《数值代数》第二次上机作业 实验报告

匡亚明学院 211240021 田铭扬

摘要

笔者使用C++语言（利用OpenBLAS库）编程，使用Gauss-Seidel算法、SOR方法、Richarson算法、Chebyshev半迭代加速方法、共轭梯度法等经典迭代算法，求解同一个线性方程组，并对算法的性能进行了比较。

正文

前言

线性方程组的求解是线性代数中最基本的需求之一，因而也成为了数值计算的重要课题。对于规模较大的线性方程组，朴素的“直接法”不再可行，而具有速度更快、可中途停止等优点的“迭代法”就显得十分重要了。其中经典的迭代法包括：Jacobi（J）算法、Gauss-Seidel（GS）算法、逐次超松弛迭代方法（SOR）、变系数Richarson（R）算法、Chebyshev半迭代加速方法、共轭梯度法（CG）、预处理共轭梯度法（PCG）等。

本次数值实验的目的即是在同一个线性方程组求解问题中应用上述算法，并对它们的运行效率进行比较。这有助于提高对于这些经典算法的认识的深度，对于《数值代数》课程的学习有重要意义。

问题

$$T_n=\begin{bmatrix}2&-1&&\\-1&2&\ddots&\\&\ddots&\ddots&-1\\&&-1&2\end{bmatrix}\quad A_{n^2}=\begin{bmatrix}T_n+2I_n&-I_n&&\\-I_n&T_n+2I_n&\ddots&\\&\ddots&\ddots&-I_n\\&&-I_n&T_n+2I_n\end{bmatrix}$$

考虑线性方程组 $A_{n^2}\boldsymbol{x}=\boldsymbol{c}$，其右端向量由真解 $\boldsymbol{x}^*=(1,1,\dots,1)^T$ 给出。取初始向量 $\boldsymbol{x}_0=\boldsymbol{0}$，用户指标 $\varepsilon=10^{-6}$。

• **第1题** 采用不同的停机标准和向量范数，观测和比较J方法和GS方法达到用户要求的迭代次数，以及停机时的真实误差。尝试给出上述信息关于矩阵阶数 n 的关系。

• **第 2 题**　以真实误差的欧式范数作为停机标准，运行 SOR 方法，观察松弛因子 ω 对于迭代次数的影响。随机取定一组关于阶数 n 的序列（建议尽可能大些），数值扫描出最佳松弛因子，并论证数值结果与理论结果是否吻合？

• **第 3 题**　采用欧式范数和半对数坐标系。考虑 J 方法、G 方法和带最佳松弛因子的 SOR 方法，进行下面的数值观察：

1）绘制误差曲线和残量曲线，比较三种算法的优劣；

2）以真实误差为停机标准，绘图并指出迭代次数同矩阵阶数的关系。

• **第 4 题**　采用欧式范数和半对数坐标系。绘制变系数 R 方法的误差曲线以及残量曲线，观测循环指标 m 的数值影响。

• **第 5 题**　运行 J 方法的半迭代加速，相应的循环指标设置为 $m = +\infty, 100, 50, 25$ 。绘制相应的误差曲线和残量曲线。它与变系数 R 方法有何区别？

• **第 6 题**　取 $n = 50, 51$ 对应不同的奇偶状态。取欧式范数，执行 CG 方法。

1）绘制相应的误差曲线和残量曲线；

2）观测迭代次数同矩阵阶数的关系，它同带有最佳松弛因子的 SOR 方法有何差异？

• **第 7 题**　以 SSOR 方法的主体部分做为预处理阵，运行相应的预处理 CG 算法。随机生成不同阶数的系数矩阵，考察迭代次数同参数 ω 的关系。

相关数学推导

本次实验的部分算法中含有参数，而参数的取值与系数矩阵或迭代矩阵的最大、最小特征值有关，因此在这里先行推导。（参考讲义[1]P48 的讨论）

矩阵 T_n 有 n 个不同的特征值 $\lambda_k = 2\left(1 - \cos\frac{k\pi}{n+1}\right)\ (k = 1,2,3,\dots,n)$，进而 $A_{n^2} = T_n \otimes I_n + I_n \otimes T_n$ 的特征值为 $\lambda_{pq} = \lambda_p + \lambda_q\ (p, q \in \{1,2,\dots,n\})$。故 A_{n^2} 的最大、最小特征值分别为 $\lambda_{min} = 4\left(1 - \cos\frac{\pi}{n+1}\right)$ 与 $\lambda_{max} = 4\left(1 - \cos\frac{n\pi}{n+1}\right)$，则有 $\lambda_{max} + \lambda_{min} = 8$ 以及 $\lambda_{max} - \lambda_{min} = 8\sin\frac{(n-1)\pi}{2n+2}$，这两个数值将在后面用到。

