学号：20002462 姓名：刘子言

实验五 线性方程组的迭代法实验

一. 实验目的

（1）深入理解线性方程组的迭代法的设计思想，学会利用系数矩阵的性质以保证迭代过程的收敛性，以及解决某些实际的线性方程组求解问题。

（2）熟悉 Matlab 编程环境，利用 Matlab 解决具体的方程求根问题。

二. 实验要求

建立 Jacobi 迭代公式、Gauss-Seidel 迭代公式和超松弛迭代公式，用 Matlab 软件实现线性方程组求解的 Jacobi 迭代法、Gauss-Seidel 迭代法和超松弛迭代法，并用实例在计算机上计算。

三. 实验内容

1. 实验题目

3-1： 分别利用 Jacobi 迭代和 Gauss-Seidel 迭代求解下列线性方程组，取 $x=(0,0,0,0,0,0)^{\mathrm{T}}$，要求精度 $\varepsilon=10^{-5}$：

$$\begin{bmatrix} -4 & -1 & 0 & -1 & 0 & 0 \\ -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & -1 & 4 & 0 & 0 & -1 \\ -1 & 0 & 0 & 4 & -1 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 \\ 0 & 0 & -1 & 0 & -1 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} 0 \\ 5 \\ 0 \\ 6 \\ 2 \\ 6 \end{bmatrix}$$

3-2： 分别取 $\omega=1$、1.05、1.1、1.25 和 1.8，用超松弛法求解上面的方程组，要求精度为 $\varepsilon=10^{-5}$。找出迭代次数最少时的 ω 值。

2. 设计思想

要求针对上述题目，详细分析每种算法的设计思想。

3. 对应程序

列出每种算法的程序。

4. 实验结果

列出相应的运行结果截图。

四. 实验分析

对实验过程进行分析总结，对比求解线性方程组的不同方法的优缺点，指出每种方法的设计要点及应注意的事项，以及自己通过实验所获得的对线性方程组求解问题的各种解法的理解。

（注：不要改变实验报告的结构，写清页码和题号，源程序以自己的姓名命名，如 3-1 题可命名为“zhangsan_3-1.m”，运行截图中应出现自己的姓名和题号）

题目 1、分别利用 Jacobi 迭代和 Gauss-Seidel 迭代求解下列线性方程组，取 $x=(0,0,0,0,0,0)^{\mathrm{T}}$，要求精度 $\varepsilon=10^{-5}$：

$$\begin{bmatrix} -4 & -1 & 0 & -1 & 0 & 0 \\ -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & -1 & 4 & 0 & 0 & -1 \\ -1 & 0 & 0 & 4 & -1 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 \\ 0 & 0 & -1 & 0 & -1 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} 0 \\ 5 \\ 0 \\ 6 \\ 2 \\ 6 \end{bmatrix}$$

1、设计思想

（1）Jacobi 方法的设计思想：是将一般形式的线性方程组的求解归结为对角方程组求解过程的重复：

$$x_i^{(k+1)} = \frac{(b_i - \sum_{\substack{j=1 \\ j\neq i}}^{n} a_{ij} x_j^{(k)})}{a_{ii}}, i = 1,2,...,n$$

Jacobi 迭代因其计算规则简单而易于编写计算程序，通常用相邻两次迭代值的偏差来刻画迭代值的精度，为了防止迭代过程不收敛或者收敛速度过于缓慢，需设置最大迭代次数 N，如果迭代次数超过 N 还不能达到精度要求，则宣告迭代失败。

（2）Gauss-Seidel 方法的设计思想：是将一般形式的线性方程组的求解归结为下三角方程组求解过程的重复：

$$x_i^{(k+1)} = \frac{(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)})}{a_{ii}}, i = 1,2,...,n$$

Gauss-Seidel 迭代的特点是一旦求出变元的新值后，老值在之后的计算中便失去使用价值，因之可将新值存放在老值所占用的单元内。

2、对应程序

（1）Jacobi 方法对应程序：

```
function [ x,k ] = liuziyan_5_1_Jacobi( A,b,x0,N,emg )
%  用 Jacobi 迭代法求线性方程组 Ax=b 的解；
% x——用迭代法求得的线性方程组的近似解；k——迭代次数
% A——线性方程组的左端矩阵；b——右端向量；x0——迭代初始值；
% N——迭代次数上限，若迭代次数大于 N 则迭代失败；
% emg——控制精度；
n = length(A);
x1 = zeros(n,1);
```

```
x2 = zeros(n,1);
x1 = x0;
k = 0;
r = max(abs(b-A*x1));
while r > emg
    for i = 1:n
        sum = 0;
        for j = 1:n
            if i~=j
                sum = sum + A(i,j)*x1(j);
            end
        end
        x2(i) = (b(i)-sum)/A(i,i);
    end
    r = max(abs(x2-x1));
    x1 = x2;
    k = k+1;
    if k > N
        disp('迭代失败，返回');
        return;
    end
end
x = x1;
```

