《数值计算方法》实验

实验报告

题目	Matlab 下实现一些算法
姓名	廖蕾
学号	16340135
班级	16 级软件工程教务 2 班

一. 实验环境：

在 win10 操作系统下，使用 Matlab R2017a 完成的。

二. 实验内容与完成情况：

1. 请实现下述算法，求解线性方程组 Ax=b，其中 A 为 n x 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。

1) **高斯消去法：**
输入的量：
系数矩阵 A 和常系数向量 b；

输出的量：
系数矩阵 A 和增广矩阵 B 的秩 RA,RB, 方程组中未知量的个数 n 和有关方程组解 X 及其解的信息.

算法描述：
首先在矩阵 A 和向量 b 中做个判断，判断 A 与 b 的秩是否相等，若不等算法结束；若相等且等于 n，那么开始高斯消去法。首先从上往下消主元，得到主元均为 1 的上三角阵，具体实现代码如下：

```
for p = 1 : n-1
    for k = p+1 : n
        m = B(k, p)/B(p, p);
        B(k, p:n+1) = B(k, p:n+1) - m * B(p, p:n+1);
    end
end
```

之后从下往上回代，得到结果 x 向量，具体实现代码如下：

```
for q = n-1: -1 : 1
    X(q) = (b(q)-sum(A(q, q+1:n)*X(q+1:n)))/A(q, q);
end
```

数值实验：

编写好函数之后在命令行中输入：

```
A = normrnd(0, 1, 10, 10);
b = normrnd(0, 1, 10, 1);
```

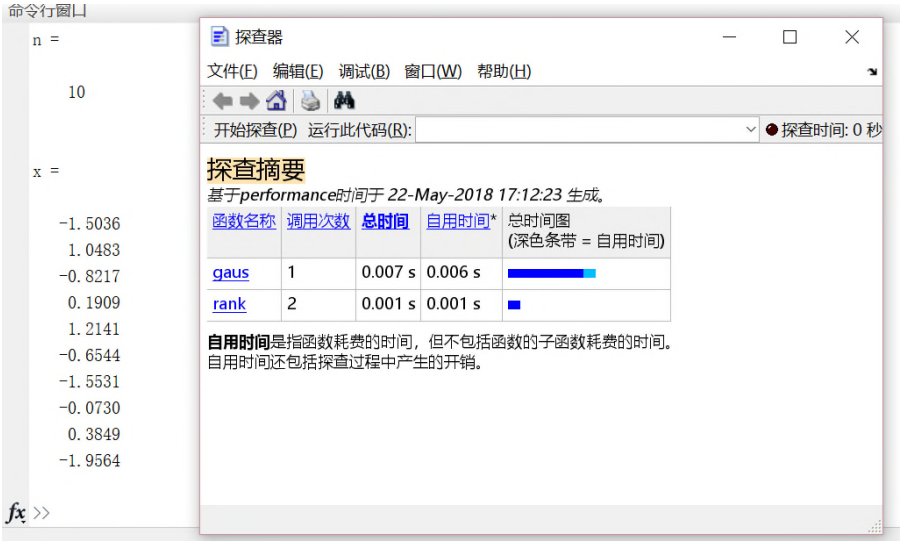
按下运行并计时，得到结果如下所示：

此时：

```
A =
-0.6140   -0.3732   -0.0286   -1.0337   -1.3140   -0.1823   -2.4516   -1.0161
-0.4836    1.0738
 0.6866    0.1014    0.2747    0.1976    1.3250   -0.1638    0.6810   -0.6970
-1.3094   -0.4895
-0.3688   -0.0439   -1.3990   -1.555    -0.5760    0.6288   -0.590     0.3858
 0.4426    0.7767
 1.3578    1.6871    0.0372   -0.7146   -1.6735   -0.4092   -0.0570   -1.6772
 0.2012   -0.4616
 0.5436    0.0770   -1.6064   -1.0215   -0.2319   -0.4135    1.2533   -0.3875
-0.1241   -0.8345
-0.4735   -0.7025    1.0537   -0.7075   -0.5512    0.1986    1.2233    1.6326
-0.2364   -0.7857
 1.0287    0.1440    1.5999   -2.1740    0.5248   -0.2509    0.5960   -0.7616
-0.6389   -0.5827
-0.6030    0.7080   -0.8727   -0.4974   -1.0328    0.6545   -1.5837    1.5258
-0.3587    0.5648
 1.5494   -0.0697   -1.1898    0.6584    0.2365    0.0156    0.5443    0.2669
-0.7873   -1.1178
-1.9684   -0.8540    0.6153    1.3906   -1.0728    1.4699    0.1297    0.2265
 0.5617   -0.3587

b =
    0.4771
    0.0486
   -0.2100
   -1.0124
    0.0441
   -2.3974
   -2.3001
    1.6941
   -0.0035
    0.2600
```

