计算流体力学作业 5

郑恒 2200011086

2025 年 5 月 22 日

1 问题介绍

在单位正方形内，求解不可压缩流动。仅上边界为水平运动边界，其余边界均为固定壁面。上边界速度分布设为

$$u(x) = \sin^2(\pi x)$$

该分布在角点处函数值和导数均为零，保证了速度场的连续性。通过数值方法计算流场，考察不同位置的速度剖面、主涡涡心位置和流函数数值等。使用 Python 语言，设运动粘度 $\nu = 0.001$，画出流线图，并对主要物理量进行讨论。

2 算法原理

对于不可压缩流体，流场满足的方程为 Navier-Stokes 方程。对于二维不可压缩流体，Navier-Stokes 方程可以写成如下形式：

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial x} + \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \tag{1}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial y} + \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) \tag{2}$$

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \tag{3}$$

其中，u 和 v 分别是流体在 x 和 y 方向的速度分量，ρ 是流体密度，p 是压力，ν 是运动粘度。在本问题中，我们需要求解流体的速度场和压力场。我们使用有限差分法来离散化 Navier-Stokes 方程，并使用迭代方法来求解离散方程。我们将流场离散化为一个网格，并在每个网格点上计算速度和压力。我们可以使用显式时间步进方法来更新速度和压力场。具体步骤如下：

1. 初始化网格和边界条件
2. 计算速度场和压力场
3. 更新速度场和压力场
4. 重复步骤 2 和 3，直到收敛
5. 计算主涡涡心位置和流函数值
6. 绘制流线图和速度剖面
7. 输出结果

在本问题中，密度 ρ 被视为常数，不随空间和时间变化。由于 ρ 为常数，实际计算时常常将其归一化处理，即令 $\rho = 1$，以简化方程和计算。

在迭代计算速度场合压力场时，我们采用投影法，以下是投影法的步骤公式：

1. 预测步：先不考虑压力项，计算临时速度场 $\mathbf{u}^*$：

$$\frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} = -(\mathbf{u}^n \cdot \nabla)\mathbf{u}^n + \nu \nabla^2 \mathbf{u}^n \tag{4}$$

   其中，$\mathbf{u}^n$ 为当前时刻速度，$\mathbf{u}^*$ 为预测速度。

2. 压力泊松方程：为保证不可压缩性，对预测速度场求解压力修正：

$$\nabla^2 p^{n+1} = \frac{1}{\Delta t} \nabla \cdot \mathbf{u}^* \tag{5}$$

3. 修正步：用新压力修正速度场，得到下一个时刻的速度：

$$\mathbf{u}^{n+1} = \mathbf{u}^* - \Delta t \nabla p^{n+1} \tag{6}$$

   这样可以保证 $\nabla \cdot \mathbf{u}^{n+1} = 0$，满足不可压缩条件。

在实际算法中，我们需要对投影法的每一步进行离散化。对于上述投影法的三个步骤，采用中心差分离散化，差分格式如下：

1. 预测步：

$$\begin{aligned}\frac{u_{i,j}^* - u_{i,j}^n}{\Delta t} = & - u_{i,j}^n \frac{u_{i+1,j}^n - u_{i-1,j}^n}{2\Delta x} - v_{i,j}^n \frac{u_{i,j+1}^n - u_{i,j-1}^n}{2\Delta y} \\ & + \nu \left(\frac{u_{i+1,j}^n - 2u_{i,j}^n + u_{i-1,j}^n}{\Delta x^2} + \frac{u_{i,j+1}^n - 2u_{i,j}^n + u_{i,j-1}^n}{\Delta y^2} \right)\end{aligned} \tag{7}$$

   对 v 分量同理。

2. 压力泊松方程：

$$\frac{p_{i+1,j}^{n+1} - 2p_{i,j}^{n+1} + p_{i-1,j}^{n+1}}{\Delta x^2} + \frac{p_{i,j+1}^{n+1} - 2p_{i,j}^{n+1} + p_{i,j-1}^{n+1}}{\Delta y^2} = \frac{1}{\Delta t} \left(\frac{u_{i+1,j}^* - u_{i-1,j}^*}{2\Delta x} + \frac{v_{i,j+1}^* - v_{i,j-1}^*}{2\Delta y} \right) \tag{8}$$

3. 速度修正步：

$$u_{i,j}^{n+1} = u_{i,j}^* - \Delta t \frac{p_{i+1,j}^{n+1} - p_{i-1,j}^{n+1}}{2\Delta x} \tag{9}$$

$$v_{i,j}^{n+1} = v_{i,j}^* - \Delta t \frac{p_{i,j+1}^{n+1} - p_{i,j-1}^{n+1}}{2\Delta y} \tag{10}$$

对第 2 步压力泊松方程，我们采用带有松弛因子的高斯赛德尔迭代法进行求解：

$$p_{i,j}^* = \frac{1}{4} \left(p_{i+1,j}^n + p_{i-1,j}^{n+1} + p_{i,j+1}^n + p_{i,j-1}^{n+1} - \Delta t \left(\frac{u_{i+1,j}^* - u_{i-1,j}^*}{2\Delta x} + \frac{v_{i,j+1}^* - v_{i,j-1}^*}{2\Delta y} \right) \right) \tag{11}$$

$$p_{i,j}^{n+1} = \alpha p_{i,j}^* + (1 - \alpha) p_{i,j}^n \tag{12}$$

其中，α 为松弛因子，通常取值在 1 到 2 之间。松弛因子可以加速收敛。

3 代码实现

4 结果分析

5 AI 工具使用说明表

AI 名称	**生成代码功能**	**使用内容**
Copilot	latex 格式框架	figure 参数调整、图片插入
Deepseek	python 绘图调整	83-104、106-118 行图绘制的具体参数调整
Deepseek	gitignore 文件忽略	全由 ai 添加

6 commit 信息