第五次作业报告 PB23000141 刘彦宏

编译环境

作业使用cmake组织项目, 编译器使用msvc1942. 开发使用的IDE是Clion, VS2022当然也可以正常构建并运行.
代码运行方式与之前的作业一致, 按先前作业的批改反馈此处不再赘述.

问题描述

根据课本给出的算法, 利用共轭梯度法求解线性方程组.

问题分析

因为只是根据课本算法直接求解线性方程组, 略掉问题分析.

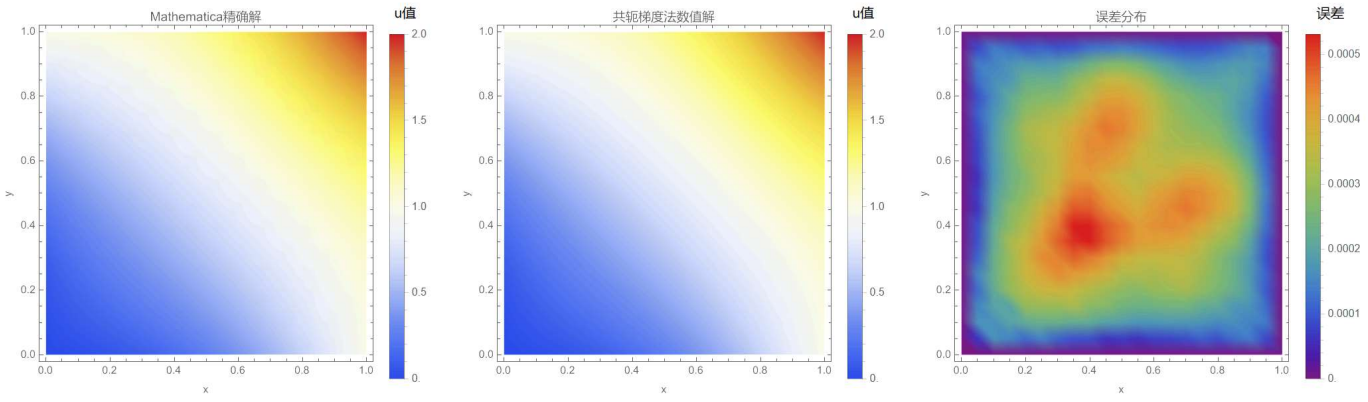
结果展示和讨论

第一题

```
D:\Code\N_A\cmake-build-debug-visual-studio\hw5\hw5_1.exe
37
2

0.021526,0.040414,0.061336,0.085231,0.112522,0.143630,0.178673,0.217781,0.261083,0.308569,0.360323,0.416294,0.476501,0.5
40871,0.609313,0.681727,0.757812,0.837258,0.919336,0.040414,0.068913,0.097397,0.127186,0.159083,0.193612,0.231127,0.2718
51,0.315971,0.363557,0.414674,0.469306,0.527418,0.588923,0.653648,0.721345,0.791606,0.863867,0.937147,0.061336,0.097397,
0.132332,0.167286,0.203338,0.241056,0.280867,0.323170,0.368099,0.415844,0.466435,0.519881,0.576126,0.635051,0.696452,0.7
59962,0.825153,0.891291,0.957491,0.085231,0.127186,0.167286,0.206706,0.246327,0.286853,0.328781,0.372488,0.418213,0.4661
23,0.516299,0.568719,0.623356,0.680034,0.738553,0.798505,0.859383,0.920530,0.981099,0.112522,0.159083,0.203338,0.246327,
0.288937,0.331817,0.375552,0.420485,0.466916,0.515034,0.564872,0.616491,0.669778,0.724600,0.780720,0.837719,0.895096,0.9
52227,1.008323,0.143630,0.193612,0.241056,0.286853,0.331817,0.376654,0.421823,0.467799,0.514833,0.563129,0.612754,0.6637
14,0.715946,0.769287,0.823476,0.878139,0.932763,0.986759,1.039450,0.178673,0.231127,0.280867,0.328781,0.375552,0.421823,
0.468131,0.514890,0.562429,0.610885,0.660362,0.710842,0.762279,0.814494,0.867252,0.920142,0.972698,1.024406,1.074591,0.2
17781,0.271851,0.323170,0.372488,0.420485,0.467799,0.514890,0.562258,0.610167,0.658751,0.708147,0.758293,0.809174,0.8606
23,0.912369,0.964067,1.015227,1.065358,1.113851,0.261083,0.315971,0.368099,0.418213,0.466916,0.514833,0.562429,0.610167,
0.658309,0.707005,0.756363,0.806339,0.856920,0.907917,0.959111,1.010118,1.060500,1.109770,1.157335,0.308569,0.363557,0.4
15844,0.466123,0.515034,0.563129,0.610885,0.658751,0.707005,0.755770,0.805171,0.855136,0.905673,0.956585,1.007643,1.0584
88,1.108679,1.157734,1.205060,0.360323,0.414674,0.466435,0.516299,0.564872,0.612754,0.660362,0.708147,0.756363,0.805171,
0.854656,0.904786,0.955526,1.006682,1.058031,1.109216,1.159805,1.209280,1.257071,0.416294,0.469306,0.519881,0.568719,0.6
16491,0.663714,0.710842,0.758293,0.806339,0.855136,0.904786,0.955242,1.006444,1.058192,1.110263,1.162286,1.213842,1.2644
01,1.313342,0.476501,0.527418,0.576126,0.623356,0.669778,0.715946,0.762279,0.809174,0.856920,0.905673,0.955526,1.006444,
1.058329,1.111005,1.164210,1.217599,1.270727,1.323014,1.373847,0.540871,0.588923,0.635051,0.680034,0.724600,0.769287,0.8
14494,0.860623,0.907917,0.956585,1.006682,1.058192,1.111005,1.164924,1.219705,1.274966,1.330277,1.385026,1.438524,0.6093
13,0.653648,0.696452,0.738553,0.780720,0.823476,0.867252,0.912369,0.959111,1.007643,1.058031,1.110263,1.164210,1.219705,
1.276469,1.334161,1.392302,1.450254,1.507278,0.681727,0.721345,0.759962,0.798505,0.837719,0.878139,0.920142,0.964067,1.0
10118,1.058488,1.109216,1.162286,1.217599,1.274966,1.334161,1.394812,1.456467,1.518457,1.579977,0.757812,0.791606,0.8251
53,0.859383,0.895096,0.932763,0.972698,1.015227,1.060500,1.108679,1.159805,1.213842,1.270727,1.330277,1.392302,1.456467,
1.522331,1.589278,1.656399,0.837258,0.863867,0.891291,0.920530,0.952227,0.986759,1.024406,1.065358,1.109770,1.157734,1.2
09280,1.264401,1.323014,1.385026,1.450254,1.518457,1.589278,1.662161,1.736204,0.919336,0.937147,0.957491,0.981099,1.0083
23,1.039450,1.074591,1.113851,1.157335,1.205060,1.257071,1.313342,1.373847,1.438524,1.507278,1.579977,1.656399,1.736204,
1.818685,
进程已结束，退出代码为 0
```

