# 数值计算报告

**16340150**

**刘俊峰**

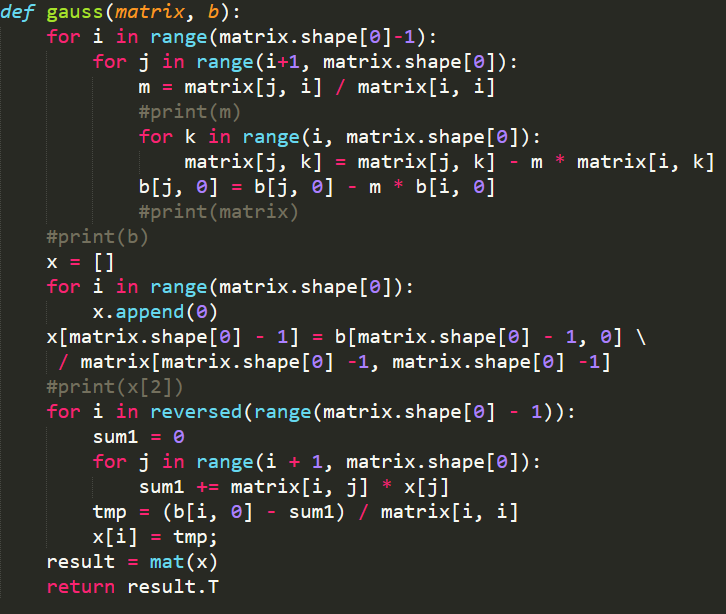
**16软件工程教务二班**

## 问题描述

1. 用直接法解线性方程组。其中A,b中的元素服从正态分布。令n=10，50，100，200，测试计算时间并绘制曲线
2. 用迭代法解线性方程组。其中A,b中的元素服从正态分布。令n=10，50，100，200，分别绘制算法的收敛曲线，横坐标时迭代步数，纵坐标是相对误差。
3. 比较上面各种方法的计算时间
4. 改变SoR算法的松弛因子，分析对收敛速度的影响
5. PageRank算法伪代码

