Numerical Computation Methods

Homework 1

学院: 数据科学与计算机学院

班级: 软工 3 班

姓名: 李天译

学号: 16340122

Contents

1 直接法

1.1 问题描述

实现高斯消去法和列主元消去法，求解线性方程组 $Ax = b$, 其中 A 为 $n \times n$ 维的已知矩阵,b 为 n 维的已知向量,x 为 n 维的未知向量.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.

1.2 Gauss 消去法

1.2.1 算法设计

根据高斯消去法的原理，算法主要分为两步：

- 消元：

$$m_{ik} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}$$

$$k = 1, 2, \ldots, n-1; i = k+1, k+2, \ldots, n$$

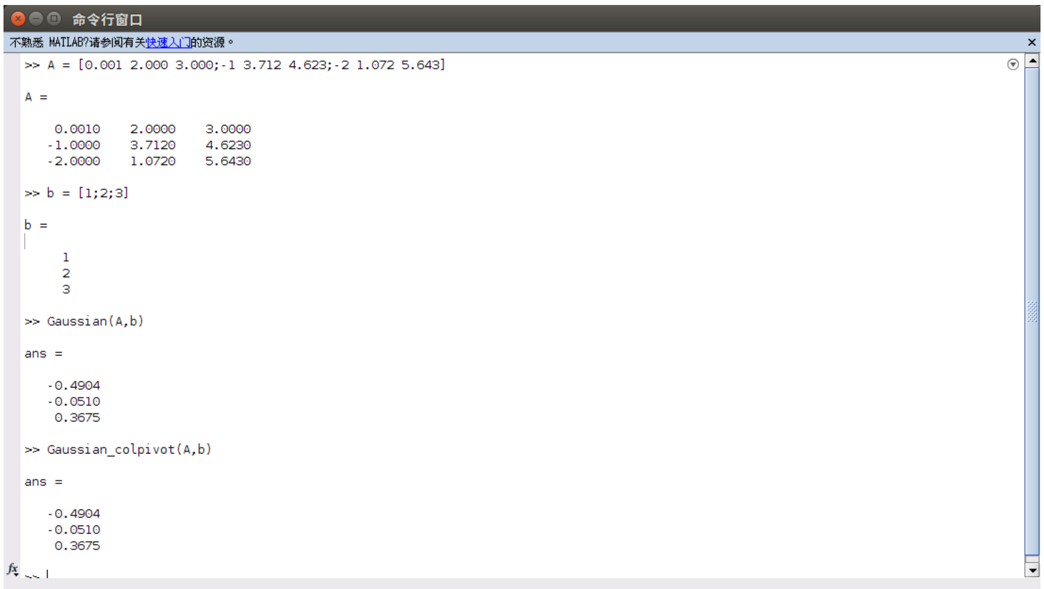
再将 $-m_{ik}$ 乘以原式的第 k 行，加到第 i 行上.

- 回代:

$$x_n = b_n^{(n)} / a_{nn}^{(n)}$$
$$x_k = \frac{b_k^{(n)} - \sum_{j=k+1}^{n} a_{kj}^{(n)} x_j}{a_{kk}^{(k)}}$$
$$k = n-1, n-2, \ldots, 1$$

1.2.2 数值实验

理论上高斯消去法的数值稳定性较差，但使用教材 5.2.3 的例子实验时并没有发现与列主元消去法的计算结果有明显差别. 结果如下:



