有限元第三次编程作业

W Huang

日期：2025 年 3 月 18 日

1 编程第一题

1.1 求解设置

求解 PDE

$$\begin{cases} -\Delta u = f, \quad \text{in } \Omega = [0,1]^2, \\ u = 0, \quad \text{on } \partial\Omega. \end{cases} \tag{1}$$

取精确解

$$u(x,y) = \sin(\pi x)\sin(\pi y), \tag{2}$$

并导出右端项，进行测试。网格如图 1 所示。

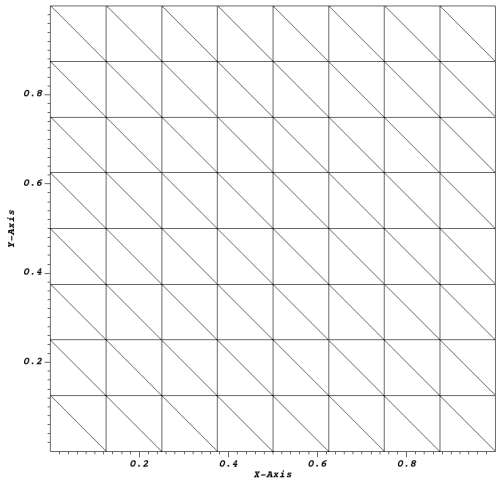

图 1: $h = \frac{1}{8}$ 时的计算网格

考虑 Hiercharchical Basis (以下简称 HB) 预优算子：

$$B = D + IA_H^{-1}I^t. \tag{3}$$

其中 I 是线性元空间下粗网格到细网格的自然延拓（即线性插值），而 A_H^{-1} 涉及到方程组的求解。我们选择使用**朴素 CG 方法**求解粗网格上的方程 $A_H u_H = f_H$。

1.2 编译说明

请安装 deal.ii 及其依赖库，见 https://www.dealii.org/developer/readme.html；安装完毕后，在本文档目录下打开终端，依次运行：

```
cd src-2D
mkdir build
cd build
cmake ..
make release
make
```

等待编译完成后，用以下命令执行测试：

```
./elliptic 10 2
```

上述测试采用 1.1 节所述的网格，规模为 $N = 2^{10}$，使用两层网格；如果需要改变网格规模，将 10 换成别的正整数；如需使用多层网格，将 2 换成别的正整数；特别地，当最后一个参数缺省时，默认使用单层网格，即朴素 CG 方法。

1.3 测试结果

用朴素 CG 方法与 HB 预优 CG 方法对比，Tolerance 设置为 $\varepsilon = 10^{-12}||f||_2$ (其中 f 是方程的右端项) 结果见图 2, 图 3 和表 1。

可以看到，朴素 CG 方法的 CG 迭代次数指数级增长；HB 预优 CG 方法的 CG 迭代次数不升反降。将预处理耗时与求解耗时相加，可以看出在 $h = 1/1024$ 时，HB 预优 CG 方法具有明显加速。

首先我们需要说明，Tolerance 设置为 $\varepsilon = 10^{-12}||f||_2$ 远远超过了实际需要，其实想要达到上表的误差，Tolerance 设置为 $\varepsilon = 10^{-4}||f||_2$ 足矣，求解几乎是瞬间完成。我们取如此夸张的 Tolerance 是为了对比出加速效果。

但是，我们只取了两层网格，因此加速效果不明显。甚至当 h 继续变小时，HB 预优 CG 方法的速度比朴素 CG 方法还要慢。这是因为 HB 预优方法虽然最外层的 CG 迭代次数得到了控制，但它需要在粗网格中进行大量的 CG 迭代，代价太大。要在实际中应用，还是需要多层网格。

