**偏微分方程现代数值解法  
数值实验报告 (第二次/HW12)**

匡亚明学院 211240021 田铭扬

**一 问题**

利用完全多重网格方法求解如下椭圆问题的线性有限元离散:  
探究不同的迭代次数l的选取对完全多重网格法所求近似解的精度的影响（与直接法的解或与V循环多重网格解比较）。

**二 程序设计**

讲义中给出的代码已经足够详细，故不再详述。但针对本文所探究的特殊问题，在程序设计时，有以下内容是需要留意的。

首先，关于计算区域的生成，由于系数函数a不连续性，笔者希望其不连续界面恰好是单元的边界——即a在同一个单元中保持连续，考虑到装配单元刚度矩阵的方式，笔者认为这样是合理的。因此，笔者是将4个边长为1的正方形组合成为计算区域，在使用initmesh指令生成初始网格后，再“手动”将多余的边界（即各正方形边界的交集）删去。

其次，笔者使用refinemesh指令进行网格加密，在数据储存时，新的结点会被“追加”在原有节点的后面。故而可以通过枚举各单元的3条边、并判断各边的2个顶点是旧有还是新增节点，由此构造延拓矩阵。由于涉及到对同一个矩阵的写入操作（其实可以改造为“max”型的归约），Matlab无法自动并行化，因此需要根据本机的CPU核心数量，使用人为技巧实现。

最后，为了保证实验结果的可比较性，笔者使用完全多重网格法结果的残差（下降率）作为V循环方法的残差限。且当完全网格法的迭代次数l取不同的值时，会分别进行一次V循环算法的求解，进而可以比较2种算法在不同精度要求下的表现。

**三 实验结果与分析**

实验结果如下表所示。

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **加密 次数** | **计算用时(s)** | | | | | | |
| **直接法** | **完全网格 l=10** | **V循环 同等TOL** | **完全网格 l=20** | **V循环 同等TOL** | **完全网格 l=30** | **V循环 同等TOL** |
| **2** | 2.22e-4 | 2.52e-3 | 2.52e-3 | 4.86e-3 | 7.08e-3 | 6.93e-3 | 3.26e-3 |
| **3** | 7.64e-4 | 7.23e-3 | 7.93e-3 | 1.62e-2 | 1.06e-2 | 2.18e-2 | 1.06e-2 |
| **4** | 3.30e-3 | 2.33e-2 | 2.76e-2 | 4.72e-2 | 4.28e-2 | 7.33e-2 | 4.19e-2 |
| **5** | 1.49e-2 | 9.51e-2 | 0.127 | 0.195 | 0.205 | 0.302 | 0.214 |
| **6** | 7.86e-2 | 0.401 | 0.484 | 0.782 | 0.876 | 1.19 | 0.969 |
| **7** | 0.385 | 1.81 | 2.89 | 3.93 | 4.39 | 5.72 | 7.39 |

表1.完全多重网格法与V循环算法 计算用时比较

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **加密次数** | **计算结果与直接法的差距 L∞** | | | | | |
| **完全网格 l=10** | **V循环 同等TOL** | **完全网格 l=20** | **V循环 同等TOL** | **完全网格 l=30** | **V循环 同等TOL** |
| **2** | 1.35e-13 | 8.97e-14 | 2.78e-17 | 3.99e-17 | 3.47e-17 | 3.47e-17 |
| **3** | 6.31e-14 | 4.21e-14 | 2.91e-16 | 3.05e-15 | 2.91e-16 | 3.05e-16 |
| **4** | 7.54e-14 | 4.52e-14 | 3.19e-16 | 3.33e-16 | 2.50e-16 | 3.33e-16 |
| **5** | 4.75e-13 | 2.99e-13 | 6.52e-16 | 5.13e-16 | 7.08e-16 | 5.13e-16 |
| **6** | 7.67e-13 | 6.50e-13 | 3.80e-15 | 2.97e-15 | 2.25e-15 | 2.45e-15 |
| **7** | 7.96e-13 | 6.20e-13 | 1.16e-14 | 1.13e-14 | 9.80e-15 | 8.40e-15 |

表2.完全多重网格法与V循环算法 计算结果与直接法的差距比较

表1&2注释：1.标红说明舍入误差的累计已占据主导；

2.标蓝说明完全多重网格法的表现比V循环算法更好。

据表1，在达到同等残差的情况下，有以下两个现象：

首先，完全多重网格法的迭代次数l较少时，完全多重网格法相比V循环算法的时间优势更明显。事实上，完全多重网格法的耗时与l基本成正比；而随着l的增加，舍入误差的累积开始成为主导（参表2），残差的下降就越来越不明显。而笔者是根据完全多重网格法的结果来设置V循环算法的残差限，因而V循环算法终止所需的迭代次数不会显著增加（这也能在一定程度上说明，V循环算法对于舍入误差的累计更加“鲁棒”——其原因是显然的，因为完全多重网格法的初值继承自上一层的结果，同时也就继承了上一次运算的误差），故此时V循环算法的速度更快。**因此，在使用多重网格算法时，可以考虑根据残差变化情况自适应地选取l，甚或时先验地设置一个残差限。**

