摘要

本次编程作业实现了多重网格求解线性方程组 $Ax = b$，其中 A 是一维 Poission 方程或二维 Poission 方程离散的矩阵。

1 程序设计原理和思路

1.1 限制算子

一维时，限制算子应该是 $\mathbb{R}^{n+1} \to \mathbb{R}^{\frac{n}{2}+1}$ 的一个线性算子；二维时，限制算子是 $\mathbb{R}^{(n+1)(n+1)} \to \mathbb{R}^{(\frac{n}{2}+1)(\frac{n}{2}+1)}$ 的一个线性算子。
设向量 $v = (v_0, v_1, ..., v_n)$，则

1.1.1 injection 算子

- 对于一维的情况，按照讲义给出的方法，定义 $v_j^{2h} = v_{2j}^h, j = 0, 1, ..., \frac{n}{2}$。
- 对于二维的情况，仍然是将细网格的值限制在粗网格上，此时用指标 (i, j) 来代表行列的编号，有 $v_{(i,j)}^{2h} = v_{2i,2j}^h$，这里 $j = 0, 1, ...\frac{n}{2}, \quad i = 0, 1, ..., \frac{n}{2}$

1.1.2 full-weighting 算子

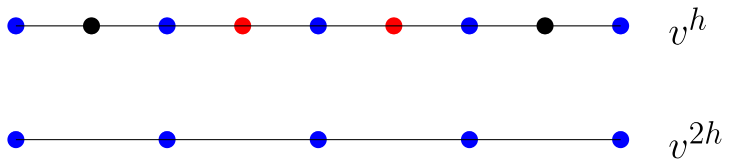
- 对于一维的情况，按照讲义给出的方法，定义 $v_j^{2h} = \frac{1}{4}v_{2j-1}^h + \frac{1}{2}v_{2j}^h + \frac{1}{4}v_{2j+1}^h, \quad j = 1, ..., \frac{n}{2} - 1$。对于端点上的值，仍然直接使用 injection 的方法。
- 对于二维的情况，使用 krocker 积，此时将权重看成一个向量 $v = \left(\frac{1}{4}, \frac{1}{2}, \frac{1}{4}\right)$，那么此时用 9 个点平均粗网格上的值，系数矩阵应该为：$v^T \otimes v$，即
$v_{(i,j)}^{2h} = \frac{1}{16}v_{(2i-1,2j-1)}^h + \frac{1}{8}v_{(2i,2j-1)}^h + \frac{1}{16}v_{(2i+1,2j-1)}^h + \frac{1}{8}v_{(2i-1,2j)}^h + \frac{1}{4}v_{(2i,2j)}^h + \frac{1}{8}v_{(2i+1,2j)}^h + \frac{1}{16}v_{(2i-1,2j+2)}^h + \frac{1}{8}v_{(2i,2j+1)}^h + \frac{1}{16}v_{(2i+1,2j+1)}^h$

1.2 插值算子

一维时，插值算子应该是 $\mathbb{R}^{\frac{n}{2}+1} \to \mathbb{R}^{n+1}$ 的一个线性算子；二维时，插值算子是 $\mathbb{R}^{(\frac{n}{2}+1)(\frac{n}{2}+1)} \to \mathbb{R}^{(n+1)(n+1)}$ 的一个线性算子。
设向量 $v = (v_0, v_1, ..., v_{n-1})$。

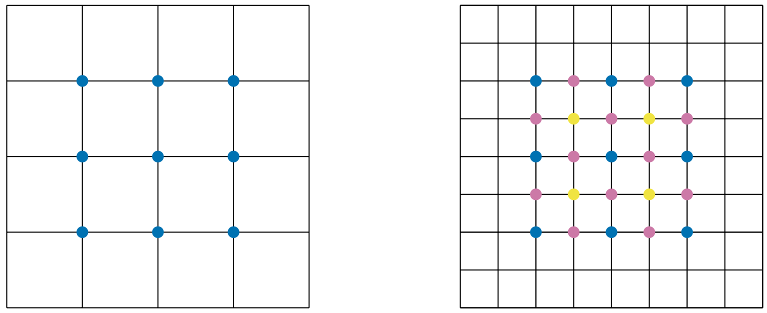
1.2.1 线性插值

- 一维时线性插值按照讲义上的公式, 对于细网格边界旁边的点（下图中黑色的点）有：



对于粗网格上的点，直接保留
$v_{2j}^{2h} = v_j^h \quad j = 0, 1, ..., \frac{n}{2}$ 对于内部的细网格点，有
$v_{2j+1}^h = \frac{1}{2}v_j^{2h} + \frac{1}{2}v_{j+1}^{2h} \quad j = 0, 1, 2, ..., \frac{n}{2} - 1$
$v_{2j-1}^h = v_{j-1}^{2h} \quad j = 1, 2, ..., \frac{n}{2} - 1$

