simple、piso算法分析

汪洋
double stay

> 本文使用 Zhihu On VSCode 创作并发布

以定常或者稳态为主讲解，等下会解释为什么不带时间项。应该是结合OpenFOAM来进行分析讲解。由于笔者研究不可压缩流体，暂且不涉及能量方程。空气动力学那边的求解也不用simple,piso等算法，本质上速度压力耦合算法是用来求解不可压缩流体的。

1. 基本NS方程

$$\nabla \cdot \mathbf{U} = 0 \tag{1}$$

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot (\mathbf{U}\mathbf{U}) = -\frac{\nabla p}{\rho} + \nabla \cdot (\mu \nabla \mathbf{U}) + \mathbf{g} \tag{2}$$

2. 基本方程离散

方程离散的最终结果应该是：$\mathbf{A}x = b$，然后才能有代数方法进行求解。所以才有OF中，`system/fvSchemes` 中的相关设置。gauss-seidal, GAMG, PCG等算法的使用。本文默认使用有线体积法，将方程(2)变为定常，去除时间项。

$$\nabla \cdot (\mathbf{U}\mathbf{U}) = -\frac{\nabla p}{\rho} + \nabla \cdot (\mu \nabla \mathbf{U}) + \mathbf{g} \tag{3}$$

用有限体积法，然后将方程(3)写为积分形式。

$$\int_V [\nabla \cdot (\mathbf{U}\mathbf{U}) + \frac{\nabla p}{\rho} + \nabla \cdot (\mu \nabla \mathbf{U}) - \mathbf{g}] dV = 0 \tag{4}$$

将各项分开写出，逐个分析。

$$\underbrace{\int_V [\nabla \cdot (\mathbf{U}\mathbf{U})] dV}_{\text{对流项}} = \underbrace{-\int_V \frac{\nabla p}{\rho} dV}_{\text{压力梯度项}} + \underbrace{\int_V \nabla \cdot (\mu \nabla \mathbf{U}) dV}_{\text{扩散项}} + \underbrace{\int_V \mathbf{g} dV}_{\text{源项，重力项}} \tag{5}$$

从最简单的开始离散。

2.1 源项

2.1.1 常数源项

例如：重力。这个很简单。

$$\int_v \mathbf{g} dV = \mathbf{g} V_p \tag{6}$$

所以在矩阵形式中 $\mathbf{A}x = b$，重力源项归到方程右边，b 中。

2.1.2 线性源项

假设方程有一个线性源项，方程形式如下：

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot (\mathbf{U}\mathbf{U}) = -\frac{\nabla p}{\rho} + \nabla \cdot (\mu \nabla \mathbf{U}) + \mathbf{g} + S\mathbf{U} \tag{7}$$

其中: S 为标量 那么源项处理如下:

$$\begin{aligned}
\int_V S\mathbf{U} dV &= S \int_V \mathbf{U} dV \\
&= S \int_V [\mathbf{U_p} + (x - x_p) * \nabla \mathbf{U_p}] dV \\
&= S\mathbf{U_p} \int_V dV + \underbrace{\int_V (x - x_p) dV}_{=0} * \nabla \mathbf{U_p} \\
&= S\mathbf{U_p} V_p
\end{aligned}$$

在矩阵形式中 $\mathbf{A}x = b$，线性化的源项可以放在系数矩阵A中，也可以放在方程右边b中。看需要，当然这里也是有逻辑的。后面会讲。

2.1.3 非线性源项

如果源项形式如下：

$$\mathbf{U}\mathbf{U}$$

这种形式，一般会做线性化处理，如下：$\mathbf{U}$

$\mathbf{U}^{\mathrm{old}}\mathbf{U}$

其中: $\mathbf{U}^{\mathrm{old}}$ 表示已知速度向量，或者上一个时间步长的速度值。由于是定常计算，这里的时间不是物理时间。 这种处理方式，有时候会成为显示处理，就是说让一个 $\mathbf{U}$ 变成已知值，这样就线性化。

