数值计算实验报告

16340219 王亮岛

第一题

请实现下述算法，求解线性方程组 Ax=b，其中 A 为 nⅹn 维的已知矩阵，b 为 n 维的已 知向量，x 为 n 维的未知向量。 （1）高斯消去法。 （2）列主元消去法。 A 与 b 中的元素服从独立同分布的正态分布。令 n=10、50、100、200，测试计算时间并绘制曲线。

**算法设计**：

首先创建n\*n的服从独立标准正态分布的矩阵A：



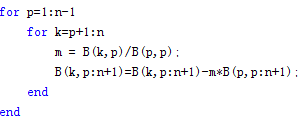
创建n\*1的服从独立标准正态分布的矩阵b：



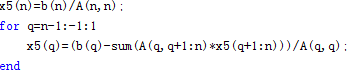
高斯消元法：

将A，b合在一起创建一个增广矩阵B

完成n-1次消元：

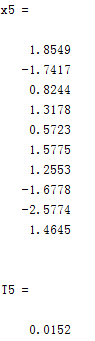


重新复制A,b矩阵，通过求解线性方程组得到回代公式：



通过n-1次回代得出结果x5。

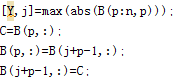
通过随机生成10\*10的矩阵A和10\*1的矩阵b得出结果：



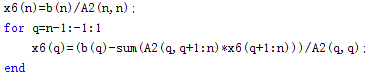
耗时0.0152秒。

列主元消去法：

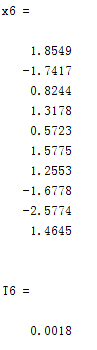
先按列挑选主元，在第k次消元之前在k\*k矩阵中找出最大主元所在行，将其与第k行交换：



然后按照高斯消元法的回代过程求解：

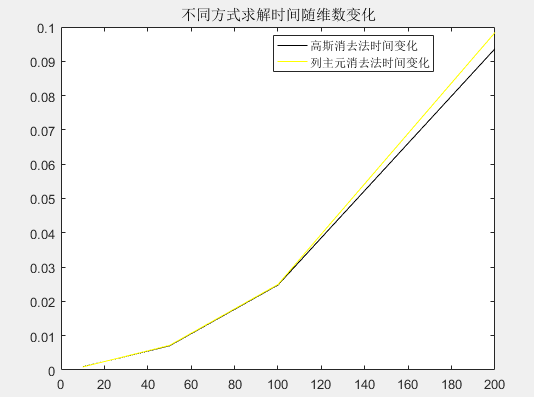


使用同一个A和b矩阵求解：



求解时间为0.0018秒。

通过生成N=10,50,100,200的矩阵画出时间随矩阵维度变化的图，比较高斯消去法和列主元消去法的求解时间：



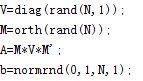
由于高斯消去法和列主元消去法的时间复杂度一样，两者时间差别主要在与列主元消去法需要机器选择主元，因此，随着维度的增加，机器选择主元的次数增加，所需的时间也就增加。

第二题

请实现下述算法，求解线性方程组 Ax=b，其中 A 为 nⅹn 维的已知矩阵，b 为 n 维的已 知向量，x 为 n 维的未知向量。 （1）Jacobi 迭代法。 （2）Gauss-Seidel 迭代法。 （3）逐次超松弛迭代法。 （4）共轭梯度法。 A 为对称正定矩阵，其特征值服从独立同分布的[0,1]间的均匀分布；b 中的元素服从独立同分布的正态分布。令 n=10、50、100、200，分别绘制出算法的收敛曲线，横坐标为迭代步数，纵坐标为相对误差。比较 Jacobi 迭代法、Gauss-Seidel 迭代法、逐次超松弛迭代法、共轭梯度法与高斯消去法、列主元消去法的计算时间。改变逐次超松弛迭代法的松弛因子，分析其对收敛速度的影响。