## 算法设计

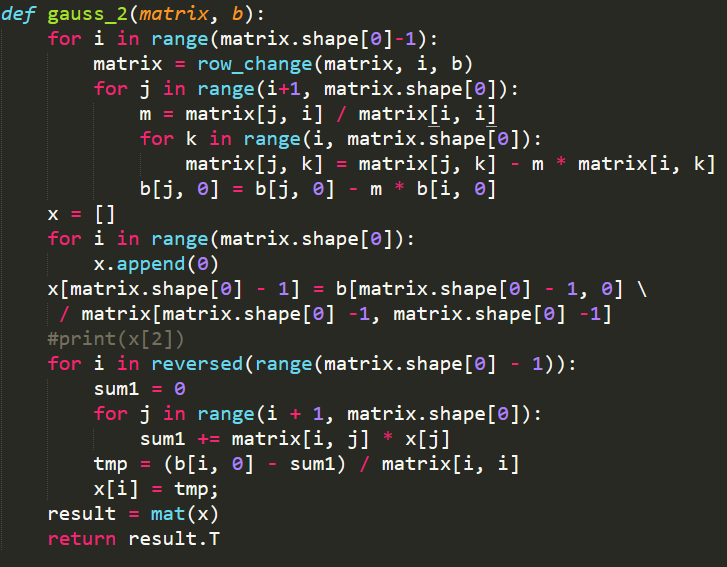
1. **直接法解线性方程组**：
2. 高斯消元



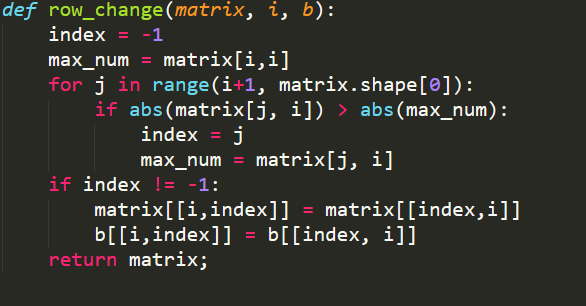
如上图源码。高斯消元法主要分两个步骤：消元和回代

进而算出结果

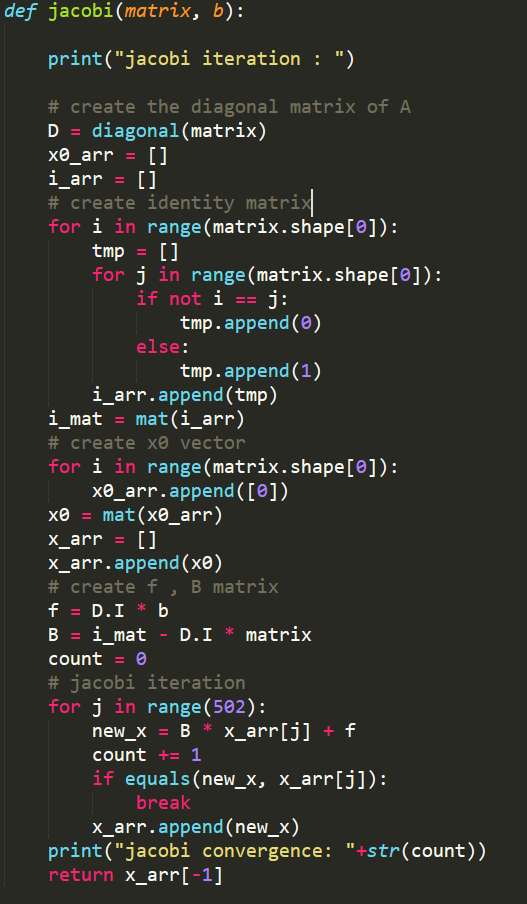
1. 列主元消去法



与高斯消元法类似。不同点在于列主元消去在每一步消元时找到主元绝对值最大的行，并用它作为关键值进行消元。行交换的代码如下



1. **迭代法解线性方程组**
2. Jacobi迭代法



雅可比迭代的关键在于找到递归迭代的表达式即：x\_k = B\_0 \* x\_(k-1) + f

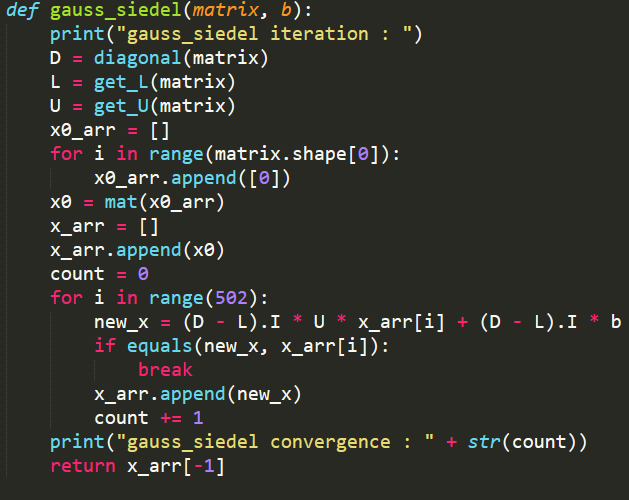
中的参数。由矩阵变换我们可以得知：

B = I – D’ \* A (D是A的对角元素组成的矩阵)

f = D’ \* b

(\*) 雅可比迭代的收敛条件很苛刻，当无法收敛时容易造成值越界错误。可以采用try/catch的结构或者对迭代次数进行限制。

1. Gauss-Seidel迭代：



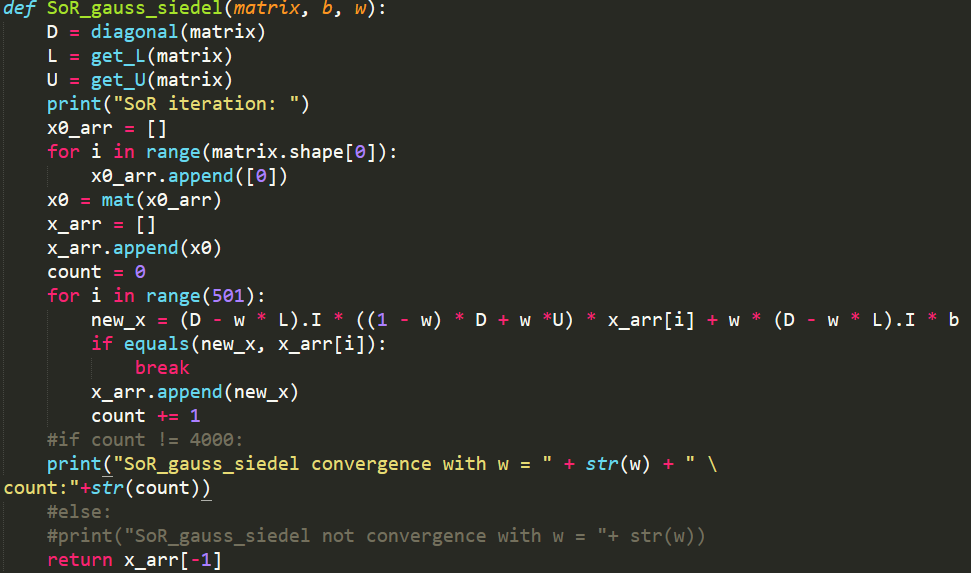
与雅可比迭代类似，但是高斯赛德尔迭代的优秀之处在于，将更新过的x的值代入到本次迭代其他元素的计算中，大大降低的迭代步数，提高收敛效率。