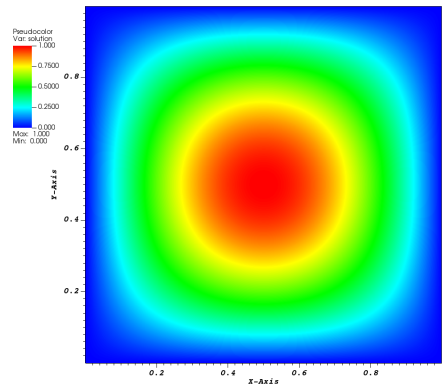

图 2: $h = \frac{1}{256}$ 时的数值解

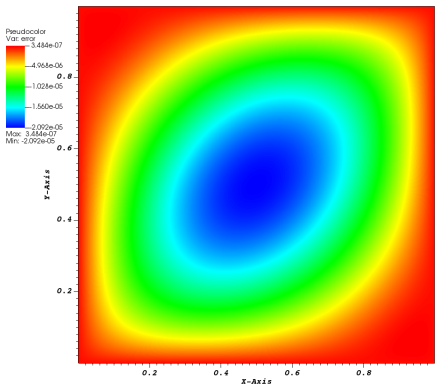

图 3: $h = \frac{1}{256}$ 时数值解与真解的误差

	h	$\frac{1}{256}$	收敛阶	$\frac{1}{512}$	收敛阶	$\frac{1}{1024}$
朴素 CG 方法	$\|\|u^h-u\|\|_{L^2}$	2.46e-05	2.00	6.16e-06	2.00	1.54e-06
	$\|\|u^h-u\|\|_{H^1}$	1.36e-02	1.00	6.82e-03	1.00	3.41e-03
	预处理耗时 (s)	0.23		0.92		3.9
	求解耗时 (s)	0.031		0.26		8.9
	CG 迭代次数	439		852		1651
HB 预优 CG 方法（两层网格）	$\|\|u^h-u\|\|_{L^2}$	2.46e-05	2.00	6.16e-06	2.00	1.54e-06
	$\|\|u^h-u\|\|_{H^1}$	1.36e-02	1.00	6.82e-03	1.00	3.41e-03
	预处理耗时 (s)	0.27		1.1		4.8
	求解耗时 (s)	0.10		0.84		3.6
	CG 迭代次数	52		94		62
HB 预优 CG 方法（三层网格）	$\|\|u^h-u\|\|_{L^2}$	2.46e-05	2.00	6.16e-06	2.00	1.54e-06
	$\|\|u^h-u\|\|_{H^1}$	1.36e-02	1.00	6.82e-03	1.00	3.41e-03
	预处理耗时 (s)	0.29		1.2		4.9
	求解耗时 (s)	0.15		0.68		3.0
	CG 迭代次数	52		50		49

表 1: $h=\frac{1}{256},\frac{1}{512},\frac{1}{1024}$ 的求解结果

2 编程第二题

2.1 求解设置

求解 PDE

$$\begin{cases} -\Delta u = f, & \text{in } \Omega=[0,1]^3, \\ u=0, & \text{on } \partial\Omega. \end{cases} \tag{4}$$

取精确解

$$u(x,y,z)=10\sin(\pi x)\sin(\pi y)\sin(\pi z), \tag{5}$$

并导出右端项，进行测试。三维空间中的四面体网格过于复杂，粗细网格之间的索引极其困难，因此为了简化，我们选择了使用六面体网格，如图 4 所示。

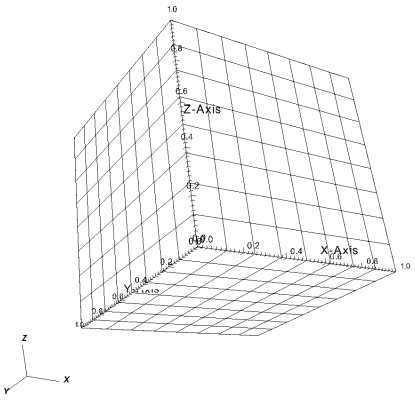

图 4: $h=\frac{1}{8}$ 时的计算网格

六面体网格对应的有限元空间是 Q_1 元，在 Q_1 元中，对于一个参考单元 $[0,1]^3$，其节点基函数为：

$$\begin{aligned}
\phi_{(0,0,0)}(x,y,z) &= (1-x)(1-y)(1-z),\\
\phi_{(0,0,1)}(x,y,z) &= (1-x)(1-y)z,\\
\phi_{(0,1,0)}(x,y,z) &= (1-x)y(1-z),\\
\phi_{(0,1,1)}(x,y,z) &= (1-x)yz,\\
\phi_{(1,0,0)}(x,y,z) &= x(1-y)(1-z),\\
\phi_{(1,0,1)}(x,y,z) &= x(1-y)z,\\
\phi_{(1,1,0)}(x,y,z) &= xy(1-z),\\
\phi_{(1,1,1)}(x,y,z) &= xyz.
\end{aligned}$$

在六面体网格和 Q_1 元下，粗细网格之间的索引非常容易，且线形插值的表示也非常简单。

考虑 Hiercharchical Basis (以下简称 HB) 预优算子：

$$B = D + IA_H^{-1}I^t. \tag{6}$$

其中 I 是线性元空间下粗网格到细网格的自然延拓（即线性插值），而 A_H^{-1} 涉及到方程组的求解。我们选择使用**朴素 CG 方法**求解粗网格上的方程 $A_H u_H = f_H$。

编译、运行方法与二维程序完全相同，这里不做重复说明。

2.2 测试结果

用朴素 CG 方法与 HB 预优 CG 方法对比，Tolerance 设置为 $\varepsilon = 10^{-6}||f||_2$ (其中 f 是方程的右端项) 结果见表 2。

结论是两层网格的 HB 预优 CG 方法一致收敛，但与朴素 CG 方法相比，毫无加速效果。另外，我们观察到朴素 CG 方法的迭代次数不随网格加细而增加，甚至稳定在 1 次，这是一个有趣的现象。

	h	$\frac{1}{32}$	收敛阶	$\frac{1}{64}$	收敛阶	$\frac{1}{128}$
朴素 CG 方法	$\|\|u^h-u\|\|_{L^2}$	2.84e-03	2.00	7.10e-04	2.00	1.77e-04
	$\|\|u^h-u\|\|_{H^1}$	5.45e-01	1.00	2.73e-01	1.01	1.36e-01
	预处理耗时 (s)	0.13		1.2		10
	求解耗时 (s)	0.00061		0.0044		0.035
	CG 迭代次数	1		1		1
HB 预优 CG 方法（两层网格）	$\|\|u^h-u\|\|_{L^2}$	2.84e-03	2.00	7.10e-04	2.00	1.77e-04
	$\|\|u^h-u\|\|_{H^1}$	5.45e-01	1.00	2.73e-01	1.01	1.36e-01
	预处理耗时 (s)	0.16		1.3		11
	求解耗时 (s)	0.0028		0.026		0.31
	CG 迭代次数	4		4		4

表 2: $h = \frac{1}{32}, \frac{1}{64}, \frac{1}{128}$ 的求解结果