数值分析 HW1-Assignment 实验报告

学号：16340041

姓名：陈亚楠

实验一

一、问题描述：

请实现下述算法，求解线性方程组 Ax = b，其中 A 为 n * 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：

（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。

二、算法设计：

1. 高斯消去法：

设有线性方程组

$$\begin{aligned}
&a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1,\\
&a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2,\\
&\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\\
&a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = b_m.
\end{aligned}$$

或写为矩阵形式

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

简记为 Ax = b。

用高斯消去法解线性方程组的基本思想是用逐次消去未知数的方法把原方程组 Ax = b 化为与其等价的三角形线性方程组，并使用回代的方法求解三角形线性方程组。

设$a_{kk}^{(k)} \neq 0$（k = 1, 2, ···, n - 1），

（1） 消元运算（k = 1, 2, ···, n - 1）

$$\begin{aligned}
&m_{ik} = a_{ik}^{(k)} / a_{kk}^{(k)} \qquad (i = k+1, \cdots, n),\\
&a_{ij}^{(k+1)} = a_{ij}^{(k)} - m_{ik} a_{kj}^{(k)} \qquad (i, j = k+1, \cdots, n),\\
&b_i^{(k+1)} = b_i^{(k)} - m_{ik} b_k^{(k)} \qquad (i = k+1, \cdots, n).
\end{aligned}$$

（2） 回代运算

$$x_n = b_n^{(n)} / a_{nn}^{(n)},$$

$$x_i = b_i^{(i)} - \sum_{j=i+1}^{n} a_{ij}^{(i)} x_j / a_{ii}^{(i)} \qquad (i = n-1, \cdots, 2, 1)$$

2. 列主元消去法：

在采用高斯消去法解方程组时，即使$a_{kk}^{(k)} \neq 0$，但其绝对值很小，那么由于舍入误差的影响，也可能引起很大的误差，因此对一般矩阵来说，最好每一步选取系数矩阵中绝对值最大的元素作为主元素，以使高斯消去法具有较好的数值稳定性。

（1） 输入数据 A 和 b，置 det = 1。

（2） 对于 k = 1，2，…，n – 1，做

① 按列选主元。令

$$|a_{rk}| = \max_{k \le i \le n} |a_{ik}|$$

如果$|a_{rk}| = 0$，则停止计算（此时 det(A) = 0）。

② 交换两行如果 r > k，则

$$a_{kj} \Leftrightarrow a_{rj} \quad (j = k, k+1, \cdots, n)$$

$$b_k = b_r$$

$$\det \Leftrightarrow -\det$$

③ 消元计算。置

$$m_{ik} = \frac{a_{ik}}{a_{kk}} \quad (I = k+1, k+2, \cdots, n)$$

$$a_{ij} = a_{ij} - m_{kj} a_{kj} \quad (i, j = k+1, k+2, \cdots, n)$$

$$b_i = b_i - m_{ik} b_{ik} \quad (i = k+1, k+2, \cdots, n)$$

④ 置 det = a_{kk} · det。

（3） 如果 $a_{mm} = 0$ 则停止计算（此时 det(A) = 0）；否则进行回代。

对于 k = n, n – 1, …, 1，置

$$x_k = \left(b_k - \sum_{j=k+1}^{n} a_{kj} x_j \right) / a_{kk}$$

当下标大于上标时，不做求和运算。

（4） 置 det = a_{nn} · det；

（5） 输出 $x_1, x_2, \cdots, x_n$ 为线性方程组 Ax = b 的解，det 为矩阵 A 的行列式的值。

三、数值实验：

1. 实验源码：

（1） 高斯消去法（Gauss.m）：

```
% 高斯消去法解线性方程组;
% A 为方程组的系数矩阵;
% b 为方程组的右端项;
% x 为方程组的解;
function x = Gauss(A, b)
    augmentedMatrix = [A b];
```

```
    n = length(b);
    ra = rank(A);
    rz = rank(augmentedMatrix);
    temp = rz - ra;
    if temp > 0
        disp('failure1')
        return
    end
    if ra == rz
        if ra == n
            x = zeros(n, 1);
            % 消元运算
            for k = 1:n - 1
                for i = k + 1:n
                    m = augmentedMatrix(i, k) /
augmentedMatrix(k, k);
                    augmentedMatrix(i, k:n + 1) =
augmentedMatrix(i, k:n + 1) - m * augmentedMatrix(k, k:n +
1);
                end
            end
            % 回代过程
            b = augmentedMatrix(1:n, n + 1);
            A = augmentedMatrix(1:n, 1:n);
            x(n) = b(n) / A(n, n);
            for j = n - 1:-1:1
                x(j) = (b(j) - sum(A(j, j + 1:n) * x(j +
1:n))) / A(j, j);
            end
        else
            disp('failure2')
        end
    end
end
```

（2） 列主元消去法（Pivot.m）：

```
% 列主元消去法解线性方程组;
% A 为系数矩阵;
% b 为方程组的右端项;
% x 为方程组的解;
% det 为系数矩阵的行列式的值;
% flag = 'failure'表示失败,flag = 'success'表示成功
function[x, det, flag] = Pivot(A, b)
    [n, m] = size(A);
```

```
nb = length(b);
if n ~= m
    disp('error')
    return;
end
if m ~= nb
    disp('error')
    return;
end
% 初始化
x = zeros(n, 1);
det = 1;
flag = 'success';
for k = 1:n - 1
    % 选主元
    max = 0;
    for i = k:n
        if abs(A(i, k)) > max
            max = abs(A(i, k));
            r = i;
        end
    end
    if max < 1e-10
        flag = 'failure';
        return;
    end
    % 交换两行
    if r > k
        for j = k:n
            z = A(k, j);
            A(k, j) = A(r, j);
            A(r, j) = z;
        end
        z = b(k);
        b(k) = b(r);
        b(r) = z;
        det = -det;
    end
    % 消元运算
    for i = k + 1:n
        m = A(i, k) / A(k, k);
        for j = k + 1:n
            A(i, j) = A(i, j) - m * A(k, j);
        end
```

```
            b(i) = b(i) - m * b(k);
        end
        det = det * A(k, k);
    end
    det = det * A(n, n);
    % 回代过程
    if abs(A(n, n) < 1e-10)
        flag = 'failure';
        return;
    end
    for k = n:-1:1
        for j = k + 1:n
            b(k) = b(k) - A(k, j) * x(j);
        end
        x(k) = b(k) / A(k, k);
    end
    x(k) = b(k) / A(k, k);
end
```