计算结果为：



结果分析：

之后分别调整 A 与 b 的大小，验证 10，50，100，200 的矩阵运算时间，得到高斯算法的时间表格如下：

n 的值	10	50	100	200
用时(s)	0.006	0.012	0.025	0.095

2) **列主元消去法：**

输入的量：

系数矩阵 A 和常系数向量 b；

输出的量：

系数矩阵 A 和增广矩阵 B 的秩 RA,RB，方程组中未知量的个数 n 和有关方程组解 X 及其解的信息.

算法描述：

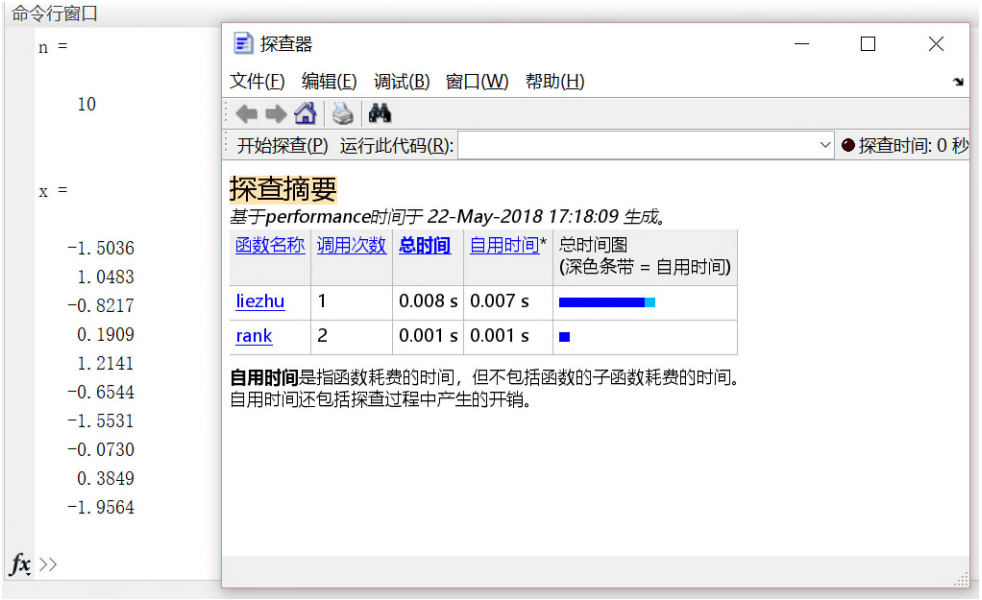
首先在矩阵 A 和向量 b 中做个判断，判断 A 与 b 的秩是否相等，若不等算法结束；若相等且等于 n，那么开始列主元消去法。先自顶向下，找出主元最大的一行，与当前行进行交换，然后最后面的每一行进行消元，得到一个主元全为 1 的上三角阵，具体实现如下：

```
for p = 1 : n-1
    [Y, j] = max(abs(B(p:n, p)));
    C = B(p, :);
    B(p, :) = B(j+p-1, :);
    B(j+p-1, :) = C;
    for k = p+1 : n
        m = B(k,p)/B(p,p);
        B(k, p:n+1) = B(k, p:n+1) - m * B(p, p:n+1);
    end
end
```

