**《数值代数》第四次上机作业 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和lapacke库）编程，对于同一个三对角矩阵的特征值求解问题，比较了幂法及其加速算法、反幂法、子空间同时迭代法、Jacobi方法、循环Jacob方法、阈值Jacobi方法、Sturm序列二分法、隐式QR方法等经典算法的表现。

**正文**

**前言**

矩阵特征信息的求解，是在实际应用中十分重要的问题，其中的经典算法包括幂法（包括反幂法、子空间同时迭代法等）、Jacobi方法、“三对角化”后使用Sturm序列二分法、QR方法等，它们的设计思想有很大的不同。

本次数值实验的目的，即是对上述算法的表现进行比较。这有助于提高对于这些经典算法——及其背后的思想——的认识深度，对于未来计算数学应用领域的进一步学习有重要意义。

**问题**

求解对象是对称三对角阵 ，其中n = 100或101，用户指标设定为 ε = 10−8。

**·第1题** 取初始向量v0 = (1, 1, 1, . . . , 1)T，用幂法计算主特征信息。  
1. 绘制特征值的误差曲线，以及特征子空间的距离曲线；  
2. 采用 Atiken 技巧和 Rayleigh 商技术进行加速，绘制相应的特征值误差曲线  
若改变初始向量，结果有什么区别吗？

**·第2题** 利用反幂法，分别求解离 q=2 和 q=3 最近的特征值及其特征向量。绘制相应的特征值和特征向量误差曲线。

**·第3题** 取非常接近某个特征值的q，观察反幂法是否呈现“一次迭代”。

**·第4题** 首先，利用幂法和降维技巧，求解前两个主特征值及其特征向量；然后，利用同时迭代方法求解前两个主特征值。比较两者的计算效果有何差异。

**·第5题** 分别用古典Jacobi方法、循环Jacobi方法和阈值Jacobi方法求解全部特征值，并绘制和对角元的收敛过程。

**·第6题** 首先，利用Strum序列二分法,求解位于开区间 (1, 2) 内的所有特征值；绘制相应的收敛过程。然后，考虑带原点位移的反幂法，观测数值精度是否得到改善？

**·第7题** 利用隐式对称 QR 方法求解全部特征值。

**·第8题** 阈值 Jacobi 方法求解（绝对值）小特征值时具有优势。考虑对称正定矩阵 ，直接计算可知其特征值1040, 9.9×1019和 9.81818×10−1。用阈值Jacobi法求解，并同Matlab命令eig()的结果比较。