1.3 列主元消去法

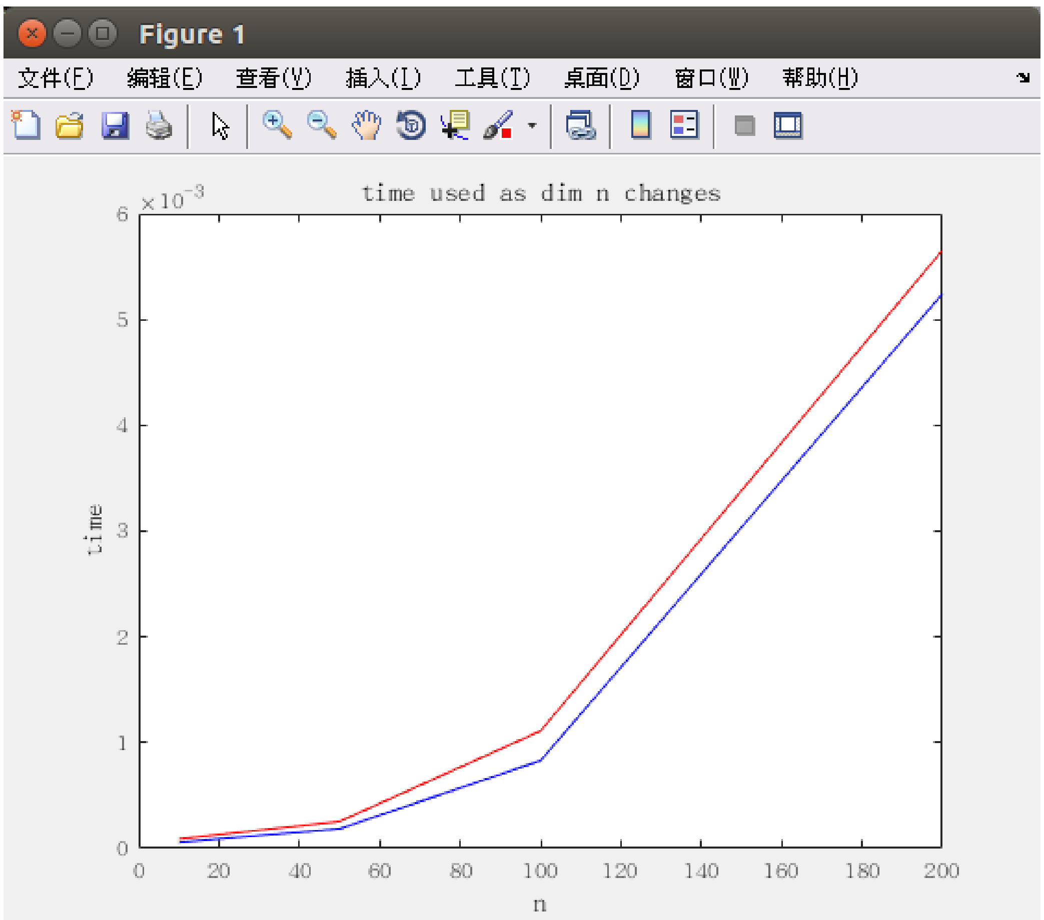
1.3.1 算法设计

在高斯消去法的基础上，为了解决高斯消去法可能出现的数值不稳定的情况，每次迭代取剩余行中绝对值最大的行作为该次迭代的列主元，其余操作与高斯消去法相同.

1.4 结果分析

使用维数 n 作为横坐标，计算时间作为纵坐标，画出了时间变化曲线。由于使用的是随机出的矩阵 A 和向量 b，每次得到的图像都有一些区别，但大体趋势都相同，且能看出列主元消去法在多数情况下要比高斯消去法耗时更长（尤其当 n 增大时），猜测这样的结果是列主元法比高斯消去法多比

较，甚至可能交换了行元素，增加了时间复杂度导致的，但时间复杂度的少许增加提高了算法的数值稳定性. 曲线的变化基本符合理论上高斯消去法的时间复杂度：$O(\frac{1}{3}n^3)$. 某次的实验结果如下图：（红色线为列主元消去法，蓝色线为高斯消去法）



2 迭代法

2.1 问题描述

实现 Jacobi 迭代法，Gauss-Seidel 迭代法，逐次超松弛迭代法和共轭梯度法，求解线性方程组 $Ax = b$, 其中 A 为 $n \times n$ 维的已知矩阵,b 为 n 维的已知向量,x 为 n 维的未知向量.A 与 b 中的元素服从独立同分布的正态分布. 令 $n = 10\ 50\ 100\ 200$, 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 ω, 分析其对收敛速度的影响.

