Rainmaker's Notebook
『求雨巫师的神奇之处在于他总是躲着不见你，却总说刚下完的雨是拜他所赐。』——《天真的人类学家》

Home
Archives
About
SiteXC

泊松方程的中心差分格式与多重网格法

这个学期上 MATH 6644 接触了多重网格（Multigrid, MG）法，觉得这是一个很有意思的方法。课程作业和期末项目的选项之一就是用 MG 加速泊松方程中心差分格式的求解。虽然我期末项目选了另一个题目，然而我也写了一下这个。这篇东西算是一点小的总结。

方程组的导出

为方便起见，1D, 2D 3D 下我们的求解范围分别是单位直线、单位正方形和单位立方体。Poisson 方程为 $-\Delta u(\vec{x}) = f(\vec{x})$, 在 1D, 2D, 3D 下分别是：

$$-\frac{\partial^2}{\partial x^2}u(x) = f(x)$$

$$-(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2})u(x,y) = f(x,y)$$

$$-(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2})u(x,y,z) = f(x,y,z)$$

我们使用均等网格来划分求解域。为方便起见，每个维度的网格数相同，均为 $n+1$，则有 $\Delta x = \Delta y = \Delta z = h = 1/n$. 中心差分格式表示的二阶导数为：$\partial^2 u/\partial x^2 = [u(x+\Delta x) - 2u(x) + u(x-\Delta x)]/\Delta x^2$. 为方便，将 $u(i\Delta x)$ 简记为 U_i, 其他维度同理增加下标。因此，我们得到下面的方程组 (1)：

$$2U_i - U_{i-1} - U_{i+1} = -h^2 f_i$$

$$4U_{i,j} - U_{i-1,j} - U_{i+1,j} - U_{i,j+1} = -h^2 f_{i,j}$$

$$6U_{i,j,k} - U_{i-1,j,k} - U_{i+1,j,k} - U_{i,j-1,k} - U_{i,j+1,k} - U_{i,j,k-1} - U_{i,j,k+1} = -h^2 f_{i,j,k}$$

假设我们使用狄利克雷边界条件，则我们有 $(n-1)^d$ 个待求解的格点，$d = 1, 2, 3$ 为问题维度。

形成方程组的时候，最外层的待求解格点的方程有一些变化，因为它们有一些相邻格点是由边界条件给出了值的边界格点，因此需要把已知量从方程等号左侧移动到右侧。由方程形式的对称性，我们很容易