**程序设计**

由于大部分算法都在讲义[1]或教材[2]上有较为详细的实现流程，故在此不再讨论。但仍有几处需要注意的地方：

1. 第2题，q=2与q=3均为T101的特征值，故执行反幂法前应加入微小扰动；

2. 第4题，书中与教材中均未明确写出，使用子空间同时迭代算法时，每一步QR分解得到的矩阵R，其对角元收敛至原矩阵的特征值。

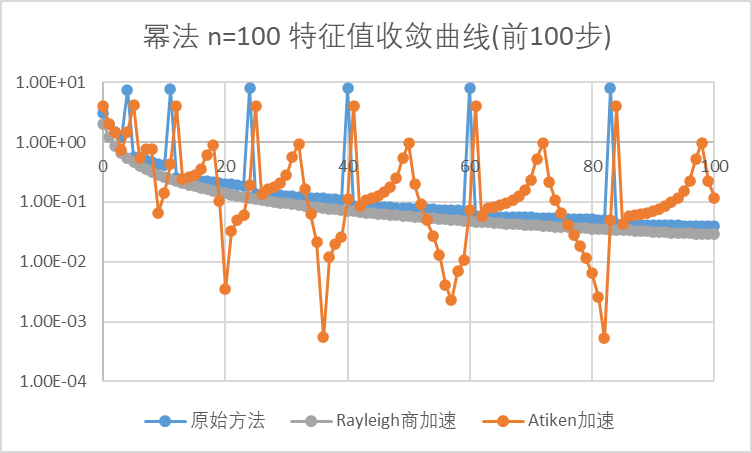
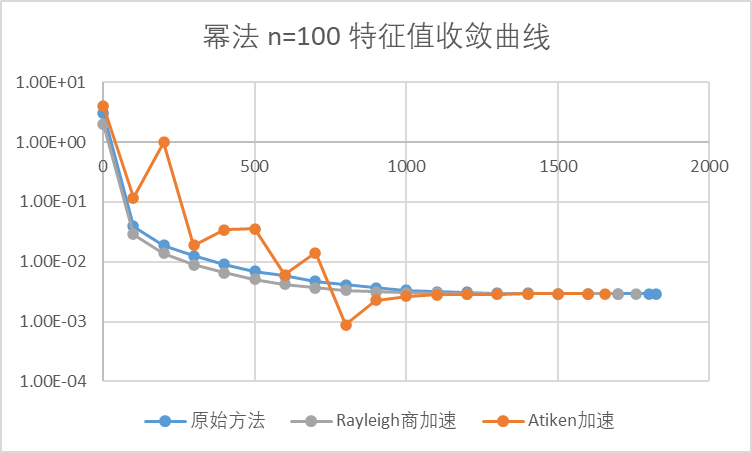
**实验环境**

部分代码使用C++编写。在虚拟机软件VMWare 17中运行deepin 20.9操作系统，设置8GB内存和8个CPU核心，开启Intel VT-x/EPT和IOMMU选项。使用了OpenBLAS、lapacke等数值代数库，对向量—向量与矩阵—向量运算进行并行加速。使用GCC 8.3.0-1编译器，未开启优化选项。

部分代码使用Matlab编写，版本为R2022a。使用Intel i5-1135G7八核CPU，使用16GB DDR4 3200MHz内存。系统为Windows 11 22H2。

**实验结果分析**

**第一题**

****

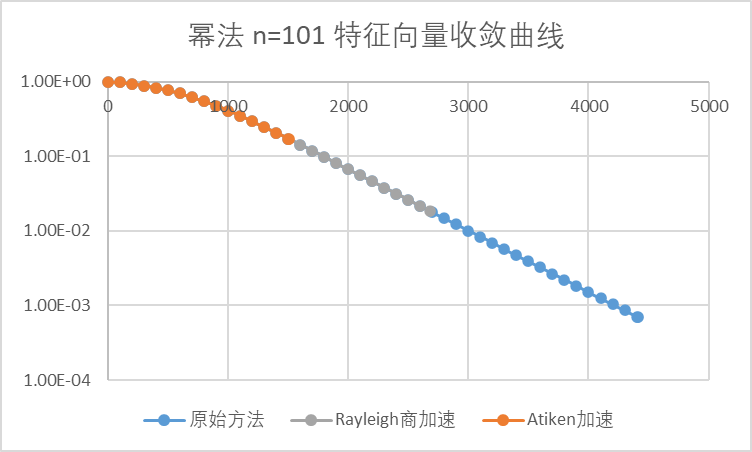
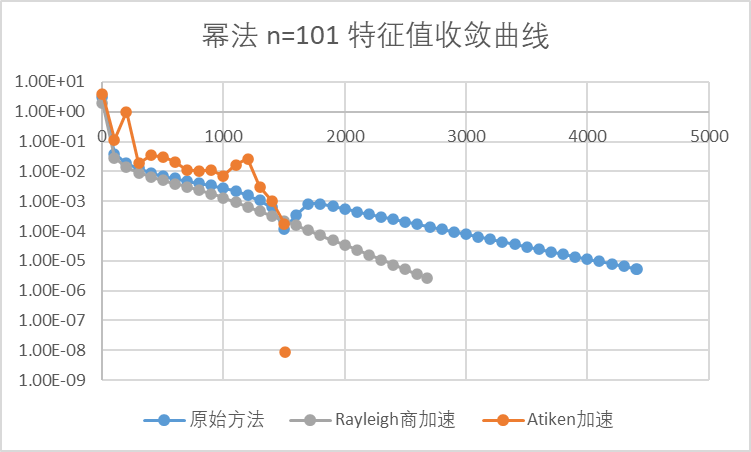
****

