数值代数大作业

吴睿林 2200010914

北京大学 数学科学学院

2025年2月

目录

1 引言

本次作业研究了 Stokes 方程的数值解法, 主要通过以下三种方法来求解离散化后的 Stokes 方程:

- 以 DGS 为磨光子的 V-cycle 多重网格法
- Uzawa 迭代法
- Inexact Uzawa 迭代法 (V-cycle 多重网格为预条件子)

Stokes 方程的定义区域为 $\Omega = (0,1) \times (0,1)$, 外力和边界条件具体如下:

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

其中, $\vec{u} = (u,v)$ 是速度, p 是压力, $\vec{F} = (f,g)$ 是外力. 边界条件为:

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

边界条件的函数由精确解给出并假设已知, 并且在边界 $x = 0, 1, y = 0, 1$ 上, $u = 0, v = 0$.

利用交错网格上的 MAC 格式进行离散可得到如下方程组:

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

本报告中将展示如何利用上述三种方法求解此方程组, 并对其数值结果进行分析.

2 基于 DGS 的 V-cycle 多重网格法

2.1 算法描述

针对此题的 DGS 迭代法基本步骤如下:

1. 初始化: 设初始值 $U_0 = (u_0^T, v_0^T)^T$ 和 P_0 (通常取为零), 令 $k = 0$, 分解矩阵 $A = D_A - L_A - U_A$.
2. 使用 Gauss-Seidel 方法更新速度分量:
   - $U_{k+1/2} = U_k + (D_A - L_A)^{-1}(F - BP_k - AU_k)$.

3. 更新速度与压力:

   - 逐个对每个单元计算散度残量, 并据此调整速度和压力值, 使其满足散度方格, 边界和顶点的情况仅考虑离散网格内的点.

以 DGS 为磨光子, V-cycle 多重网格法的基本步骤如下:

1. 网格生成与初始化
2. 预光滑操作 (使用 DGS 进行 v_1 次磨光)
3. 限制算子操作
4. 粗网格求解, 重复 V-cycle 的步骤直到到达底层最粗网格, 此时增大磨光次数得到较为精确的解
5. 插值算子操作
6. 后光滑操作 (使用 DGS 进行 v_2 次磨光)

2.2 算法实现

基于 DGS 的 V-cycle 多重网格方法的实现主要包含如下步骤:

- Gauss-Seidel 迭代中使用分量和 for 循环进行更新
- DGS 迭代的第二步更新压力和散度时按照课件上的顺序, 先更新内部单元, 再更新边界单元, 最后更新顶点
- 计算 Laplace 项 Δu 时使用 matlab 自带矩阵卷积函数 conv2 和卷积核加速计算, 卷积核如下:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

2.3 数值结果

此处将展示不同网格尺寸 ($N = 64, 128, 256, 512, 1024, 2048$) 下的数值结果, 包括每个网格的迭代次数, CPU 时间以及误差的收敛行为. 其中 v_1 为前光滑迭代次数, v_2 为后光滑迭代次数, L 为最底层网格相比于顶层网格的限制倍数, N/L 为底层网格每边的网格数, 具体结果如下表:

N	64	128	256	512	1024	2048
迭代次数	7	7	6	6	6	6
运行时间	0.0233	0.0596	0.1686	0.9458	4.6490	29.9837
误差	1.4951e-03	3.7363e-04	9.3402e-05	2.3353e-05	5.8410e-06	1.4635e-06

表 1: $v_1 = 2, v_2 = 2, L = N/2$

N	64	128	256	512	1024	2048
迭代次数	12	12	12	12	12	12
运行时间	0.0390	0.0739	0.2360	1.3739	7.1399	72.0763
误差	1.4951e-03	3.7363e-04	9.3402e-05	2.3349e-05	5.8372e-06	1.4593e-06

表 2: $v_1 = 1, v_2 = 1, L = N/2$

N	64	128	256	512	1024	2048
迭代次数	5	5	5	5	5	5
运行时间	0.0382	0.0505	0.1775	1.2420	6.4683	38.3802
误差	1.4951e-03	3.7362e-04	9.3391e-05	2.3342e-05	5.8297e-06	1.4519e-06

表 3: $v_1 = 4, v_2 = 4, L = N/2$

N	64	128	256	512	1024	2048
迭代次数	7	7	7	6	6	6
运行时间	0.0446	0.0572	0.1862	0.9409	4.9561	32.9805
误差	1.4951e-03	3.7363e-04	9.340e-05	2.3350e-05	5.8384e-06	1.4605e-06

