**《数值代数》第一次上机作业 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库和lapack库）编程，用Gauss列主元消元法、Cholesky分解法、Crout算法、追赶法等算法，分别解决一个“示例问题”，并藉此讨论了上述算法的运行效率和误差。

**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，其中诞生了Gauss消元法（以及列主元Gauss消元法和Gauss-Jordan消元法）、Cholesky分解法（LLT与LDLT算法）、LU分解法（Crout算法和Doolittle算法，追赶法为其特例）等经典算法。

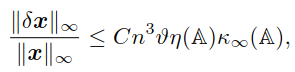
本次数值实验的目的即是“实机”应用上述其中几个算法解决问题，并对它们的运行效率(CPU时间)、误差等进行讨论。因为这些算法的经典性，此次实验对于《数值代数》课程的深入学习有重要的意义。

**问题**

**第1题** 利用列主元 Gauss 消元法、LL⊤ 法和 LDL⊤ 法求解线性方程组，真解取为 ，右端向量由利用真解计算出来。

1.1 绘制数值误差同矩阵阶数 n 的关系，其中数值误差采用对数坐标；

1.2 绘制 CPU 时间同 n 的关系；

1.3 绘制矩阵条件数与 n 的关系。请问：摄动理论给出的右式是否完美刻画了相对误差的大小？

**第2题** 考虑行列重排后相等的两个 n 阶矩阵

执行相应的 Crout 算法；利用 Matlab 命令 spy() 绘制它们在三角分解后的非零元素分布（或结构图），并比较相应的 CPU 时间。

利用矩阵的元素分布特点，修改 Crout 算法，删除那些无用的运算时间。重复上述操作，观察 CPU 时间是否得到节省？

**第3题** 计算三对角阵 或块三对角阵 的逆矩阵，并观测它们的运行效率（关于 n 的计算复杂度）。

**第4题** 设 ，定义 ；考虑两个同解的三对角线性方程组 ，其右端项均由真解 生成。用追赶法求解它们，观测数值误差同 n 的关系。

**程序设计**

第1题用到的LLT法和LDLT法、第2题用到的Crout算法、第3题用到的Gauss-Jordan算法和第4题用到追赶法，分别参照讲义（参考文献[1]）P15、P16(右下)、P14、P11和P19的“代码块”进行实现，不再详述。部分算法在讲义中并未完全实现（如追赶法仅实现了矩阵变形，而未实现右端项变形和回代求解；LLT及LDLT法仅实现了矩阵分解，而未实现回代求解），笔者进行了相应的补全。此外，讲义中的代码是基于朴素的“点对点”操作实现的，读者利用OpenBLAS库，对“axpy”、向量点乘、求向量2-范数等操作进行了优化。

第1题还使用了列主元Gauss消元法，笔者在讲义P2的Gauss算法“代码块”基础上进行改动，实现了此算法。为保证代码易读性，笔者并未使用数组保存行交换情况，而是略微牺牲效率，用OpenBLAS库中的向量交换操作实现。