图1-4：幂法收敛曲线

如图，这是取初始向量v0 = (1, 1, 1, . . . , 1)T时幂法及其两种加速方法的收敛表现（每100步输出一次结果）。算法表现与理论不尽相同。

n=100时，三种方法以差不多的速度收敛，且三种方法求出的特征向量始终保持与真实特征向量垂直（故没有画图）。猜测是由于初值的特殊性导致的。

n=101时，Rayleigh商加速方法以比原始方法快不到两倍（以迭代步数记）的速度收敛，Atiken方法则在中段出现了近乎“奇异”的迅速收敛，猜测也与初值的特殊性有关。另外注意到，原始方法与两种加速方法在特征向量的收敛速度上没有差异，其原因是显然的：两种加速方法均没有对与特征向量有关的部分进行改动。这也导致了，原始方法由于收敛最慢，可以迭代更多步数，最终得到的特征向量反而更好。**因此，在实际应用中，如果只需求解主特征值，可以考虑使用加速算法；如果还要求相应特征向量，应使用原始方法；或在加速算法的基础上调低用户指标（停机指标）。**

此外在两种情形下，Atiken加速算法的特征值收敛曲线，一直到“中段”都还有一定幅度的“波动”。猜测这是由其“两步递推”的形式导致的，而与初始向量的特殊性无关。

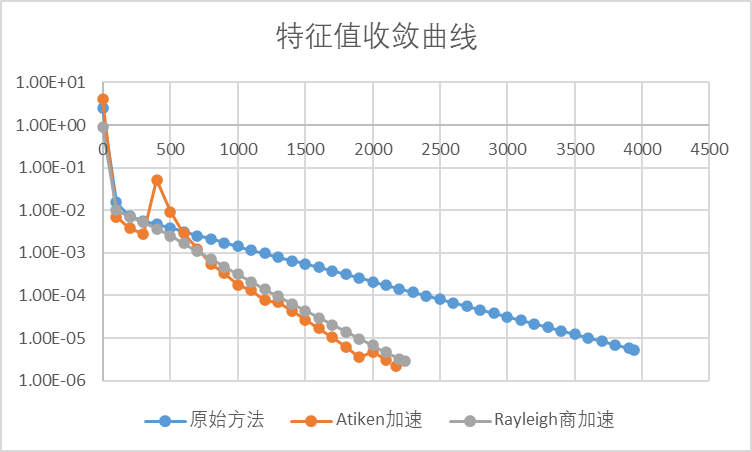
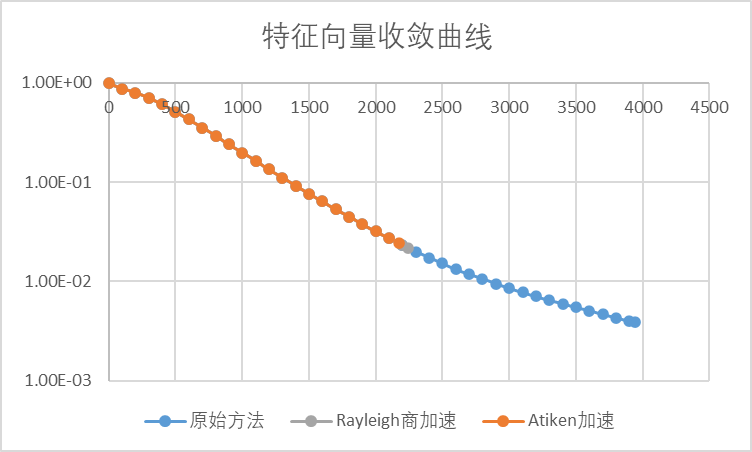


图5、6：(随机选取初值)幂法的收敛曲线

由于前面出现的奇异现象可能与初值的特殊性有关，故笔者随机选取初值重新进行多次计算，仅以n=100时为例，图中是一次计算的结果。

改为随机选取初值，对n=100时使用幂法，特征向量是收敛的。而特征值的收敛上，Atiken加速也和Rayleigh商方法一样，相对于原始方法有不到2倍的速度提升。这说明前面所描述的反常表现，确实是因为题中给定的初始向量v0 = (1, 1, 1, . . . , 1)T过于特殊导致的。（此外，Atiken方法特征值误差的“波动”现象并未消失，说明这是方法本身而非初值选取的原因。）

