**实验五 线性方程组的迭代法实验**

学号：19001531 姓名：陈正江

**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解决具体的线性方程组求解问题。

**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**

**5-1：**分别利用Jacobi迭代和Gauss-Seidel迭代求解下列线性方程组，取，要求精度：

**2、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：



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：



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。

**3、对应程序**

1. **Jacobi方法代码如下：**

function [ x,k ] = czj5\_1\_1\_f( A,b,x0,N,emg )

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;

1. **Gauss-Seidel方法代码如下：**

function [ x,k ] = czj5\_1\_2\_f( A,b,x0,N,emg )

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;

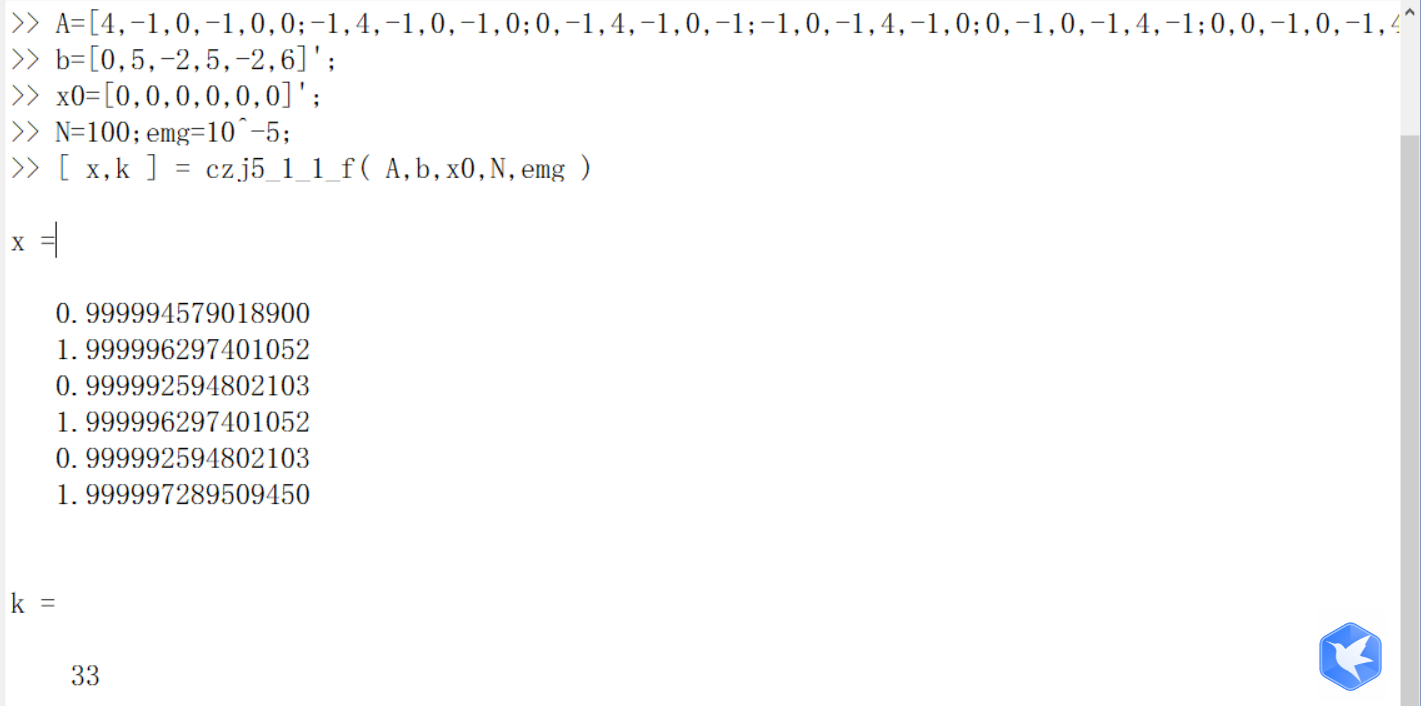
end

end

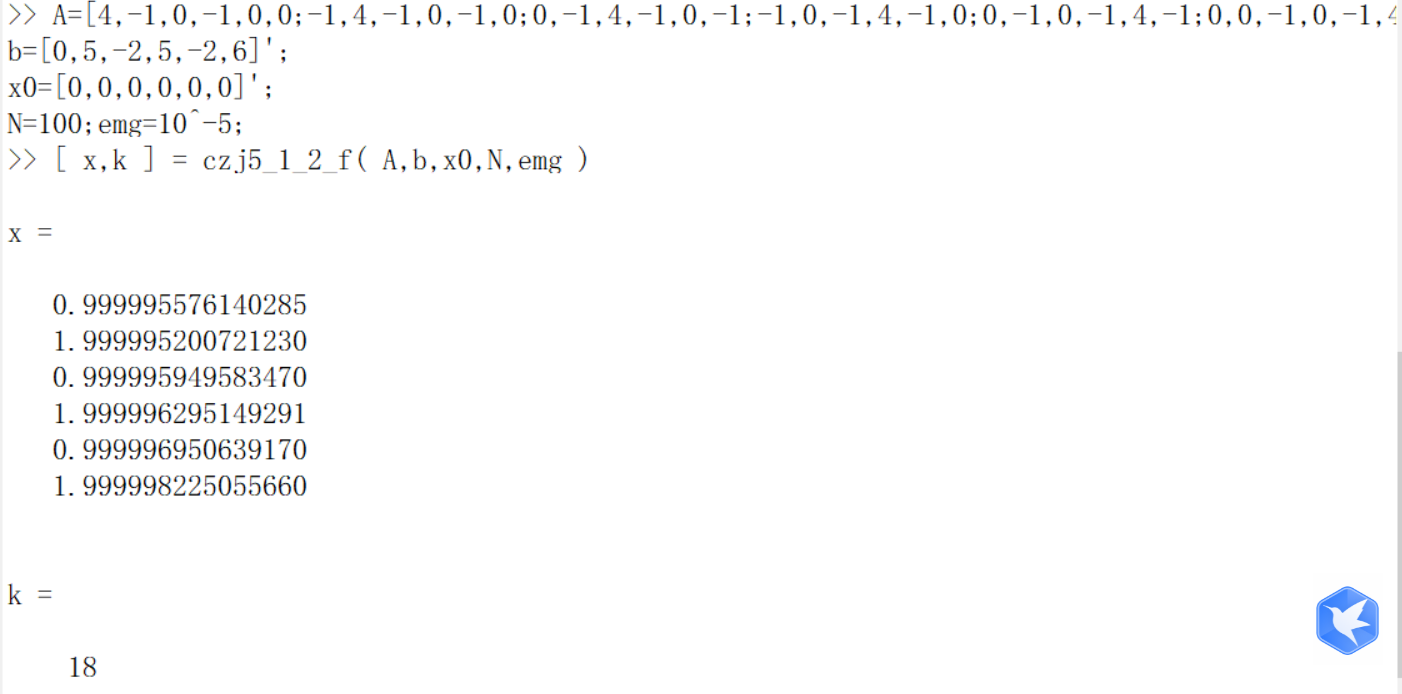
x = x1;

**4、实验结果**

**(1) Jacobi方法的运行结果：**

****

**(2) Gauss-Seidel方法的运行结果：**

****

**1、实验题目**

**5-2：**分别取、1.05、1.1、1.25、1.5和1.8，用超松弛法求解上面的方程组，要求精度为。找出迭代次数最少时的值。

**2、设计思想**

SOR迭代法新值通常优于老值，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。





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。

**3、对应程序**

1. **SOR迭代法代码：**

function [ x,k ] = czj5\_2\_1\_f( A,b,x0,N,emg,w )

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');

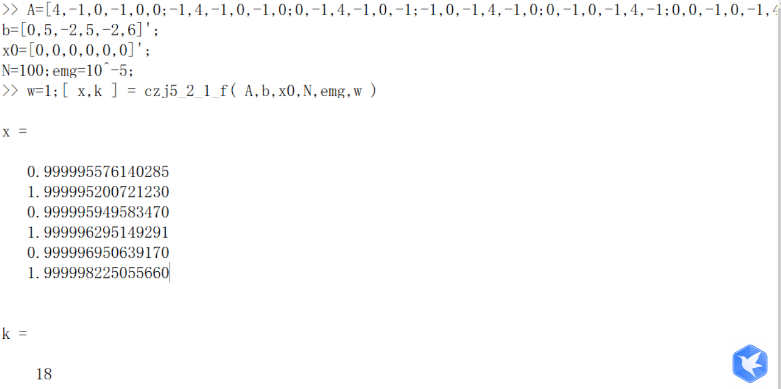
return;

