大作业

郭宇扬

本次作业来自课件《**V-cycle**》。

1 Stokes 方程与交错网格上的 MAC 格式

考虑 Stokes 方程

$$\begin{cases} -\Delta \vec{u} + \nabla p = \vec{F}, & (x,y) \in (0,1)\times(0,1), \\ \operatorname{div} \vec{u} = 0, & (x,y) \in (0,1)\times(0,1), \end{cases} \tag{1}$$

边界条件为

$$\begin{aligned} &\frac{\partial u}{\partial \vec{n}} = b, \quad y = 0, \qquad &&\frac{\partial u}{\partial \vec{n}} = t, \quad y = 1, \\ &\frac{\partial v}{\partial \vec{n}} = l, \quad x = 0, \qquad &&\frac{\partial v}{\partial \vec{n}} = r, \quad x = 1, \\ &u = 0, \quad x = 0,1, \qquad &&v = 0, \quad y = 0,1, \end{aligned}$$

其中 $\vec{u} = (u,v)$ 为速度, p 为压力, $\vec{F} = (f,g)$ 为外力, $\vec{n}$ 为外法向方向.

利用交错网格上的 MAC 格式离散 Stokes 方程 (1), 可得到如下线性方程组

$$\begin{pmatrix} A & B \\ B^T & 0 \end{pmatrix} \begin{pmatrix} U \\ P \end{pmatrix} = \begin{pmatrix} F \\ 0 \end{pmatrix}. \tag{$*$}$$

2 数值算例

在区域 $\Omega = (0,1)\times(0,1)$ 上, 外力为

$$\begin{aligned} f(x,y) =& -4\pi^2(2\cos(2\pi x) - 1)\sin(2\pi y) + x^2, \\ g(x,y) =& 4\pi^2(2\cos(2\pi y) - 1)\sin(2\pi x). \end{aligned}$$

Stokes 方程 (1) 的真解为

$$\begin{aligned}u(x,y)=&(1-\cos(2\pi x))\sin(2\pi y),\\v(x,y)=&-(1-\cos(2\pi y))\sin(2\pi x),\\p(x,y)=&\frac{x^3}{3}-\frac{1}{12}.\end{aligned}$$

利用这些真解可以计算出所有边界条件.

3 大作业要求

1. 分别取 $N=64,128,256,512,1024,2048$, 以 DGS 为磨光子, 用基于 V-cycle 的多重网格方法求解离散问题 $(*)$, 停机标准为 $\|r_h\|_2/\|r_0\|_2\le 10^{-8}$, 对不同的 ν_1,ν_2,L, 比较 V-cycle 的次数和 CPU 时间, 并计算误差

$$e_N=h\left(\sum_{j=1}^{N}\sum_{i=1}^{N-1}\left|u_{i,j-\frac{1}{2}}-u(x_i,y_{j-\frac{1}{2}})\right|^2+\sum_{j=1}^{N-1}\sum_{i=1}^{N}\left|v_{i-\frac{1}{2},j}-v(x_{i-\frac{1}{2}},y_j)\right|^2\right)^{\frac{1}{2}}.$$

2. 分别取 $N=64,128,256,512$, 以 Uzawa Iteration Method 求解离散问题 $(*)$, 停机标准为 $\|r_h\|_2/\|r_0\|_2\le 10^{-8}$, 并计算误差

$$e_N=h\left(\sum_{j=1}^{N}\sum_{i=1}^{N-1}\left|u_{i,j-\frac{1}{2}}-u(x_i,y_{j-\frac{1}{2}})\right|^2+\sum_{j=1}^{N-1}\sum_{i=1}^{N}\left|v_{i-\frac{1}{2},j}-v(x_{i-\frac{1}{2}},y_j)\right|^2\right)^{\frac{1}{2}}.$$