（2）Gauss-Seidel 方法对应程序：

```
function [ x,k ] = liuziyan_5_1_Gauss_Seidel( A,b,x0,N,emg )
%  用 Gauss-Seidel 迭代法求线性方程组 Ax=b 的解；
% x——用迭代法求得的线性方程组的近似解；k——迭代次数
% A——线性方程组的左端矩阵；b——右端向量；x0——迭代初始值；
% N——迭代次数上限，若迭代次数大于 N 则迭代失败；
% emg——控制精度；
n = length(A);
x1 = zeros(n,1);
x2 = zeros(n,1);
x1 = x0;
k = 0;
r = max(abs(b-A*x1));
while r > emg
    for i = 1:n
        sum = 0;
        for j = 1:n
            if j>i
```

```
                sum = sum + A(i,j)*x1(j);
            elseif j<i
                sum = sum + A(i,j)*x2(j);
            end
        end
        x2(i) = (b(i)-sum)/A(i,i);
    end
    r = max(abs(x2-x1));
    x1 = x2;
    k = k+1;
    if k > N
        disp('迭代失败，返回');
        return;
    end
end
x = x1;
```

3、实验结果

（1）Jacobi 方法的运行结果如下：

```
>> A=[-4,-1,0,-1,0,0;-1,4,-1,0,-1,0;0,-1,4,0,0,-1;-1,0,0,4,-1,0;0,-1,0,-1,4,-1;0,0,-1,0,-1,4];
>> b=[0,5,0,6,2,6]';x0=[0,0,0,0,0,0]';N=100;emg=10^-5;
>> [ x,k ] = liuziyan_5_1_Jacobi( A,b,x0,N,emg )

x =

  -0.882351329244557
   1.764702159765875
   0.999997208054992
   1.764704519795487
   1.941173246726976
   2.235289032483706


k =

    18
```

（2）Gauss-Seidel 方法的运行结果如下：

```
>> A=[-4,-1,0,-1,0,0;-1,4,-1,0,-1,0;0,-1,4,0,0,-1;-1,0,0,4,-1,0;0,-1,0,-1,4,-1;0,0,-1,0,-1,4];
>> b=[0,5,0,6,2,6]';x0=[0,0,0,0,0,0]';N=100;emg=10^-5;
>> [ x,k ] = liuziyan_5_1_Gauss_Seidel( A,b,x0,N,emg )

x =

  -0.882352088451626
   1.764705299933894
   0.999999655495174
   1.764705669173054
   1.941176072788437
   2.235293932070903


k =

    11
```

题目 2、分别取 $\omega=1$、1.05、1.1、1.25 和 1.8，用超松弛法求解上面的方程组，要求精度为 $\varepsilon=10^{-5}$。找出迭代次数最少时的 ω 值。

1、设计思想

SOR 迭代法可以看做是对的改进：由于新值通常优于老值，在将两者加工成松弛值时自然要求取松弛因子大于 1，以尽量发挥新值的优势。

$$\begin{cases} \tilde{x}_i^{(k+1)} = \dfrac{(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)})}{a_{ii}}, i = 1,2,...,n \\ x_i^{(k+1)} = \omega \cdot \tilde{x}_i^{(k+1)} + (1-\omega)\cdot x_i^{(k)} \end{cases}$$

SOR 迭代法设计时需要注意：一是每一步迭代的迭代值与松弛值是交替生成的；二是在计算迭代值时，会用松弛值取代相应的迭代值，因此可将迭代、松弛两个环节归并在一起。

2、对应程序

SOR 超松弛法的对应程序：

```
function [ x,k ] = liuziyan_5_2_SOR( A,b,x0,N,emg,w )
%  用 SOR 迭代法求线性方程组 Ax=b 的解；
% x——用迭代法求得的线性方程组的近似解；k——迭代次数
% A——线性方程组的左端矩阵；b——右端向量；x0——迭代初始值；
% N——迭代次数上限，若迭代次数大于 N 则迭代失败；
% emg——控制精度；w——松弛因子
n = length(A);
x1 = zeros(n,1);
x2 = zeros(n,1);
x1 = x0;
k = 0;
r = max(abs(b - A*x1));
while r > emg
    for i = 1:n
        sum = 0;
        for j = 1:n
            if j>=i
                sum = sum + A(i,j)*x1(j);
            elseif j<i
                sum = sum + A(i,j)*x2(j);
            end
        end
        x2(i) = x1(i) + w*(b(i)-sum)/A(i,i);
    end
    r = max(abs(x2-x1));
    x1 = x2;
    k = k+1;
    if k > N
        disp('迭代失败，返回');
        return;
    end
end
x = x1;
```