但是，上述结果其实是笔者在三次计算种选择了一次较好的结果。另外两次的结果（仅展示特征值的收敛曲线）如下。

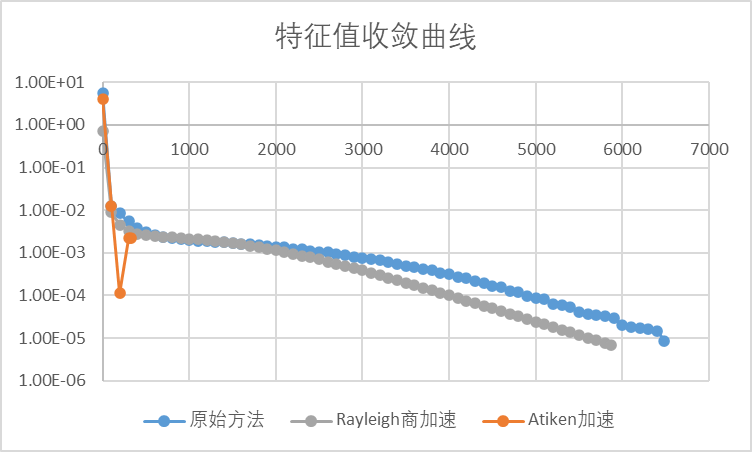
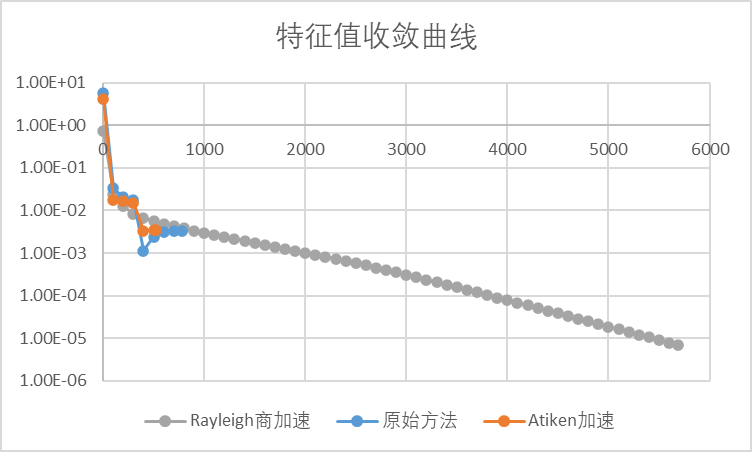


图7、8：(随机选取初值)幂法的收敛曲线（续）

左图中，Rayleigh商方法达到了很好的精度，却一直没有停机；原始方法一度达到了比Atiken方法更好的表现，却在停机前出现了反弹。右图中，原始方法与Rayleigh商方法的表现都不尽如人意，Atiken加速有相对两种方法好很多的表现，却也在停机前大幅反弹。

这表明，幂法的表现受到初始向量选取的影响很大，应当多计算几次后取最好的结果。**但在实际应用中不会预先知道要求解的特征信息，其实是很难比较哪一次的结果“最好”的。这对幂法的实际应用带来了很大的限制。**