2. **实验结果：**

（1） **高斯消元法（GaussTest.m）：**

```
% 高斯消去法的验证程序
A = [2 2 3; 4 7 7; -2 4 5];
b = [3 1 -7]';
x = Gauss(A, b)
```

输出结果：

```
>> GaussTest

x =

     2
    -2
     1
```

（2） **列主元消元法：**

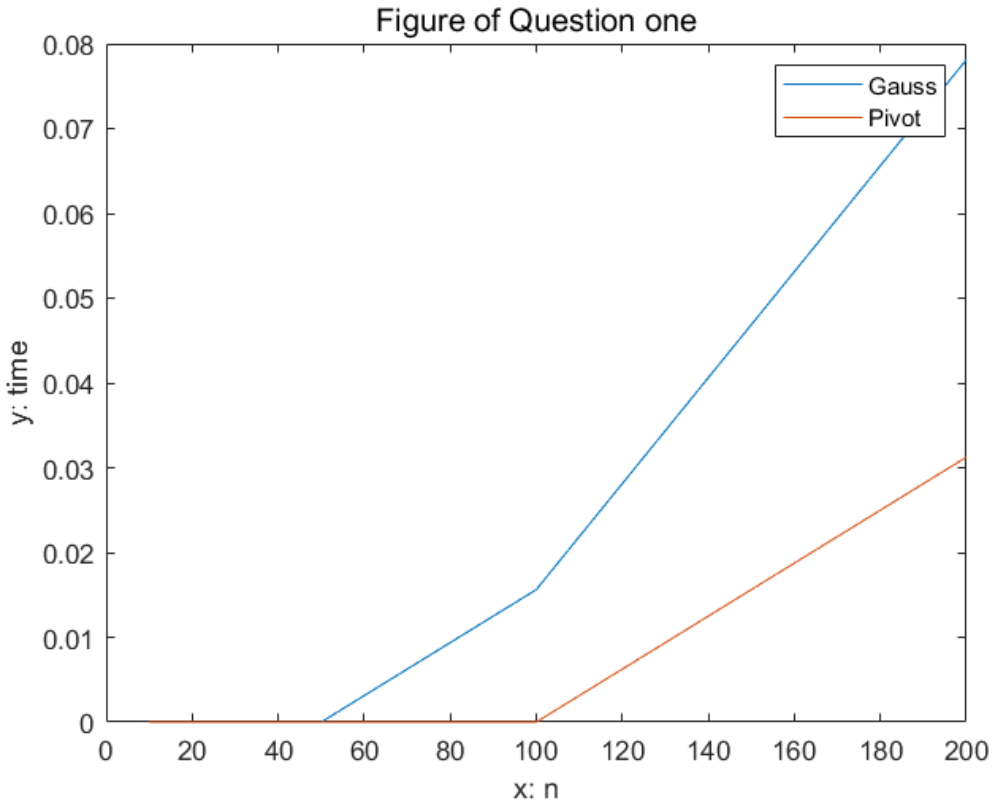
```
% 列主元消去法的验证程序
A = [-3 2 6; 10 -7 0; 5 -1 5];
b = [4 7 6];
[x, det, flag] = Pivot(A, b)
```

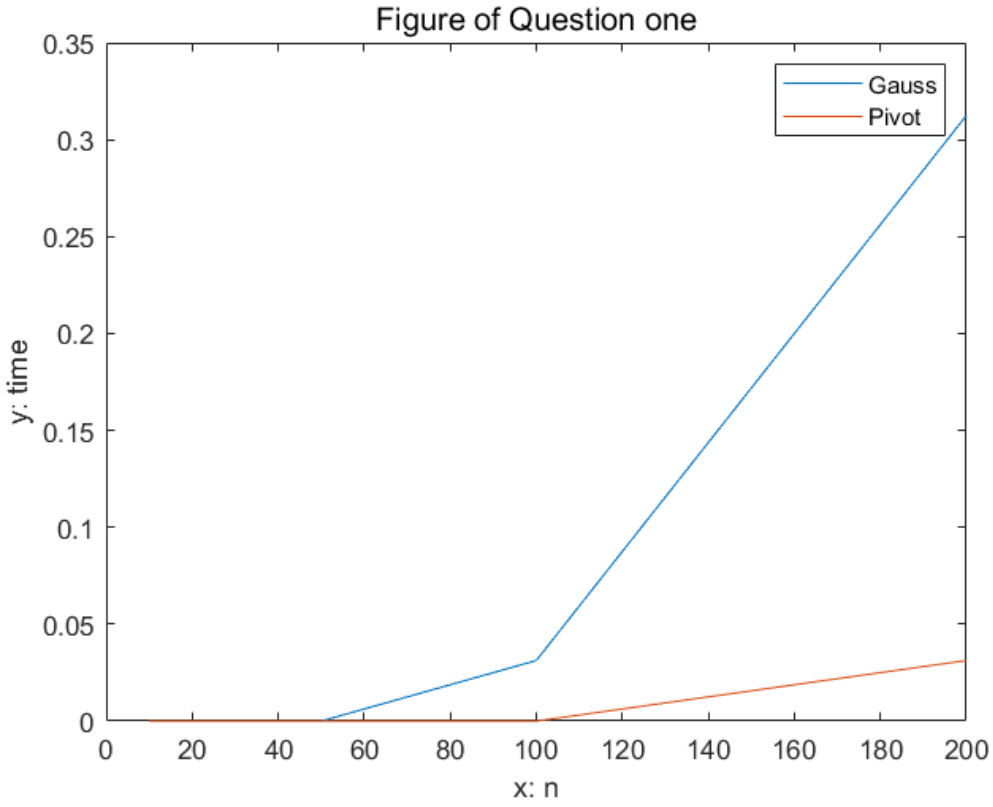
输出结果：

```
>> PivotTest

x =

    0.0000
   -1.0000
    1.0000


det =

   155


flag =

    'success'
```