而 Jacobi 方法的迭代矩阵 $B_{n^2} = I - D^{-1}A_{n^2}$ 的特征值为 $\mu_{pq} = \frac{\lambda_{pq}}{4} - 1$，有 $\rho(B) = \cos\frac{\pi}{n+1}$，$\mu_{max} + \mu_{min} = 0$ 以及 $\mu_{max} - \mu_{min} = 2\sin\frac{(n-1)\pi}{2n+2}$。

符号说明

后文中若无专门说明，均用 n 为系数矩阵A_{n^2}的参数，而用 $N=n^2$ 表示矩阵的阶数。SOR 方法与 PCG 方法的“因子”均用 ω 表示。

程序设计

由于大部分算法均在讲义[1]上以流程或伪代码的方式给出，故在此不再详述。这里仅就讲义中未注明的细节，或是与其中不太一致的地方，进行补充说明：

1. Jacobi 算法的公式 $x_k^{(i)}=\frac{1}{a_{ii}}\left[b^{(i)}-\sum_{j\neq i}a_{ij}x_{k-1}^{(j)}\right]$ [1]中，右侧的累加项可以直接将系数矩阵A_{n^2}的第 i 行与第 k-1 步的解向量 $\mathbf{x}$ 进行内积（便于使用 BLAS 库加速）得到，然后将多减的 j=i 时的项补回来即可。GS 法同理可改造为两次内积。

2. 由前面的结果得，SOR 方法的最佳松弛因子为 $\omega=\frac{2}{1+\sqrt{1-\rho^2(B)}}=\frac{2}{1+\sin\frac{\pi}{n+1}}$ 。

3. 非定常 Richardson 方法 $y_k=y_{k-1}+\tau_k(b-Ay_{k-1})$ 的其最佳参数设置为 $\tau_k^*=\left[\frac{\lambda_{\max}-\lambda_{\min}}{2}\cos\left(\frac{2k-1}{2m}\pi\right)+\frac{\lambda_{\max}+\lambda_{\min}}{2}\right]^{-1}$ [1]，由前文结果得$\tau_k=(4\sin\frac{(n-1)\pi}{2n+2}\cos\frac{(2k-1)\pi}{2m}+4)^{-1}$，注意$\sin\frac{(n-1)\pi}{2n+2}$与 k 无关，可以只在最初计算一次，保存在全局变量中供后续使用。

4. 同理，Jacobi 方法版迭代加速的两个参数为 $\nu=1,\ \xi=(\sin\frac{(n-1)\pi}{2n+2})^{-1}$。另外还有一点需要注意，BLAS 库中的“axpy”操作 $(\boldsymbol{y}\leftarrow\alpha\boldsymbol{x}+\boldsymbol{y})$ 仅支持对一个向量进行数乘。因而对于公式 $y_{k+1}=\rho_{k+1}(\nu x_{k+1}+(1-\nu)g_k)+(1-\rho_{k+1})y_k$，在使用 Jacobi 方法计算 $\mathbf{x}_{k+1}$ 时，除以a_{ii}的一步就可以同时多乘上$\rho_{k+1}\nu$，以简化计算。

5. CG 法的计算过程中，数值$r_k{}^Tr_k$与向量Ap_k都需要使用两次，故可以将第一次的计算结果保存下来留给第二次使用，节省了约 N^2 次乘法操作。PCG 法也是同理。另外由于它们计算过程中都用到了残量，可以直接使用残量准则作为终止条件，也节省了一定量的计算。

