HIT Beamer Theme

多重网格作业

ZY-HIT

2021 年 2 月 21 日

1. 离散格式回顾
2. 多重网格简介
3. 求解器实现

1 离散格式回顾

2 多重网格简介

3 求解器实现

离散格式

- 二维情况下二阶导中心差分离散：

$$\left(\frac{d^2\phi}{dx^2}\right)\Big|_{x=O} = \frac{2\phi_E}{\Delta x_E(\Delta x_E+\Delta x_W)} - \frac{2\phi_O}{\Delta x_E \Delta x_W} + \frac{2\phi_W}{\Delta x_W(\Delta x_E+\Delta x_W)} + \frac{2\phi_N}{\Delta x_N(\Delta x_N+\Delta x_S)} - \frac{2\phi_O}{\Delta x_N \Delta x_S} + \frac{2\phi_S}{\Delta x_S(\Delta x_N+\Delta x_S)} \tag{1}$$

- 使用 Gauss-Seidel 迭代方法

$$\phi_O^{n+1}\left(\frac{2}{\Delta x_E \Delta x_W}+\frac{2}{\Delta x_N \Delta x_S}\right) = S_R - \phi_W^{n+1}\frac{2}{\Delta x_W(\Delta x_E+\Delta x_W)} - \phi_S^{n+1}\frac{2}{\Delta x_S(\Delta x_N+\Delta x_S)} - \phi_N^{n}\frac{2}{\Delta x_N(\Delta x_N+\Delta x_S)} - \phi_E^{n}\frac{2}{\Delta x_E(\Delta x_E+\Delta x_W)} \tag{2}$$

1 离散格式回顾

2 多重网格简介

3 求解器实现

多重网格理论模型

- 求解如下线性代数方程：

$$\boldsymbol{A}\cdot\boldsymbol{x}=\boldsymbol{B} \tag{3}$$

其中，数组 $\boldsymbol{x}$ 是系统的真实解。

- 如果我们采用迭代法求解该方程组，定义残差向量 $\boldsymbol{r}$

$$\boldsymbol{A}\cdot\boldsymbol{y}=\boldsymbol{B}-\boldsymbol{r} \tag{4}$$

- 定义误差向量，表征真实解与数值解的差值

$$\boldsymbol{e}=\boldsymbol{x}-\boldsymbol{y} \tag{5}$$

- (式. (3) - 式. (4))，可得

$$\boldsymbol{A}\cdot\boldsymbol{x}-\boldsymbol{A}\cdot\boldsymbol{y}=\boldsymbol{A}\cdot\boldsymbol{x}-\boldsymbol{y}=\boldsymbol{A}\cdot\boldsymbol{e}=\boldsymbol{B}-(\boldsymbol{B}-\boldsymbol{r})=\boldsymbol{r} \tag{6}$$

即 $\boldsymbol{A}\cdot\boldsymbol{x}-\boldsymbol{A}\cdot\boldsymbol{y}=\boldsymbol{r}$

多重网格理论模型

- 残差向量可以通过迭代求解模型式. (4) 获得。在此，想象利用迭代模型求解线性方程组式. (6) 并获得误差向量，此求解器可写为如下形式

$$\boldsymbol{e}^{(k)} = \boldsymbol{T} \cdot \boldsymbol{e}^{(k-1)} + \boldsymbol{c} \tag{7a}$$

由于系数矩阵 $\boldsymbol{A}$ 在式. (3) 和式. (6) 中是相同的，因此系数矩阵 $\boldsymbol{T}$ 也与所选择的迭代模型系数矩阵相同。但是另一个常数向量 $\boldsymbol{c}$ 是不同的，

$$c_i = \frac{r_i}{a_{ii}} \tag{7b}$$

- 多重网格方法旨在利用不同网格下误差的衰减特性来加速收敛，在最细网格上，短波误差得到有效的减小；而在最粗网格上，长波误差迅速减小。此外，在细网格上迭代的计算量要比在粗网格上迭代的计算量大，因此在粗网格上迭代所产生的额外计算消耗将被多重网格方法所提高的收敛速度所抵消。

多重网格算法

- 在细网格迭代 (Fine grid iterations)(前光顺，pre-smoothing)

  在细网格上迭代得到 $\boldsymbol{y}^h$。

$$\boldsymbol{A}^h \cdot \boldsymbol{x} = \boldsymbol{b} \tag{8}$$

  选择合适的迭代次数，可以有效地减小误差的短波震荡分量，但是并不能消除长波分量。

  残差向量

$$\boldsymbol{r}^h = \boldsymbol{b} - \boldsymbol{A}^h \cdot \boldsymbol{y}^h \tag{9}$$