四、结果分析：

时间曲线：





分析以上时间曲线可知，在当线性方程组矩阵的维数较小时，高斯消元法与列主元消元法二者的时间效率差别较小，但在维数变大时，列主元消元法明显优于高斯消元法。并且列主元消元法解决了在逐次消元的过程中，主对角元素相对同列的其他元素绝对值很小，舍入误差很大，影响计算精度，甚至消元过程无法进行的问题。

实验二

一、问题描述：

请实现下述算法，求解线性方程组 Ax = b，其中 A 为 n * 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：

（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。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。

二、算法设计：

1. Jacobi 迭代法：

对于线性方程组 Ax = b，其中 A = $(a_{ij})_{n*n}$非奇异，且$a_{ii} \neq 0$，i = 1, 2, ⋯, n。

记

$$\boldsymbol{D}=\begin{bmatrix} a_{11} & & & \\ & a_{22} & & \\ & & \ddots & \\ & & & a_{nn} \end{bmatrix},\begin{bmatrix} 0 & & & \\ -a_{21} & 0 & & \\ \vdots & \vdots & & \\ -a_{n1} & -a_{n2} & \cdots & 0 \end{bmatrix},\begin{bmatrix} 0 & -a_{12} & \cdots & -a_{1n} \\ & 0 & \cdots & -a_{2n} \\ & & & \vdots \\ & & & 0 \end{bmatrix}$$

则 A = D – L – U。方程组 Ax = b 可改写为

$$\mathrm{x}=\mathrm{D}^{-1}(\mathrm{L}+\mathrm{U})\mathrm{x}+\mathrm{D}^{-1}\mathrm{b}$$

相应的矩阵形式的雅可比迭代公式为

$$\mathrm{x}^{(\mathrm{k}+1)}=\mathrm{D}^{-1}(\mathrm{L}+\mathrm{U})\mathrm{x}^{\mathrm{k}}+\mathrm{D}^{-1}\mathrm{b}$$

简记为

$$\mathrm{x}^{(\mathrm{k}+1)}=\mathrm{B}_{\mathrm{J}}\mathrm{x}^{(\mathrm{k})}+\mathrm{f}_{\mathrm{J}}$$

其中

$$\mathrm{B}_{\mathrm{J}}=\mathrm{D}^{-1}(\mathrm{L}+\mathrm{U}),\ \mathrm{f}_{\mathrm{J}}=\mathrm{D}^{-1}\mathrm{b}。$$

2. Gauss-Seidel 迭代法：

对于线性方程组 Ax = b，其中 A = $(\mathrm{a_{ij}})_{\mathrm{n*n}}$非奇异，且$\mathrm{a_{ii}}\neq 0$，i = 1, 2, ···, n。数组 x(n)开始存放$\mathrm{x}^{(0)}$，后存放$\mathrm{x}^{(\mathrm{k})}$，$\mathrm{N_0}$为最大迭代次数：

（1）$\mathrm{X_i}\leftarrow 0.0$（i = 1, ···, n）；

（2）对于 k = 1, 2, ···, $\mathrm{N_0}$

对于 i = 1, 2, ···, n

$$x_i \leftarrow b_i - \sum_{j=1}^{i-1} a_{ij}\, x_j - \sum_{j=i+1}^{n} a_{ij}\, x_j \Big/ a_{ii}$$

。

3. 逐次超松弛迭代法：

SOR 迭代法是高斯-塞德尔迭代法的一种修正，可由下述思想得到：

设已知$\mathrm{x}^{(\mathrm{k})}$及已计算$\mathrm{x}^{(\mathrm{k}+1)}$的分量$\mathrm{x_j}(\mathrm{k}+1)$（j = 1, 2, ···, i - 1），

（1） 首先用高斯-塞德尔迭代法定义辅助量$\tilde{\mathrm{x}}_{\mathrm{i}}^{(\mathrm{k}+1)}$，

$$\tilde{\mathrm{x}}_{\mathrm{i}}^{(\mathrm{k}+1)}=(b_i-\sum_{j=1}^{i-1}a_{ij}X_j^{(k+1)}-\sum_{j=i+1}^{n}a_{ij}X_j^{(k)})\ /\ \mathrm{a_{ii}}$$

（2） 再由$X_i^{(k)}$与$\tilde{\mathrm{x}}_{\mathrm{i}}^{(\mathrm{k}+1)}$加权平均定义$X_i^{(k+1)}$，即

$$X_i^{(k+1)}=(1-\ \omega)X_i^{(k)}+\ \omega\tilde{\mathrm{x}}_{\mathrm{i}}^{(\mathrm{k}+1)}=\ X_i^{(k)}+\ \omega(\tilde{\mathrm{x}}_{\mathrm{i}}^{(\mathrm{k}+1)}-\ X_i^{(k)})$$

将第一个式子带入第二个式子即得到解线性方程组 Ax = b 的 SOR 迭代式。

4. 共轭梯度法：

CG 算法：

（1） 任取$\mathrm{x}^{(0)}\in \mathrm{R}^{\mathrm{n}}$，计算$\mathrm{r}^{(0)}=\mathrm{b}-\mathrm{Ax}$，取$\mathrm{p}^{(0)}=\mathrm{r}^{(0)}$；

（2） 对 k = 0, 1, ···，计算

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

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

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

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

(3) 若$r^{(k)} = 0$，或$(p^{(k)}, Ap^{(k)}) = 0$，计算停止，则$X^{(k)} = X^*$。由于 A 正定，故当$(p^{(k)}, Ap^{(k)}) = 0$ 时，$p^{(k)} = 0$，而$(r^{(k)}, r^{(k)}) = (r^{(k)}, p^{(k)}) = 0$，也即$r^{(k)} = 0$。

三、数值实验：

1. Jacobi 迭代法：

（1） 实验源码（Jacobi.m）：

```
% 雅可比迭代法求解线性方程组；
% A 为方程组的系数矩阵；
% b 为方程组的右端项；
% delta 为精度要求，默认 1e-5；
% max 为最大迭代次数，默认 100；
% x 为方程组的解；
% k 为迭代次数；
% flag = 'failure'表示失败,flag = 'success'表示成功
function [x, k, flag] = Jacobi(A, b, delta, max)
    n = length(A);
    k = 0;
    x = zeros(n, 1);
    y = zeros(n, 1);
    flag = 'success';
    while 1
        for i = 1:n
            y(i) = b(i);
            for j = 1:n
                if j ~= i
                    y(i) = y(i) - A(i, j) * x(j);
                end
            end
            if abs(A(i, i)) < 1e-10 | k == max
                flag = 'failure';
                return;
            end
            y(i) = y(i) / A(i, i);
        end
        if norm(y - x, inf) < delta
```

```
            break;
        end
        x = y;
        k = k + 1;
    end
end
```