6. A_{n^2}较简单，故 PCG 法的预优化矩阵可“纸笔”进行 Cholesky 分解 $Q=LL^T$，$L=(D-\omega DL)D^{-1/2}$，故可在初始化时直接生成，计算时每步“回代求解”即可。

实验环境

使用 VMWare 17 虚拟机运行 deepin 20.9 操作系统，并为其分配 4 个 CPU 核心及 6GB 内存。使用了 OpenBLAS 库对向量内积、赋值、求范数等运算进行并行加速。使用 GCC 8.3.0-1 release 编译器。

实验结果分析

第一题

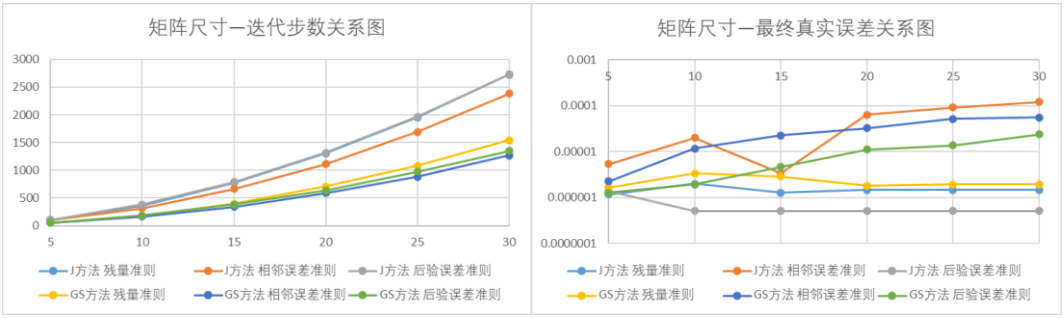

图 1、2：J 方法与 GS 方法在不同停止准则下的迭代步数、误差表现

从图 1 中可以看到，Gauss-Seidel 算法的收敛速度明显快于 Jacobi 算法，平均迭代步数约为 Jacobi 算法的 0.6 倍。由于 GS 方法用到了系数矩阵的更多信息（下三角部分，J 方法只用到了对角部分），更好的表现是符合预期的。

再考察三种终止准则。一方面，相邻误差准则的迭代步数略少，但终止时的真实误差更大——比残量准则差 2 个数量级。另一方面，残量准则虽然误差表现好，迭代步数也仅略多，但每次计算残量时还有 N^2 次乘法运算，在实际耗时上不一定占优。而对于后验误差准则，在 J 方法中迭代步数较多但最终误差最小，在 GS 方法中则均“中等”，属于一种“折中”的方式。

使用残量收敛准则也有一种“折中”方式，即每经过 m 步只判断一次残差，可以减少这一部分计算量。（不过对于后续共轭斜量法等算法，它们在迭代过程中本来就需要计算残量，此时用残量准则就很合适了。）

第二题

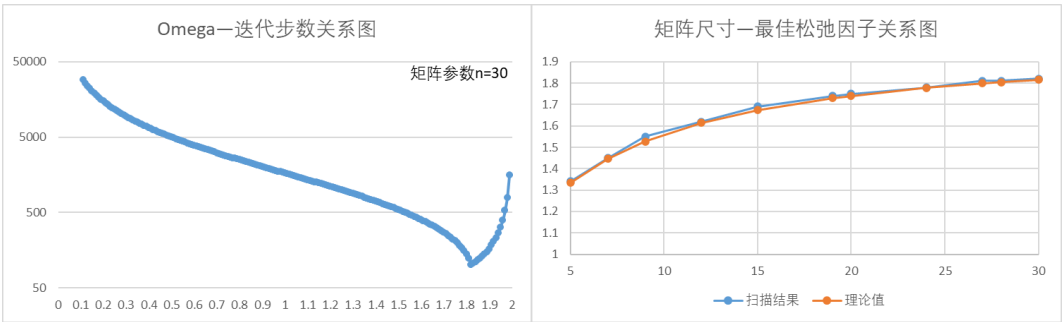

图 3：SOR 方法松弛因子与迭代步数的关系、图 4：SOR 矩阵尺寸与最佳松弛因子的关系

从图 3 中可以看到，松弛因子与迭代步数的关系与课堂所讲的图像吻合：有一个最小值点，且最小值点左侧较陡、右侧较缓。而从图 4 可以看到，数值

扫描得到的最佳松弛因子比理论值稍大。这是因为扫描有一定“间隙”（本次实验中为 0.01），最佳松弛因子可能落在两个扫描值之间；而由于前述“左陡右缓”的性质，因子略偏大时表现比略小时好，故扫描结果相比理论偏大。