**第二题、第三题**

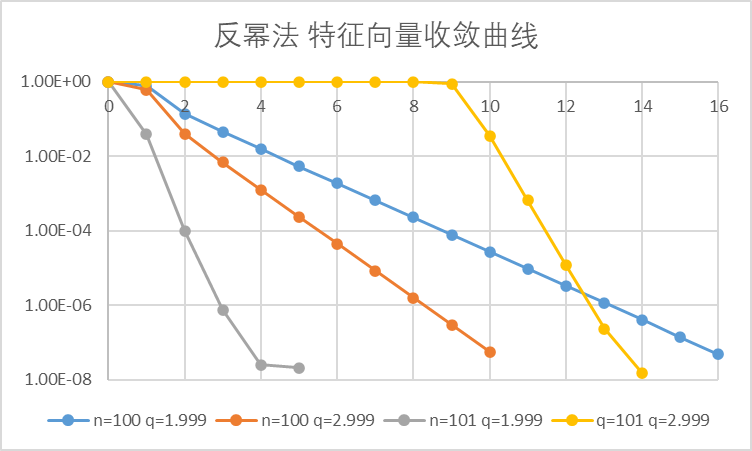
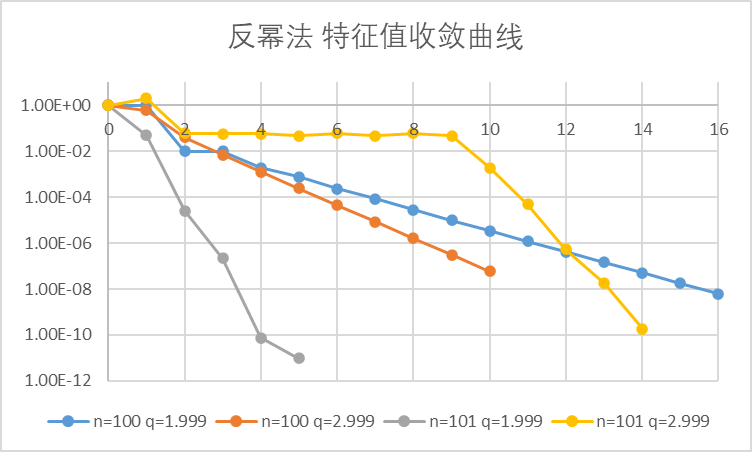


图9、10：反幂法的收敛曲线

图中是反幂法的收敛曲线，初值是随机选取的。可以看到反幂法的表现不是很稳定，如在n=101 q=1.999的情形收敛速度很快；而在n=101 q=2.999的情形出现了很长的“平台期”，之后再快速收敛。原因有待进一步探究。

但无论如何，其收敛步数比幂法减少了2个数量级。而考虑到解线性方程组与矩阵乘向量的复杂度差距(O(n3)与O(n2))，以及本题中n=100/101，可以认为在本题的条件下，幂法与反幂法的计算速度大致相同。

此外，本题的代码是使用Matlab编写的，其矩阵“左除”运算性能较好。使用C++编写时反幂法的表现都很差（特征值的误差只能达到10-2量级）。**这是由于其子问题的矩阵A-qI接近奇异，因而对线性方程组求解算法的精度要求较高。**在第六题的结果分析中会继续讨论这一问题。