- 对于二维的情况，我们的点可以分成下面的几类：



参考 William L.Briggs 的 A Multigrid Tutoral，有下面的插值格式

$$v^h_{2i,2j} = v^{2h}_{ij},$$

$$v^h_{2i+1,2j} = \frac{1}{2}(v^{2h}_{ij} + v^{2h}_{i+1,j}),$$

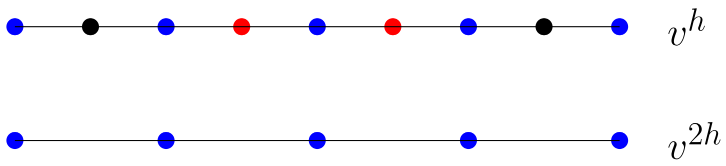
$$v^h_{2i,2j+1} = \frac{1}{2}(v^{2h}_{ij} + v^{2h}_{i,j+1}),$$

$$v^h_{2i+1,2j+1} = \frac{1}{4}(v^{2h}_{ij} + v^{2h}_{i+1,j} + v^{2h}_{i,j+1} + v^{2h}_{i+1,j+1}), \quad 0 \le i,j \le \frac{n}{2} - 1.$$

对于蓝色的粗网格上的点，我们直接用粗网格的值；对于紫色的点，用一维的插值去平均；对于黄色的点，用周围四个蓝色的点去平均；

1.2.2 二次插值

- 对于一维的情况，



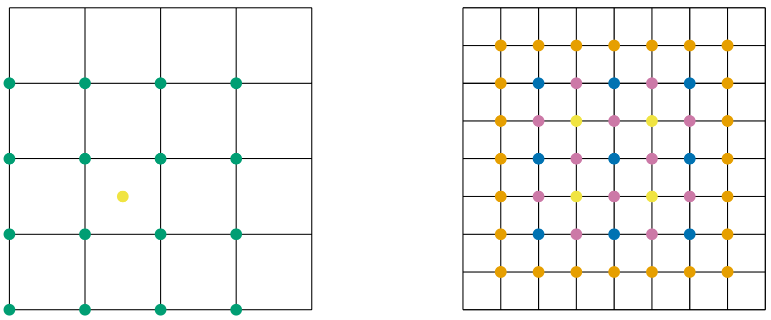
对黑色的点用三个点插值，红色的点用四个点插值，可以得到：

$$v^h_0 = \frac{3}{8}v^{2h}_0 + \frac{6}{8}v^{2h}_1 - \frac{1}{8}v^{2h}_2 \quad v^h_n = \frac{6}{8}v^{2h}_{\frac{n}{2}-2} + \frac{6}{8}v^{2h}_{\frac{n}{2}-1} - \frac{1}{8}v^{2h}_{\frac{n}{2}}$$

$$v^h_{2j} = v^{2h}_j \quad j = 0, 1, ..., \frac{n}{2} - 2$$

$$v^h_{2j+1} = \frac{9}{16}v^{2h}_j + \frac{9}{16}v^{2h}_{j+1} - \frac{1}{16}v^{2h}_{j+2} - \frac{1}{16}v^{2h}_{j-1}$$

- 对于二维的情况：



对于蓝色的点，直接使用粗网格上的值即可；对于紫色的点，使用一维的二次插值；对于橙色的点，令 $v = v = \left(\frac{-1}{16}, \frac{9}{16}, \frac{9}{16}, \frac{-1}{16}\right)$，$u = \left(\frac{3}{8}, \frac{6}{8}, \frac{-1}{8}\right)$，则插值格式应为 $v^T \otimes u$；对于右图中黄色的点，由于一维时的插值格式是 $v = \left(\frac{-1}{16}, \frac{9}{16}, \frac{9}{16}, \frac{-1}{16}\right)$，相应的选择二维的插值格式是 $v^T \otimes v$ 用左图中的 16 个点插值，

2 拉普拉斯算子的离散

- **Dirichlet 边界**：直接离散。
- **Neumann 边界**：在边界上利用 ghost cell 逼近导数，然后在边界上求离散拉普拉斯算子。此时离散出来的矩阵不可逆，在做 w_Jacobi 迭代时，每次都取与零空间正交的解，最后在加上相差的常数。
- **Mixed 边界条件**：二维时为了避免正方形四个角上定义冲突，直接给出解在四个角上的值。

3 粗网格上的求解和 Relaxation 中权重的选择

3.1 Relaxation 中权重的选择

- **一维情况**：对于 Dirichlet 边界，离散得到的矩阵对称正定，此时特征值设为 $\lambda_k, \quad k = 1, 2, ..., n+1$，其中 $k_1 = k_2 = 2$，其余的特征值课本已经给出，选择松弛因子 $\omega = \frac{2}{3}$。对于其他边界条件，离散出来的矩阵不对称，但是仍然选择 $\omega = \frac{2}{3}$ 作为松弛因子。
- **二维情况**：选择 $\omega = \frac{4}{5}$ 作为松弛因子。

4 结果分析

所有的结果都保存在 output 文件夹中，在 `json` 文件中，我输出了误差向量的 1-范数、2-范数、无穷范数和迭代次数，在 `csv` 文件中我输出了每次求解的运行时间。

设置当迭代次数大于用户给出的最大迭代次数或者当 $\|x_{k+1} - x_k\|_2 < \epsilon$ 时，停止迭代。这里 x_{k+1} 表示第 $k+1$ 次迭代得到的解。

可以发现一次插值的稳定性比二次插值要好。当网格很大时，二次插值对 Neumann 边值问题得到的解发散。对 Mixed 边界条件，得到的解误差也不如 Dirichlet 问题精确。且当网格为 256 时，求解速度明显很慢，已经接近 $3000ms$。