第三题

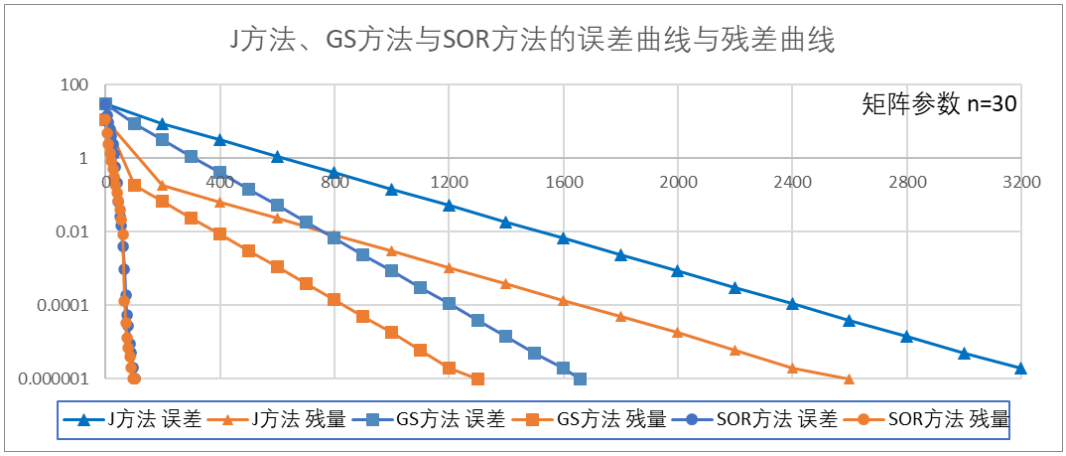

图 5：J 方法、GS 方法与 SOR 方法的误差曲线与残量曲线

从图中可以看到，三种方法的误差、残量都是单调下降的，且 GS 方法好于 J 方法，逐次超松弛算法（SOR 方法）则远好于前两种方法。

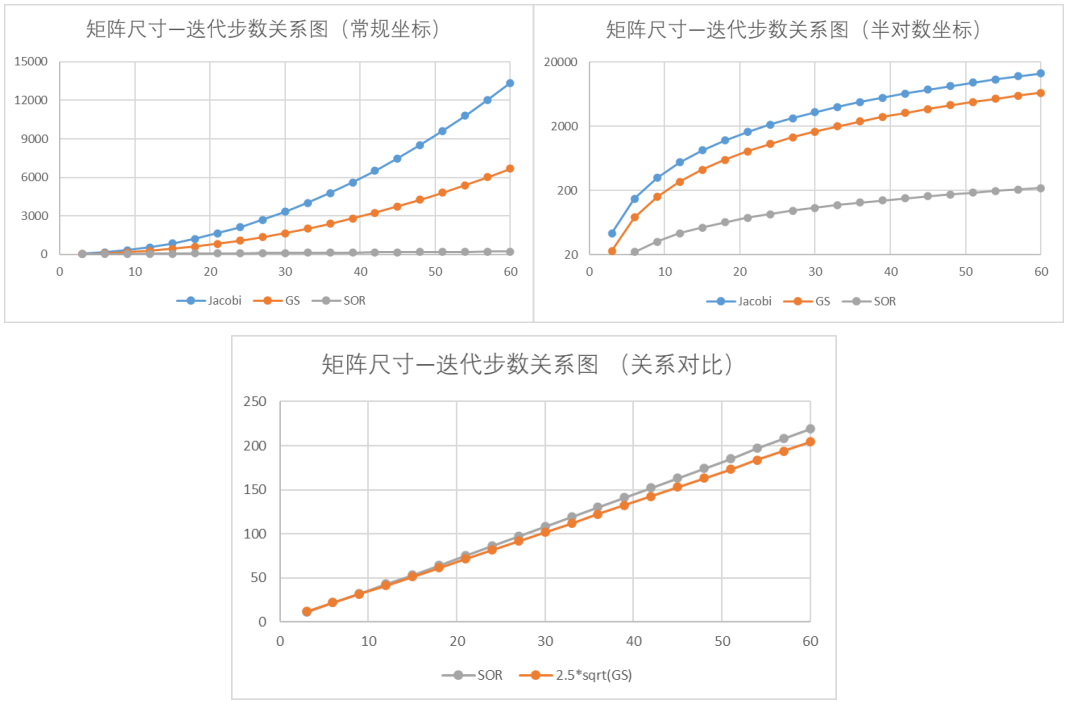

图 6、7、8：J 方法、GS 方法与 SOR 方法的迭代步数与矩阵尺寸的关系

从迭代步数与矩阵尺寸的关系图中更能看出 SOR 算法相比 J 方法与 GS 方法的优越性。