2.2 Jacobi

2.2.1 算法设计

根据公式有:

$$x^{(0)} = \left(x_1^{(0)}, x_2^{(0)}, \cdots, x_n^{(0)}\right)^T$$

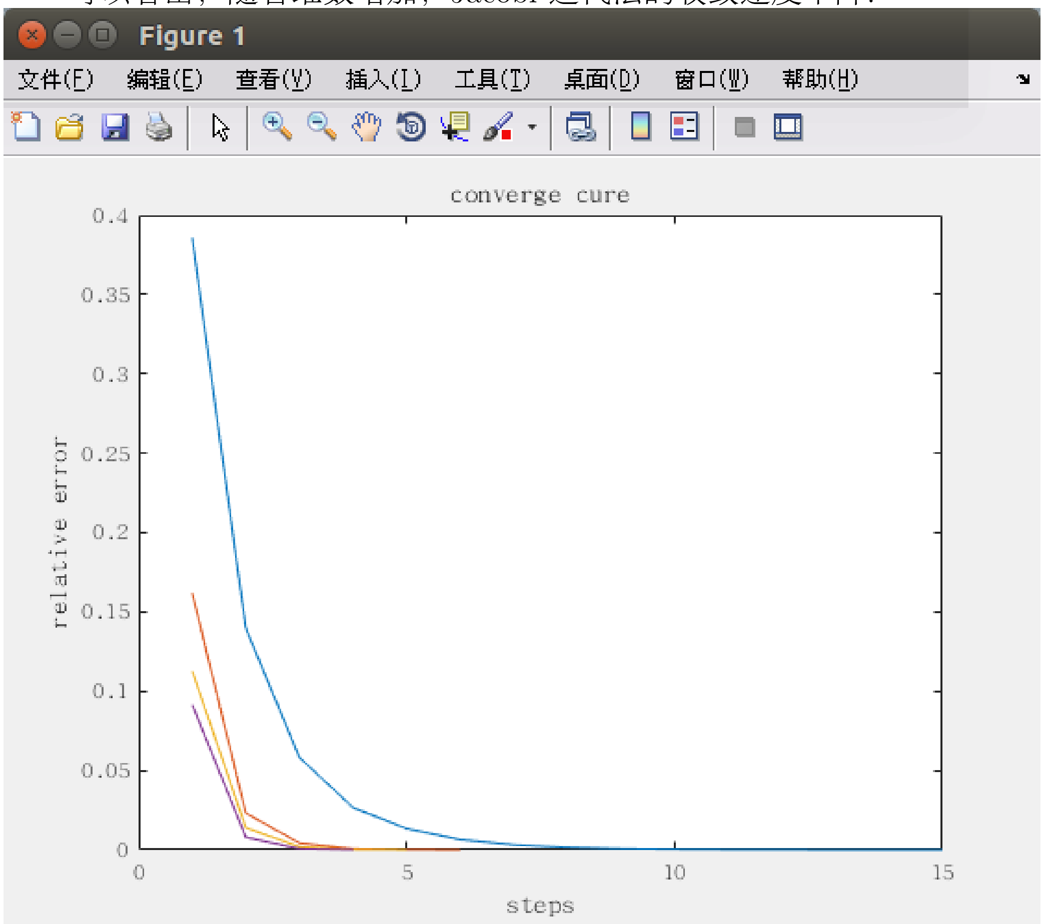
$$x_i^{(k+1)} = \frac{b_i - \sum_{j=1,j\neq i}^{n} a_{ij} x_j^{(k)}}{a_{ii}}$$

$$i = 1, 2, \ldots, n; k = 0, 1, \ldots$$

直到误差很小时停止.

2.2.2 数值实验

可以看出，随着维数增加，Jacobi 迭代法的收敛速度下降.