（2） 实验结果（JacobiTest.m）：

```
% 雅可比迭代法的验证程序
A = [4 1 -1; 1 -5 -1; 2 -1 -6];
b = [13 -8 -2]';
delta = 1e-5;
max = 100;
[x, k, flag] = Jacobi(A, b, delta, max)
```

输出结果：

```
>> JacobiTest

x =

    3.0000
    2.0000
    1.0000


k =

     9


flag =

    'success'
```

2. Gauss-Seidel 迭代法：

（1） 实验源码：

```
% 高斯塞德尔迭代法求解线性方程组：
% A 为方程组的系数矩阵；
% b 为方程组的右端项；
% delta 为精度要求，默认 1e-5；
% max 为最大迭代次数，默认 100；
% x 为方程组的解；
% k 为迭代次数；
% flag = 'failure'表示失败,flag = 'success'表示成功
function [x, k, flag] = GaussSeidel(A, b, delta, max)
```

```
    n = length(A);
    k = 0;
    x = zeros(n, 1);
    y = zeros(n, 1);
    flag = 'success';
    while 1
        y = x;
        for i = 1:n
            z = b(i);
            for j = 1:n
                if j ~= i
                    z = z - A(i, j) * x(j);
                end
            end
            if abs(A(i, i)) < 1e-10 | k == max
                flag = 'failure';
                return;
            end
            z = z / A(i, i);
            x(i) = z;
        end
        if norm(y - x, inf) < delta
            break;
        end
        k = k + 1;
    end
end
```

(2) 实验结果：

```
% 高斯塞德尔迭代法的验证程序
A = [10 -2 -2; -2 10 -1; -1 -2 3];
b = [1 0.5 1]';
delta = 1e-5;
max = 100;
[x, k, flag] = GaussSeidel(A, b, delta, max)
```

输出结果：

```
>> GaussSeidelTest

x =

    0.2311
    0.1471
    0.5084


k =

     8


flag =

    'success'
```

3. **逐次超松弛迭代：**

（1） 实验源码：

```
% 逐次超松弛迭代法求解线性方程组;
% A 为方程组的系数矩阵;
% b 为方程组的右端项;
% delta 为精度要求，默认 1e-5;
% max 为最大迭代次数，默认 100;
% w 为超松弛因子，默认为 1;
% x 为方程组的解;
% k 为迭代次数;
% flag = 'failure'表示失败,flag = 'success'表示成功
function [x, k, flag] = SOR(A, b, delta, w, max)
    n = length(A);
    k = 0;
    x = zeros(n, 1);
    y = zeros(n, 1);
    flag = 'success';
    while 1
        y = x;
        for i = 1:n
            z = b(i);
            for j = 1:n
                if j ~= i
                    z = z - A(i, j) * x(j);
                end
            end
```

```
            if abs(A(i, i)) < 1e-10 | k == max
                flag = 'failure';
                return;
            end
            z = z / A(i, i);
            x(i) = (1 - w) * x(i) + w * z;
        end
        if norm(y - x, inf) < delta
            break;
        end
        k = k + 1;
    end
end
```

（2） 实验结果：

```
% 逐次超松弛迭代法的验证程序
A = [2 -1 0 0; -1 2 -1 0; 0 -1 2 -1; 0 0 -1 2];
b = [1 0 1 0]';
delta = 1e-5;
w = 1.46;
max = 100;
[x, k, flag] = SOR(A, b, delta, w, max)
```

输出结果：

```
>> SORTest

x =

    1.2000
    1.4000
    1.6000
    0.8000


k =

    16


flag =

    'success'
```

4. 共轭梯度法：

（1） 实验源码（CG.m）：

```
% 共轭梯度法求解线性方程组：
% A 为方程组的系数矩阵；
% b 为方程组的右端项；
% delta 为精度要求，默认 1e-5；
% max 为最大迭代次数，默认 100；
% x 为方程组的解；
% k 为迭代次数；
% flag = 'failure'表示失败,flag = 'success'表示成功
function [x, k, flag] = CG(A, b, x0, delta, max)
    n = length(A);
    k = 0;
    x = zeros(n, 1);
    flag = 'success';
    r0 = b - A * x0;
    p0 = r0;
    while 1
        a0 = dot(r0, r0) / dot(p0, A * p0);
        x1 = x0 + a0 * p0;
        r1 = r0 - a0 * A * p0;
        b0 = dot(r1, r1) / dot(r0, r0);
        p1 = r1 + b0 * p0;
        if dot(p0, A * p0) == 0
            flag = 'success';
            break;
        end
        if abs(p0) < 1e-10 | k == max
            flag = 'failure';
            break;
        end
        x0 = x1;
        r0 = r1;
        p0 = p1;
        k = k + 1;
    end
    x = x0;
end
```

（2） 实验结果（CGTest.m）：

```
% 共轭梯度法的验证程序
A = [3 1; 1 2];
b = [5 5]';
delta = 1e-5;
x0 = [0 0]';
max = 100;
```