此外，可以在图 8 中看到，SOR 方法的迭代步数与 GS 方法迭代步数的平方根成正比，这也证明了课堂上所讲的“平方级别的改善”。

第四题

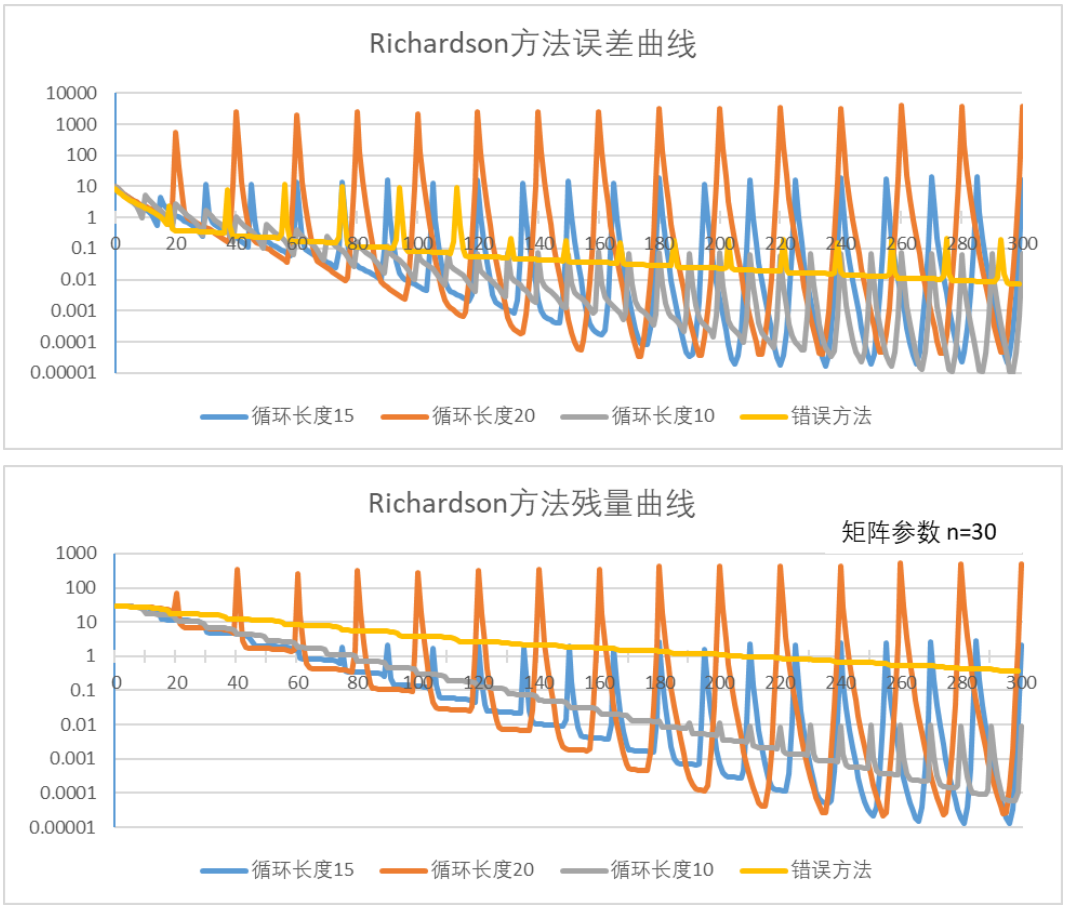

图 9、10：变系数 Richardson 方法在不同循环长度下的误差曲线、残量曲线

如图可以看到，重启的变系数 Richardson 方法表现了很明显的单调性。取循环长度较长时收敛速度更快，而在周期的末尾误差上升的也更多。但是注意到，随着迭代的进行，误差与残量的“上界”将会固定在一个值，且“下界”都会趋于稳定。猜测是受到了机器精度的限制。