迭代公式为：

x\_n = (D – L)’ \* U \* x\_(n-1) + (D – L)’ \* b

其中D是对角元素组成的矩阵。L是下三角元素的相反数组成的矩阵，U则相反。

1. SoR迭代



逐次超松驰迭代是高斯赛德尔迭代的加权形式，通过改变参数w的值改变算法的收敛速度。迭代公式：

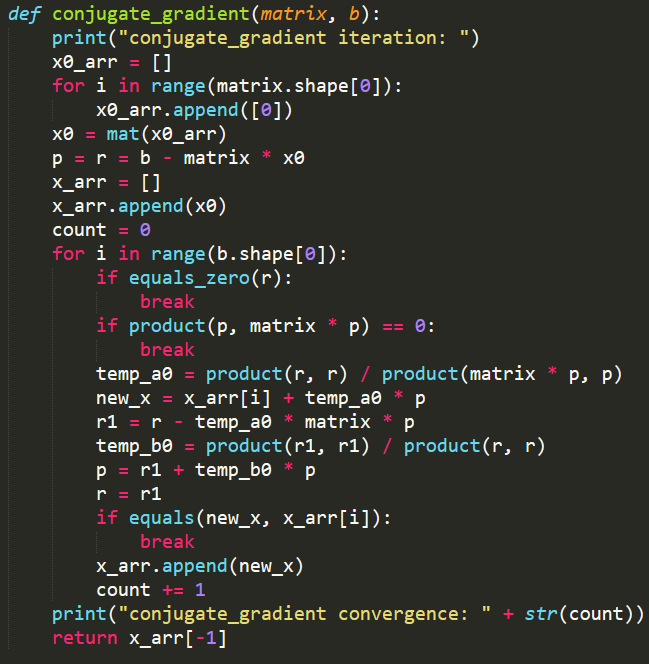
x\_n = (D – wL)’ \* ((1-w) \* D + w \* U) \* x\_(n-1) + w \* (D – w \* L)’ \* b

当w=1时，收敛速度和高斯一样

当w<1时，收敛速度慢

当w>1时，收敛速度快

1. CG方法



共轭梯度法和最速下降法类似，用于解对称正定的矩阵十分有效。找到一组搜索向量，并通过此找到下一个迭代的x

x\_0=[0]

取p\_0=r\_0=b-A\*x\_0,

Temp\_a = (r\_k, r\_k) / (p\_k, A \* p\_k)

x\_(k+1) = x\_k + Temp\_a \* p\_k

r\_(k+1) = r\_k – Temp\_a \* A \* p\_k

Temp\_b = (r\_(k+1), r\_(k+1)) / (r\_k, r\_k)