2.2 对流项

对流项处理通常使用散度定理，对于一个封闭的体积内有一个向量场 U ，则:

$$\int_V \nabla\cdot\mathbf{U}dV=\int_S\mathbf{U}\cdot\hat{\mathbf{n}}dS$$

所以

$$\int_V \nabla\cdot\mathbf{U}\mathbf{U}dV=\int_S\mathbf{U}(\mathbf{U}\cdot\hat{\mathbf{n}})dS$$

积分本质是求和，所以可以写成如下形式。

$$\int_S\mathbf{U}(\mathbf{U}\cdot\hat{\mathbf{n}})dS=\Sigma_{i=1}^m\int_S\mathbf{U}_i(\mathbf{U}_i\cdot\hat{n}_i)dS\approx\sum_{i=1}^m\mathbf{U}_{fi}(\mathbf{U}_{fi}\cdot\hat{n}_{fi})dS$$

实际上这里有一个非常重要的过程，就是讲体心的值插值到面心上。所以 `system/fvSchemes` 中会有很多插值算法。 `upwind、Second-order/Linear upwind, Central Differencing, QUICK` 。

对流项的处理远远不止这一点。离散化之后，对流项会成为系数矩阵中A的对角线元素和非对角线元素。

2.3 扩散项

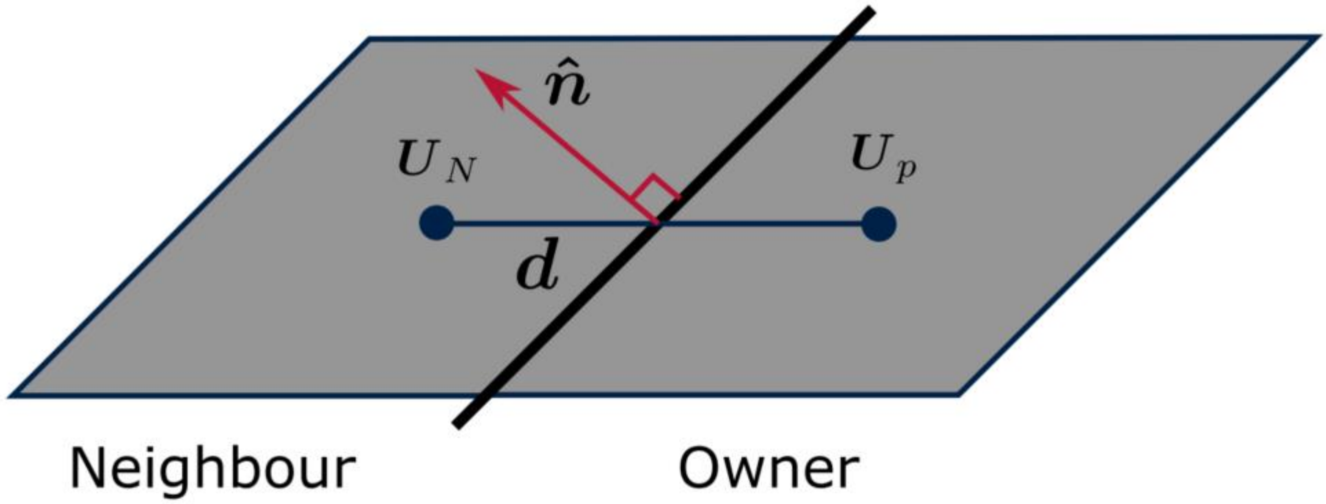
用散度定理处理对流项。 扩散项通常如下:

$$\int_V\nabla\cdot(\nu\nabla\mathbf{U})dV=\int_V\nu\nabla\mathbf{U}\cdot\hat{n}dS=\sum_{f=1}^m[\nu_f(\nabla\mathbf{U})_f\cdot n_f]S_f$$