之后和高斯消去法一样，从下往上回代，计算得到 x 的值。

数值实验：

利用和高斯实验中相同的 10×10 矩阵，用列主元法进行计算得到结果为：



对比得到：列主元消去法与高斯消去法对于这个 10×10 的矩阵来说结果相同。

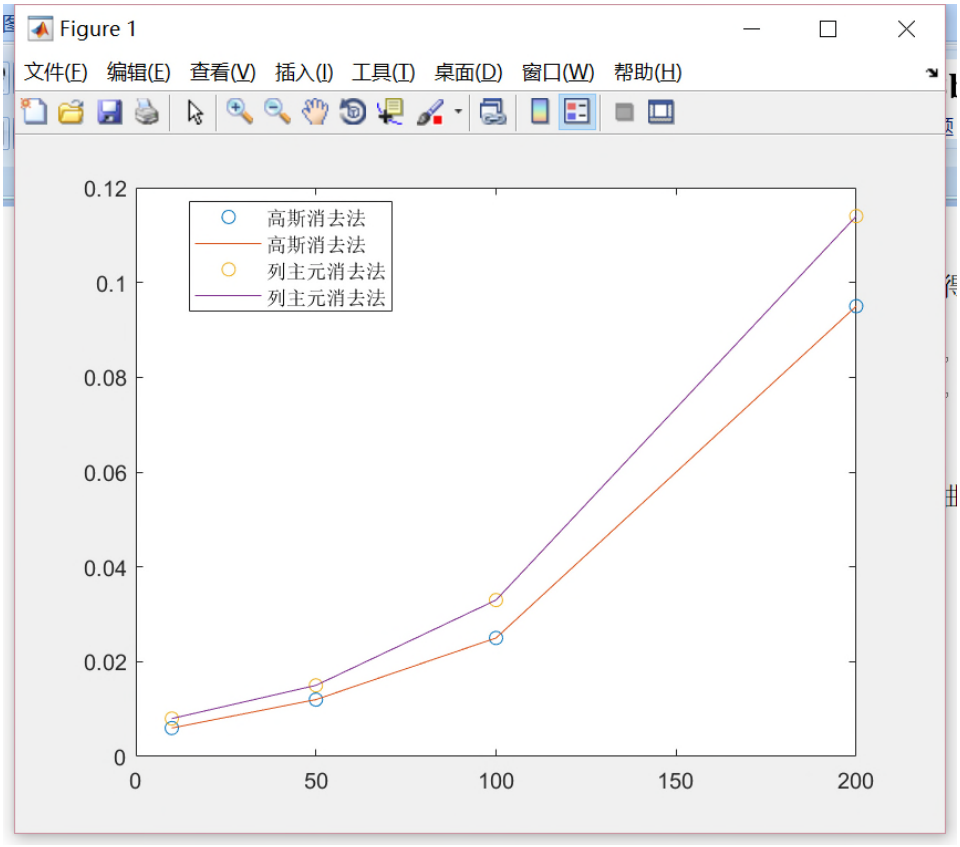
结果分析：

之后分别调整 A 与 b 的大小，验证 10，50，100，200 的矩阵运算时间，得到高斯算法的时间表格如下：

n 的值	10	50	100	200
用时(s)	0.008	0.015	0.033	0.114

3) **总结：**

在高斯消去法和列主元消去法中，我将时间放在了一张图中，得到时间曲线如下：



2. 请实现下述算法，求解线性方程组 Ax=b，其中 A 为 n x 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。

1) **Jacobi 迭代法：**

输入的量：

系数矩阵 A 和常系数向量 b；

输出的量：

方程组的解 x，以及迭代次数 k；

算法描述：

首先在矩阵 A 和向量 b 中做个判断，判断 A 与 b 的秩是否相等，若不等算法结束；若相等且等于 n，那么开始 Jacobi 迭代。计算出 A 矩阵对应的对角阵 D，上三角阵 U，下三角阵 L。计算出 $G = D^{-1}(L+U)$ 和 $d = D^{-1}b$。判断 G