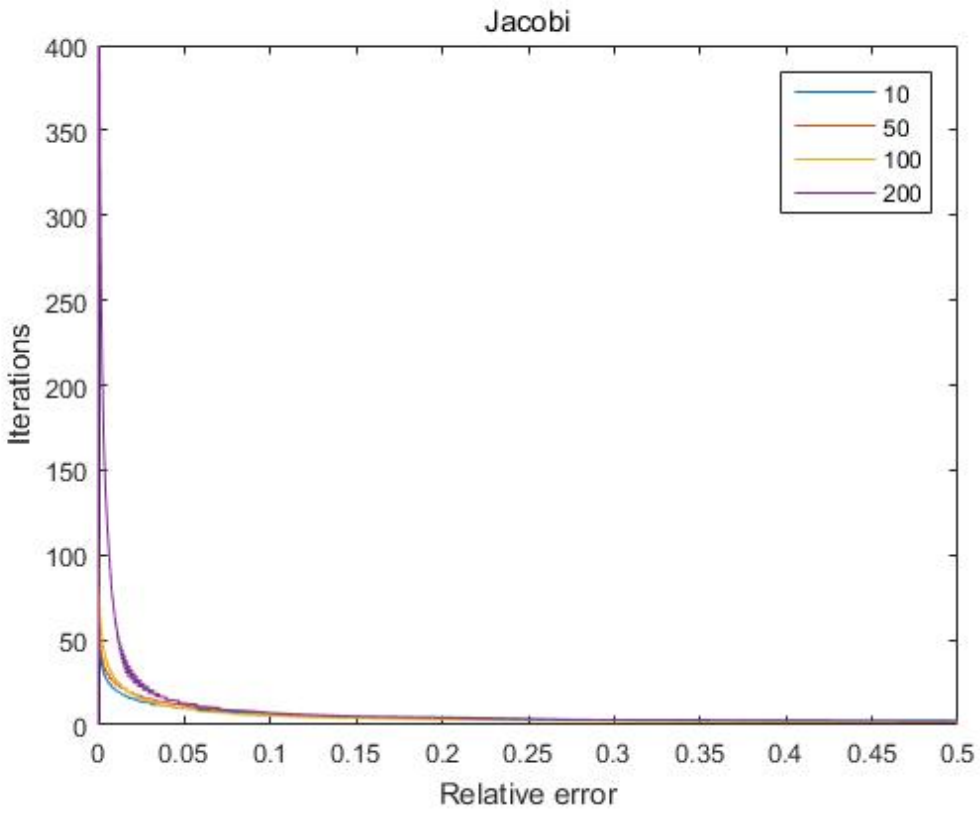
```
[x, k, flag] = CG(A, b, x0, delta, max)
```

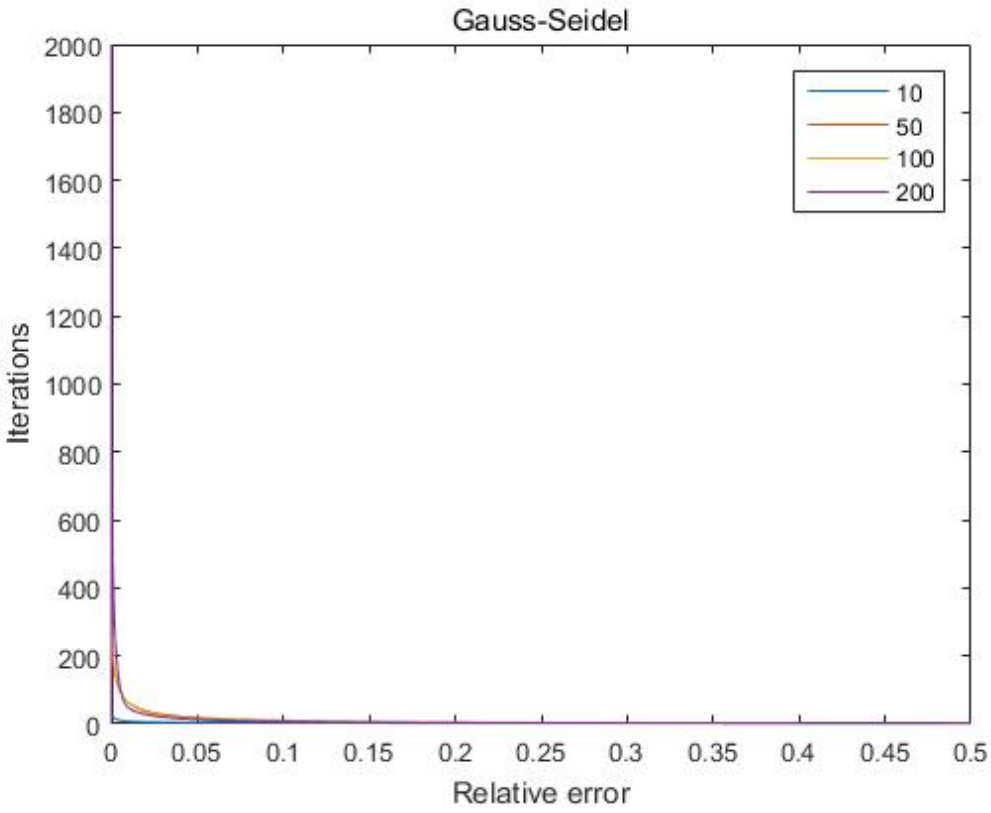
输出结果：

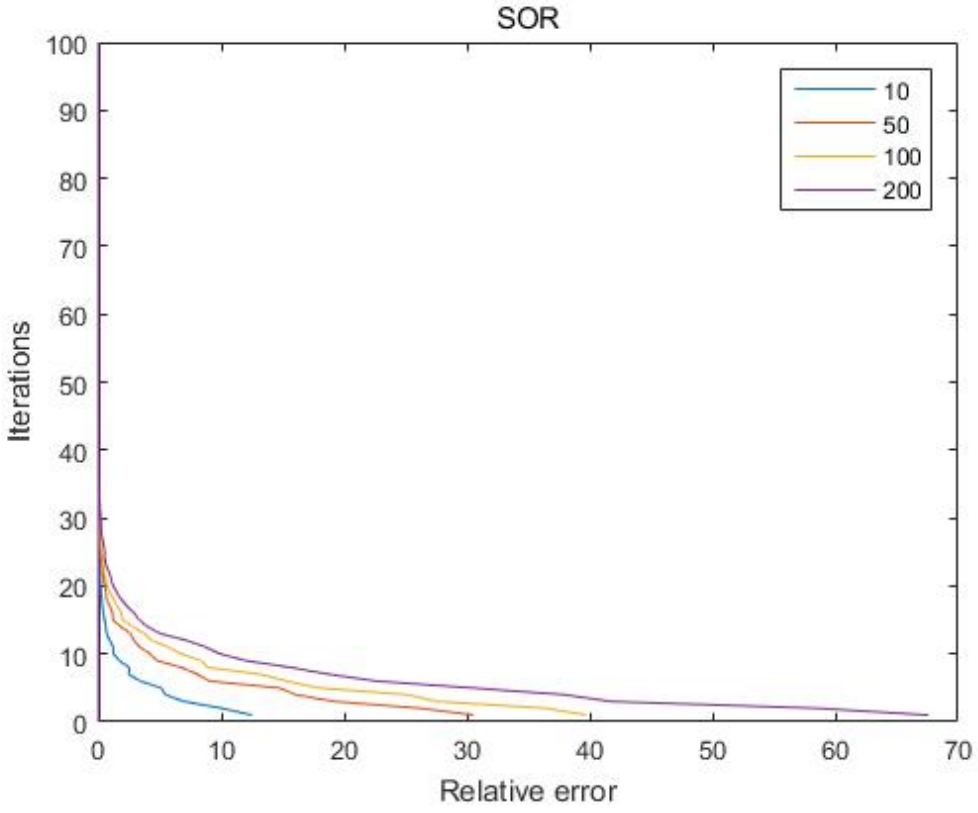
```
>> CGTest

x =

    1.0000
    2.0000


k =

     2


flag =

    'success'
```