3、实验结果

（1）SOR 方法松弛因子 w = 1 时的运行结果：

```
>> A=[-4,-1,0,-1,0,0;-1,4,-1,0,-1,0;0,-1,4,0,0,-1;-1,0,0,4,-1,0;0,-1,0,-1,4,-1;0,0,-1,0,-1,4];
>> b=[0,5,0,6,2,6]';x0=[0,0,0,0,0,0]';N=100;emg=10^-5;w=1;
>> [ x,k ] = liuziyan_5_2_SOR( A,b,x0,N,emg,w )

x =

  -0.882352088451626
   1.764705299933894
   0.999999655495174
   1.764705669173054
   1.941176072788437
   2.235293932070903


k =

    11
```

（2）SOR 方法松弛因子 w = 1.05 时的运行结果：

```
>> A=[-4,-1,0,-1,0,0;-1,4,-1,0,-1,0;0,-1,4,0,0,-1;-1,0,0,4,-1,0;0,-1,0,-1,4,-1;0,0,-1,0,-1,4];
>> b=[0,5,0,6,2,6]';x0=[0,0,0,0,0,0]';N=100;emg=10^-5;w=1.05;
>> [ x,k ] = liuziyan_5_2_SOR( A,b,x0,N,emg,w )

x =

  -0.882351915407469
   1.764705768490600
   0.999999895219948
   1.764705913856286
   1.941176394684010
   2.235294091650618


k =

     9
```

（3）SOR 方法松弛因子 w = 1.1 时的运行结果：

```
>> A=[-4,-1,0,-1,0,0;-1,4,-1,0,-1,0;0,-1,4,0,0,-1;-1,0,0,4,-1,0;0,-1,0,-1,4,-1;0,0,-1,0,-1,4];
>>  b=[0,5,0,6,2,6]';x0=[0,0,0,0,0,0]';N=100;emg=10^-5;w=1.1;
>> [ x,k ] = liuziyan_5_2_SOR( A,b,x0,N,emg,w )

x =

  -0.882352180898984
   1.764706065804214
   0.999999999975546
   1.764706065793144
   1.941176559041766
   2.235294160299121


k =

    11
```

（4）SOR 方法松弛因子 w = 1.25 时的运行结果：

```
>> A=[-4,-1,0,-1,0,0;-1,4,-1,0,-1,0;0,-1,4,0,0,-1;-1,0,0,4,-1,0;0,-1,0,-1,4,-1;0,0,-1,0,-1,4];
>> b=[0,5,0,6,2,6]';x0=[0,0,0,0,0,0]';N=100;emg=10^-5;w=1.25;
>> [ x,k ] = liuziyan_5_2_SOR( A,b,x0,N,emg,w )

x =

  -0.882355625282821
   1.764704969556319
   0.999999999979323
   1.764704969599195
   1.941175849802975
   2.235293695482973


k =

    18
```

（5）SOR 方法松弛因子 w = 1.8 时的运行结果：

```
>> A=[-4,-1,0,-1,0,0;-1,4,-1,0,-1,0;0,-1,4,0,0,-1;-1,0,0,4,-1,0;0,-1,0,-1,4,-1;0,0,-1,0,-1,4];
>> b=[0,5,0,6,2,6]';x0=[0,0,0,0,0,0]';N=100;emg=10^-5;w=1.8;
>> [ x,k ] = liuziyan_5_2_SOR( A,b,x0,N,emg,w )
迭代失败，返回
```

对比上述不同松弛因子的实验结果可知，迭代次数最少时 w=1.05；同时可以看出 SOR 方法的计算量与松弛因子 w 的具体选择密切相关，w=1.05、1/1.1、1.25 的迭代次数 k 依次增加，当 w=1.8 时，迭代次数超过 100 次，宣告迭代失败。

四．实验分析

对实验过程进行分析总结，对比求解线性方程组的不同方法的优缺点，指出每种方法的设计要点及应注意的事项，以及自己通过实验所获得的对线性方程组求解问题的各种解法的理解。

答：

（1）由实验结果可以看出，在同种精度下，Gauss-Seidel 迭代法比 Jacobi 迭代法收敛速度快。一般来说，Gauss-Seidel 迭代法比 Jacobi 迭代法收敛要快，但有时反而比 Jacobi 迭代法要慢，甚至可能 Jacobi 迭代收敛而 Gauss-Seidel 迭代发散；而且 Jacobi 迭代法更易于并行化。因此两种方法各有优缺点，实际应用时要根据所需适当选取。

（2）由实验结果可知，当松弛因子 w=1 时，SOR 迭代法等同于 Gauss-Seidel 迭代法，这也符合理论推导的情况。此外，SOR 迭代法的收敛速度完全取决于松弛因子 w 的选取，选取不当可能导致迭代次数过大而迭代失败，但一个适当的松弛因子又能够大大的提高收敛速度。

（3）SOR 迭代具有计算公式简单、程序实现容易等特点，他是求解大型稀疏方程组的一种有效方法。如果松弛因子 w 选择合适，SOR 方法能够有效的提高收敛速度。