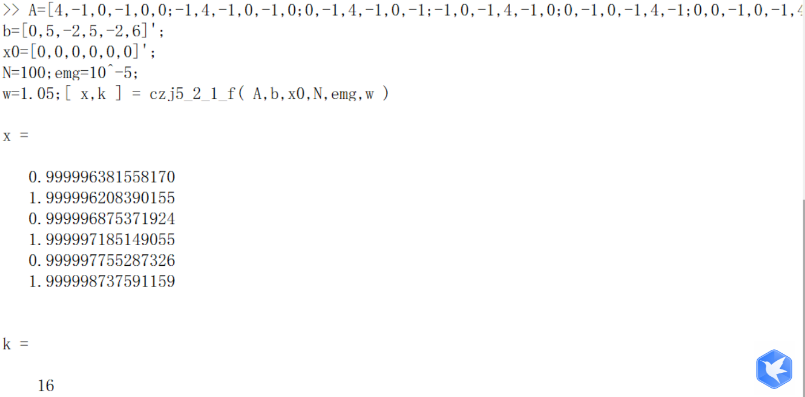
end

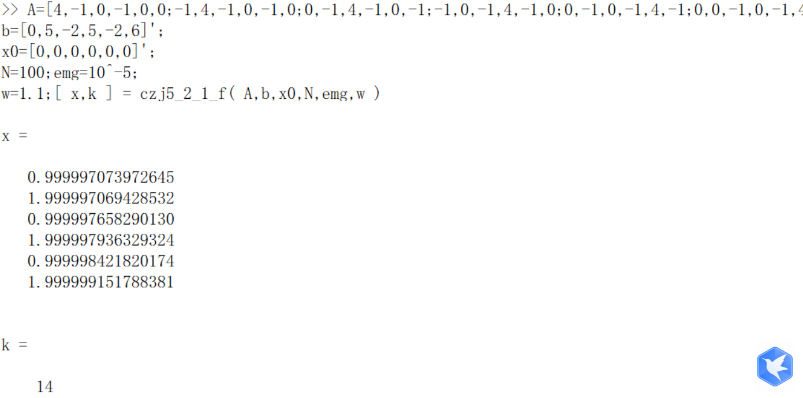
end

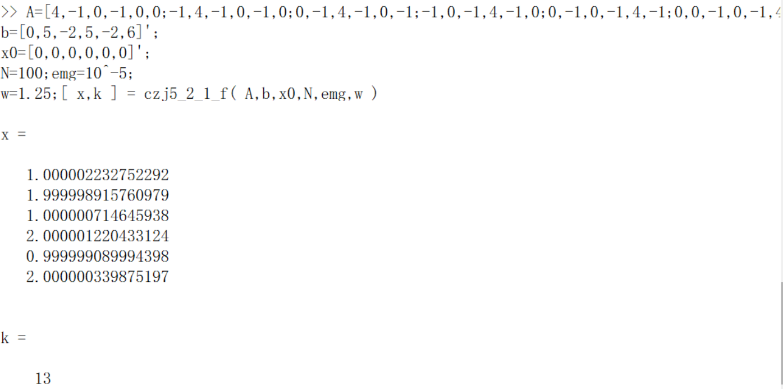
x = x1;

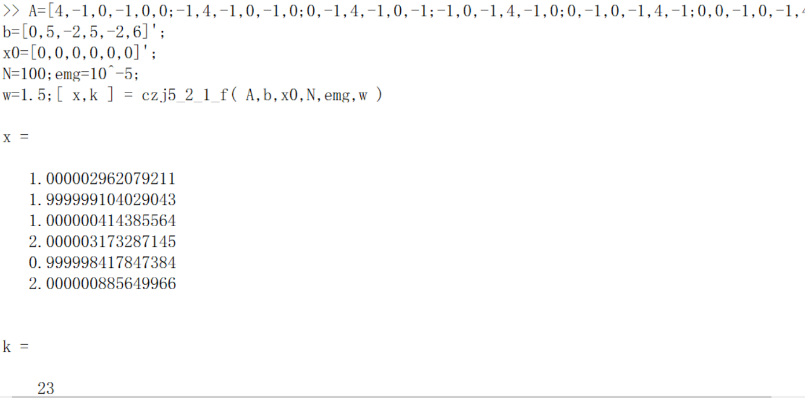
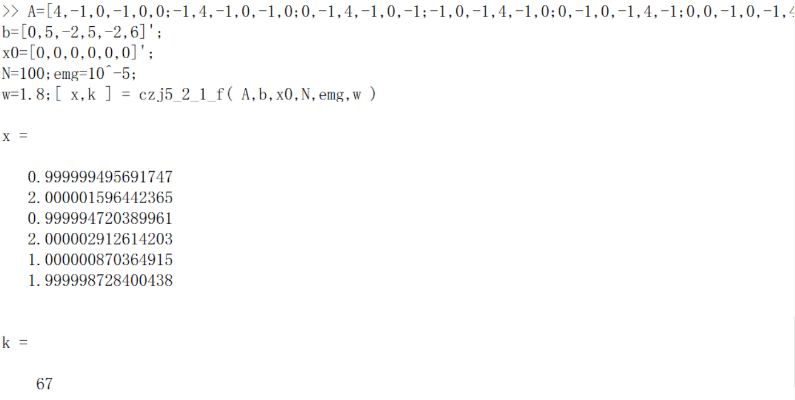
**4、实验结果**

**（1）SOR方法松弛因子w = 1时的运行结果： （2）w = 1.05时的运行结果：**

****

**（3）w = 1.1时的运行结果： （4）w = 1.25时的运行结果：**

****

**（5）w =1.5时的运行结果： （5）w =1.8时的运行结果：**

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=1.25；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、1.05、1.1和1.25的迭代次数k依次减少，w=1.5和1.8的迭代次数k依次增加。

**四. 实验体会**

通过以上实验，我有以下总结和收获：

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。两种方法各有优缺点，实际应用时要根据所需适当选取。当松弛因子w=1时，SOR迭代法等同于Gauss-Seidel迭代法。
2. SOR迭代具有计算公式简单、程序实现容易等特点，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。而且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。