2.3 Gauss-Seidel

2.3.1 算法设计

与 Jacobi 法类似，但在每一步迭代中用到当前已更新值的信息加速收敛. 根据公式有:

$$x^{(0)} = \left(x_1^{(0)}, x_2^{(0)}, \cdots, x_n^{(0)}\right)^T$$

$$x_i^{(k+1)} = x_i^{(k)} + \Delta x_i$$

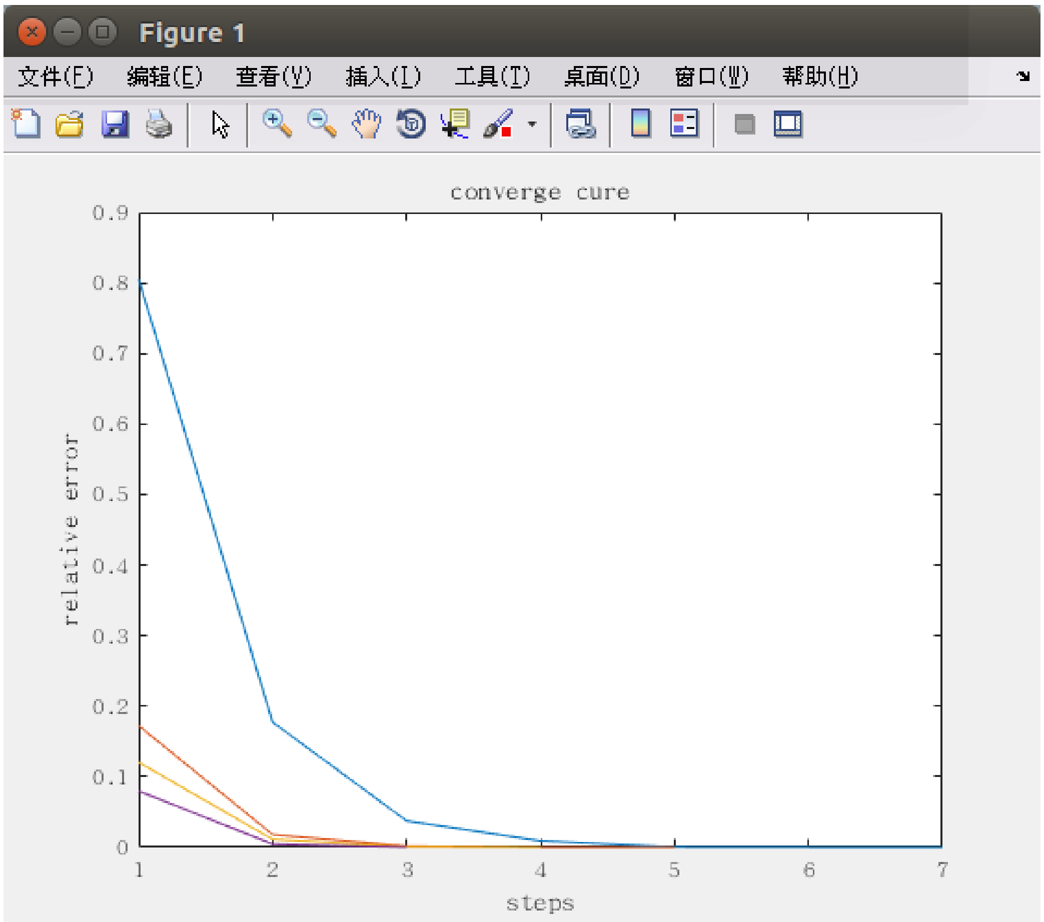
$$\Delta x_i = \frac{b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i}^{n} a_{ij} x_j^{(k)}}{a_{ii}}$$

$$i = 1, 2, \ldots, n; k = 0, 1, \ldots$$

直到误差很小时停止.

2.3.2 数值实验

可以看出，随着维数增加，Gauss-Seidel 迭代法的收敛速度下降，并且对比发现，本方法误差随步长的变化较 Jacobi 迭代法更大.