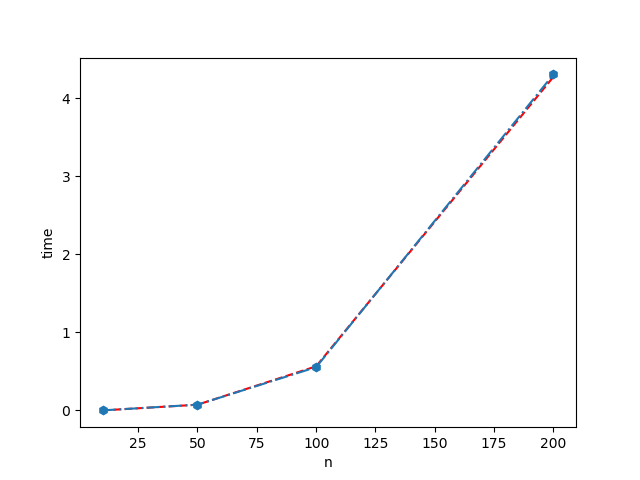
p\_(k+1) = r\_(k+1) + Temp\_b \* p\_k

当r\_k=0或者(p\_k, A \* p\_k)=0时停止。或者找到符合精度的x也可以停止。

CG方法是迭代法中收敛较快的方法。

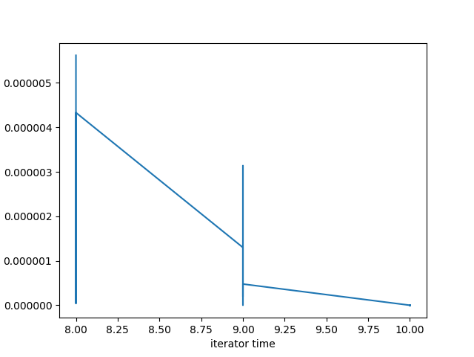
## 数值实验

1. **直接法的时间对比**

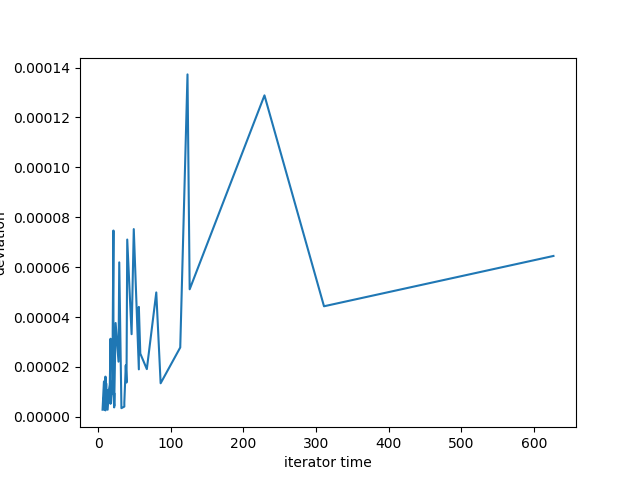


蓝色线是高斯消元，红色线是列主元消去。

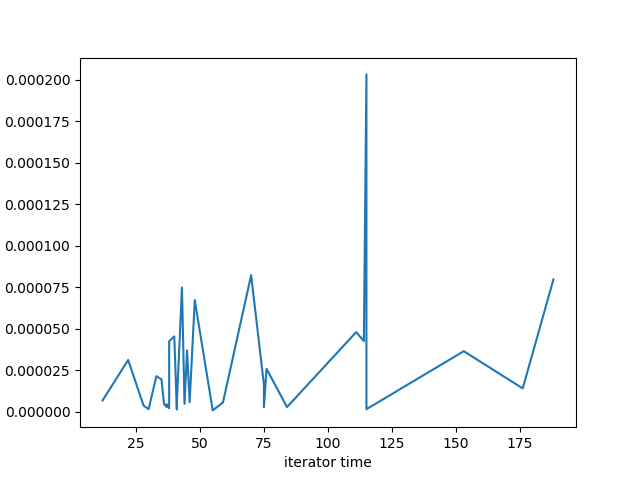
1. **迭代算法的迭代步数和相对误差的关系**
2. n=10
3. CG方法