第2题要求针对题中矩阵，对Crout算法进行特别优化，为保证行文连贯，此部分留待“实验结果分析”段落详述。

最后，第1题需要计算矩阵的条件数，使用了OpenBLAS库自带指令实现。

**实验环境**

使用VMWare 17虚拟机运行deepin 20.9（基于Debian10）操作系统，并为其分配4个CPU核心及6GB内存。

使用OpenBLAS库实现的CBLAS，并使用lapack库。

使用GCC 8.3.0-1版本编译器，未开启编译优化选项。

**实验结果分析**

**第1题**

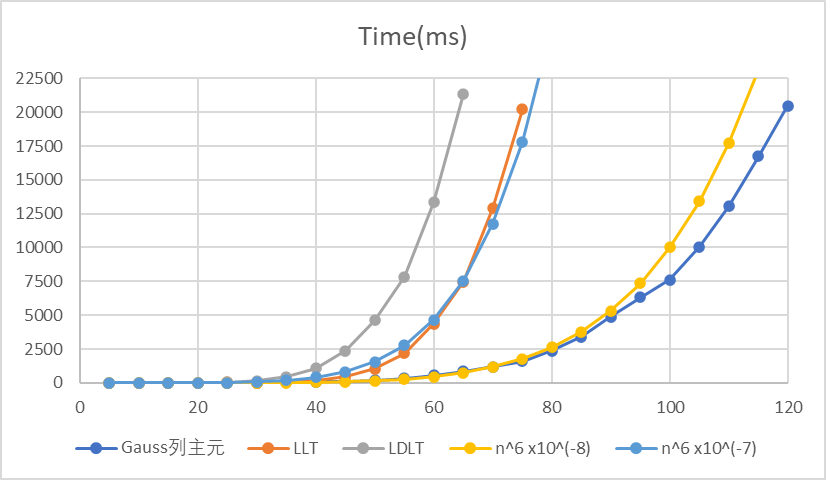


图1.1 第1题运行效率(CPU时间)图表

如图表所示，三种算法运行耗时均与同阶。注意到矩阵的阶数为，故实验结果证明了三种算法的时间复杂度与理论一致，均为。

注意到LLT算法较Gauss列主元算法慢10倍左右，可能是因为LLT法涉及较慢的开方运算。但是不涉及开方运算的LDLT算法却比LLT的表现还差，可能是因为笔者是按照讲义中的“代码块”实现的算法：讲义P15中 LLT算法是基于“逐列次序”实现的，笔者使用了OpenBLAS库中的向量点乘指令，而该库会对其进行并行优化，故速度较快；讲义P16中 LDLT算法是基于“逐行次序”实现的，无法进行这一优化，故速度较慢。预计若基于“逐列次序”重写LDLT算法并进行相应优化，运行速度将会快于LLT算法。

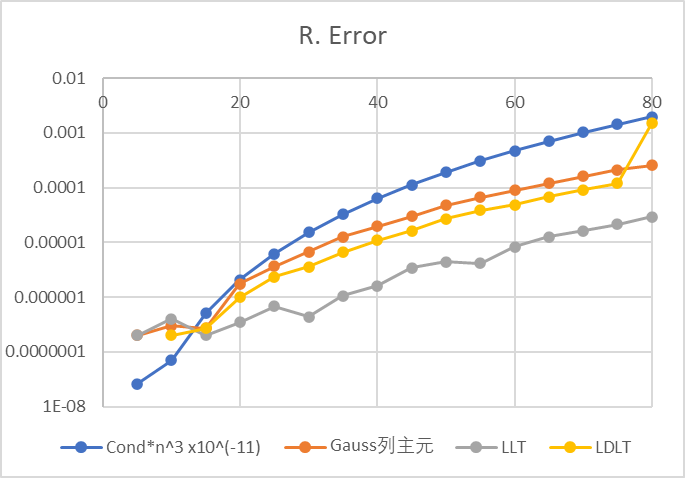
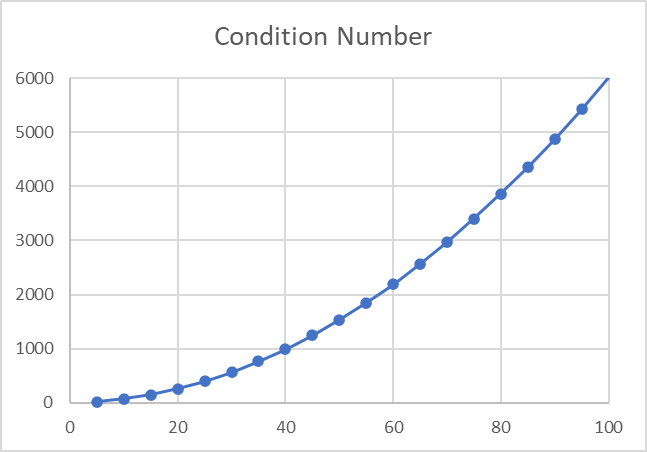
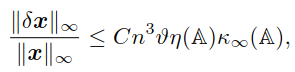
**

图1.2 第1题相对误差图表-对数坐标 / 图1.3 第1题条件数图表

如图表所示，注意矩阵的阶数为，故三种算法的相对误差均与系数矩阵的 同阶。由于课上并未学习主元增长因子的相关内容，可以“反向”讨论第3小问，即：假设公式成立，则矩阵的主元增长因子。