另外还注意到一个有趣的现象：在循环的中段，误差有一定程度的下降，而残量却几乎“持平”，暂时不知道出现此现象的原因。

而关于图中的黄色曲线，是笔者“自作聪明”，在每次残差上升时立即进行重启，而忽略掉后面的几步。这样虽然保证了残差的单调下降，却使得收敛速度明显下降。原因在于这相当于缩短了每次循环的长度，使得循环使用的参数（相应次数切比雪夫多项式的零点）不再正确，自然表现变坏了。这也提醒我们，对于本身就“非单调”的算法，强行“单调化”不一定能带来好的结果。

第五题

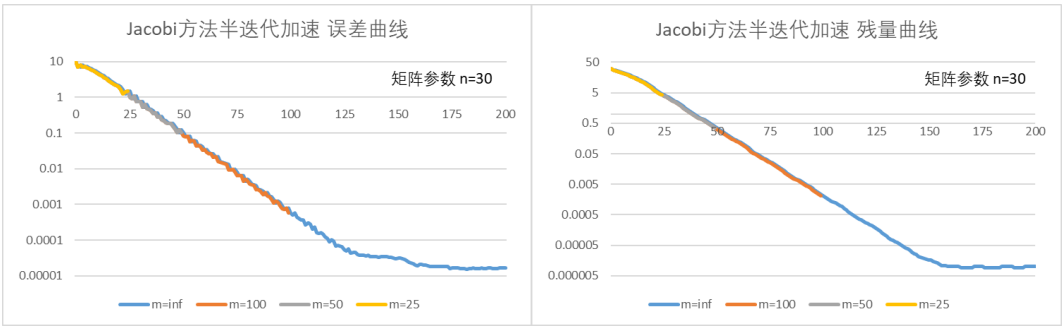

图 11、12：Jacobi 方法半迭代加速的误差曲线、残量曲线

从图中可以看出，对 Jacobi 方法进行半迭代加速时，循环参数 m 的选取不会对算法得表现造成显著的影响。

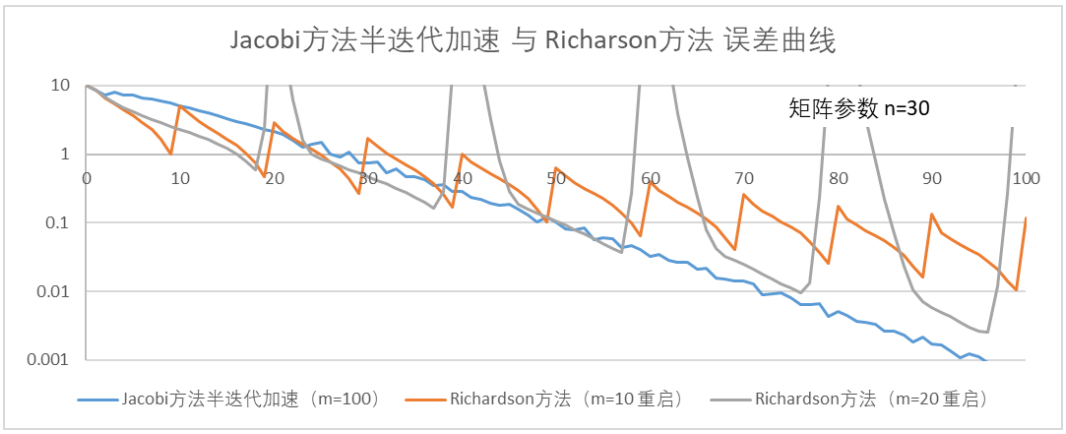

图 13：Jacobi 方法半迭代加速与变系数 Richardson 的误差曲线

而从这张对比图中可以看出对 Jacobi 方法进行半迭代加速的收敛速度，比循环长度 m=10 的 Richardson 方法要好，而只比 m=20 的方法略好。推测选取更大的循环长度 m，能使得两种方法的收敛速度表现接近。这说明两种方法的本质（利用切比雪夫多项式进行加速）是相似的。