**算法设计：**

首先按要求生成矩阵A，b矩阵服从标准正态分布：



Jacobi 迭代法：

生成迭代矩阵J：



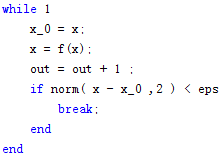
由于特征值要服从[0,1]间均匀分布，所以我加入判断条件：若迭代矩阵最大特征值的绝对值大于1，则迭代不收敛：



接着生成f = INV(D)\*b，并定义迭代函数:

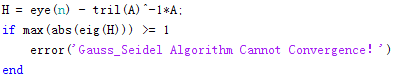


开始迭代，在这里，我定义了误差小于10的-6次方：

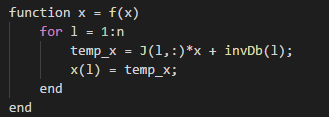


高斯赛德尔迭代：

对高斯赛德尔矩阵是否收敛的判断，我参考雅克比迭代，若迭代矩阵的最大特征值的绝对值大于1，则不收敛：



在这里，高斯赛德尔迭代其实是雅克比迭代的改进，高斯赛德尔迭代在计算每一个x分量时用新值不用旧值，同样，对迭代矩阵判断其最大特征值是否大于1，但此时的迭代矩阵为：H = eye(n) - tril(A)^-1\*A。而针对每次使用新值的函数为：

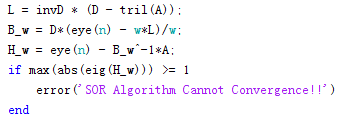


每次通过l次的雅克比迭代用x的最新值迭代。

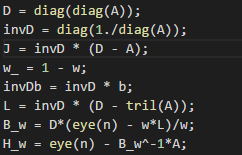
开始迭代时，与雅克比迭代相同，同样的，我设置了10的-6次方作为误差。

SOR迭代：

同样，对逐次超松弛迭代是否收敛的判断与雅克比迭代相同：

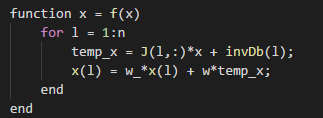


逐次超松弛迭代又是对高斯赛德尔迭代的一种改进，因此需要用到高斯赛德尔迭代的方法。首先求迭代矩阵H\_w：



通过判断H\_w的最大特征值是否大于1，确定矩阵是否收敛。

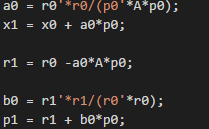
在迭代函数中，首先用高斯赛德尔迭代定义辅助量temp\_x，然后：x(l)=(1-w)\*x(l)+w\*temp\_x，迭代函数如下：



CG迭代：

用共轭梯度法迭代：

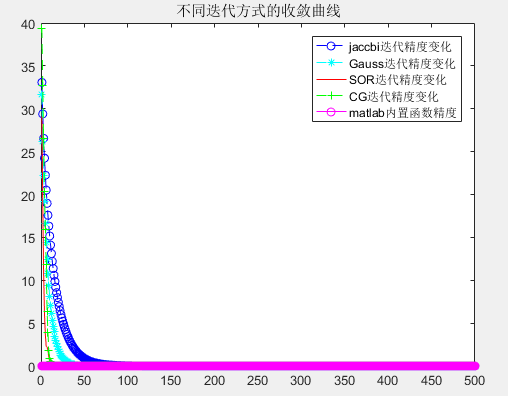
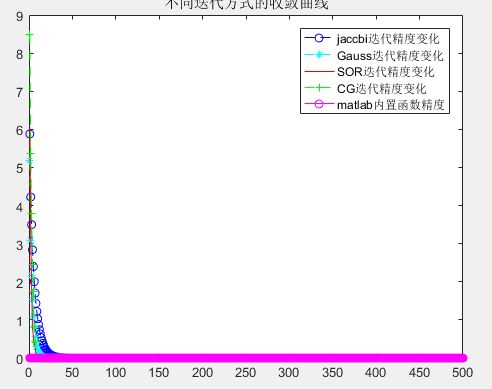
1. 首先取x(0)属于R(n)，计算r0 = b - A\*x0; p0 = r0;
2. 对k=0,1...计算：