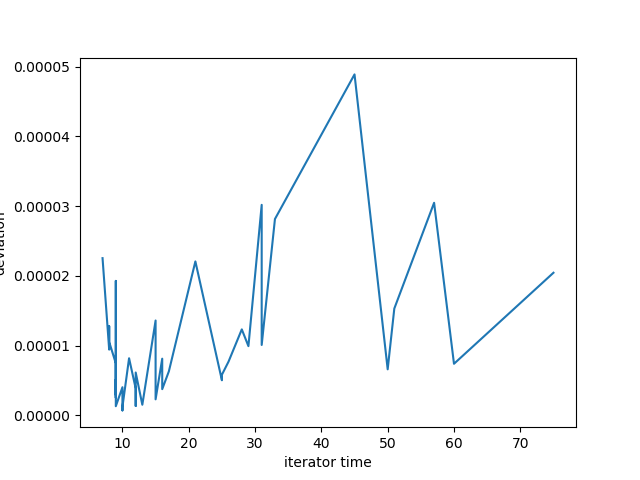
1. Gauss-Siedel



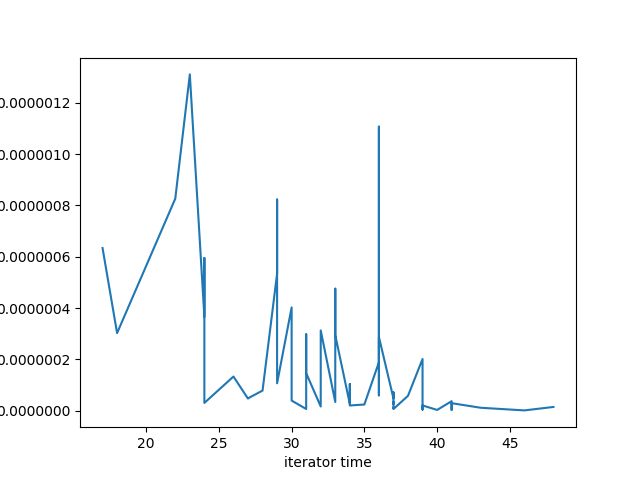
1. Jacobi



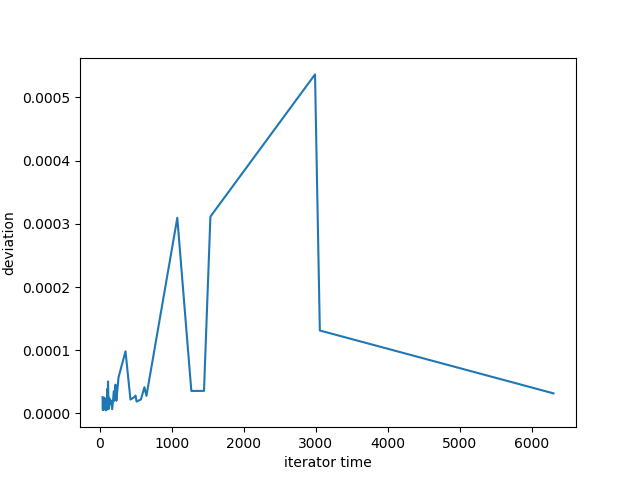
1. SoR(w=1.1)