Rainmaker's Notebook

『求雨巫师的神奇之处在于他总是躲着不见你，却总说刚下完的雨是拜他所赐。』——《天真的人类学家》

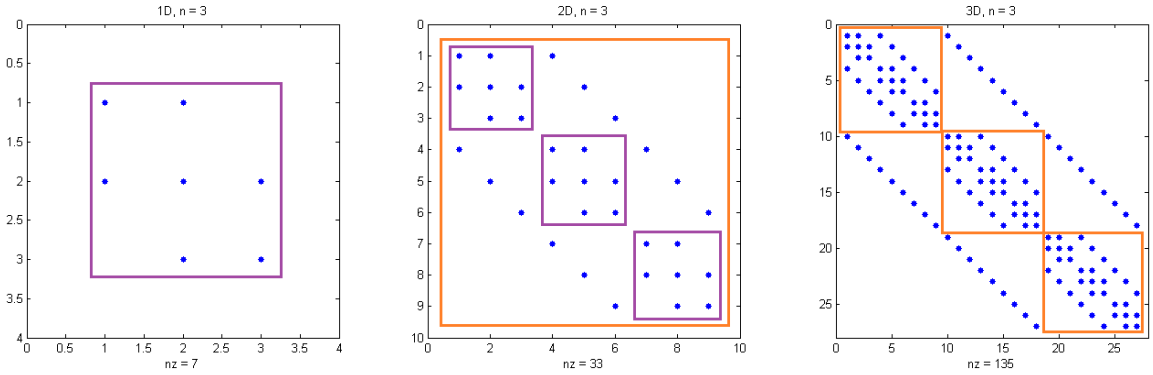
Home

Archives

About

SiteXC

知道形成的方程组系数矩阵也是对称的，并且是半正定的。下面这张图是对于 $n-1=3$， 1D, 2D, 3D 下的系数矩阵非零元分布图以及它们之间的关系：



很容易可以看出，2D 系数矩阵里面嵌套 1D 系数矩阵，3D 系数矩阵里面嵌套了 2D 系数矩阵，只是主对角线的元素值变了。

多重网格法简介

古典迭代法（Jacobi, Gauss-Seidel, SOR）在求解上述形成的方程组的时候收敛速度很慢。然而，它们有一个优点，即能快速消除误差中的高频分量（emmm 其实我现在还没搞懂什么这一句话）。多重网格法的思想是在一个细网格上用迭代法消去误差高频分量，然后将残差（residual，定义为$b-Ax$）限制投影到粗网格上，使得细网格上相对低频的误差分量在粗网格上变成高频误差分量并被消去。粗网格上求解得到的修正值再插值投影到细网格上，对原来的解进行修正。由于这一投影会重新引入高频误差，所以还需要再上用迭代法消去误差高频分量。

这里有一个问题：如何构造粗网格。一般来说，粗网格在每个维度上的格点数量都是细网格的一半。也就是说，如果某个维度有 n 个格点，那么粗网格上的同一个维度应该有 $\lfloor \frac{n}{2} \rfloor$ 个格点。同时，粗网格上的格点在细网格上对应的位置，互相之间隔着一个细网格的格点。因此，我们要求 n 是奇数，这样对于这一个维度，粗网格上格点的位置分别是原来细网格上的第 $2, 4, \cdots, n-1$ 个格点的位置。

二重网格已被证明在理论上是收敛的，并且收敛速度与网格尺度无关。很自然地，我们可以使用更多重的网格，直到最粗的网格只有很少数量的未知数，可以使用直接法进行求解。多重网格的网格变换路径一般有如下几种：

1. V-cycle: 从最细的网格开始不断向粗网格变换，到规定的最粗的网格以后，一层层返回到最细的网格。

Rainmaker's Notebook
『求雨巫师的神奇之处在于他总是躲着不见你，却总说刚下完的雨是拜他所赐。』——《天真的人类学家》

Home

Archives

About

SiteXC

2. W-cycle: 从最细的网格开始不断向粗网格变换，到规定的最粗的网格以后，往上返回若干层，然后再次向粗网格变换，再逐层返回到最细的网格。
3. Full Multigrid (FMG): 从最粗的网格开始，向细网格变换，再向粗网格变换，再向更细的网格变换，重复直到达到最细的目标网格。

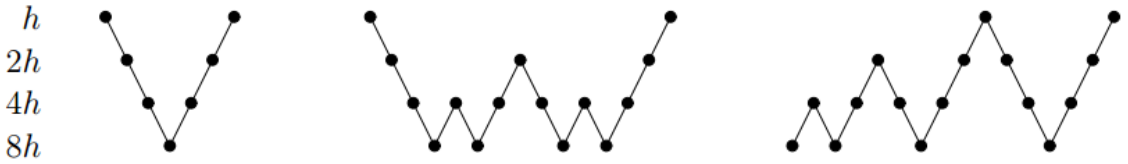


Figure 6.11: V-cycles and W-cycles and FMG use several grids several times.

我学的和用的是最简单的 V-cycle。一个 V-cycle 主要有以下几个部分：

1. 如果待求解系统足够小，直接求解并返回结果；
2. Pre-smoothing: 前光滑处理，用迭代法对给定的初始解 x 进行迭代，得到 $x^{'}$；
3. 限制投影：构造 restriction operator R，将残差 $r = b - Ax^{'}$ 投影到粗网格上：$r^H = Rr$;
4. 构造粗网格系数矩阵 A^H, 并递归调用 V-cycle 求解 $A^H e^H = r^H$, 其中 e^H 的初始解为 0;
5. 插值投影：将求得的 e^H 投影到细网格上并修正 $x^{'}$: $x^{''} = x^{'} + Pe^H$;
6. Post-smoothing: 后光滑处理，用迭代法对 $x^{''}$ 进行迭代，得到 $\bar{x}$, 作为返回的结果。

这里引入了两个问题：如何构造 restriction operator R 和如何构造粗网格系数矩阵 A^H.