而关于反幂法的一步收敛性，使用常规坐标能更好体现，见下图。

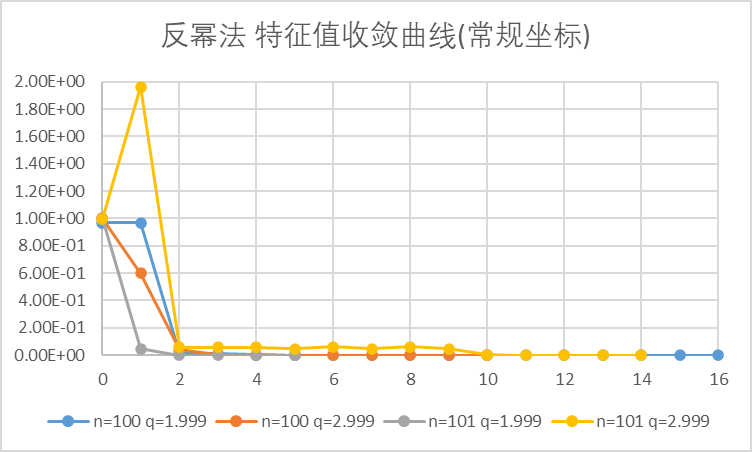


图11：反幂法的收敛曲线（续）

**第四题**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| n=100  (三次平均) | 幂法+降维收缩 | | 幂法+降维收缩 (Rayleigh商加速) | | 子空间同时迭代法 | |
| 主特征值 | 次特征值 | 主特征值 | 次特征值 | 主特征值 | 次特征值 |
| 特征值误差 | 1.23E-04 | 1.21E-03 | 6.89E-06 | 2.77E-06 | 2.90E-03 | 2.90E-03 |
| 特征向量误差 | 8.85E-02 | 2.34E-01 | 4.87E-02 | 2.91E-02 | 5.28E-03 | 5.81E-03 |
| 用时(ms) | 381.64 | | 114.42 | | 88.70 | |
| n=101  (三次平均) | 幂法+降维收缩 | | 幂法+降维收缩 (Rayleigh商加速) | | 子空间同时迭代法 | |
| 主特征值 | 次特征值 | 主特征值 | 次特征值 | 主特征值 | 次特征值 |
| 特征值误差 | 4.07E-04 | 1.72E-01 | 7.04E-06 | 2.84E-06 | 2.84E-03 | 2.84E-03 |
| 特征向量误差 | 8.85E-02 | 2.33E-01 | 4.97E-02 | 2.98E-02 | 5.03E-03 | 4.96E-03 |
| 用时(ms) | 89.88 | | 54.25 | | 49.06 | |

表1-2：幂法与子空间同时迭代法的收敛表现对比

对比上表中的数据可知，在只求解前两个特征信息时：

1.对于主特征值的求解，无论是否应用Rayleigh商加速，幂法+降维收缩的误差表现均好于子空间迭代法；而在求解次特征值时，未进行加速的幂法出现了比较明显的误差积累现象；子空间迭代法的表现很稳定。

2.对于特征向量的求解，子空间迭代法的表现好于幂法+降维收缩的方法。

3.无论是否使用Rayleigh商加速方法，子空间迭代法的用时表现均好于幂法；如果需要求解更多的特征信息，同时迭代法的时间优势会更加明显。

**第五题**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 矩阵大小 | n=100 | | | n=101 | | |
| 方法 | 古典 | 循环 | 阈值(δ=101) | 古典 | 循环 | 阈值(δ=102) |
| 消元次数 | 14865 | 36985 | 23075 | 15194 | 37790 | 23905 |
| 扫描次数 | — | 11 | 16 | — | 11 | 16 |
| 用时(ms) | 112.46 | 4.23 | 2.73 | 51.89 | 5.85 | 2.93 |

表3：三种Jacobi方法性能表现对比

上表中呈现的是三种Jacobi方法的性能表现。虽然循环Jacobi方法和阈值Jacobi方法需要进行更多次Givens变换，它们的用时却远少于古典Jacobi方法。这表明在古典Jacobi方法中，最大元素的选取确实消耗的过多的时间，以至于出现“喧宾夺主”的情况，拖累了算法的效率。

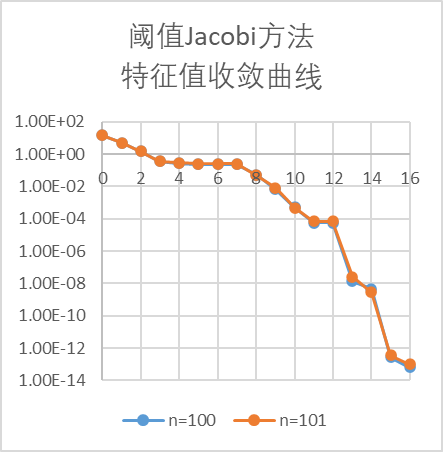
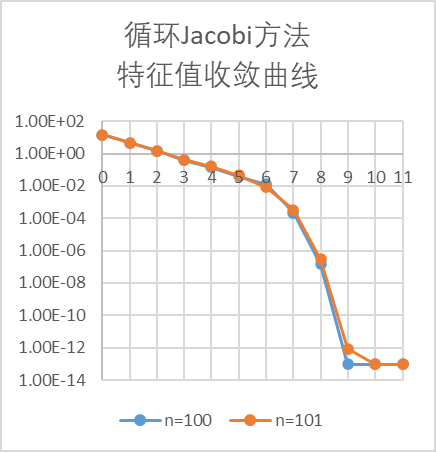
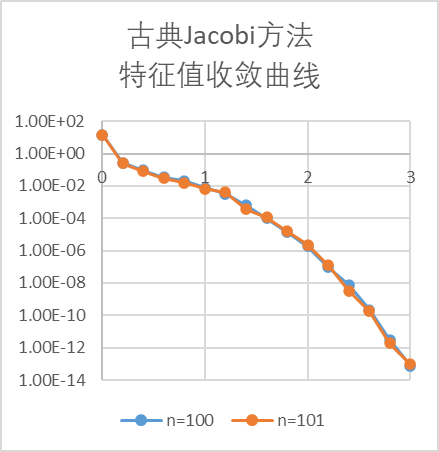