  误差向量

$$\boldsymbol{e}^h = \boldsymbol{x} - \boldsymbol{y}^h \tag{10}$$

$$\boldsymbol{A}^h \cdot \boldsymbol{e}^h = r^h \tag{11}$$

多重网格算法

- 限制 (Restriction)

  求解过程从细网格 h 转移到粗网格 $ch, (c > 1)$ 上。由于网格间距变粗，原先的一部分长波误差以短波的形式在粗网格中表征，这部分误差可以迅速收敛。

  求解粗网格上的误差向量

$$\boldsymbol{A}^{ch} \cdot \boldsymbol{e}^{ch} = \boldsymbol{r}^{ch} \tag{12}$$

  其中，粗网格上的残差向量 $\boldsymbol{r}^{ch}$ 可以通过某种数学上的平均格式计算得到。系数矩阵 $\boldsymbol{A}^{ch}$ 可以利用差分格式重新计算，或者采用某种数学平均格式来计算。

  在粗网格上的迭代开销是比较小的，因此可以采用合适的迭代次数来获得 $\boldsymbol{e}^{ch}$ 的收敛解。

多重网格算法

- 延拓 (Prolongation)

  在粗网格上计算得到的点相比于细网格是少的，我们需要采用一个合适的插值格式来获得细网格上的误差向量 $\boldsymbol{e}'^h$。

- 校正与后光顺 (Correction and final iterations)

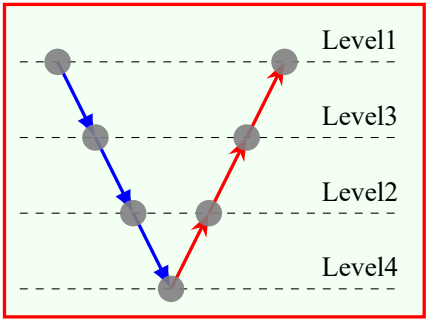
  当我们获得了细网格上的误差向量 $\boldsymbol{e}'^h$，我们可以校正细网格上的解，

$$\boldsymbol{y}^{improved} = y^h + \boldsymbol{e}'^h \tag{13}$$

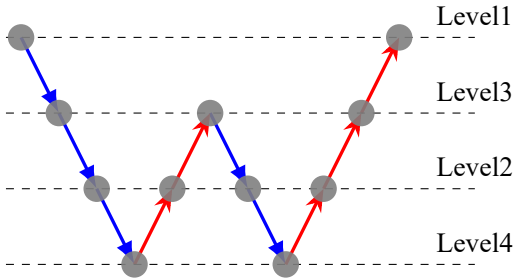
  由于此时长波误差被消除，这个改进（校正）的解更接近于真实解。但是在延拓和限制的过程中引入了误差，所以需要更多的迭代次数用来抹平引入的任何误差。

- Illustration of different multigrid cycle strategies:

V-cycle



W-cycle



1 离散格式回顾

2 多重网格简介

3 求解器实现

题目

偏微分方程

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = 50000 \left[100 \left((1-x)^2 + y^2\right) - 2\right] e^{-50\left((1-x)^2+y^2\right)} \tag{14}$$

作用域为 $[0,1] \cup [0,1]$，解析解为

$$\phi(x,y) = 500e^{-50\left((1-x)^2+y^2\right)} + 100x(1-y) \tag{15}$$

使用二阶中央差分格式，编程求解上述问题，使用 GMG(几何多重网格)方法。仅使用两套网格 81×81 和 41×41。在细网格使用 One-sweep Gauss-Seidel。对于粗网格光顺，使用两种不同方法:

(a)GaussSeidel 光顺，扫略从 1 一次增加到 5 次。
(b) 使用直接求解器 (高斯消元)。画出残差图。

求解器策略

- (1) 使用 Gauss Seidel 迭代方法，现在细网格上迭代一定次数，根据系数矩阵关系求出残差。
- (2) 将残差转移到粗网格上，可以采用直接赋值或者其他加权方法。
  * 利用残差与误差的关系再进行迭代求解，此时系数矩阵的构造需要特别注意，采用不同的系数矩阵构造方法会对编程实现难易程度产生较大的影响。
- (3) 将上一步残差继续在更粗糙网格上迭代，即再进行 (2)(3)。
- (4) 将求解得到的误差向量从粗网格延拓到细网格，一般采用双线性插值。
- (5) 误差向量与原先的解相加，对每一次残差迭代结果进行一次校正。
- (6) 当过程进行到原始细网格级别，校正过程即误差与解相加，对之前的结果进行校正。

求解器实现 (限制)