的谱半径是否小于 1，若不是，说明该矩阵对 Jacobi 算法不收敛，跳出程序；若是，则开始迭代。迭代的具体实现如下：

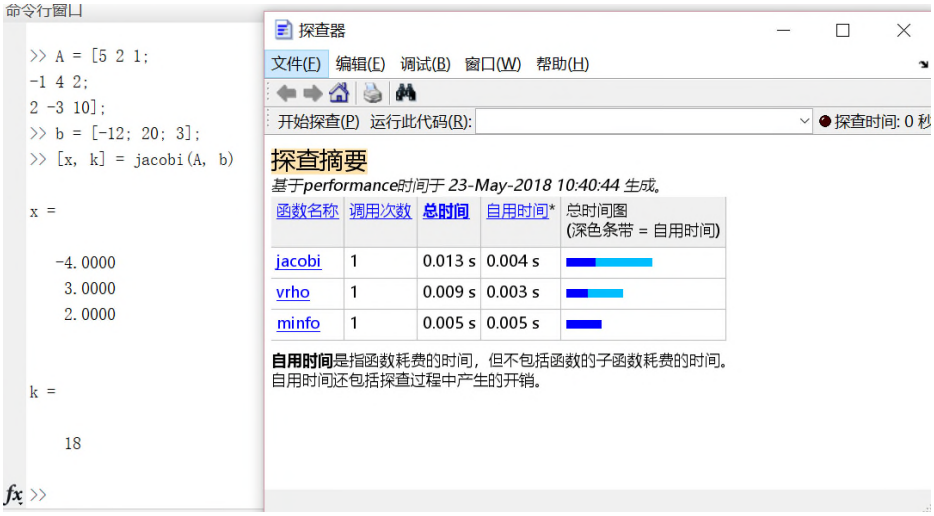
```
x = zeros(n, 1); %x初始设为0向量
x1 = x;
x2 = G * x + d;
while norm(x2 - x1, inf) > 10^(-4)
    x1 = x2;
    x2 = G * x2 + d;
    N = N + 1;
end
```

数值实验：

对以下矩阵进行计算：

$$\begin{cases} 5x_1 + 2x_2 + x_3 = -12 \\ -x_1 + 4x_2 + 2x_3 = 20 \\ 2x_1 - 3x_2 + 10x_3 = 3 \end{cases}$$

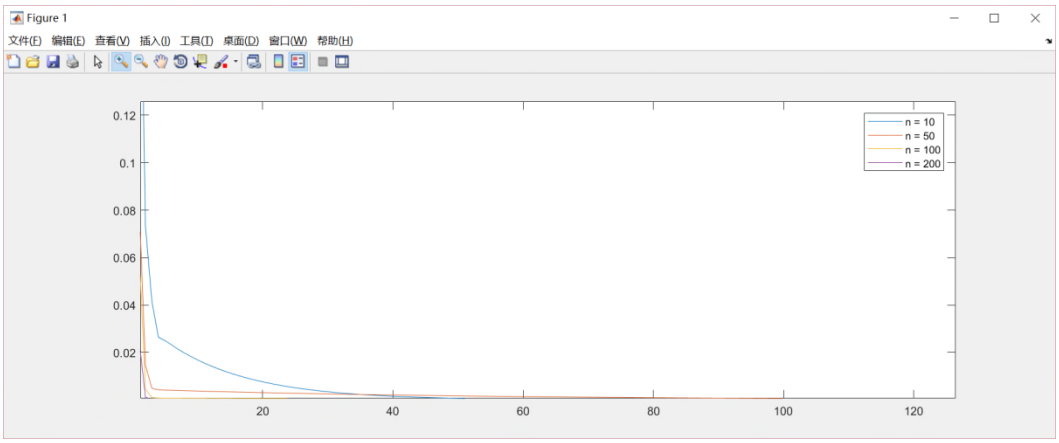
得到结果：



结果分析：

通过生成严格的对角占优矩阵，令 n = 10, 50, 100, 200 绘制 Jacobi 的收敛曲线如下：

其中测试数据来自随机生成的矩阵 A 和向量 b。



2) **Gauss-Seidel 迭代法：**

输入的量：

系数矩阵 A 和常系数向量 b；

输出的量：

方程组的解 x，以及迭代次数 k；

算法描述：

首先在矩阵 A 和向量 b 中做个判断，判断 A 与 b 的秩是否相等，若不等算法结束；若相等且等于 n，程序继续。计算出 A 矩阵对应的对角阵 D，上三角阵 U，下三角阵 L。计算出 $B=(D-L)^{-1}U$ 和 $g=(D-L)^{-1}$b。判断 B 的谱半径是否小于 1，若不是，说明该矩阵对 Gauss-Seidel 算法不收敛，跳出程序；若是，则开始迭代。迭代的具体实现如下：

```
k = 1;
x = B * x0 + g;
while norm(x - x0, inf) > 10^(-4)
      x0 = x;
      x = B * x + g;
      k = k + 1;
    if(k >= M)
          disp('Warning: 迭代太多次，可能不收敛');
          return
      end
end
```