2.4 SOR

2.4.1 算法设计

以 ω 作为松弛因子，在 Gauss-Seidel 方法上加以变化，得到逐次超松弛迭代法公式.

$$x^{(0)} = \left(x_1^{(0)}, x_2^{(0)}, \cdots, x_n^{(0)}\right)^T$$

$$x_i^{(k+1)} = x_i^{(k)} + \Delta x_i$$

$$\Delta x_i = \frac{\omega(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i}^{n} a_{ij}x_j^{(k)})}{a_{ii}}$$

$$i = 1, 2, \ldots, n; k = 0, 1, \ldots$$

直到误差很小时停止.

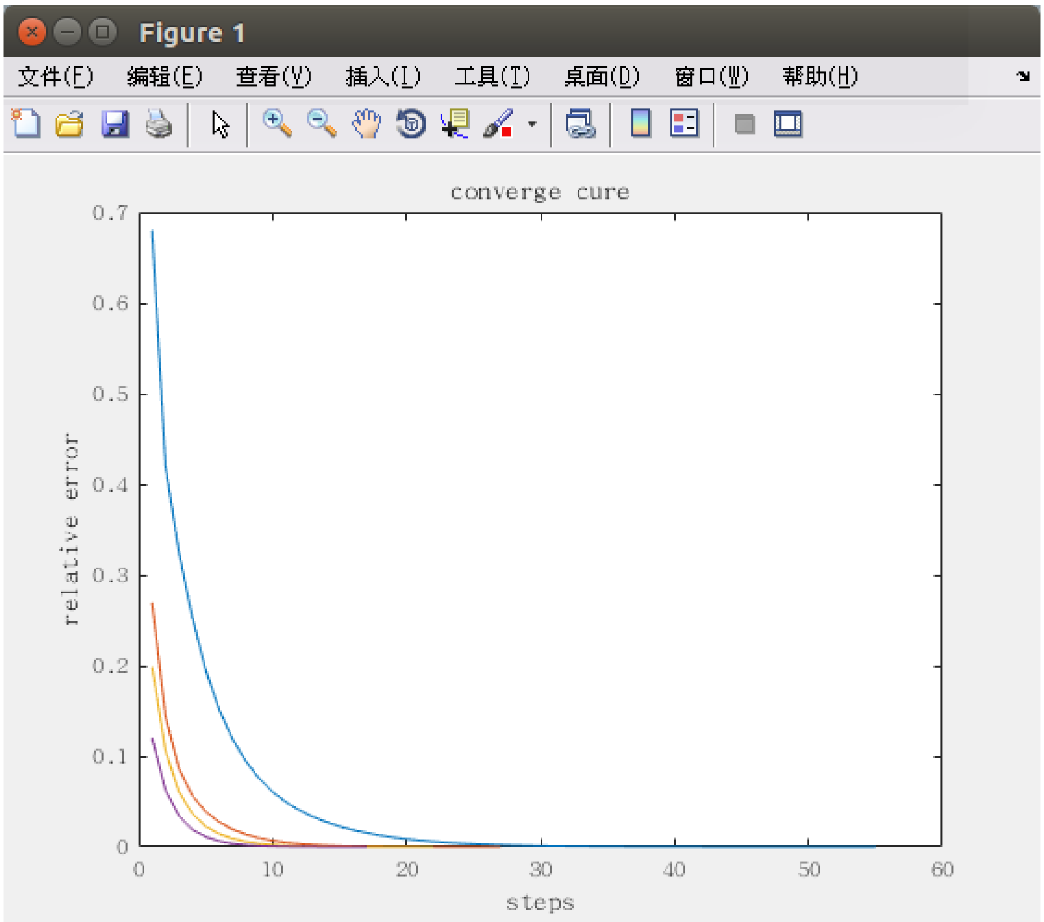
可以看出,高斯-塞德尔迭代实质上就是 $\omega = 1$ 时的逐次超松弛迭代法(SOR)

.