其次，网格越密（结点数越多），完全多重网格法的时间优势越明显。原因也是显然的，这是理论分析中log(nk) 之差距的体现。

**综上可得出结论，当网格结点数较多、要求精度不太高时，完全多重网格算法（相比V循环算法）在计算速度上表现更佳。**

由于笔者设置V循环算法残差限的方式，表2仅具有参考用途。

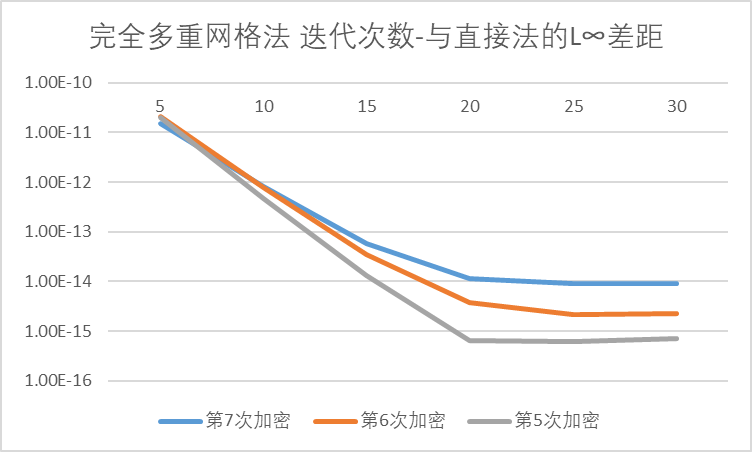
 将表格中的结果绘制成折线图，我们还能得到如下的结果：

图1.完全多重网格法与V循环算法 计算结果与直接法的差距比较

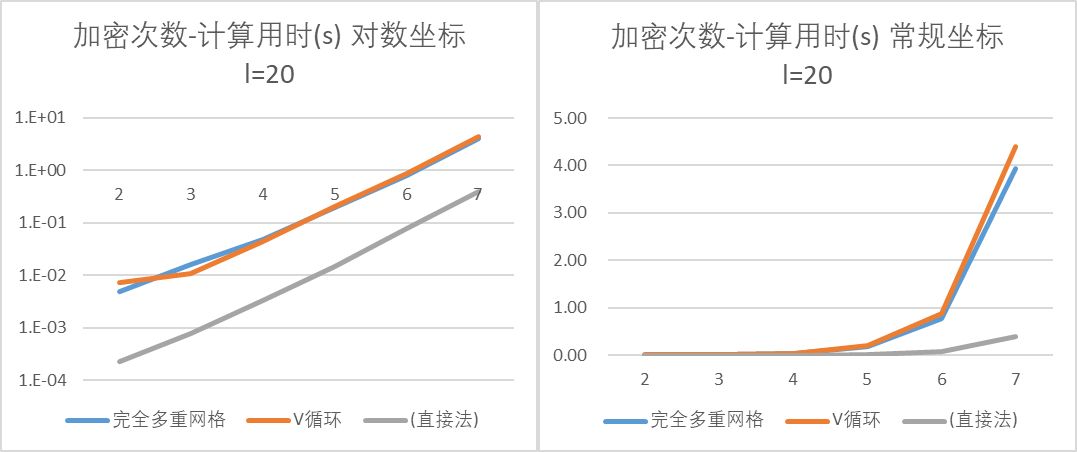
 首先，在舍入误差占据主导以前，完全多重网格法的误差下降是“指数级别”的（上图的纵坐标为对数坐标）。此外，虽然网格越密、舍入误差的影响就越明显，但是图线出现“拐点”的位置却是一致的（l=20），**因此猜测迭代次数的选取主要与问题本身有关，而与网格的疏密无关。**

图2.l=20 完全多重网格法、V循环算法与直接法 计算用时比较

其次，当迭代次数选取图1的“拐点”（l=20）时，据图2，完全多重网格法和V循环算法的用时处于同一数量级；而随着节点数增多，完全多重网格法逐渐体现出了优势。（另外，图像显示计算用时为指数增长，而节点数量随着网格加密也是指数增长的，说明算法的复杂度达到了最优。）

虽然笔者计算机性能较弱，无法在更密的网格中完成计算（主要瓶颈是延拓矩阵的构造），但是可以预测，网格更密时完全多重网格法的优势会更加明显。**而考虑到网格密度的要求主要与实际问题（在“物理世界”）的尺度有关，也可以说，完全多重网格法更适用于物理尺度较大的问题。**

此外，上图中的时间并没有包含构造延拓矩阵的用时——实际上，这反而消耗了大部分的计算时间（见下表）。因此，**两种方法相比直接法的优势是存疑的，或许需要在更加病态问题中才能够体现。**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **加密次数** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| **内部节点数** | **41** | **177** | **737** | **3009** | **12161** | **48897** | **196097** |
| **延拓矩阵 构造用时(s)** | **0.0352** | **0.0187** | **0.0182** | **0.0190** | **0.0508** | **0.432** | **14.8** |

表3.延拓矩阵构造用时

**四 结论**

综上所述，在本文所探究的问题中，完全多重网格法与V循环算法各有优劣，应当根据实际要求选择；此外，两者对于直接法均未表现出优势。

**附录：Matlab代码**

由于本次实验涉及的代码较多，故不在此处列出。如有需要，请参见随本文提交的“code.zip”压缩文件。谢谢理解！

以下列出各文件的用处：  
main.m 主程序 pdeasmpoi.m 刚度矩阵装配

mg\_prolangation.m 与 mg\_prolangation\_sub.m 延拓矩阵生成

mg\_vcycle.m V循环算法主程序 mgp\_vcycle.m V循环算法迭代子

fmg.m 多重网格算法 mgs\_gs G-S磨光

output.txt 原始输出