数值实验：

使用 Jacobi 算法中的方程组进行实验：

向量的处置为 n 维空间中的零向量。

$$\begin{cases} 5x_1 + 2x_2 + x_3 = -12 \\ -x_1 + 4x_2 + 2x_3 = 20 \\ 2x_1 - 3x_2 + 10x_3 = 3 \end{cases}$$

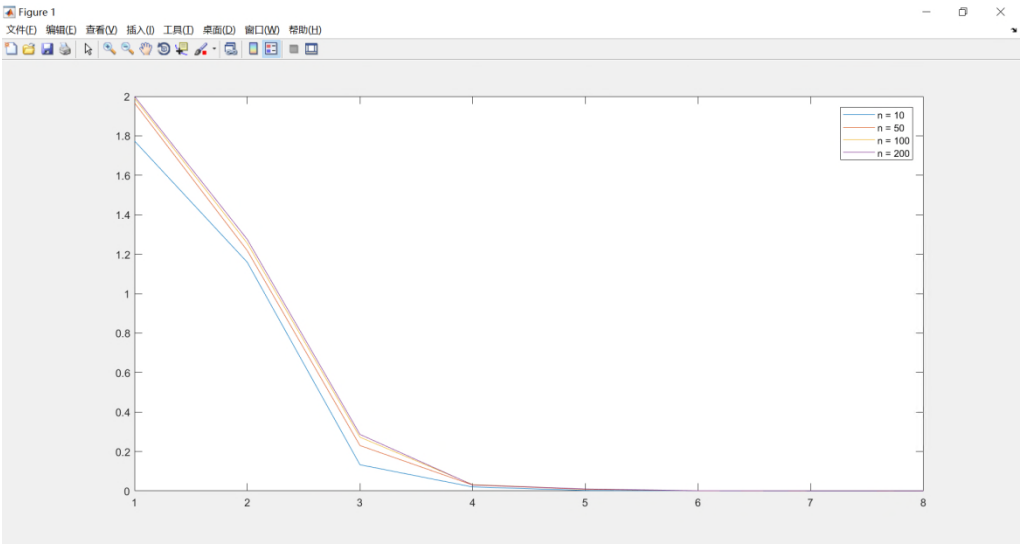
得到结果如下：

可以看出，这里的计算结果和 Jacobi 迭代中计算结果一样。

结果分析：

通过生成严格的对角占优矩阵，令 n = 10, 50, 100, 200 绘制 Gauss-Seidel 的收敛曲线如下：

其中测试数据来自随机生成的严格对角占优矩阵 A 和向量 b。



在实验中，我发现，高斯赛德尔迭代的收敛速度比 Jacobi 迭代要快很多，在 200x200 的矩阵的迭代中，也能在 10 次迭代内解出结果。

3) **逐次超松驰迭代法：**

输入的量：

系数矩阵 A，常系数向量 b 以及松弛因子 w；

输出的量：
方程组的解 x，以及迭代次数 k；

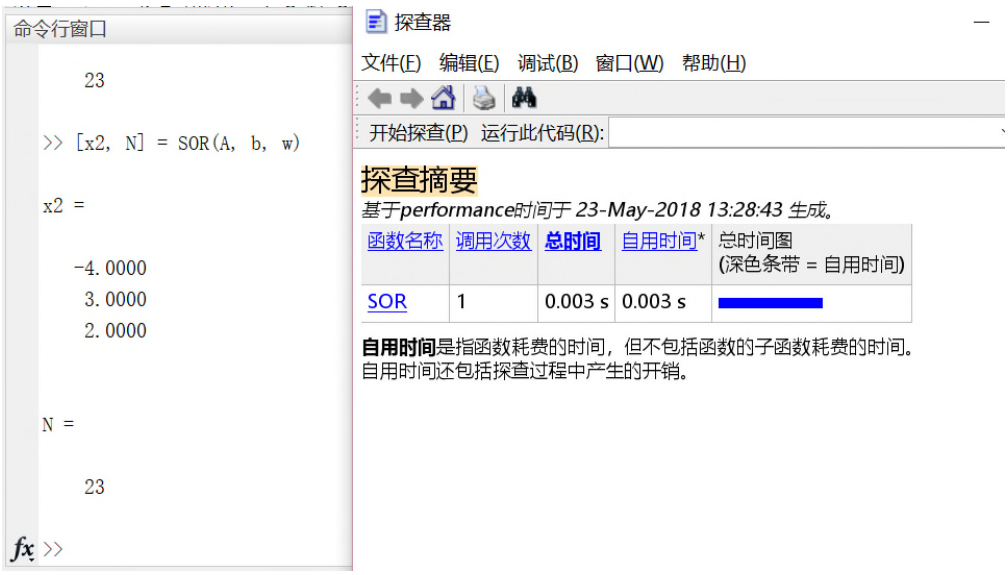
算法描述：
首先在矩阵 A 和向量 b 中做个判断，判断 A 与 b 的秩是否相等，若不等算法结束；若相等且等于 n，程序继续。再判断输入的松弛因子 w 的值是否在[0, 2]之间，若不是跳出函数，程序结束；若是，程序继续。计算出 A 矩阵对应的对角阵 D，上三角阵 U，下三角阵 L。计算出 $B=(D-\omega L)^{-1}((1-\omega)D+\omega U)$ 和 $f=\omega(D-\omega L)^{-1}b$，开始迭代，具体的接待如下：

```
x0 = zeros(n, 1);
x = B * x0 + f;
k = 1;
while norm(x - x0, inf) > 10^(-4)
    x0 = x;
    x = B * x0 + f;
    k = k + 1;
    if(k >= M)
        disp('Warning: µü´úÌ«¶à´Î£¬¿ÉÄÜ²»ÊÕÁ²');
        return
    end
end
```

数值实验：
继续使用上面的方程组：

$$\begin{cases} 5x_1+2x_2+x_3=-12 \\ -x_1+4x_2+2x_3=20 \\ 2x_1-3x_2+10x_3=3 \end{cases}$$