2.4.2 数值实验

- 收敛随维数变化:

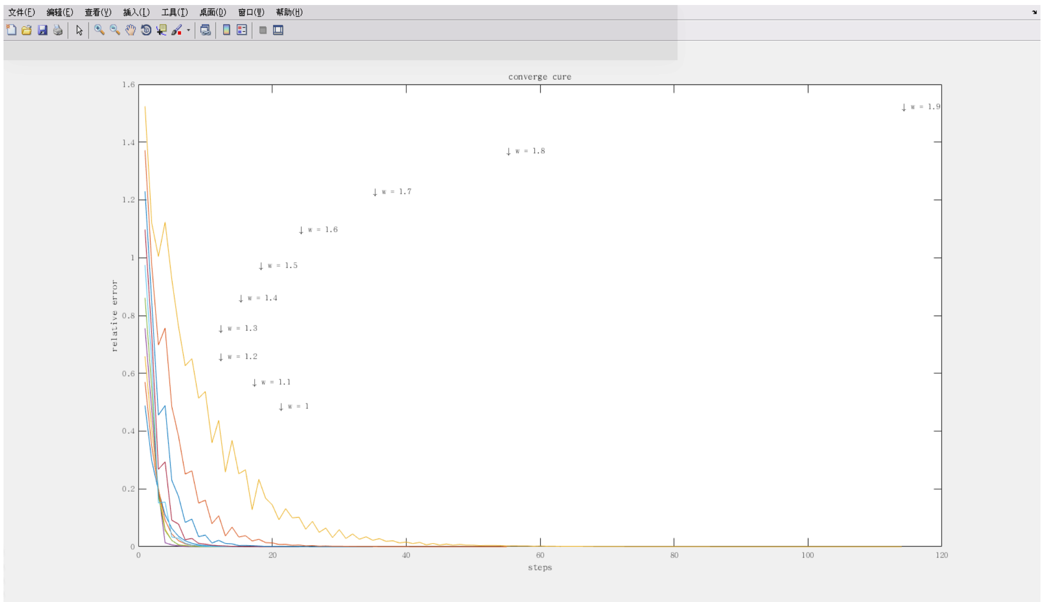
  测试中取 $\omega = 1.5$，测试的迭代速度较慢.



- 松弛因子影响迭代步长:

  使用教材中 A,b 的值为例，以 0.1 为迭代步长，在 $(1,2)$ 范围内（超松弛的参数范围）改变参数 ω 的值，测试收敛所需步数，并标注在横坐标的对应位置.

  可以看出 $\omega = 1.3$ 是求解这个方程的近似最优松弛因子，且收敛所需步数随松弛因子的值增加而先减少再增加.



2.5 CG

2.5.1 算法设计

根据公式可得:

取 $x^{(0)} = 0$, $r^{(0)} = b - Ax^{(0)}$, 取 $p^{(0)} = r^{(0)}$

对 $k = 0, 1, \ldots,$ 计算:

$$\alpha_k = \frac{(r^{(k)}, r^{(k)})}{(p^{(k)}, Ap^{(k)})}$$

$$x^{(k+1)} = x^{(k)} + \alpha_k p^{(k)}$$

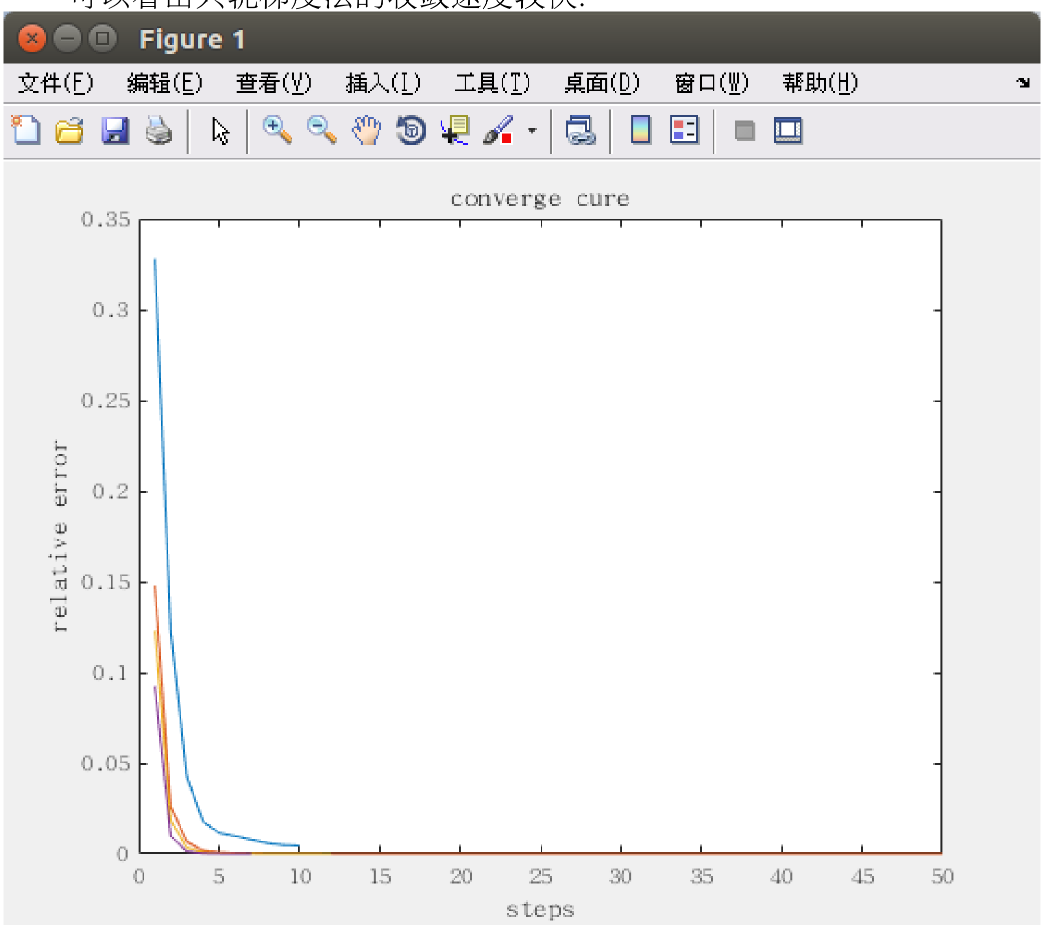
$$r^{(k+1)} = r^{(k)} - \alpha_k Ap^{(k)}, \beta_k = \frac{(r^{(k+1)}, r^{(k+1)})}{(r^{(k)}, r^{(k)})}$$

$$p^{(k+1)} = r^{(k+1)} + \beta_k p^{(k)}$$

直到误差很小时停止.

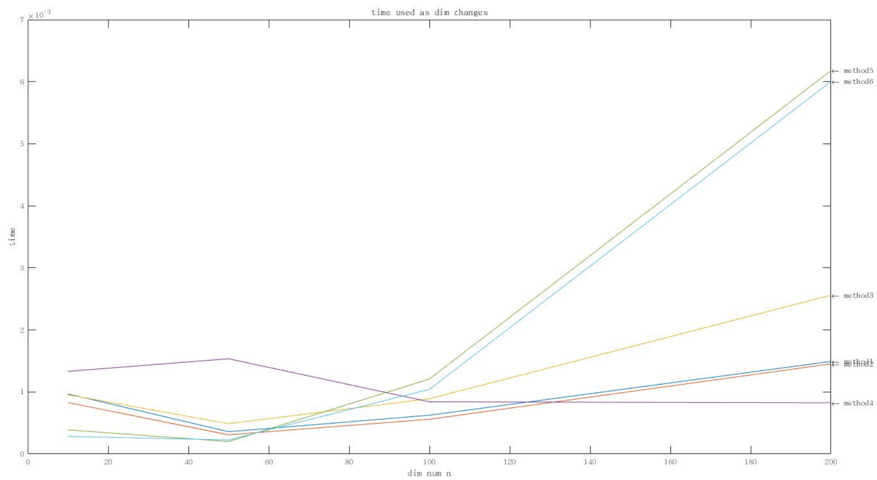
2.5.2 数值实验

可以看出共轭梯度法的收敛速度较快.