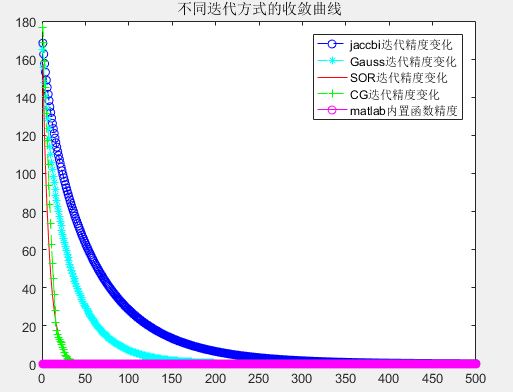
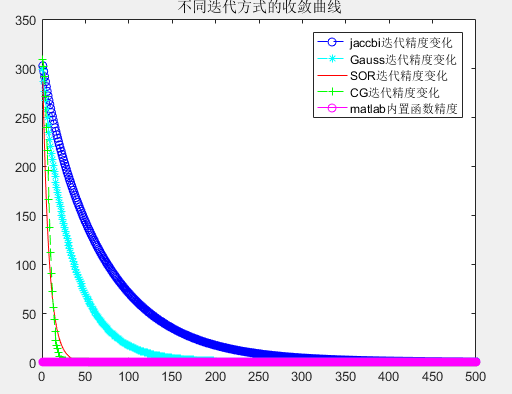
1. 然后将x1,r1,p1重新赋给x0,r0,p0循环，直到迭代结束。

收敛曲线如下：

10\*10矩阵，50\*50收敛：



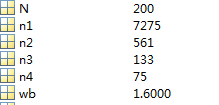
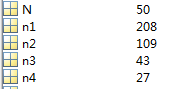
100\*100收敛，200\*200收敛：



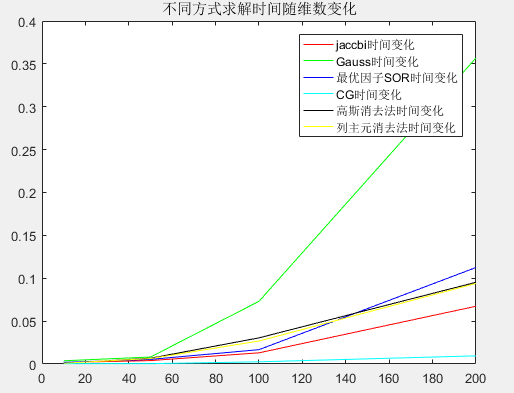
可以看出收敛速度最快的是共轭梯度法，而最慢的是雅克比迭代。

迭代步数比较，n1,n2,n3,n4分别为雅克比迭代，高斯赛德尔迭代，逐次超松弛迭代，共轭梯度法：

10\*10矩阵，50\*50矩阵，100\*100矩阵，200\*200矩阵



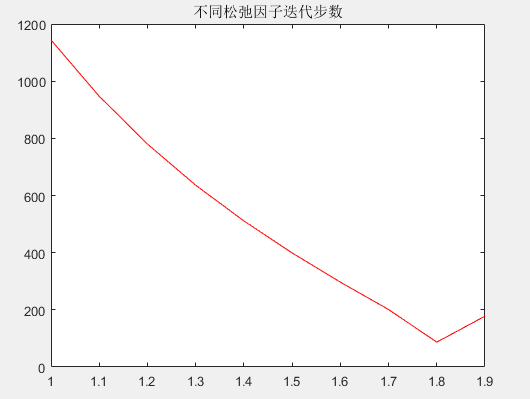
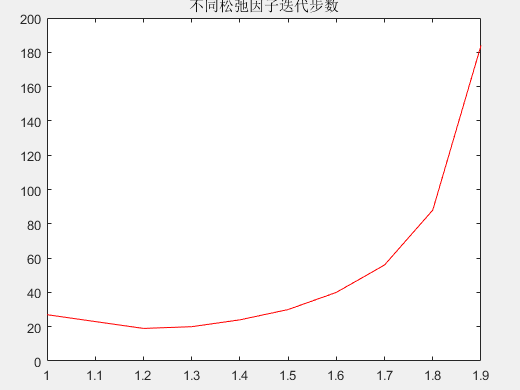
计算时间比较：