构造 Restriction Operator

R 有两种常见的构造方法。最简单的构造方法是粗网格上某个格点的值直接取细网格上对应格点的值，即 injection operator, 然而对应 P 需要以 full weighting operator 的形式构造。因此，一般直接用 full weighting operator 来构造 R. 在 1D 的时候，它的形式如下：
$r_i^H = 0.25r_{2i-1} + 0.5r_{2i} + 0.25r_{2i+1}$, 我们可以将其视作一个卷积核 $K_{1D} = [0.25, 0.5, 0.25]$. 与其对应的是 $P_{1D} = 2R_{1D}^T$. 在 2D 的时候，R 对应的 2D 卷积核是 1D 卷积核的 Kronecker 张量积：$K_{2D} = K_{1D} \otimes K_{1D}$, 即

$$K_{2D} = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

在 3D 的时候，R 对应的卷积核 K_{3D} 是这样的：

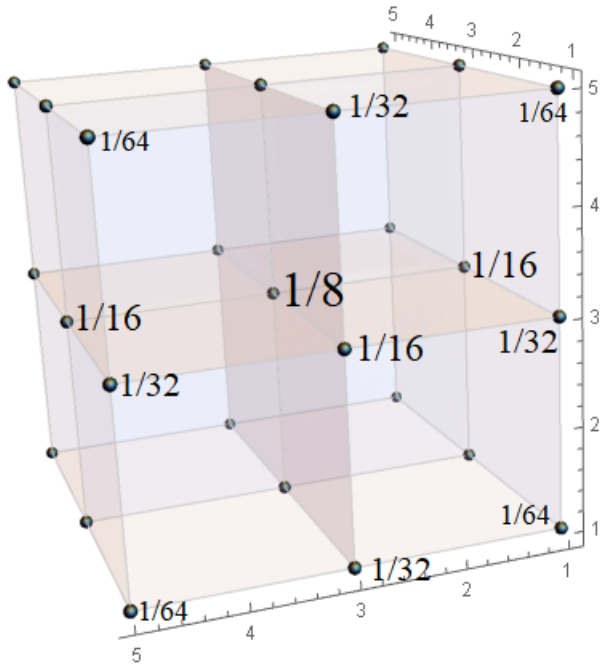
Rainmaker's Notebook
『求雨巫师的神奇之处在于他总是躲着不见你，却总说刚下完的雨是拜他所赐。』——《天真的人类学家》

Home

Archives

About

SiteXC



与其对应的 $P_{3D} = 8R_{3D}^T$. 实际上我们很容易发现，P_{1D}, P_{2D}, P_{3D} 之间也有张量积关系：$P_{2D} = P_{1D} \otimes P_{1D}$, $P_{3D} = P_{1D} \otimes P_{1D} \otimes P_{1D}$ （R_{1D}, R_{2D}, R_{3D} 同理）。

构造粗网格系数矩阵

构造粗网格系数矩阵 A^H 也有两种方法。第一种方法 rediscretization on coarse grid 最简单直接，即直接在粗网格上根据 (1) 导出系数矩阵。我使用的是第二种方法，Galerkin projection on coarse grid，即 $A^H = RAP$. 这种方法有时候收敛需要的 V-cycle 数量会比第一种方法要少一些，然而需要计算稀疏矩阵-矩阵乘法。

构造 restriction operator 和粗网格系数矩阵都只需要构造一次，因为在 V-cycle 中这些矩阵是不变的。对于 pre-smoother 和 post-smoother，实验表明 Gauss-Seidel 的效果比 Jacobi 要好。