第六题

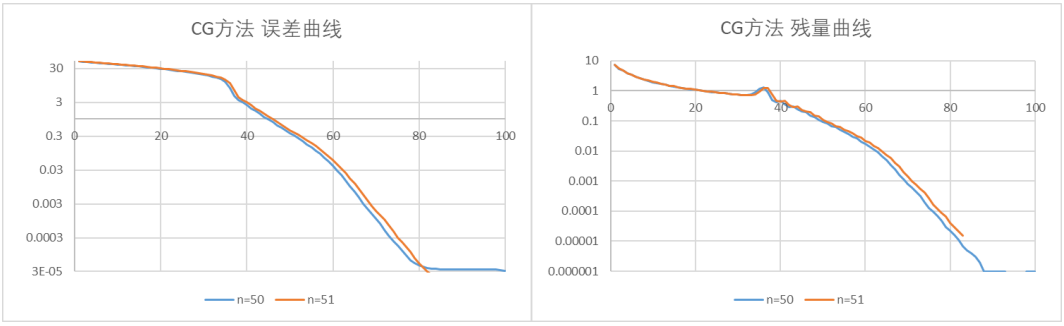

图 14、15：共轭梯度法在不同奇偶状态下的误差曲线、残量曲线

如图，在不同奇偶状态下，共轭梯度法（CG 法）再开始时的表现是一致的。但随着迭代的进行，$N=n^2$ 为偶数时算法的收敛速度略微变快，却更快达到了误差的下限（但是高于终止准则的界限）；N 为奇数时收敛略慢，却能够保持更长时间的误差快速下降，且从右侧的残量曲线图来看，如是将停机准则从真实误差改为残量准则，“最终误差”还有进一步减小的空间。

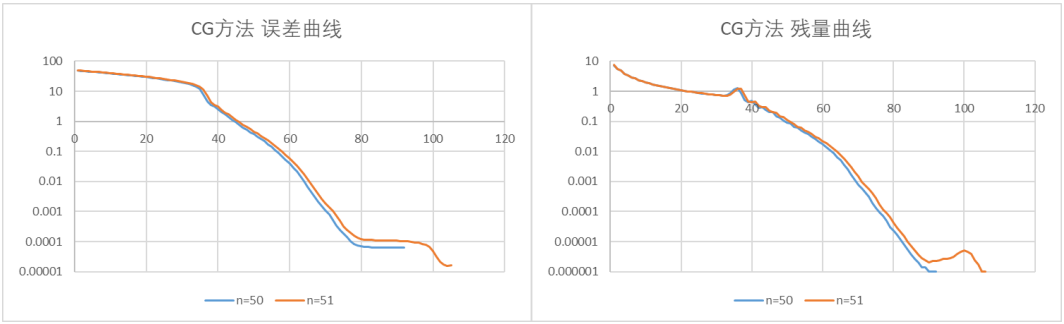

图 16、17：共轭梯度法（使用残量准则）在不同奇偶状态下的误差曲线、残量曲线

经过实验，如上图所示，改用残量准则后 N 为奇数时的最终“最终误差”确实变小了。但实际表现却与估计的现象——即 $n=51$ 时的误差曲线将会“穿过”$n=50$ 时的曲线——不服，而且残量曲线还略有反复。这种现象出现的原因还有待研究。