难点还是面心上的速度梯度和面心上的单位法向量。

2.3.1 非正交修正

扩散项离散之后的难点，主要是面心上的速度梯度和面心上的单位法向量点乘的问题。



image

这里很明显的$U_n - U_p \nparallel \hat{n}$ 难点就是这样

$$\nu_f (\nabla \mathbf{U})_f \cdot \hat{n}_f$$

通常ν_f 可以通过体心插值得到。

如果速度梯度和面心法向量平行，简单。

$$(\nabla \mathbf{U})_f \cdot \hat{n}_f = \frac{\mathbf{U}_n - \mathbf{U}_p}{|d|} \cdot |\hat{n}|$$

非正交修正 通常有三种，具体可以参考Jasak的博士论文。主要是把面心法向量进行分解，所以里面有三种分解方式。

$$\sum_{f=1}^{m} [\nu_f (\nabla \mathbf{U})_f \cdot n_f] S_f$$

$$\sum_{f=1}^{m} [\nu_f (\nabla \mathbf{U})_f \cdot (\Delta + k)] S_f$$

$$\underbrace{\sum_{f=1}^{m} [\nu_f (\nabla \mathbf{U})_f \cdot \Delta_f] S_f}_{implicit} + \underbrace{\sum_{f=1}^{m} [\nu_f (\nabla \mathbf{U})_f \cdot k_f] S_f}_{explicit}$$

`OpenFOAM` 中，非正交修正项可以用一个参数γ来抑制。通常其小于1，牺牲精度得到稳定性。

> 如果网格质量好，那么在 `SIMPLE` 算法中，外部循环足够了。
> 如果网格质量不好，那么需要设置非正交修正循环。实际上是在压力修正上进行多次修正。一般设置为0，压力修正只计算一次。如果设置为1，压力修正计算两次。一般情况下，设置为即可。
> 进行非正交修正会牺牲精度来保证计算稳定性。
> 显示项有时候会归结到源项去，用上一次迭代结果计算。无界、发散
> 有时候会放弃这部分内容，坏的网格只是让我们得到一个答案。over-relaxed方法误差较大，但是最容易收敛。

3. 梯度修正

一般有三种方式，Gauss node based, Guass cell based, least square method. 在OF中，一般这样填写。对于静态网格，最小二乘法最好，因为算一次即可。这里面有许多细节。对于skewness网格，没有误差。而基于体心的结果，误差较大。

```
gradSchemes
{
    default         none;
    grad(U)         Gauss linear; // cell based;
}
```

```
gradSchemes
{
    default         none;
    grad(U)         pointLinear; // node based;
}
```

```
gradSchemes
{
    default         none;
    grad(U)         leastSquares; // least squares gradient;
}
```

4 PISO算法

首先还是进一步明确不可压缩流体的方程，一个压力速度耦合问题

$$\nabla U = 0$$

$$\frac{\partial U}{\partial t} + \nabla \cdot (UU) = -\frac{1}{\rho}\nabla p + \nu \nabla \cdot \nabla U + g$$

> 四个方程四个未知量 (p, u_x, u_y, u_z)
> 但是没有压力方程。
> 对于不可压缩流体，不能使用状态方程直接计算压力
> 连续性方程实际上是在计算动量方程时的限制条件。

当前面的方程离散之后，得到形式 $Ax = b$，就是本节主要内容，分析PISO算法如何计算的。

尽量和OpenFOAM版本接近。

$$MU = -\nabla p$$

系数矩阵M，都是上面离散后的系数。

通常来说，压力梯度可以使用初始条件或者上一个迭代步的结果，计算出当前的速度，这一步也称为 `动量预测` 。但是求出来的速度不一定满足连续性方程。

OpenFOAM

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

分解系数矩阵M

使用公式

$$MU = AU - H$$

这里的重点就是矩阵A和矩阵H。 其中A为系数矩阵M的对角元素组成，其他元素为零，所以是对称矩阵，易于求导。

$$H = AU - MU$$

$$A^{-1} = \begin{pmatrix} 1/A_{1,1} & 0 & 0 & \dots & 0 \\ 0 & 1/A_{2,2} & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1/A_{n,n} \end{pmatrix}$$