测试

我的程序在这里。 为了方便，方程右端项用的是随机数，没有按什么条件去构造。程序使用 0 向量作为 x 的初识猜测解，在 relative residual norm $|b - Ax_k|_2 / |b|_2$ 小于等于 10^{10} 的时候停止。程序默认使用 Gauss-Seidel 作为 smoother。我写的 Gauss-Seidel 是原始实现，没有进行染色划分和并行，所以跑得比较慢。

Rainmaker's Notebook

『求雨巫师的神奇之处在于他总是躲着不见你，却总说刚下完的雨是拜他所赐。』——《天真的人类学家》

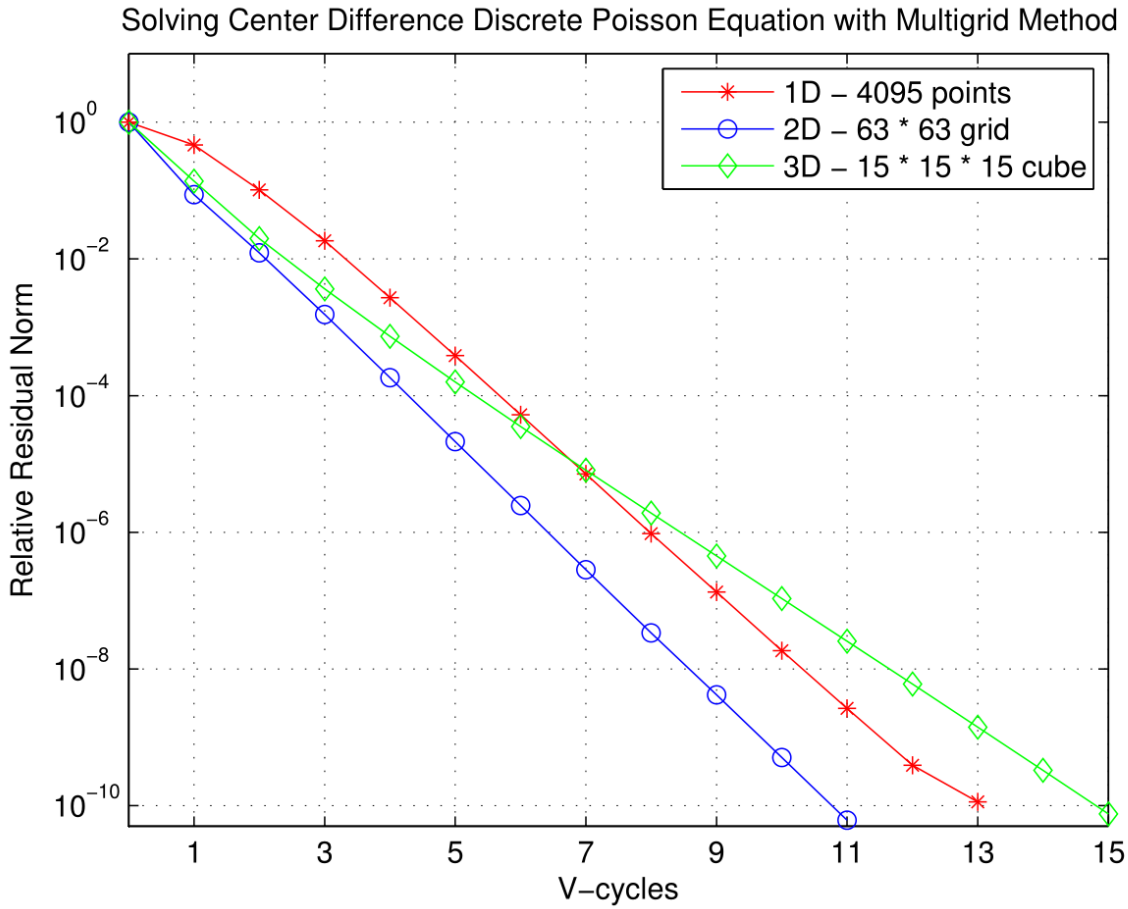
Home

Archives

About

SiteXC

我在 1D, 2D, 3D 下各测了几组数据。测试表明，对相同维度问题的不同网格大小，迭代收敛需要的 V-cycle 数基本上是不变的。下面是某几组数据的测试结果。作为对比，我使用了没有 preconditioner 的 CG 来求解同样的方程组。



