有限元第四次编程作业

W Huang

日期：2023 年 12 月 25 日

1 Stokes 方程

1.1 求解设置

求解无滑移边界条件的不可压 Stokes 方程：

$$\begin{cases} -\Delta \mathbf{u} + \nabla p = \mathbf{f} & , \text{in } \Omega, \\ \nabla \cdot \mathbf{u} & , \text{in } \Omega, \\ \mathbf{u} = 0 & , \text{on } \partial\Omega. \end{cases} \tag{1}$$

可以导出弱形式：

$$(\nabla \mathbf{u}, \nabla \mathbf{v}) + (\nabla \cdot \mathbf{v}, \nabla p) + (\nabla \cdot \mathbf{u}, \nabla q) = (\mathbf{f}, \mathbf{v}), \quad \forall \mathbf{v} \in H_0^1(\Omega), q \in L_0^2(\Omega). \tag{2}$$

这里

$$L_0^2(\Omega) = \left\{ q \in L^2(\Omega) : \int_\Omega q \, dx = 0 \right\}. \tag{3}$$

现在我们在有限元空间 $\mathcal{P}_2^d \times \mathcal{P}_1$ 中取逼近。取精确解

$$\mathbf{u}(\mathbf{x}) = \pi(\sin^2(\pi x_1)\sin(2\pi x_2), -\sin(2\pi x_1)\sin^2(\pi x_2)), \tag{4}$$

$$p(\mathbf{x}) = \cos(\pi x_1)\sin(\pi x_2), \tag{5}$$

并导出右端项，进行测试。网格如图 1 所示。离散系统的稀疏模式 (sparsity pattern) 具有明显的块状结构，如图 2 所示（我们对自由度进行了重编号：先用 Cuthill McKee 算法对所有自由度进行重排，然后再保序地按 u_1, u_2, p 的顺序重排）。

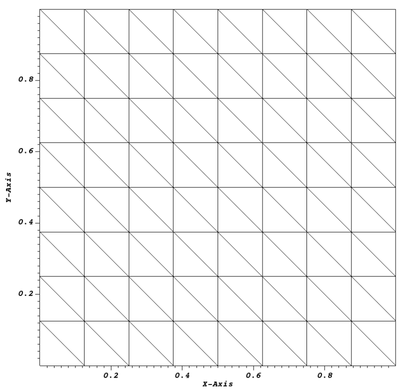

图 1: $h = \frac{1}{8}$ 时的计算网格

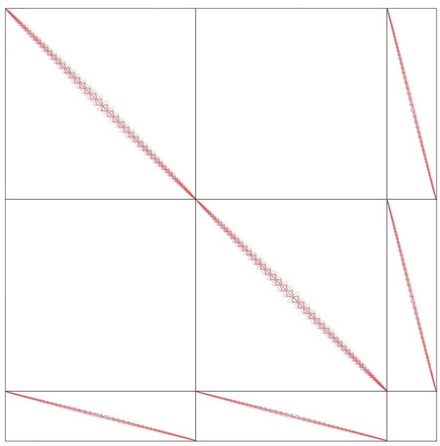

图 2: $h = \frac{1}{32}$ 时离散系统的稀疏模式

按照题目要求，我们使用 MINRES 方法求解，`deal.ii` 中为我们提供了 `SolverMinRes`，可直接调用。现在考虑题目所述的预优算子：

$$B = \begin{pmatrix} B_\Delta & & \\ & B_\Delta & \\ & & (\mathrm{diag}\ M_Q)^{-1} \end{pmatrix}. \tag{6}$$

其中 B_Δ 是使用 V-多重网格进行一轮松弛。考虑到 $\mathcal{P}_2$ 元的索引非常复杂，粗细网格插值极其困难，因此我们决定使用代数多重网格 (AMG)。`Trilinos` 为我们提供了相关的库，`deal.ii` 将其引入并进行了用户友好的封装，我们直接调用 `TrilinosWrappers::PreconditionAMG` 即可。

最后，我们还需要计算误差。我们采用 $n = 3$ 的高斯求积公式（二维情形下，每个三角形中有 9 个积分节点，具有 5 阶代数精度）来近似计算误差的 L^2 范数。

1.2 编译说明

请安装依赖库：

- `openMPI`（也可以用 `MPICH` 等其它 MPI 软件包代替）；
- `Trilinos` （`deal.ii` 的 README 中提及了安装方式）；
- `deal.ii` （请确保 `DEAL_II_WITH_TRILINOS` 开关是开启状态）。

在确保依赖库正确安装后，请输入以下命令编译。

```
cd src
mkdir build
cd build
cmake ..
make release
make
```