OpenFOAM

```
volScalarField rUA = 1.0/UEqn().A();
```

本质上说，矩阵H是提出对角元素之后的残余。用这个矩阵求解压力方程的源项?

$$
\begin{aligned}
AU - H &= -\nabla p \\
A^{-1}AU - A^{-1}H &= -A^{-1}\nabla p \\
U &= A^{-1}H - A^{-1}\nabla p
\end{aligned}
$$

$$
\text{由连续性方程可知，}\nabla \cdot U = 0
$$

$$
\begin{aligned}
\nabla \cdot A^{-1}H - \nabla \cdot A^{-1}\nabla p &= 0 \\
\nabla \cdot A^{-1}\nabla p &= \nabla \cdot A^{-1}H
\end{aligned}
$$

$$
\boxed{\nabla \cdot \mathcal{A}^{-1}\nabla p = \nabla \cdot A^{-1}H}
$$

在OpenFOAM中喜欢如下形式:

$$
\nabla \cdot (\frac{1}{a_p}\nabla p) = \nabla \cdot (\frac{\mathcal{H}(\mathcal{U})}{a_p})
$$

OpenFOAM

```
volScalarField rUA = 1.0/UEqn().A();
HbyA = rAU*UEqn().H();
fvm::laplacian(rAU,p) == fvc::div(phiHbyA);
```

最后一句语法还有不明确的地方。左边laplacian很对，右边did也很对，就是phi要明确，通量吗？还有fvm,fvc要知道。

压力速度耦合算法

SIMPLE算法

从上到下循环。四个方程。通常称为outer corrector

$$
\begin{aligned}
\mathcal{M}U &= -\nabla p \\
\mathcal{H} &= \mathcal{A}U - \mathcal{M}U \\
\nabla \cdot (A^{-1}\nabla p) &= \nabla (A^{-1}H) \\
U &= A^{-1}H - A^{-1}\nabla p
\end{aligned}
$$

PISO算法

$$
\mathcal{M}U = -\nabla p
$$

动量预测不在循环，只循环下面三个方程。

$$
\begin{aligned}
\mathcal{H} &= \mathcal{A}U - \mathcal{M}U \\
\nabla \cdot (A^{-1}\nabla p) &= \nabla(A^{-1}H) \\
U &= A^{-1}H - A^{-1}\nabla p
\end{aligned}
$$

通常称为inner loops

simple算法通常使用稳态流动，这样没有时间项。

> 由于时间项的存在，时间项又很小，所以系数矩阵中很容易做到的对角占优。
> 稳态流动计算那种，通常使用under-relaxation方法来做对角占优。

公式如下：

$$
a_p U_p + \sum_N a_n U_n = R_p
$$

认为添加左右两边添加一项如下

$$
\frac{1-\alpha}{\alpha} a_p U_p + a_p U_p + \sum_N a_n U_n = R_p + \frac{1-\alpha}{\alpha} a_p U_p
$$

又有用到显隐式的思想，右侧用上个时间步，左侧用当前。

$$
\frac{1}{\alpha} a_p U_p + \sum_N a_n U_n = R_p + \frac{1-\alpha}{\alpha} a_p U_p^{old}
$$

这样当 α 较小的时候，可以保证对角占优。 同样的处理方式可以处理 p, k, T 等物理量。

经验：

> 当库朗数小于1，我们可以做一次动量预测和两次压力修正(内循环).
> 使用低松弛保证对角占优。
> 可以通过多做几次压力修正来消除非正交的影响，也就是所谓的非正交修正。

这个就是大体的思路。inner,outer,non-orthogonal corrector. OpenFOAM

```
PIMPLE
{
    momentumPredictor           yes;
    nOuterCorrectors            1;// 选择为1的时候就是PISO
```

```
    nCorrectors                 3;//sets the number of times the algorithm solves
    nNonOrthogonalCorrectors    1;// 非正交修正，就是修正刚才说的压强项。
}
```

发布于 05-22