表 4: $v_1 = 2, v_2 = 2, L = N/8$

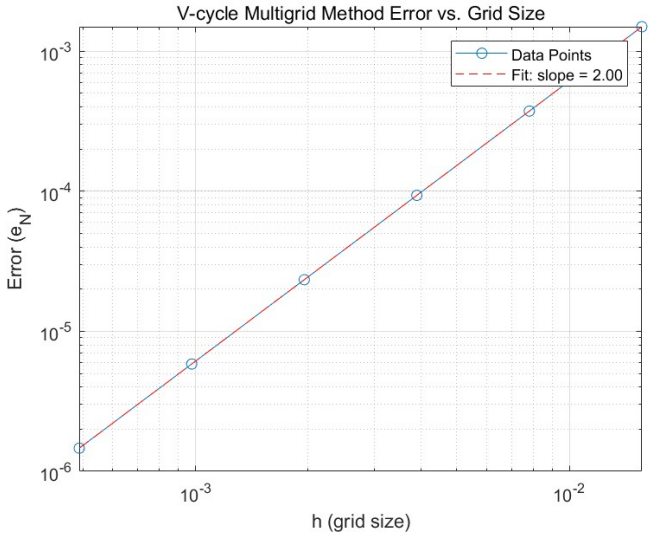

图 1: DGS为光滑子的V-cycle多重网格法的**误差收敛阶**

2.4 结果分析

误差随网格尺寸变化的双对数图的参数为 $v_1 = v_2 = 2, L = N/2$, 由于不同参数计算得到的误差变化不大, 所以就只放一张, 由图可以看到, **误差收敛阶**为 2, 且十分稳定.

对比不同的 v_1, v_2, L 可以发现

- $v_1 = 2, v_2 = 2, L = N/2$ 时运行速度最快.
- 加大 v_1, v_2 会减少迭代次数, 但运行时间也会增大, 此时磨光次数太多, 多余的磨光次数浪费了算力.
- 减小 v_1, v_2, 此时迭代次数明显增加, 运行时间仍然增大, 此时可解释为每次 V-cycle 没有充分磨光, 所以需要迭代更多次 V-cycle, 增大了运行时间.
- 减小 L, 即提升底层最粗网格, 此时迭代次数和求解时间变化不明显, 因为我在底层最粗网格选择了精确求解 (固定迭代 100 次), 所以结果变化不大.

2.5 二阶收敛性分析

MAC 格式采用交错网格, 将 u, v 定义在网格边界的中心, 而 p 定义在网格中心. 以 h 表示网格步长, 离散格式如下:

拉普拉斯算子 Δu 采用中心差分离散:

$$\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h^2}.$$

梯度项 ∇p 采用二阶中心差分:

$$\frac{p_{i+1,j} - p_{i-1,j}}{2h}, \quad \frac{p_{i,j+1} - p_{i,j-1}}{2h}.$$

散度条件 $\nabla \cdot \mathbf{u} = 0$ 采用中心差分:

$$\frac{u_{i+1,j} - u_{i-1,j}}{2h} + \frac{v_{i,j+1} - v_{i,j-1}}{2h} = 0.$$

为了验证 MAC 格式的二阶收敛性, 我们进行截断误差分析.

对 Stokes 方程在 $h \to 0$ 进行泰勒展开, 并与离散格式比较, 得到:

拉普拉斯算子的截断误差为 $O(h^2)$.

梯度算子的截断误差为 $O(h^2)$.

散度条件的截断误差为 $O(h^2)$.

综上, 离散方程对原方程的截断误差严格为二阶, 即 $O(h^2)$, 说明 MAC 格式在理论上具有二阶准确性.

3 Uzawa 迭代法

3.1 算法描述

给定 P_0, 令 $k = 0$

1. 精确求解 $AU_{k+1} = F - BP_k$
2. 更新压力 $P_{k+1} = P_k + \alpha(B^T U_{k+1})$
3. 判断误差是否小于允许的值: 如果小于, 则停止迭代, 否则回到第一步

3.2 数值结果

此处展示使用 Uzawa 迭代法求解的数值结果, 包括迭代次数, 误差, 运行时间等.