运行结果如图. 当然这明显是看不出来对不对的, 所以我使用Mathematica求解并可视化做了对照.



可以看出来大致上是没什么问题的.
当然因为我还是不清楚这个有效数字是如何保留的, 所以停止迭代的条件实际上就是用的课本上的实用共轭梯度法的 $\epsilon < 10^{-4}$ 这样子, 想必问题不大.

关于最佳松弛因子, 利用课本上的

$$\omega = \omega_{\text{opt}} = \frac{2}{1 + \sqrt{1 - \rho(B)^2}}$$

和

$$\text{是 } \rho(B) = \cos\frac{\pi}{n} = \cos h\pi. \text{ 这}$$

知道是 $\frac{2}{1+\sin\frac{\pi}{20}}$, 这大约是1.729左右. SOR迭代法对矩阵的要求比较低, 但是想要快速收敛需要预先知道好一些的 ω, CG法虽然要求矩阵对称正定, 但是稳定的快, 不需要手动寻找参数.

第二题

```
D:\Code\N_A\cmake-build-debug-visual-studio\hw5\hw5_2.exe
9
0
[
  [ 0.333351, 0.333008, 0.334491, 0.33272, 0.332373, 0.33289, 0.333527, 0.333953, 0.334105, 0.334035, 0.333825, 0.333551
, 0.333272, 0.333026, 0.332836, 0.332712, 0.332655, 0.332661, 0.33272, 0.332822, 0.332957, 0.333113, 0.333279, 0.333446,
 0.333605, 0.333749, 0.333871, 0.333964, 0.334026, 0.334052, 0.334038, 0.333984, 0.333887, 0.333746, 0.333561, 0.333331,
 0.333057, 0.332739, 0.332377, 0.331973 ]
]
CG Method Error: 0.00407986
```

这个没什么好说的, 但是可以发现这个CG法还是比较强大. 同样的40阶要是用LU, PLU, QR, LDLT这些早就炸飞了, CG法10000阶都不会炸飞, 非常酷.

第三题

```
D:\Code\N_A\cmake-build-debug-visual-studio\hw5\hw5_3.exe
————————————————————CG————————————————————————
5
0
[
  [ 1, -2, 3, -2, 1 ]
]
————————————————————GS————————————————————————
44
0
[
  [ 1, -2, 3, -2, 1 ]
]
————————————————————JACOBI————————————————————
79
0
[
  [ 1, -2, 3, -2, 1 ]
]

进程已结束，退出代码为 0
```

感觉没什么特别之处, 大家都算出来了也都算对了. 矩阵太小, 大家都用不上1ms就结束了, 然后可以看出来CG法用了5次迭代, 理论上来说最多就5次, 其他两个迭代次数多一些也很合理.