得到结果如下：

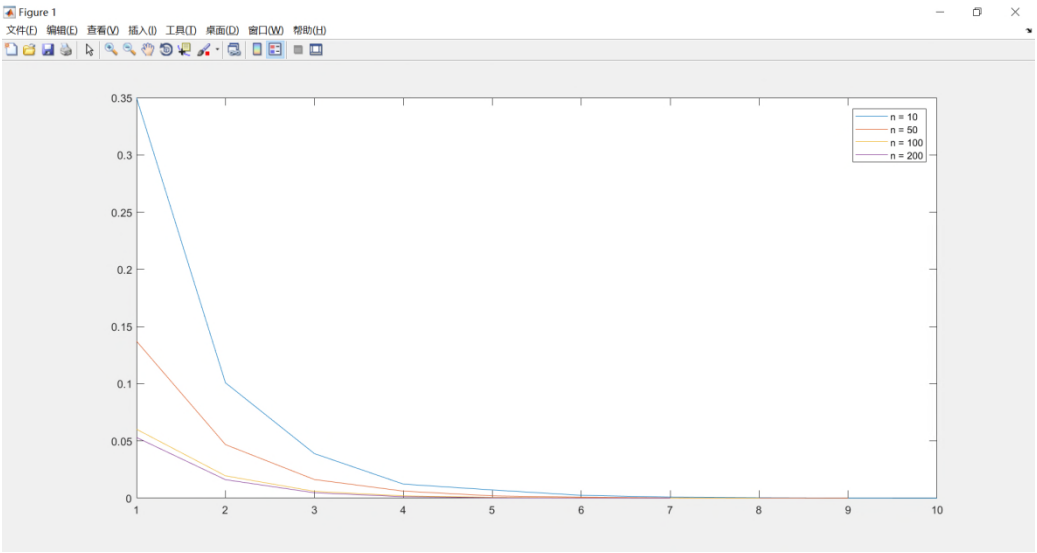


结果分析：

利用上面随机生成的严格对角占优矩阵，筛选出谱半径小于 1 的矩阵，来分析超松驰迭代的收敛情况。

得到结果如下：

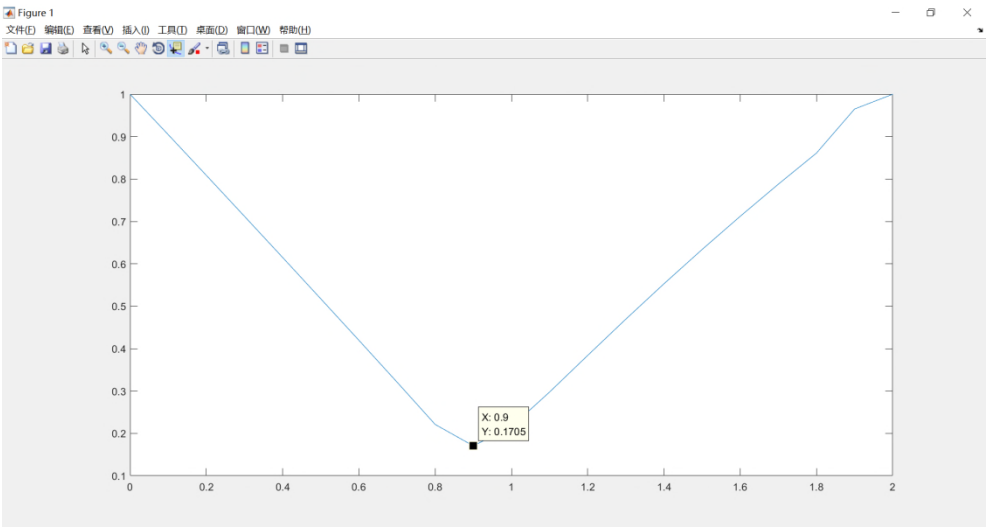
其中，矩阵大小分别为 10，50，100，200，且在每次 SOR 算法中，松弛因子都取 w=1.3。



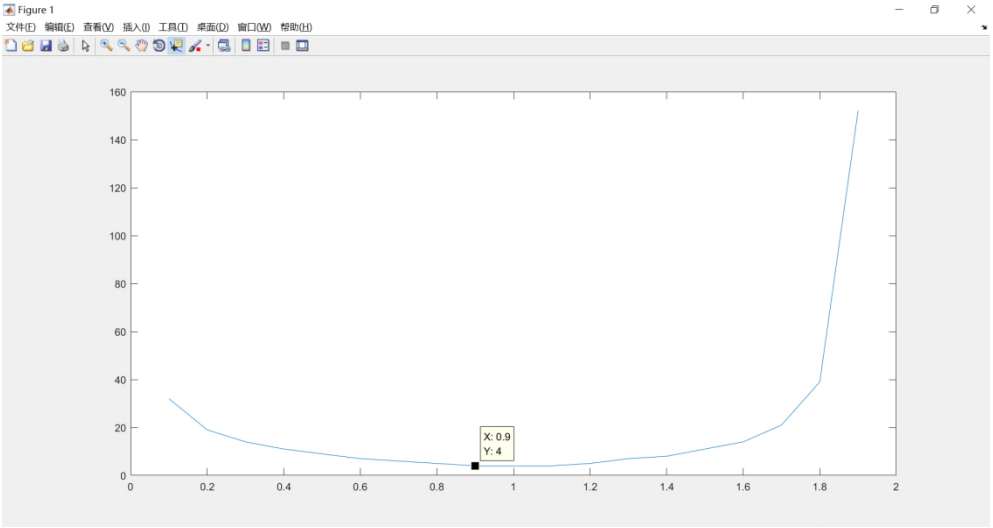
松弛因子的分析：

对于一个 200x200 的矩阵 A，计算 $B=(D-\omega L)^{-1}((1-\omega)D+\omega U)$ 矩阵的谱半径，w 取值为从 0 到 2，步长为 0.1 的值。

得到曲线如下（横坐标为 w，纵坐标为 B 的谱半径）：



不同的 w 对应的迭代步数如下：



4) **共轭梯度法：**

输入的量：

系数矩阵 A 和常系数向量 b；

输出的量：

方程组的解 x，以及迭代次数 k；

算法描述：

首先在矩阵 A 和向量 b 中做个判断，判断 A 与 b 的秩是否相等，若不等算法结束；若相等且等于 n，程序继续。令 x 矩阵的初始值为 n 维空间中的零向量，计算出第一步迭代中的 $r_0 = b - A \times x_0$ 和 $p_0 = r_0$，然后开始迭代，具体的代码如下：

```
while abs(p0) > 10^(-4)
    if(abs(r0) < 10^(-4))
        break;
```

```
    end
    a0 = r0'* r0 / (p0'* A*p0);
    x1 = x0 + a0 * p0;
    r1 = r0 - a0 * A * p0;
    b0 = r1'*r1 / (r0'*r0);
    p1 = r1 + b0 * p0;
    x0 = x1;
    r0 = r1;
    p0 = p1;
    k = k + 1;
end
```