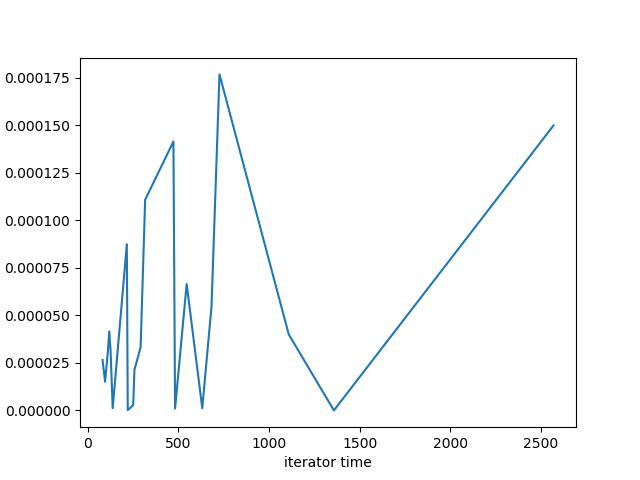
1. n = 50
2. CG方法



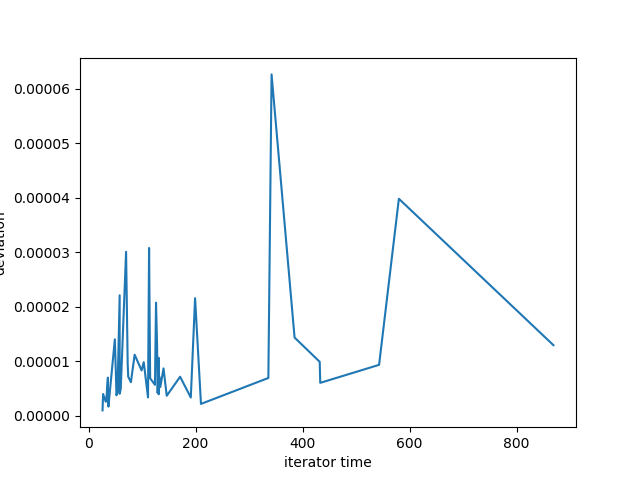
1. Gauss-Siedel



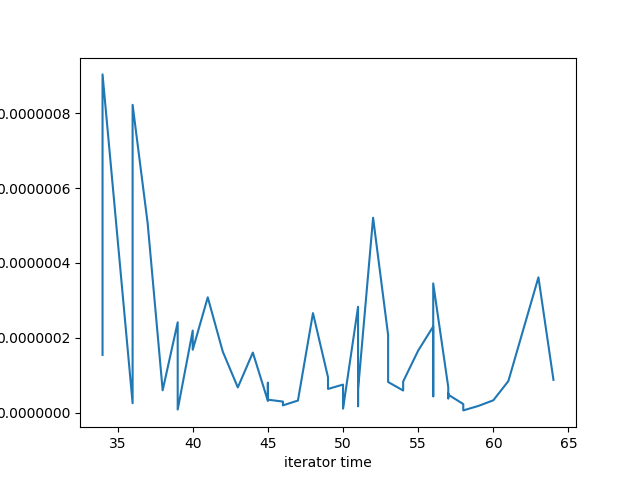
1. Jacobi



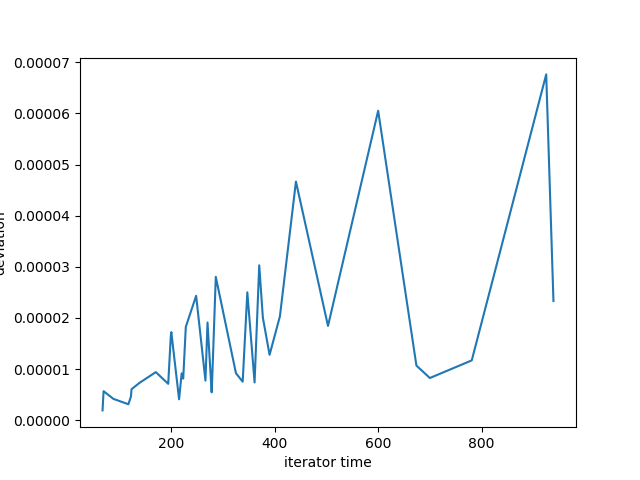
1. SoR(w=1.1)