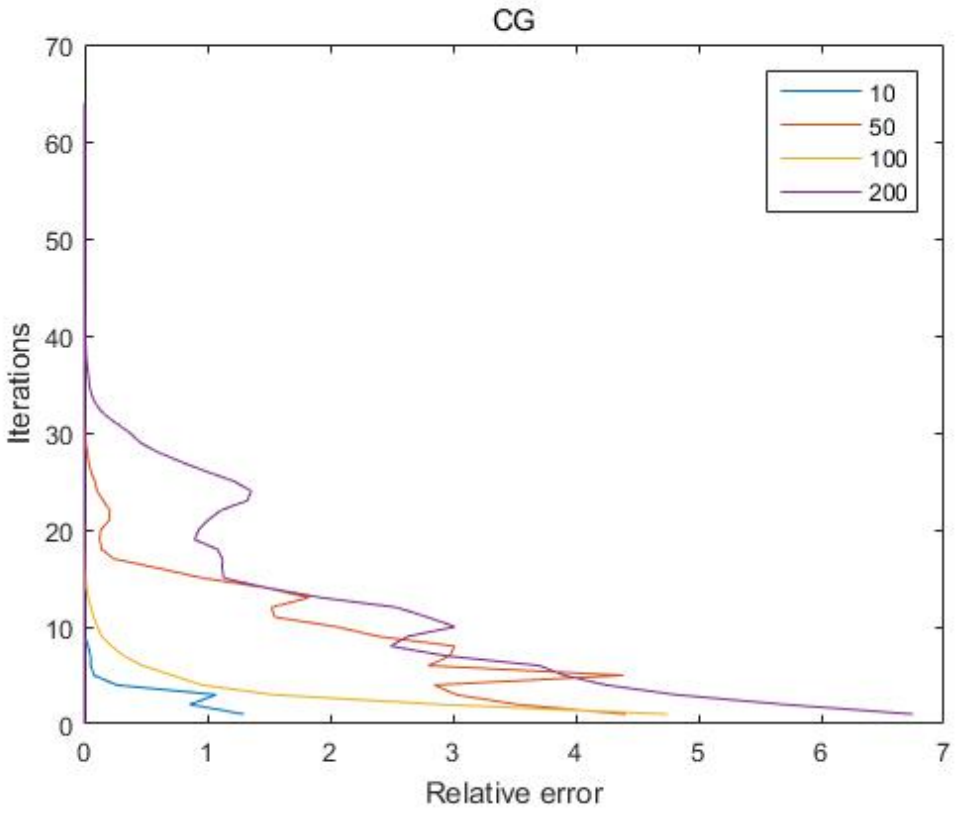
四、结果分析：

（1） 收敛曲线：

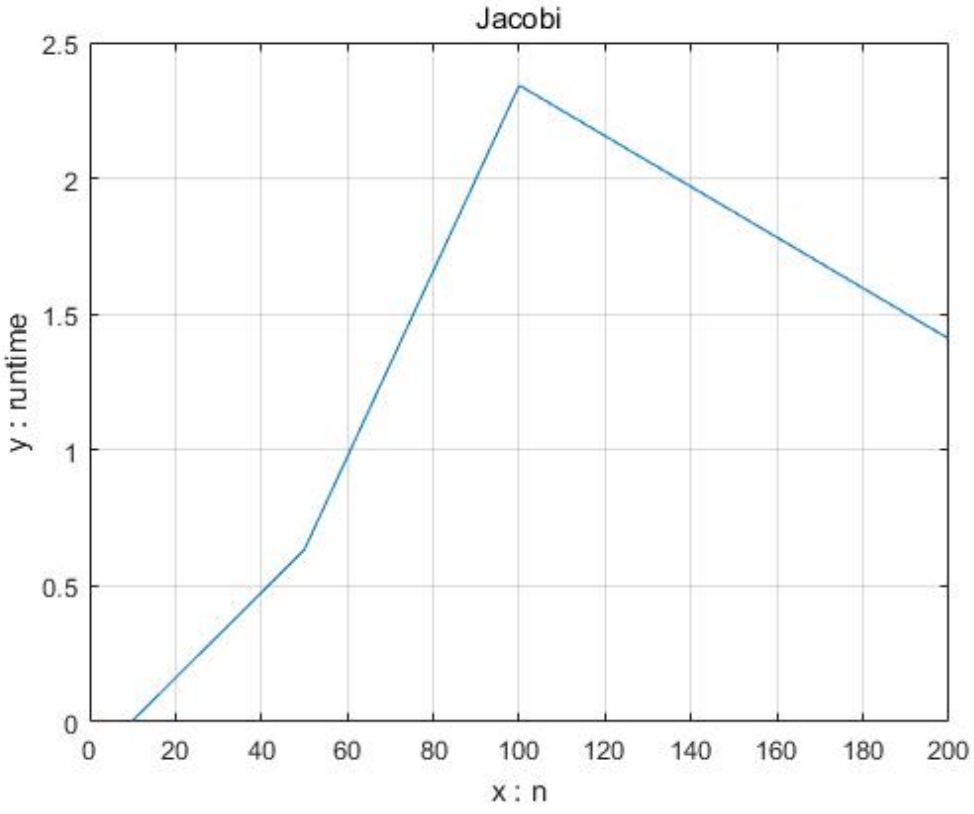


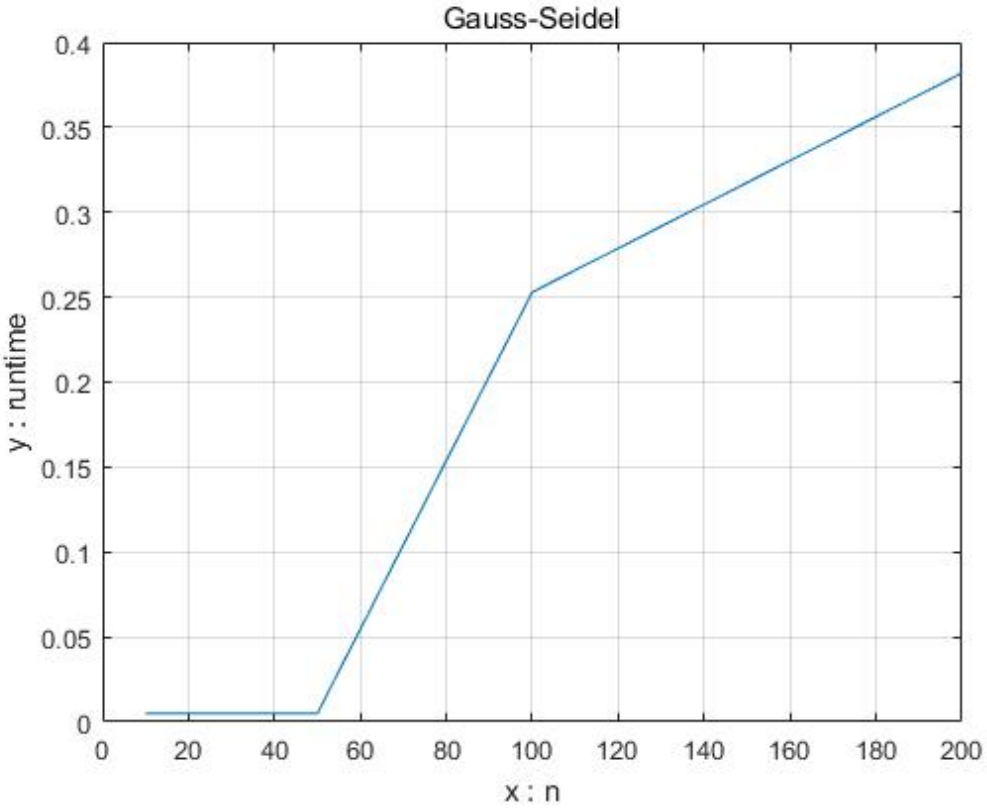

Gauss-Seidel
Iterations
Relative error
10
50
100
200
0
200
400
600
800
1000
1200