**第2题**

对两个矩阵，分别取a=0.001与a=1000执行Crout算法，运行时间如下表：

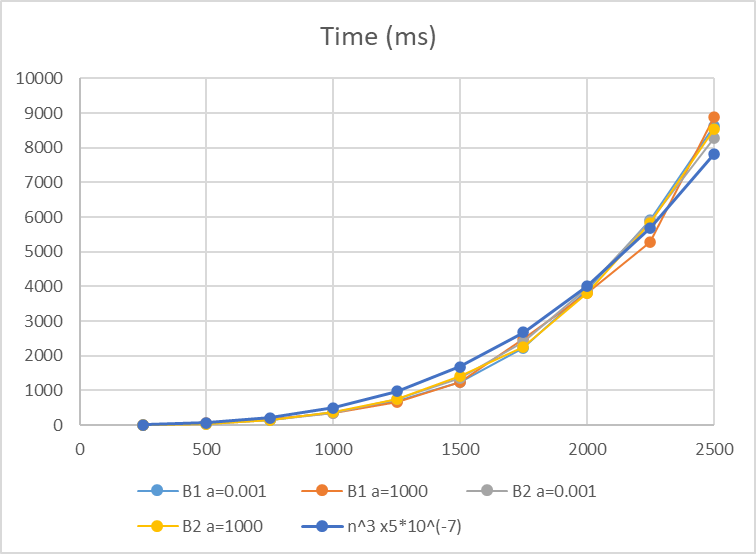


图2.1 第2题运行效率(CPU时间)图表

如表所示，Crout算法对两个矩阵、两种a的取值的运行效率相近，且均与同阶，与理论相符。而为了观察两者分解后“0”的分布情况，取n=20,a=10并输出执行Crout算法后的结果，得到如下图所示的输出：