Rainmaker's Notebook

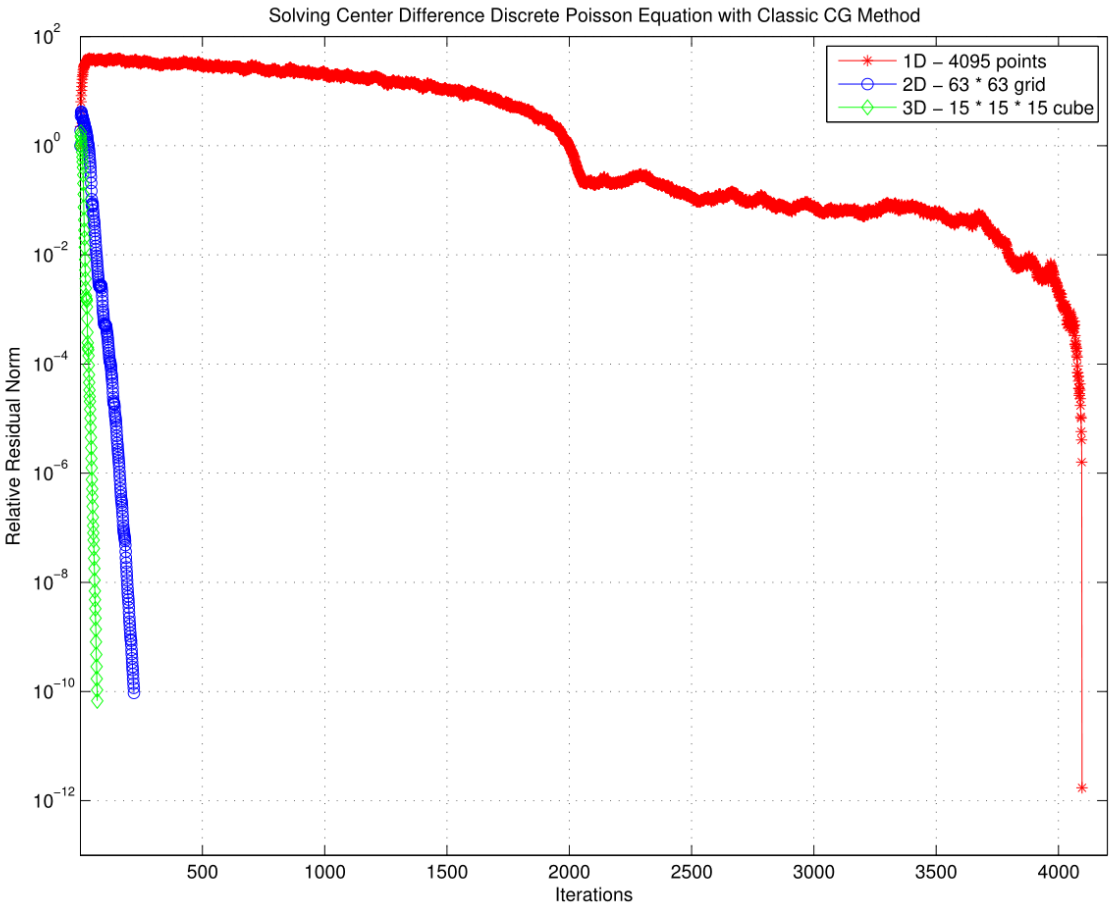
『求雨巫师的神奇之处在于他总是躲着不见你，却总说刚下完的雨是拜他所赐。』——《天真的人类学家》

Home

Archives

About

SiteXC



这三组数据只能纵向对比，因为横向对比的话虽然系数矩阵的大小接近但是条件数差了太多。可以看出，Multigrid 的收敛速度相当快。

发布于 2017-12-01

tags: { Multigrid } { LinearAlgebra } { PDE }

Related Issues not found

Please contact @EnigmaHuang to initialize the comment

Login with GitHub