图12-14：三种Jacobi方法的特征值收敛曲线

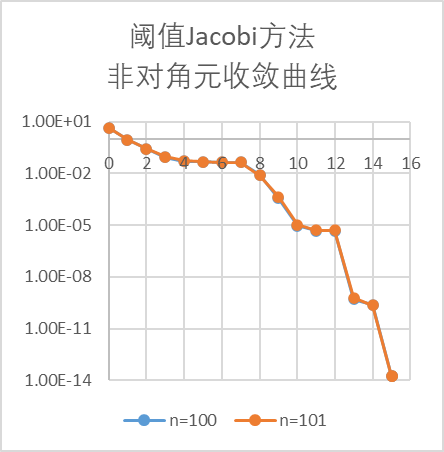
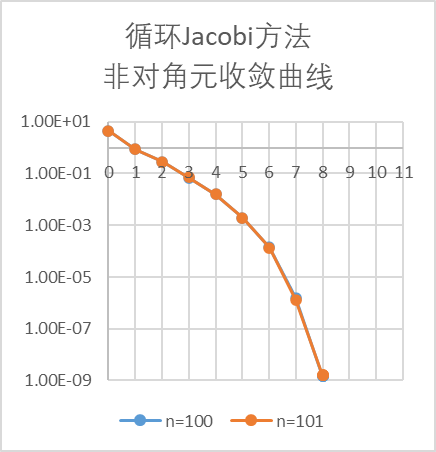
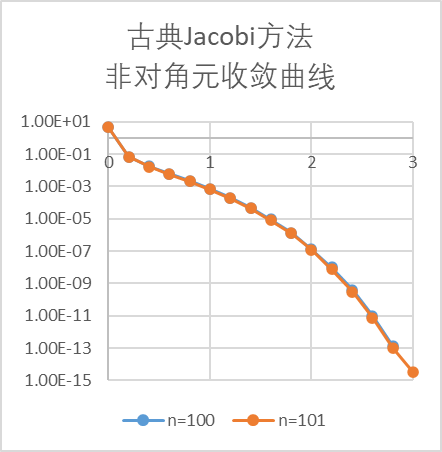


图15-17：三种Jacobi方法的非对角元收敛曲线

注1：古典Jacobi方法的横坐标为，以和另两种方法的“扫描次数”对应；

注2：程序输出格式为15位小数，误差小于5E-16后输出为0，无法在对数图上绘制。

而从收敛曲线中可见，三种Jacobi方法都有很好的收敛性，因此在决定选用何种方法时，只需考虑耗时的因素，选择阈值Jacobi法即可。

此外，关于阈值Jacobi法，在数值实验时注意到：选择较大的阈值时，算法能更快停机，但停机时的误差表现会变差，**因此需要根据实际需求（需要精度还是速度）选择阈值。**另外由于方法的特性（达到阈值后将其降低），阈值Jacobi方法的收敛曲线中出现了比较明显的“阶梯”，十分有趣。

**第六题**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 矩阵大小 | n=100 | | | n=101 | | |
| 方法 | Sturm序列二分法 | 附加一步反幂法 | | Sturm序列二分法 | 附加一步反幂法 | |
| C++(PLU) | Matlab | C++(PLU) | Matlab |
| 特征值 平均误差 | 4.06E-09 | 1.52E-05 | 4.16E-09 | 3.76E-09 | 4.80E-06 | 3.70E0-9 |