数值实验：

仍然使用上面的方程组：

$$\begin{cases} 5x_1 + 2x_2 + x_3 = -12 \\ -x_1 + 4x_2 + 2x_3 = 20 \\ 2x_1 - 3x_2 + 10x_3 = 3 \end{cases}$$

在运行过程中，我发现程序死循环了，通过计算 A 的所有特征值，我发现，这不是一个正定矩阵，所以在共轭梯度迭代中，不能收敛。

```
>> eig(A)

ans =

   9.7927 + 0.0000i
   4.6037 + 2.1545i
   4.6037 - 2.1545i

>>
```

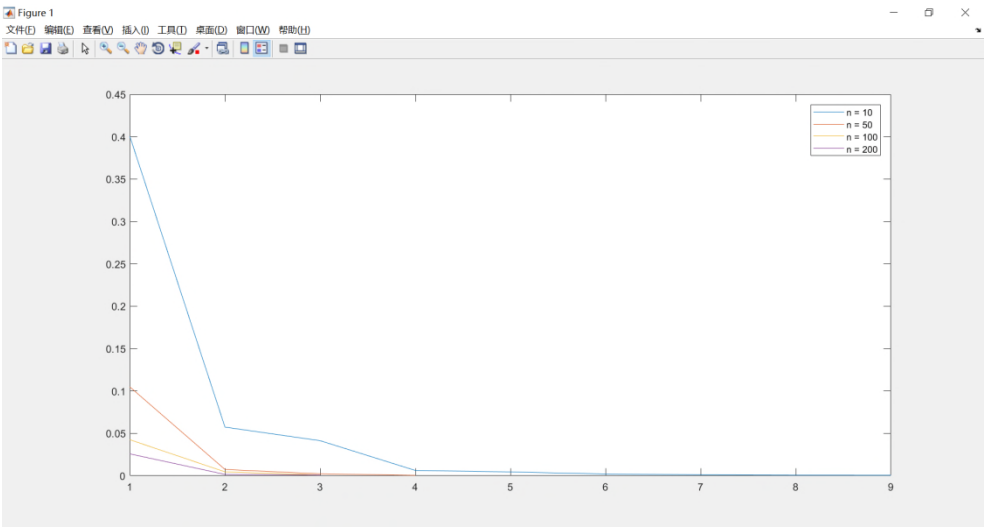
改用下面这个正定矩阵：

结果分析：

利用上面随机生成的严格对角占优矩阵，筛选出谱半径小于 1 的矩阵，来分析共轭梯度迭代法的收敛情况。

得到结果如下：

其中，矩阵大小分别为 10，50，100，200，



5) **总体分析：**

对于同一个 200x200 的正定矩阵，分别用上面六个算法进行计算，得到下面的时间比较：

算法	高斯消去法	列主元消去法	Jacobi 迭代法	Gauss-Seidel 迭代法	逐次超松驰迭代法	共轭梯度法
时间(s)	0.226	0.163	0.052	0.035	0.046	0.007

结论：共轭梯度法在时间上最优，高斯消去法在时间上最差。

3. 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。

算法描述：

先将数据导入到一个矩阵中，获得两列的节点编号，生成 n×n 的邻接矩阵。通过邻接矩阵，计算 pagerank 的矩阵，接下来开始迭代，迭代初向量为：$x = 1/n \times ones(n,1)$，系数设为 deta = 0.05，迭代公式为：newv = (1 - beta) * D * x + beta * 1/n * ones(n, 1)；

伪代码如下：

```
X = 1/n * ones(n, 1);
Eps = 10e – 5;
For i from 1 to n
    D(:, i) = D(:, i) / Sum(D(:, i);
For k from 1 to 10e+5
    newv = (1 - beta) * D * x + beta * 1/n * ones(n, 1);
    x = newv;
    if norm(newv – x) < eps
        break;
```

结果分析：