3. 分别取 $N=64,128,256,512,1024,2048$, 以 Inexact Uzawa Iteration Method 为迭代法求解离散问题 $(*)$, 停机标准为 $\|r_h\|_2/\|r_0\|_2\le 10^{-8}$, 其中以 V-cycle 多重网格方法为预条件子, 利用共轭梯度法求解每一步的子问题 $AU_{k+1}=F-BP_k$, 对不同的 $\alpha,\tau,\nu_1,\nu_2,L$, 比较外循环的迭代次数和 CPU 时间, 并计算误差

$$e_N=h\left(\sum_{j=1}^{N}\sum_{i=1}^{N-1}\left|u_{i,j-\frac{1}{2}}-u(x_i,y_{j-\frac{1}{2}})\right|^2+\sum_{j=1}^{N-1}\sum_{i=1}^{N}\left|v_{i-\frac{1}{2},j}-v(x_{i-\frac{1}{2}},y_j)\right|^2\right)^{\frac{1}{2}}.$$

此外, 要求大家要把所有误差结果以**表格**或者**图像**呈现, 要能体现**误差收敛阶**.

4 提示与注意事项

1. 注意到原微分方程的解中压力 p 不唯一: 会相差一个常数. 故离散之后的代数方程组解中压力 P 也在相差一个常数意义下唯一. 为了确定一个解可以令 p 的积分平均为零（这是真解 p 中 $-1/12$ 的来源）, 对应到离散解 P 即其平均值为零.

2. 为提高程序运行速度, 减小内存及存储开销, 在实现迭代法时可不直接存储线性方程组 $(*)$ 的系数矩阵, 也可将 U, P, F 等向量以矩阵的形式存储, 此时矩阵向量乘法 AU, BP 可看作 U, P 与某些矩阵的矩阵卷积. 而在底层网格上求解可以使用只需矩阵向量乘法运算的共轭梯度法.

3. 如果 B 是离散的梯度算子 ∇, 那么 B^T 应该离散梯度算子的伴随算子, 即负的散度算子 $-\nabla\cdot$. 而在对残量方程做提升或限制后, 得到的新线性方程组一般并不满足数值散度为 0 的条件, 此时待求解的线性方程组将形如

$$\begin{pmatrix} A & B \\ B^T & 0 \end{pmatrix} \begin{pmatrix} U \\ P \end{pmatrix} = \begin{pmatrix} F \\ D \end{pmatrix}. \tag{$**$}$$

其中 $D = [d_{i,j}]$ 为负的数值散度. 在利用 DGS 迭代法求解线性方程组 $(**)$ 时, 散度方程的残量 $r_{i,j}$ 将变为

$$r_{i,j} = -\frac{u_{i,j-\frac{1}{2}}^{k+\frac{1}{2}} - u_{i-1,j-\frac{1}{2}}^{k+\frac{1}{2}}}{h} - \frac{v_{i-\frac{1}{2},j}^{k+\frac{1}{2}} - v_{i-\frac{1}{2},j-1}^{k+\frac{1}{2}}}{h} - d_{i,j}.$$

即 $r = B^T U^{k+\frac{1}{2}} - D = -\nabla_h \cdot U^{k+\frac{1}{2}} - D$. 在利用 Uzawa 和 Inexact Uzawa 迭代法求解线性方程组 $(**)$ 时, 更新的压力 P_{k+1} 将变为

$$P_{k+1} = P_k + \alpha(B^T \hat{U}_{k+1} - D).$$

特别地, 当 $D = 0$ 时, 得到的迭代格式即为课件上给出的迭代格式.

4. 通过算子运算可以部分得到矩阵 $B^T A^{-1} B$ 的特征信息, 这将有助于选取 Uzawa 迭代法中的最优参数 α^*.