表4：Sturm序列二分法、及附加一步反幂法后的计算结果

从表中数据来看，Sturm序列二分法是本次实验中特征值误差表现最好的算法。但本次实验所求解的是矩阵是三对角阵，若对一般矩阵应用此方法，还要考虑“三对角化”时的舍入误差造成的影响，方法的表现会在一定程度上变差。

对于算法的结果执行一步反幂法，使用C++编写的程序（使用lapacke库中的“?gesv”函数解线性方程组，此函数内部由PLU分解实现）计算，反而导致了特征值误差增加；而使用Matlab程序计算（解线性方程组由“左除”运算实现，经查找资料[3]，它对于上Hessenberg矩阵使用了特别的算法），特征值的误差大体不变。此外，笔者也用自己编写的高斯-列主元消元算法进行了尝试，其表现比使用lapacke库函数更差

**因而，如果想要用反幂法来改善特征值精度，需要寻找更健壮的解线性方程组的算法，或是使用精度更高的浮点数据类型。**

**第七题**

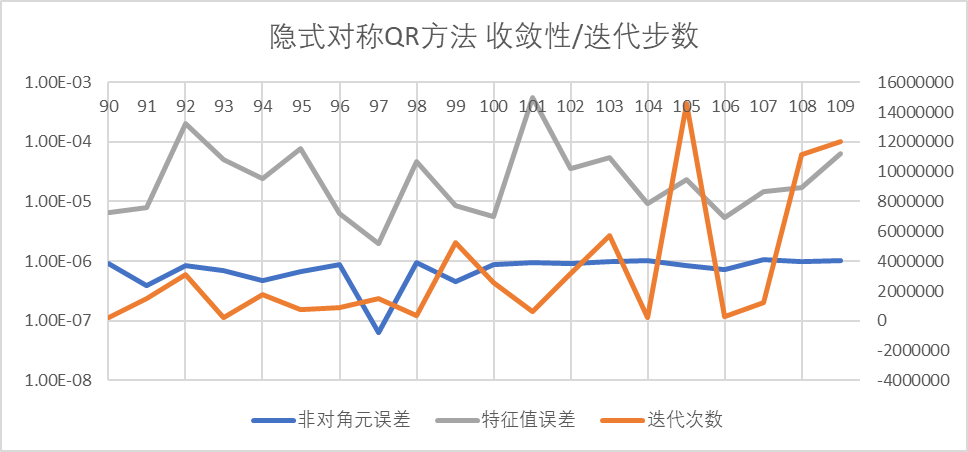


图18：隐式对称QR方法 收敛曲线迭代步数曲线

从图中可以看到，三对角矩阵的隐式QR方法（讲义[1]P123）能达到较好的收敛度。但无论是迭代步数还是特征值误差，都随着矩阵尺寸有较大幅度的波动，且并未体现出相关性及周期性等，其原因有待进一步探究。

**第八题**

|  |  |  |  |
| --- | --- | --- | --- |
| 真实值 | 1040 | 9.9×1019 | 9.81818×10−1 |
| Jacobi法计算结果 | 1.00000E+40 | 9.90000E+19 | 9.81818E-01 |
| eig()计算结果 | 1.0000E+40 | -1.9286E+23 | 0.9900 |

表5：Jacobi方法与“eig”命令（QR方法）计算结果对比

如表，Jacobi方法很准确地给出了六位有效数字地特征值，而Matlab的eig()命令的结果不尽如人意，尤其是在第二大特征值出现了很严重的偏差。尤其是在Matlab的反幂法给出相对较好的结果后，这也提醒了笔者，不能迷信于某一个软件的计算结果，应当多方比较。

**结语**

在本次实验中，笔者使用了幂法、反幂法、同时迭代法、三种Jacobi方法、Sturm序列二分法、隐式QR方法等算法，比较了它们求解同一个三对角矩阵的特征值问题的表现，尤其是发现了一些有待进一步探究的现象现象（例如反幂法不稳定的表现、QR方法收敛速度的很大波动等），加深了笔者对于这些算法的认识，更有利于以后的学习。

**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

[3] <https://ww2.mathworks.cn/help/matlab/ref/mldivide.html>