N	64	128	256	512	1024	2048
迭代次数	2	2	2	2	2	2
运行时间	0.0114	0.0547	0.2646	1.3050	5.9646	57.5572
误差	1.4951e-03	3.7363e-04	9.340e-05	2.3350e-05	5.8372e-06	1.4593e-06

表 5: $\alpha = 1$

3.3 结果分析

因为在 Uzawa 迭代法第一步精确求解 $AU_{k+1} = F - BP_k$ 时, 我直接选用 matlab 中自带的左除函数, 这步运行速度较慢, 所以此方法总体运行时间比基于 DGS 的 V-cycle 多重网格法要慢. 从误差-步长的图像中可以看到误差收敛阶仍是精确等于 2.

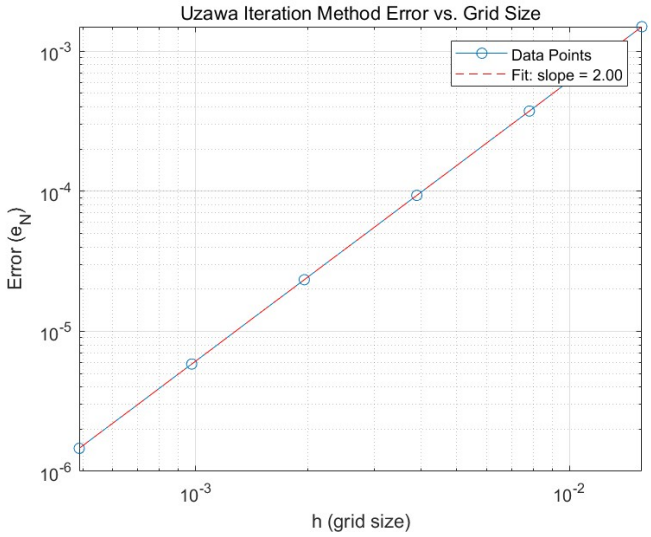

图 2: Uzawa 迭代法的**误差收敛阶**

3.4 $B^TA^{-1}B$ 的特征值

令 $S = B^TA^{-1}B$, 首先每个分量均为 1 的向量为 B 一个特征向量, 所以 0 是 B 的一个特征值, 容易验证 0 是 B 的单特征值.

通过计算可得 $AB = BB^TB$, 于是有 $(I - S)B^TB = 0$, 又 $\mathrm{rank}(B^TB) = \mathrm{rank}(B) = N^2 - 1$, 故有 1 为 S 的 $N^2 - 1$ 重特征值.

故 $B^TA^{-1}B$ 的特征值仅有 0, 1, 所以在 Uzawa 迭代法中对应的迭代矩阵 $I - \alpha B^TA^{-1}B$ 特征值为 $1, 1 - \alpha$, 此时 α 取 $[0, 2]$ 中的数都不会增大迭代矩阵的谱半径, 实际上 $\frac{2}{\lambda_{min}(B^TA^{-1}B)+\lambda_{max}(B^TA^{-1}B)}$ 为使得最大最小特征值绝对值相同的情况. 为了使得零化更多特征向量, 并加以数值验证, 发现此题中 α 取 1 表现良好.

4 Inexact Uzawa迭代法

4.1 算法描述

Inexact Uzawa 迭代法是对 Uzawa 迭代法的改进, 主要思想是子问题求解线性方程组时不精确求解, 仅求得近似解, 减少每次迭代的计算量. 在本作业中, 我们使用 V-cycle 多重网格作为预条件子, 利用共轭梯度法来进行每次子问题的求解. 算法步骤如下

给定 P_0, 令 $k = 0$

1. 近似求解 $AU_{k+1} = F - BP_k$ 得到 $\hat{U}_{k+1}$, 采用共轭梯度法近似求解, 共轭梯度法中使用 V-cycle 为预条件子

2. 更新压力 $P_{k+1} = P_k + \alpha(B^T\hat{U}_{k+1})$

3. 判断误差是否小于允许的值: 如果小于, 则停止迭代, 否则回到第一步

令 $\hat{U}_{k+1}$ 为方程 $AU_{k+1} = F - BP_k$ 的近似解, 定义

$$\delta_k = A\hat{U}_{k+1} - F + BP_k$$

要求近似解满足

$$\|\delta_k\|_2 \leq \tau \|B^T \hat{U}_k\|$$

4.2 数值结果

此处展示 Inexact Uzawa 迭代法的数值结果, 包括迭代次数, CPU时间, 误差等.