等待编译完成后，用以下命令执行测试：

```
./stokes
```

上述测试将从 $h = \frac{1}{4}$ 开始，逐次加密，一直运行到 $h = \frac{1}{1024}$。

1.3 测试结果

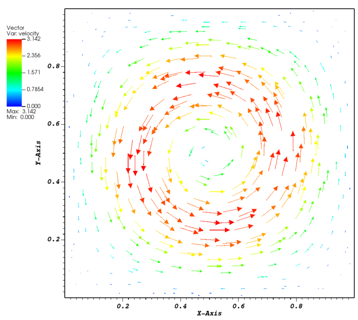

图 3: $h = \frac{1}{256}$，速度场

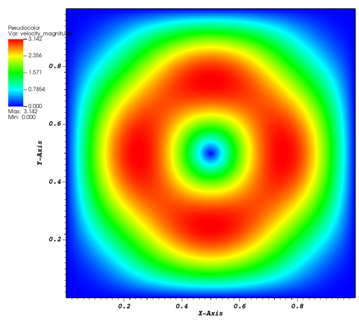

图 4: $h = \frac{1}{256}$，速度的大小 $|\mathbf{u}|$

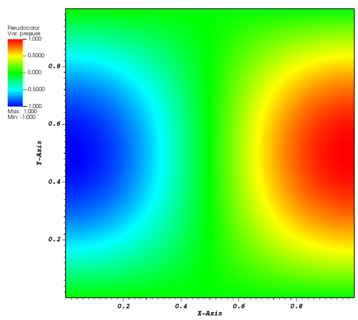

图 5: $h = \frac{1}{256}$，压强 p

我们将求得的数值解用 `VisIt` 绘制，如图 $3-5$ 所示，其图像与我们构造的解析解一致。

用朴素 MINRES 方法与预优 MINRES 方法对比，见表 1。表中的“装配耗时”是指从稀疏模式生成完毕开始，到刚度矩阵计算完成耗费的时间；“求解耗时”是指 AMG 初始化和 MINRES 迭代的耗时之和。可以看到朴素 MINRES 方法的迭代次数随着网格加密而指数增长，求解时间也逐渐变得令人难以接受；而预优 MINRES 方法的迭代次数涨幅非常小，求解时间明显快于朴素 MINRES 方法，甚至对 $h=\frac{1}{1024}$ 的网格也能轻松求解。

此外，我们还输出了预优 MINRES 方法求解误差的 L^2 范数，见表 2。可以看到 $\mathbf{u}$ 保持了 $\mathcal{P}_2$ 元的 L^2 范数三阶收敛性质；p 保持了 $\mathcal{P}_1$ 元的 L^2 范数二阶收敛性质。

	h	$\frac{1}{128}$	$\frac{1}{256}$	$\frac{1}{512}$	$\frac{1}{1024}$
预优 MINRES 方法	装配耗时 (s)	0.134	0.613	2.38	9.44
	求解耗时 (s)	1.32	6.42	32.0	140
	MINRES 迭代次数	80	85	100	105
朴素 MINRES 方法	装配耗时 (s)	0.168	0.601	2.37	9.59
	求解耗时 (s)	9.61	93.5	791	>1h, killed
	MINRES 迭代次数	2898	5449	10628	

表 1: 预优 MINRES 方法与朴素 MINRES 方法，$h=\frac{1}{128}$ 到 $\frac{1}{1024}$ 的 CPU 耗时。迭代终止条件均为残差的 2 范数小于等于右端项 2 范数的 10^{-6} 倍，即 $||\text{res}||_2 \le 10^{-6}||\text{rhs}||_2$。

h	$\frac{1}{256}$	收敛阶	$\frac{1}{512}$	收敛阶	$\frac{1}{1024}$
$\|\|u_1^h-u_1\|\|_{L^2}$	2.03e-07	2.99	2.56e-08	2.91	3.40e-09
$\|\|u_2^h-u_2\|\|_{L^2}$	2.03e-07	2.98	2.57e-08	2.89	3.46e-09
$\|\|p^h-p\|\|_{L^2}$	6.29e-06	1.99	1.58e-06	1.97	4.02e-07

表 2: 预优 MINRES 方法，$h=\frac{1}{256}$ 到 $\frac{1}{1024}$ 的 L^2 误差

2 题外话：投影方法

事实上，假如我们额外地知道 $\Delta\mathbf{u}$ 在边界上的法向分量，我们可以使用投影方法来解决 Stokes 方程。只需对 Stokes 方程两侧应用散度算子，然后将不可压条件代入即可。第一步是求解 Neumann 边界条件的 Poisson 方程：

$$\begin{cases} \Delta p = \nabla\cdot\mathbf{f} & ,\text{in } \Omega, \\ \mathbf{n}\cdot\nabla p = \mathbf{n}\cdot(\mathbf{f}+\Delta\mathbf{u}) & ,\text{on } \partial\Omega. \end{cases} \tag{7}$$

这里我们并不需要真的去计算 $\nabla\cdot\mathbf{f}$，因为在弱形式下，我们有：

$$(\nabla p,\nabla q)_\Omega = (\mathbf{f},\nabla q)_\Omega + (\mathbf{n}\cdot\Delta\mathbf{u},q)_{\partial\Omega}. \tag{8}$$

第二步是计算 $\mathbf{u}=\mathbf{f}-\nabla p$，然后第三步是解 u_1 与 u_2 的齐次 Dirichlet 边界条件 Poisson 方程。

投影法的优势在于速度快，而且 $\mathbf{u}$ 与 p 可以使用一样的有限元，从而拥有一致的高阶收敛阶。我们用 $\mathcal{Q}_2$ 元实现了 Stokes 方程的投影法，表 3 展示了投影法的误差和 CPU 耗时，代码见：

```
https://github.com/EbolaEmperor/Study/tree/main/FEM/09-Stokes-System/Projection-
    Method
```

h	$\frac{1}{256}$	收敛阶	$\frac{1}{512}$	收敛阶	$\frac{1}{1024}$
$\|\|u_1^h - u_1\|\|_{L^2}$	1.12e-07	3.00	1.40e-08	2.96	1.80e-09
$\|\|u_2^h - u_2\|\|_{L^2}$	1.12e-07	3.00	1.40e-08	2.95	1.81e-09
$\|\|p^h - p\|\|_{L^2}$	7.04e-09	3.12	8.10e-10	3.03	9.93e-11
CPU 耗时 (s)	2.45		10.4		43.1

表 3: 投影法，$h = \frac{1}{256}$ 到 $\frac{1}{1024}$ 的 L^2 误差、CPU 耗时