1400
1600
1800
2000
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0.45
0.5

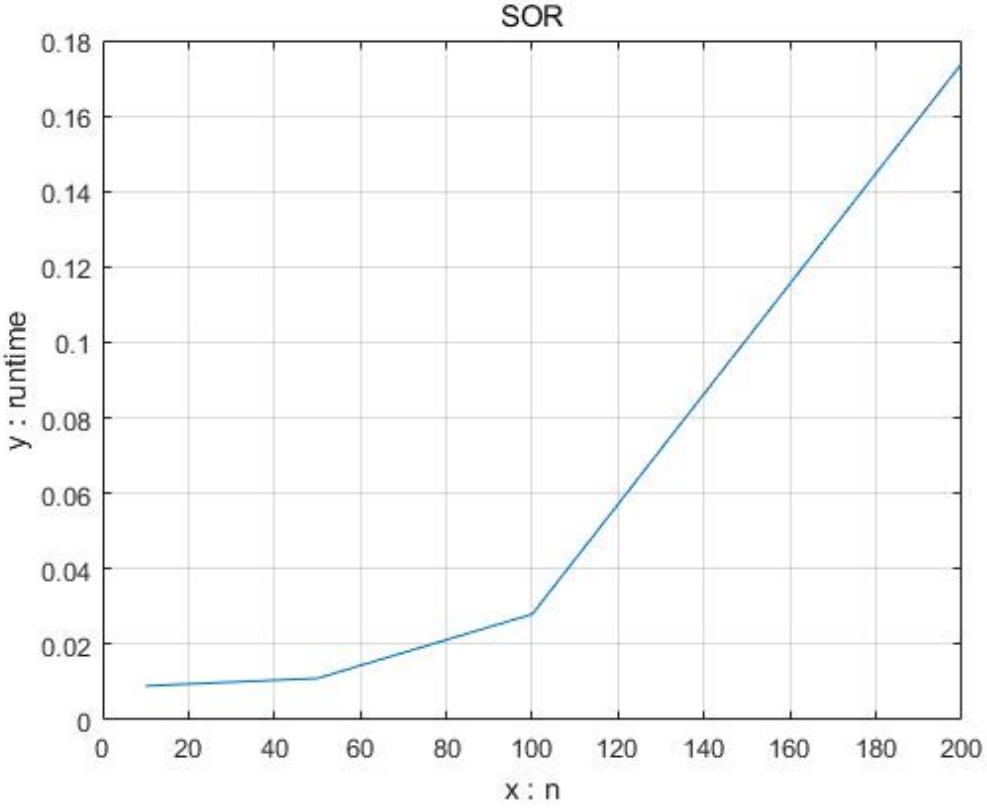

SOR
Iterations
Relative error
10
50
100
200
0
10
20
30
40
50
60
70
80
90
100
0
10
20
30
40
50
60
70

