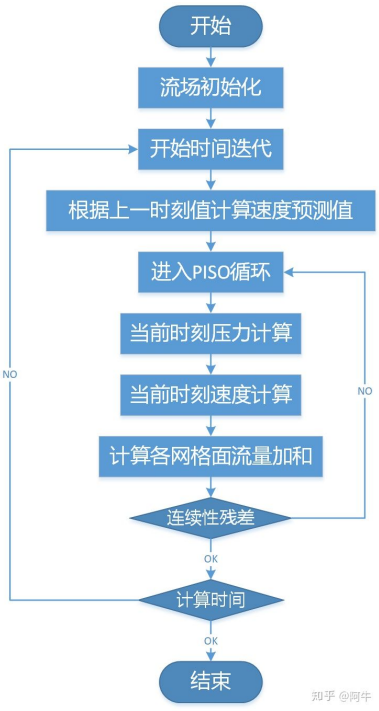
$$F=\vec{s}_f\cdot\vec{u}$$

将方程2代入则有

$$F=-\left(a_P^{\vec{u}}\right)^{-1}\vec{s}_f\cdot\nabla p+\left(a_P^{\vec{u}}\right)^{-1}\vec{s}_f\cdot H(\vec{u})$$

2. 求解流程

OpenFOAM的PISO基本求解流程描述如下



3. 代码分析

```
#include "fvCFD.H"

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

int main(int argc, char *argv[])
{
    #include "setRootCase.H"

    #include "createTime.H"
    #include "createMesh.H"
    #include "createFields.H"
    #include "initContinuityErrs.H"

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

    Info<< "\nStarting time loop\n" << endl;

    while (runTime.loop())
    {
        Info<< "Time = " << runTime.timeName() << nl << endl;

        #include "readPISOControls.H"

        #include "CourantNo.H"

        fvVectorMatrix Ueqn
        (
            fvm::ddt(U)
          + fvm::div(phi, U)
          - fvm::laplacian(nu, U)
        );

        solve(UEqn == -fvc::grad(p));

        // --- PISO loop

        for (int corr=0; corr<nCorr; corr++)
        {
            volScalarField rAU(1.0/UEqn.A());

            volVectorField HbyA("HbyA", U);
            HbyA = rAU*UEqn.H();
            surfaceScalarField phiHbyA
            (
                "phiHbyA",
                (fvc::interpolate(HbyA) & mesh.Sf())
              + fvc::interpolate(rAU)*fvc::ddtCorr(U, phi)
            );

            adjustPhi(phiHbyA, U, p);

            for (int nonOrth=0; nonOrth<=nNonOrthCorr; nonOrth++)
            {
                fvScalarMatrix pEqn
                (
                    fvm::laplacian(rAU, p) == fvc::div(phiHbyA)
                );

                pEqn.setReference(pRefCell, pRefValue);
                pEqn.solve();

                if (nonOrth == nNonOrthCorr)
                {
                    phi = phiHbyA - pEqn.flux();
                }
            }

            #include "continuityErrs.H"

            U = HbyA - rAU*fvc::grad(p);
            U.correctBoundaryConditions();
        }

        runTime.write();

        Info<< "ExecutionTime = " << runTime.elapsedCpuTime() << " s"
            << "  ClockTime = " << runTime.elapsedClockTime() << " s"
            << nl << endl;
    }

    Info<< "End\n" << endl;

    return 0;
}
```

初始化：设置Case目录、初始化时间和网格
createFields.H 自定义场和参考压力
initContinuityErrs.H 初始化计算误差

初始化PISO

计算Courant数

构建速度方程Ueqn
根据上一时刻的phi和p求解U预测值

方程（3）中的$\left(a_P^{\vec{U}}\right)^{-1}$

计算方程（3）中的$\left(a_P^{\vec{U}}\right)^{-1}H(\vec{u})$

fvc::ddtCorr(U, phi)通过速度和通量误差来修正 面散度

调整进出口通量保证连续性

构建压力方程pEqn

非正交校正

根据方程（2）进行速度修正
修正边界区域速度

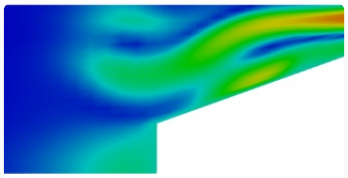
后记：

1. PISO收敛判断是否应该考虑面流量的和达到收敛判据，但是代码中循环判据为corr<nCorr。按照这种方式，如何能够准确的给定nCorr?
2. 已经了解了非正交判定的必要性和基本判定方法，但是代码中nNonOrthCorr代表的是什么呢？如何实现呢？

编辑于 2018-07-18

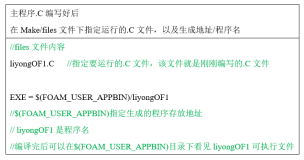
openfoam

推荐阅读



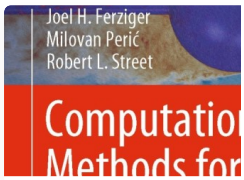
OpenFOAM的动网格

史浩 发表于OpenF...



OpenFOAM编程第3课：按模板写程序

火成岩 发表于OpenF...



笔记|非定常问题|CFD

ANFLK 发表于

1 条评论 切换为时间排序

写下你的评论...

汪洋 2018-09-15

好文章，继续！

赞

▲ 赞同 9 ▼ 1 条评论 分享 喜欢 收藏 申请转载 ...