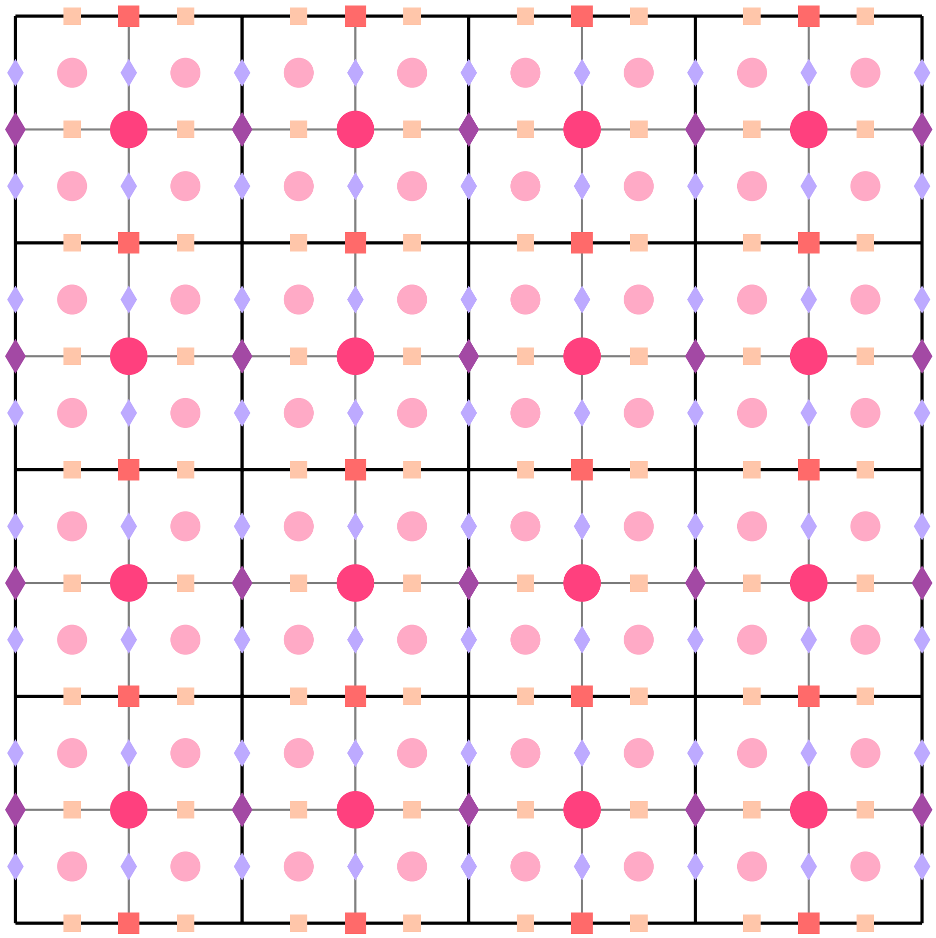
5. 在计算粗网格上的 A_{2h}, B_{2h} 时, 按照课件上的方法, A_{2h}, B_{2h} 应满足

$$\begin{pmatrix} A_{2h} & B_{2h} \\ B_{2h}^T & 0 \end{pmatrix} = I_h^{2h} \begin{pmatrix} A_h & B_h \\ B_h^T & 0 \end{pmatrix} I_{2h}^h,$$

但这样计算得到的 A_{2h}, B_{2h} 形式会比较奇怪. 在实际应用中, 可以直接利用在粗网格上的 MAC 格式离散 Stokes 方程得到的系数矩阵去近似 A_{2h}, B_{2h}, 以方便算法的实现.

6. 在 DGS 迭代中, 扫描全部单元的顺序会对收敛性产生一定的影响, 但在本问题中不会影响是否收敛, 在实现时可以采用课件上的顺序, 也可以尝试按行列遍历或红黑格迭代. (选用其中一种方便实现的方法实现即可)

7. 提升限制算子的选择有很多种, 除课件上给出的提升限制算子外, 还有一些易于实现的提升限制算子可供选择, 如下图所示. 经过数值实验, 该提升限制算子的效率远高于课件中给出的提升限制.



限制算子

◆ = 最近两个◇/4 + 次近四个◇/8

■ = 最近两个□/4 + 次近四个□/8

● = 最近四个○/4

提升算子

◇ (粗网格线) = 最近◆

◇ (细网格线) = 最近两个◆/2

□ (粗网格线) = 最近■

□ (细网格线) = 最近两个■/2

○ = 最近●