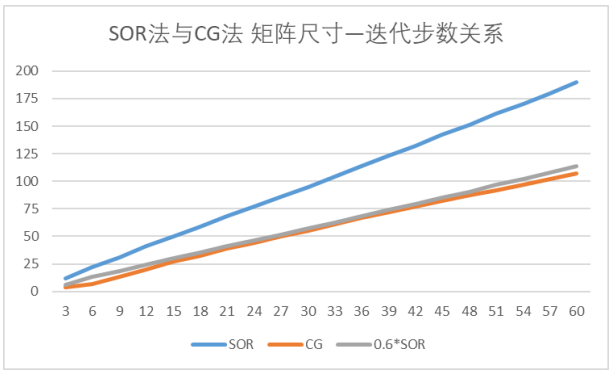

图 18：共轭梯度法与 SOR 方法的迭代步数与矩阵尺寸的关系

最后是 CG 算法与 SOR 算法的比较，可以看到在系数矩阵的尺寸相同时，使用 CG 算法所需的步数只有 SOR 算法的 0.6 倍，得到了较好的改善。

单就收敛速度而言，注意到前文已经说明 SOR 算法相对于 Gauss-Seidel 等朴素算法是“开平方”级别的改良，这说明 CG 也是“开平方级别”，与 SOR 仅有系数上的优势。但是用作对比的 SOR 算法需要在已知迭代矩阵A_{n^2}的特征值情况下，提前计算出“最佳收敛因子”，才能达到这样的改良；而 CG 算法不需要提前知道任何信息就能达到如此效果，是其巨大的优点。

第七题

图 20：PCG 方法迭代步数与最佳参数 ω 的关系

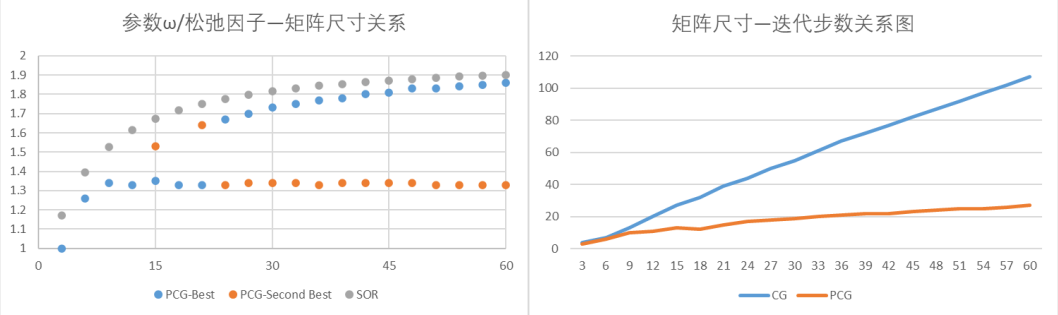

图 21：PCG 与 SOR 法的最佳参数与矩阵尺寸关系

图 22：PCG 法与 CG 法迭代步数与矩阵尺寸关系

首先如图 22，进行预优化后的共轭梯度法相比优化前，达到停止准则所需的迭代步数大幅度减少。而如图 20，预优化时的参数 ω 即使没有取到最优，$\omega>1$ 时 PCG 的表现仍要好于 CG 法，这是与 SOR 算法很大的不同！

此外，如前文所述，由于本题预优化矩阵的 LL^T 分解是用纸笔计算出来的。因次，预优化后的每步只多消耗了两次“回代求解”的计算量，共 N^2 次乘法运算，这相对于迭代步数的大幅减少是微不足道的。

一个有趣的现象是，参数 ω 有一般两个局部最优值。其中一个固定在 1.33 附近，另一个会随矩阵尺寸的增加而逐渐增大，并接近 SOR 算法的最佳松弛因子。且在系数矩阵阶数较小时，固定的局部最优值表现更好，而阶数较大时另一个最优值表现更好。这个奇特现象背后的数学原理还有待进一步的研究。

结语

在本次数值实验中，笔者使用了 Jacobi 算法、Gauss-Seidel 算法、逐次超松弛迭代方法、变系数 Richarson 算法、Chebyshev 半迭代加速方法、共轭梯度法和预处理共轭梯度法，求解同一线性方程组并分析其性能，锻炼了笔者的实践能力，且极大地加深了笔者对于上述算法的理解。

在实验中也出现了几个“遗留问题”，比如 Richardson 方法残量曲线、PCG 方法参数的有趣现象等，留下了进一步研究的空间。

实验代码

由于篇幅限制，代码及原始数据不在实验报告中列出，可以在笔者 github 仓库中查看。网址为：https://github.com/lk758tmy/NA2-Codes。

参考文献

[1]《数值代数》讲义. 张强

[2]数值计算方法-下册. 林成森. 科学出版社. 2005-1 第二版