N	64	128	256	512	1024	2048
迭代次数	12	12	12	12	12	12
运行时间	0.0216	0.0553	0.2256	1.0848	5.0259	47.8447
误差	1.4951e-03	3.7363e-04	9.340e-05	2.335e-05	5.838e-06	1.460e-06

表 6: $\alpha = 1, \tau = 10^{-5}, v_1 = 2, v_2 = 2, L = N/2,$

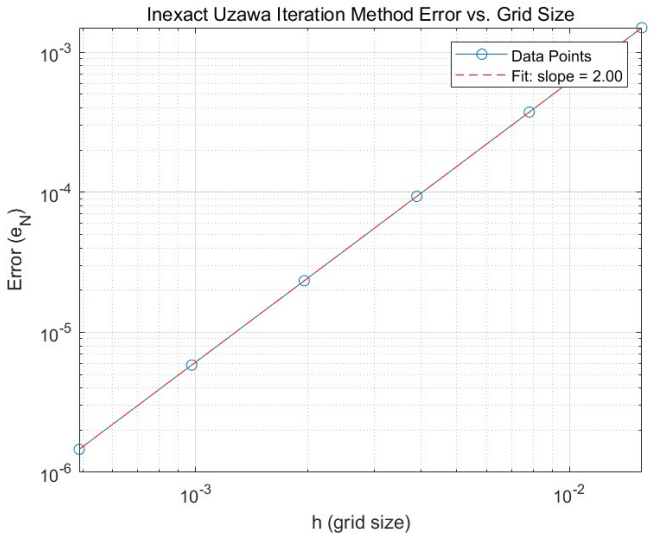

图 3: Inexact Uzawa 迭代法的误差收敛阶

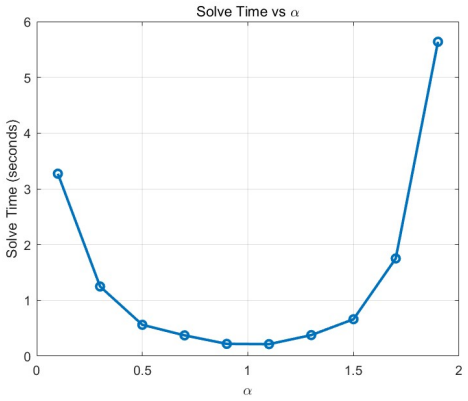

图 4: α 取不同值对应的运行时间

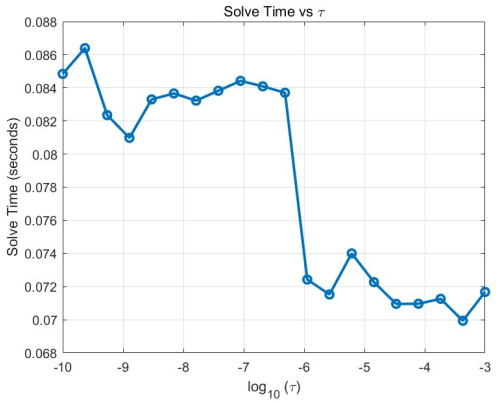

图 5: τ 取不同值对应的运行时间

4.3 结果分析

Inexact Uzawa 迭代法表现出较传统 Uzawa 法更高的计算效率, 尽管迭代次数相比于传统 Uzawa 方法更多, 但每次迭代的速度和效率会高很多, 所以总运行时间会更少, 尤其在大规模问题 ($N = 2048$) 中, 收敛速度显著提高.

观察图 3 可知误差收敛阶仍然精确为 2, 对于 Inexact Uzawa 迭代法, 变化前后磨光次数 v_1, v_2 以及修改底层网络的尺寸与第一题中的影响类似, 这里主要分析变化 α 和 τ 对运行时间的影响, 固定 $N = 128$, $v_1 = v_2 = N/L = 2$, 由图 4 可知 α 为 1 时运行时间最少, 并且影响显著, 而对于 α 取 1 最优的原因, 在前文第二题对矩阵 $B^T A^{-1} B$ 特征值的分析可以说明. 而变化 τ 对运行时间的影响则是 τ 太大时 (大于 10^{-3}) 结果可能不收敛, 而从图 5 中可以看到 τ 也不应太小 (小于 10^{-6}), 否则也会因为求解"过度"影响求解效率, 此时相当于回归传统 Uzawa 算法, 在精确求解 $AU_{k+1} = F - BP_k$;