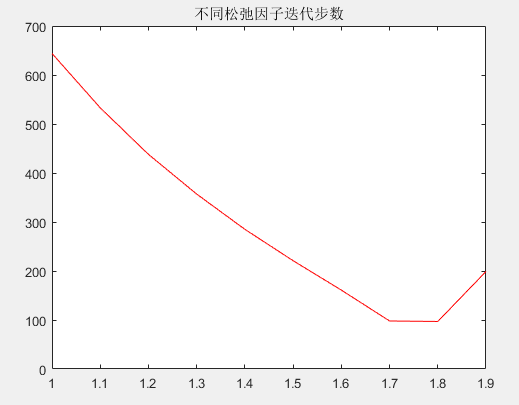
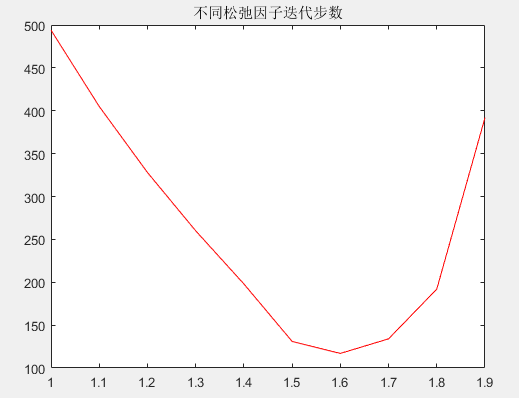
横坐标为维度，纵坐标为时间，雅克比迭代虽然收敛速度慢，但是其运算简单，因此计算时间会比高斯赛德尔迭代快，由于每次测试时间变化不一样，我只是选取了其中一次测试结果。

改变逐次超松弛因子判断其对收敛速度影响：

10\*10矩阵，50\*50矩阵：



100\*100矩阵，200\*200矩阵：



我选取的松弛因子的区间为[1,1.9]，因为实验发现超过2的松弛因子的CG迭代是不收敛的。可以看出对于不同维度的矩阵的最适松弛因子不同，随着松弛因子的值的增加，曲线总体呈现先减后增。

第三题:

在 Epinions 社交数据集（https://snap.stanford.edu/data/soc-Epinions1.html）中，每个网络节点可以选择信任其它节点。借鉴Pagerank 的思想编写程序，对网络节点的受信 任程度进行评分。在实验报告中，请给出伪代码。

伪代码如下：

L=邻接矩阵

N=网页节点数;

q=阻尼系数;

构建矩阵D为对角线上元素D(k,k)表示第k个点出度的对角矩阵：

d=sum(L,2);

D=diag(d);

构建基本转移矩阵：M= L' \* D^(-1);

处理悬挂网页的随机修正，其中矩阵a为第i个网页是否为悬挂网页，若是则为1：

e=ones(N,1);

a=(d==0);

S=M+e\*a'/N;

构造概率转移矩阵：

G= qS + (1-q)e\*e'/N;

H0=初始化全零矩阵；

H1=初始化权重向量全1矩阵；

While H1和H0的误差大于设置误差 //使用幂法

H0=H1；

H1=G\*H0；

End

最后将H1倒序排序得出rank和index。

附录：

文件介绍：

CG.m文件为实现共轭梯度法的函数。

gaus.m文件为实现高斯消元法的函数。

liezhu.m文件为实现列主元消去法的函数。

gauss.m文件为实现高斯赛德尔迭代的函数。

jaccbi.m文件为实现雅克比迭代的函数。

SOR.m文件为实现逐次超松弛迭代的函数。

generic.m文件为生成矩阵的函数。

comTime.m文件为实现不同维度因不同方法消耗时间的函数。

test.m文件为为实现不同方式的收敛曲线。

test\_wb.m文件为实现不同维度的不同收敛因子对收敛步数影响的函数。

PageRank.m文件为实现PageRank方法对节点排序。