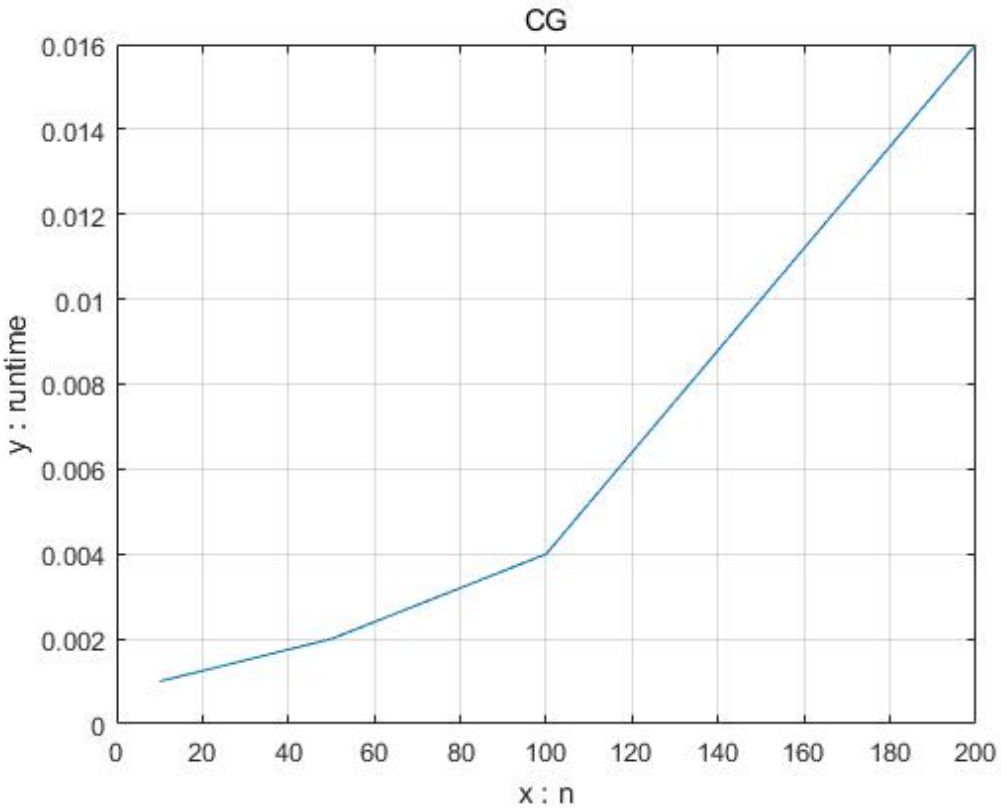

CG
10
50
100
200
Iterations
Relative error
0
10
20
30
40
50
60
70
0
1
2
3
4
5
6
7

(2) 时间曲线：

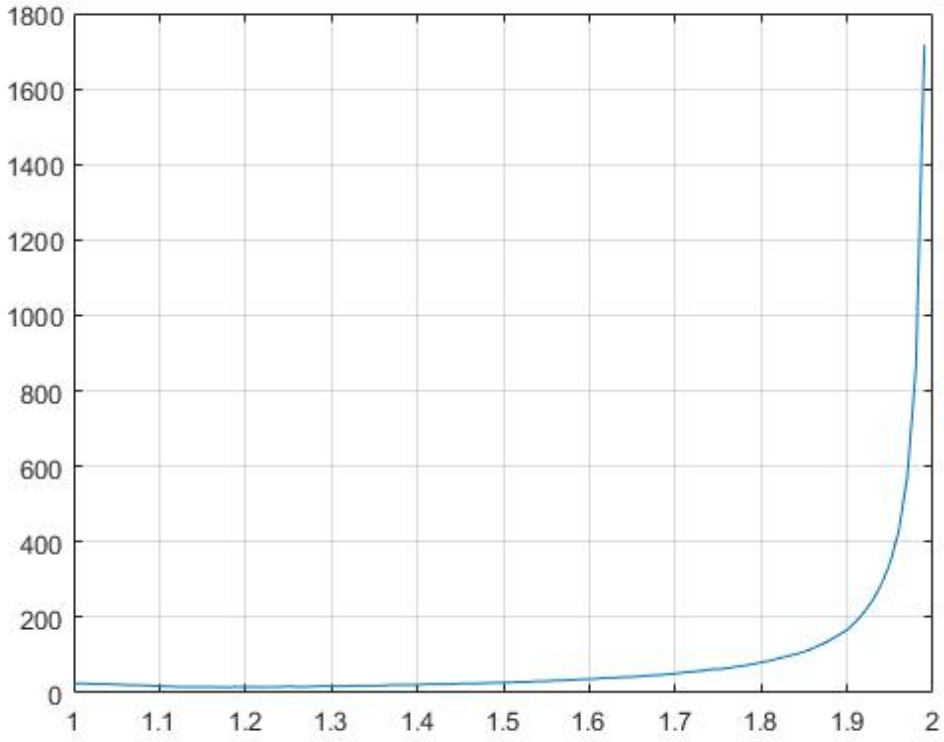

Jacobi
y : runtime
x : n
0
0.5
1
1.5
2
2.5
0
20
40
60
80
100
120
140
160
180
200


Gauss-Seidel
y : runtime
x : n
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0
20
40
60
80
100
120
140
160
180
200


SOR
y : runtime
x : n
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0.18
0
20
40
60
80
100
120
140
160
180
200



(3)　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：



实验三

一、问题描述：

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。

二、算法原理：

PageRank 算法的基本思想是网页的重要性排序是由网页间的链接关系所决定的。算法是依靠网页间的链接结构来评价每个页面的等级和重要性，一个网页的 PR 值不仅考虑指向它的链接网页数，还有指向它的网页的其他网页本身的重要性。

PageRank 具有两大特性：

① PR 值的传递性：网页 A 指向网页 B 时，A 的 PR 值也部分传递给 B；

② 重要性的传递性：一个重要网页比一个不重要网页传递的权重要多；

计算公式：

$$PR(p_i) = \frac{1-d}{n} + d \sum_{p_j \in M(i)} \frac{PR(p_j)}{L(j)}$$

三、实现代码：

```
nodes = load('soc-Epinions1.txt');
maxSize = max(max(nodes)) + 1;
sparseM = sparse(maxSize, maxSize);
delta = 0.0001;
beta = 0.85;
edgeSum = zeros(maxSize, 1);
for i = 1:size(nodes, 1)
    edgeSum(nodes(i, 1) + 1) = edgeSum(nodes(i, 1) + 1) + 1;
end
for i = 1:size(nodes, 1)
    sparseM(nodes(i, 2) + 1, nodes(i, 1) + 1) = 1 / edgeSum(nodes(i,
1) + 1);
end
num = size(sparseM, 2);
v = rand(num, 1);
v = v ./ norm(v, 1);
vlast = ones(num,1) * inf;
while(max(abs(v - vlast)) > delta)
    vlast = v;
    v = beta * sparseM * v + (1 - beta) / num * ones(num, 1);
end;
node = 0:75887;
```

```
plot(node,v);
[maximum, pos] = max(v);
disp(maximum);
disp(pos);
```

四、实验结果：