2.6 结果分析

比较上面提到的六种求线性方程解的方法，在维数改变时的计算时间，实验结果如下：



几点结论：

- 高斯消去法和列主元消去法的计算时间差别不大，Jacobi 和 Gauss-Seidel 迭代法的计算时间差别不大.
- 随着维数增加，直接法的两种方法计算时间变得显著高于迭代法的几种方法，且能看出直接法的两种方法受到维数影响变化更大.
- CG 法的计算时间受维数影响不大，且在维数增加时显著优于其他算法.
- 实验中选定的松弛因子 $\omega = 1.3$ 下，SOR 比其他迭代法的方法计算时间更长，该因子的选取未能体现出 SOR 方法的优势.

3 Page Rank

3.1 问题描述

在Epinions 社交数据集中, 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.

3.2 算法设计

$$v^{(k+1)} = (1-\beta)Av^{(k)} + \beta\frac{1}{n}I$$

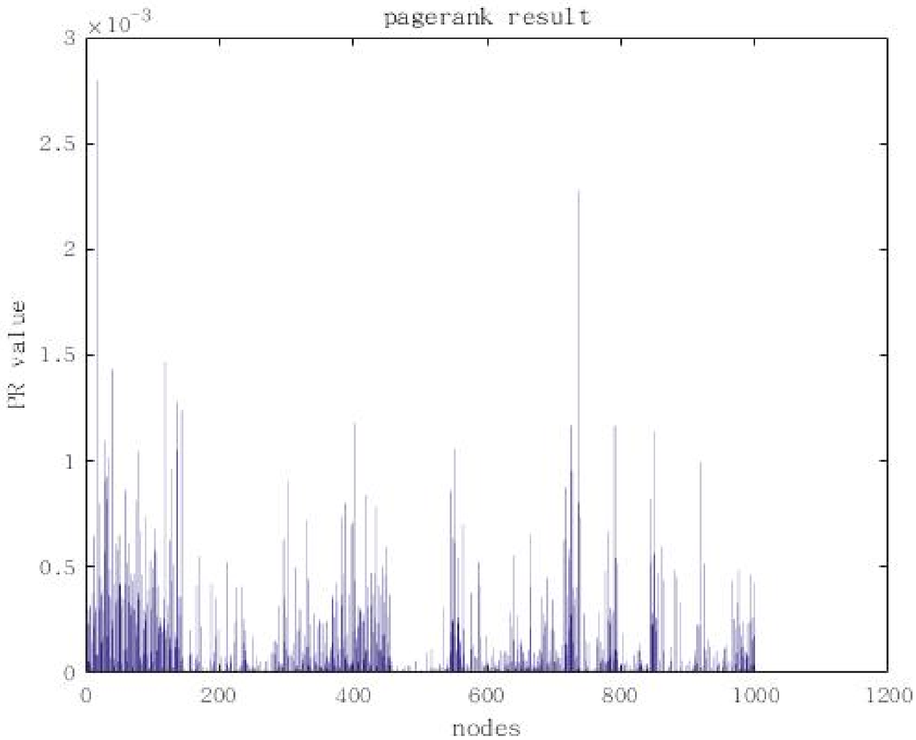
$k = 1, 2, \ldots,$ 当误差足够小时为止.

```
% Pseudocode :
edge E(x,y)
nodes N
matrix A
min_error error
factor beta = 0.05
n x 1 matrix, all the elements are 1 I

for every edge E(x,y)
    A(x,y) = 1;
end
```

```
for every column c
    A(:,c) = A(:,c) / (sum of c);
end

while error of this step > error
    newv = (1-beta) * A * v + beta * 1/N * I;
    v = newv;
end
```

3.3 结果分析

前 500 个节点的排序结果如下:



但测试发现, 这些节点的 pr 值之和并不为 1, 猜测是由于多次迭代的舍入误差导致的.