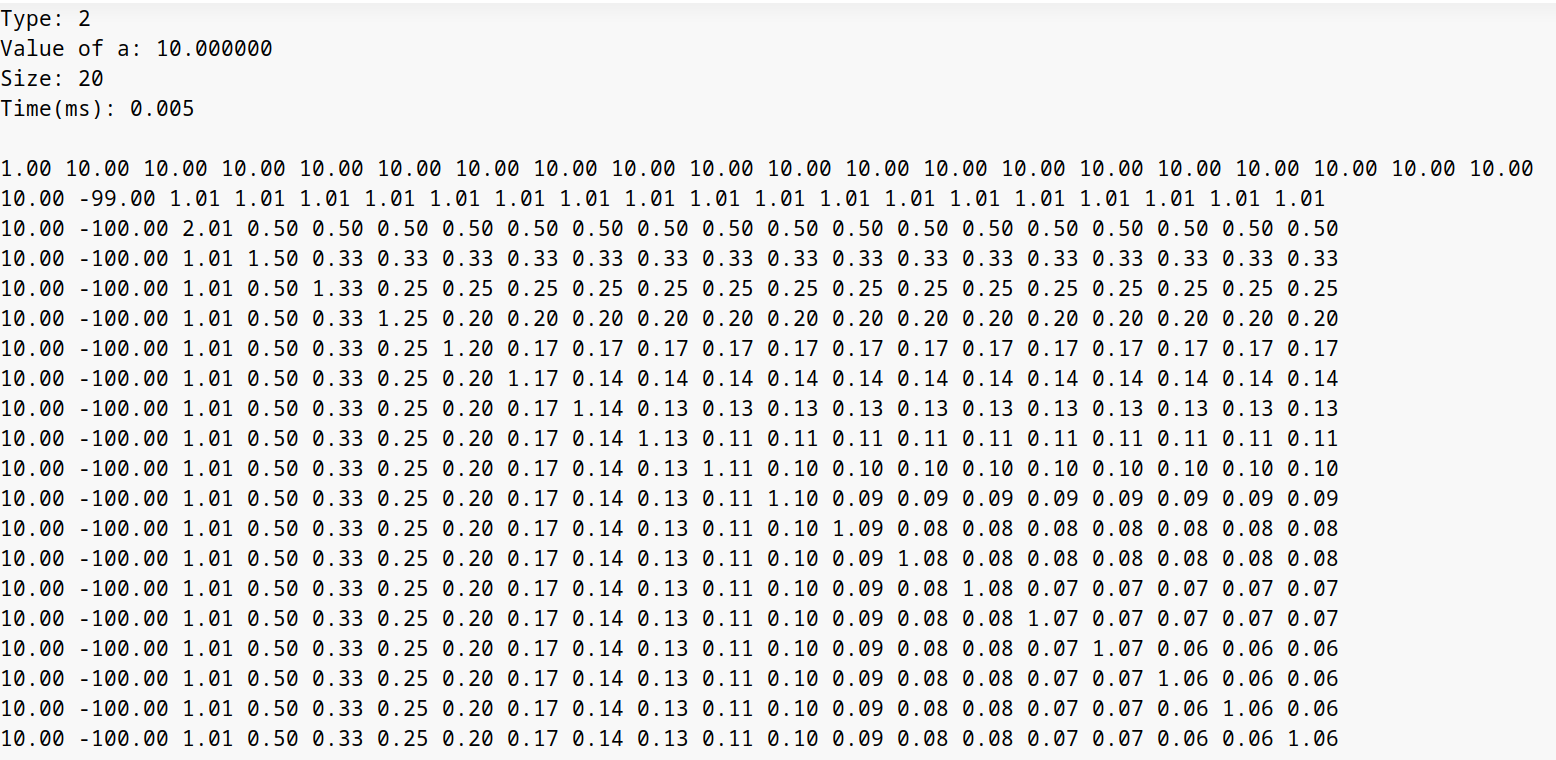
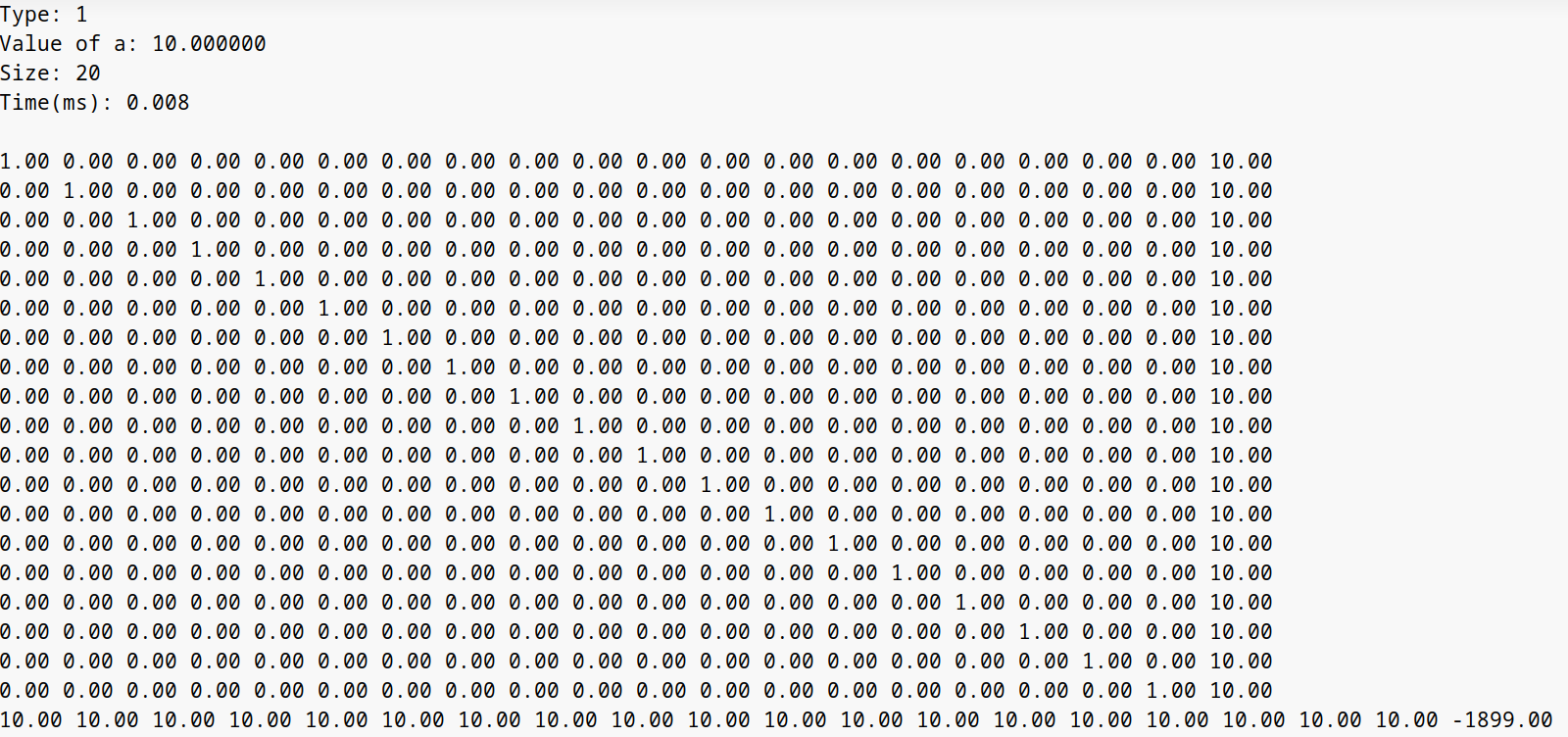
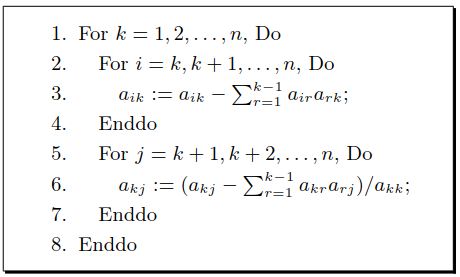