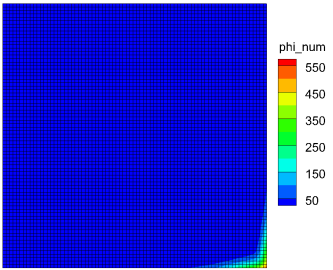
```
1  // ****************** 计算残差 rh = Bh - Ah * Xh
2  residual(rh0, N0, Ah0_coeff, Xh0, Bh0);
3  // ****************** 限制，将细网格残差转移到粗网格
4  Restriction(rh1, rh0, N1, N0);
5  // ****************** 重新构造迭代矩阵，使其满足原来迭代形式
6  get_Right_term_resi(N1 + 2, &Xh1_delta[0], &Yh1_delta[0], rh1, Bh1);
7
8  // ****************** 残差迭代获得误差
9  for (int steph0 = 0; steph0 < 10; ++steph0)
10 {
11     Gauss_Seidel(eh1, Ah1_coeff, rh1, N1);
12 }
```

求解器实现 (延拓)

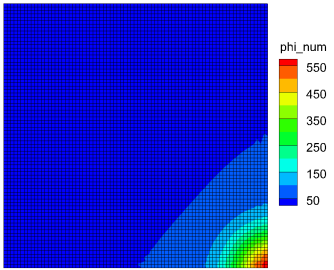
```
1
2  // ****************** 误差向量延拓，双线性插值，粗网格 --> 细网格
3  Prolongation(eh3, rh2, N3, N2, X23_weight, Y23_weight);
4
5  // ****************** 误差校正
6  eh2 += rh2;
7
8  // ****************** 校正后的误差再迭代 d
9  for (int steph0 = 0; steph0 < 5; ++steph0)
10 {
11     Gauss_Seidel(eh2, Ah2_coeff, Bh2, N2);
12 }
```

均匀网格

- 只进行 5 此 GS 迭代，然后将 5 次迭代后的残差做一次 V-cycle，结果对比如下



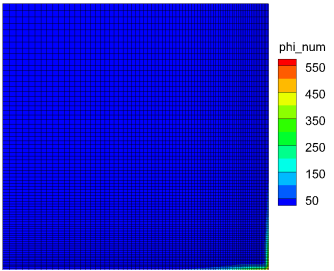
(a) 直接 GS 迭代



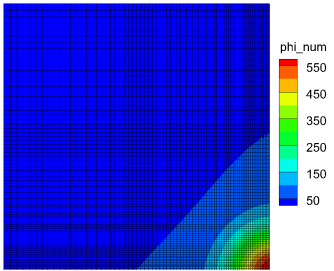
(b) 经过一次 V-cycle

非均匀网格

- 只进行 5 此 GS 迭代，然后将 5 次迭代后的残差做一次 V-cycle，结果对比如下

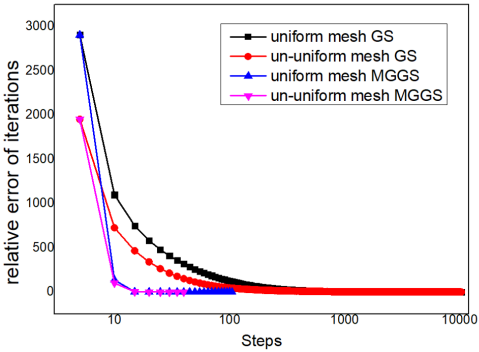


(c) 直接 GS 迭代



(d) 经过一次 V-cycle

- 给出 MultiGrid 迭代和非 MultiGrid 的 GS 迭代的收敛结果，该 V-cycle 的 MG 方法对于收敛速度有着指数级别的提升，大概提升了两个数量级。
- 虽然每次 V-cycle 会带来一些包括限制、延拓、以及粗网格迭代的开销，但是迭代次数的明显下降足以抵消这些开销。



小结

- 受限于个人能力和时间，本作业实现了一个非常 Low 的 Poisson 方程全第一类边界条件的 MG 求解器，只采用了最简单的 V-cycle。
- MultiGrid Gauss Seidel 迭代求解器相比于原始 Gauss Seidel 求解，多了许多额外步骤，比如残差计算、限制、延拓，甚至还需要根据残差的性质再构建一个迭代方程组。
- 相比于原始 GS 迭代，这是一个更复杂的系统，出错的概率大大增加，保证粗细网格能够一一对应，限制操作和延拓操作都保证正确性是算法得以实现的关键，这也是最头疼的地方。
- 算法实现复杂度大大增加，迭代次数可以显著减少（好的东西来的都不容易）。

Thank Dr. Wang Yang!