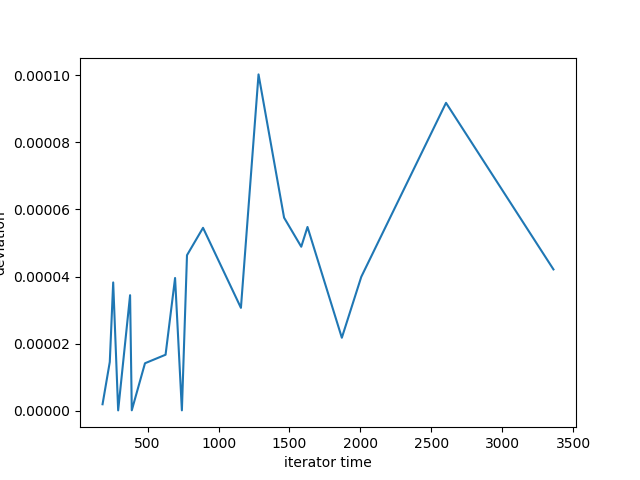
1. n = 100
2. CG方法



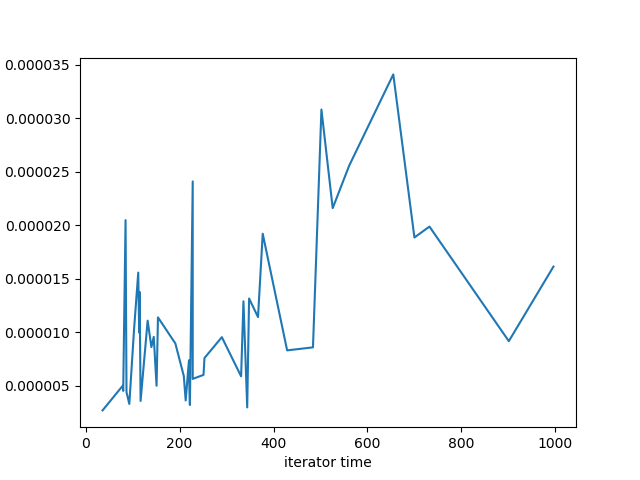
1. Gauss-Siedel



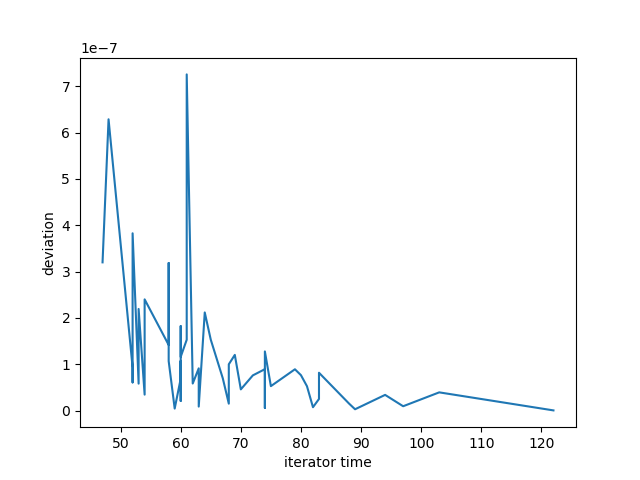
1. Jacobi



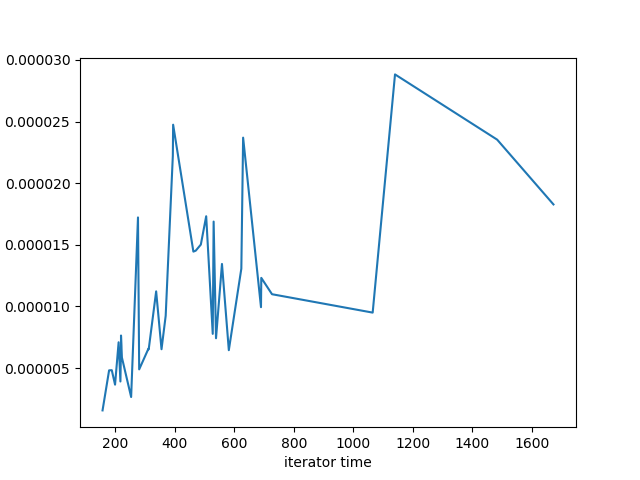
1. SoR(w=1.1)



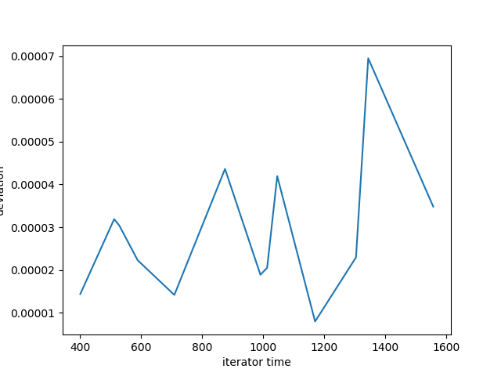
1. n = 200
2. CG方法



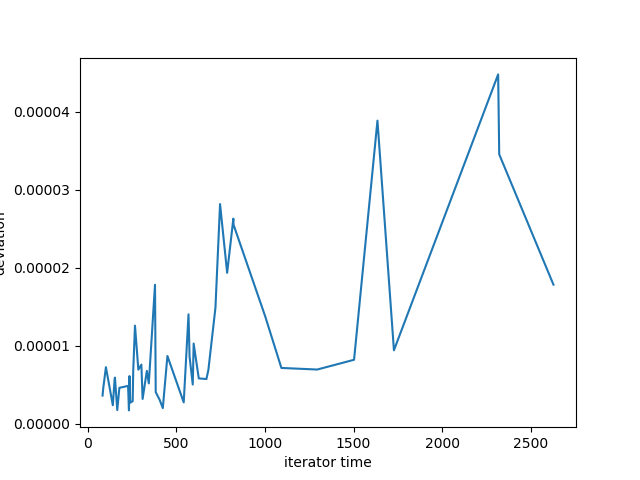
1. Gauss-Siedel



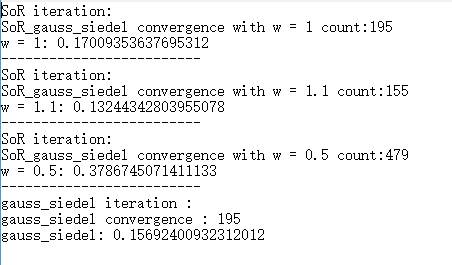
1. Jacobi



1. SoR(w=1.1)



1. SoR算法改变松弛因子的结果



1. 六种算法的时间消耗：（gauss2为列主元消去法）

n=10

jacobi iteration :

jacobi convergence: 34

jacobi time: 0.0

-------------------------

gauss\_siedel iteration :

gauss\_siedel convergence : 19

gauss\_siedel time: 0.0

-------------------------

SoR iteration:

SoR\_gauss\_siedel convergence with w = 1.1 count:14

SoR time: 0.015630245208740234

-------------------------

conjugate\_gradient iteration:

conjugate\_gradient convergence: 9

conjugate time: 0.0

-------------------------

gauss time: 0.0

-------------------------

gauss2 time: 0.0

n=50

jacobi iteration :

jacobi convergence: 176

jacobi time: 0.015631437301635742

-------------------------

gauss\_siedel iteration :

gauss\_siedel convergence : 89

gauss\_siedel time: 0.07534527778625488

-------------------------

SoR iteration:

SoR\_gauss\_siedel convergence with w = 1.1 count:71

SoR time: 0.047452449798583984

-------------------------

conjugate\_gradient iteration:

conjugate\_gradient convergence: 30

conjugate time: 0.015640974044799805

-------------------------

gauss time: 0.07815361022949219

-------------------------

gauss2 time: 0.07439517974853516

n=100

jacobi iteration :

jacobi convergence: 335

jacobi time: 0.10940694808959961

-------------------------

gauss\_siedel iteration :

gauss\_siedel convergence : 183

gauss\_siedel time: 1.142120599746704

-------------------------

SoR iteration:

SoR\_gauss\_siedel convergence with w = 1.1 count:151

SoR time: 0.3421316146850586

-------------------------

conjugate\_gradient iteration:

conjugate\_gradient convergence: 37

conjugate time: 0.026874542236328125

-------------------------

gauss time: 0.5314581394195557

-------------------------

gauss2 time: 0.5472836494445801

n=150

jacobi iteration :

jacobi convergence: 429

jacobi time: 0.12505006790161133

-------------------------

gauss\_siedel iteration :

gauss\_siedel convergence : 230

gauss\_siedel time: 1.2827227115631104

-------------------------

SoR iteration:

SoR\_gauss\_siedel convergence with w = 1.1 count:187

SoR time: 0.8039429187774658

-------------------------

conjugate\_gradient iteration:

conjugate\_gradient convergence: 42

conjugate time: 0.06252741813659668

-------------------------

gauss time: 1.759505271911621

-------------------------

gauss2 time: 1.7984991073608398

n=200

jacobi iteration :

jacobi convergence: 481

jacobi time: 0.1821136474609375

-------------------------

gauss\_siedel iteration :

gauss\_siedel convergence : 247

gauss\_siedel time: 1.4903826713562012

-------------------------

SoR iteration:

SoR\_gauss\_siedel convergence with w = 1.1 count:223

SoR time: 1.326357364654541

-------------------------

conjugate\_gradient iteration:

conjugate\_gradient convergence: 55

conjugate time: 0.10910391807556152

-------------------------

gauss time: 4.156708002090454

-------------------------

gauss2 time: 4.206740140914917

## 结果分析

1. 对于高斯消元和列主元消去法，两者的时间复杂度相差不大，在n足够大的时候才能体现出细微的差别。列主元消去法比高斯消元法消耗时间多的原因在于列主元消去法需要进行行交换操作。但是造成不了数量级上的差距，而且列主元消去法在处理主元元素较小的行时能降低误差，因此比高斯消元法要优秀。
2. 在四种迭代法迭代步数-误差图中，我们不难发现CG算法是收敛较快而且能保证误差在一定水平的优秀算法。然而CG算法也有它的不足，它只能对对称正定矩阵进行求解。

而在四种迭代算法中，雅可比迭代收敛条件最为苛刻，而且收敛速度较慢。可以看到在n=200时，很少矩阵能够用雅可比迭代进行求解。而SoR和Gauss-Siedel收敛速度较快，而且收敛条件没有那么苛刻。在n=200图中我们可以看到两者收敛步数普遍在1000以内，如果不在这个范围内，说明很可能随机生成的矩阵是不收敛的。

1. SoR松弛因子改变的实验中基本符合预期结果。在w=1时迭代收敛步数和高斯迭代一样，而当w>1时，收敛速度就会变快。反之则更慢
2. 比较6种方法的时间消耗。迭代法的速度比普遍比消元法要快，这是由算法决定的，迭代法的复杂度比消元法要低一个量级。而在4种迭代法中，收敛最快的是CG方法，然后是SoR和高斯迭代，最慢的是雅可比迭代。然而，时间消耗上雅可比方法由于求逆乘法等操作比较少，因此略微领先高斯迭代，但两者的时间复杂度是一个量级的。

## PageRank算法

create matrix A where aij = 1 / sum of nodes that node\_j trusts

FOR i FROM 0 TO N DO

if node\_i not trust any nodes except itself DO

node\_i trust all nodes

END IF

END FOR

SET 0 TO K

SET A \* [1/N ....]^T to P\_0

WHILE K=0 OR INF\_NORM(P\_K - P\_(K-1)) > 0.00005 DO

P\_(K+1) = A \* P\_K

END WHILE

PRINT P\_K

(\*) 源码由python3.6编写，用到的第三方库：numpy，scipy，matplotlib

测试文件是test.py 和test\_time.py 可以自由添加代码对不同的方法进行测试。

直接法的源码在m.py和matrix2.py，迭代法在iteration.py