图2.2/2.3 B1及B2矩阵分解后的0元素分布

 注意到B1矩阵分解后，除对角线元素与最后一行、最后一列元素外全为0，而B2矩阵分解后几乎没有0元素。故可针对B1进行优化：右图是讲义中的伪代码，第3行改为当且仅当(i==k||i==n||k==n)时才执行，否则置aik=0，第7行同理。如此优化后运行结果如下表，实际的时间复杂度介于与之间。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Size** | 500 | 1000 | 1500 | 2000 | 2500 |
| **未优化** | 50.873 | 381.984 | 1240.887 | 3908.282 | 8619.499 |
| **优化** | 0.212 | 1.037 | 4.992 | 23.328 | 49.293 |

表2.4 B1优化前后对比（取a=0.001）

**第3题**

由于矩阵阶数较少，效果不明显，笔者在此选用来测试Gauss-Jordan算法求逆矩阵的效率。运行结果如下：

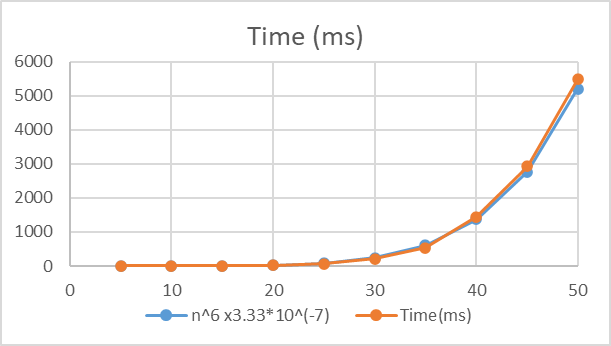


图3.1 第3题运行效率(CPU时间)图表

如图表所示，运行时间与同阶，即时间复杂度为，与理论相符。

**第4题**

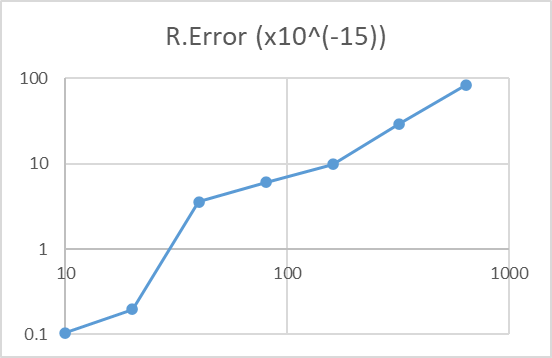
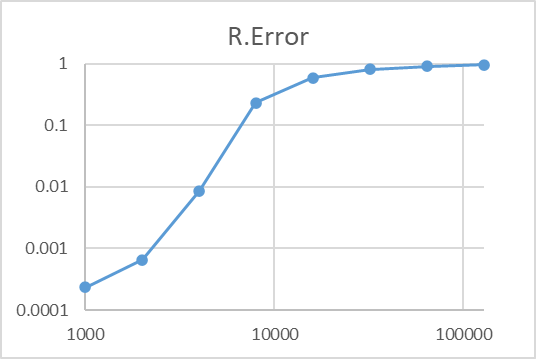


图4.1/4.2 第4题误差图表-双对数坐标

上左是追赶法求解的误差图表，上右是求解的误差图表注意到由于不是严格对角占优矩阵，追赶法并不稳定，在阶数较大时相对误差接近1，表现很差。

另一方面，是对角占优的，但由于其中元素绝对值较小，笔者不得已使用double变量（前文所有实验均使用float类型变量）以保证矩阵元素不会虽阶数增长很快低于机器精度。但即便如此，仍然不能得到阶数很大时的数据，且使用double类型会导致相对误差大幅降低，不具有可比性。这是本次实验的遗憾，留待以后的实验中改进。

**结语**

在本次数值实验中，笔者使用列主元Gauss消元法、LLT法和LDLT法、Crout算法、Gauss-Jordan算法和追赶法，对示例问题进行了求解与分析，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。

**实验代码**

由于篇幅限制，代码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-上册. 林成森. 科学出版社. 2005-1第二版