云南大学数学与统计学实验教学中心

实验报告

课程名称：数值计算方法实验	**学期**：2015—2016 学年第一学期	**成绩**：
指导教师： 李耀堂	**学生姓名**：金洋	**学生学号**：20131910023
实验名称： **Jacobi 和 Gauss-Seidel 方法**		
实验编号：No. 5	**实验日期**： **2015.12.11**	**实验学时**： **3**
学院： 数学与统计学院	**专业**： **信息与计算科学**	**年级**：2013 级

一、实验目的

练习利用 Jacobi 和 Gauss-Seidel 方法解线性方程组；

二、实验内容

用Gauss-Seidel迭代法解下列线性方程组，要求当$\left\|x^{(k+1)}-x^{(k)}\right\| \leq 10^{-5}$时迭代终止。

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

三、实验环境

PC 计算机；

MATLAB R2014a；

四.实验方法：Jacobi 和 Gauss-Seidel 迭代法

方法简述：

(1)**Jacobi** 方法：

用迭代法解线性方程组 $Ax=b$，设 A 非奇异，且对角线元素 $a_{ii} \neq 0 \quad (i=1,2\cdots,n)$，把 A 分裂成三个矩阵之和 A=L+D+U，其中

$$L=\begin{bmatrix}0 & & & & \\ a_{21} & 0 & & & \\ a_{31} & a_{32} & 0 & & \\ \vdots & \vdots & \ddots & \ddots & \\ a_{n1} & a_{n2} & \cdots & a_{n,n-1} & 0\end{bmatrix},D=\begin{bmatrix}a_{11} & & & & \\ & a_{22} & & & \\ & & a_{33} & & \\ & & & \ddots & \\ & & & & a_{nn}\end{bmatrix},U=\begin{bmatrix}0 & a_{12} & a_{13} & \cdots & a_{1n} \\ & 0 & a_{23} & \cdots & a_{2n} \\ & & 0 & \cdots & a_{3n} \\ & & & \ddots & \vdots \\ & & & & 0\end{bmatrix}$$

由线性方程组进行变换：

$$\begin{pmatrix}a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & \ddots & \\ a_{n1} & a_{n2} & \cdots & a_{nn}\end{pmatrix}\begin{pmatrix}x_1 \\ x_2 \\ \vdots \\ x_n\end{pmatrix}=\begin{pmatrix}b_1 \\ b_2 \\ \vdots \\ b_n\end{pmatrix}\to\begin{pmatrix}1 & \frac{a_{12}}{a_{11}} & \cdots & \frac{a_{1n}}{a_{11}} \\ \frac{a_{21}}{a_{22}} & 1 & \cdots & \frac{a_{2n}}{a_{22}} \\ \vdots & & \ddots & \\ \frac{a_{n1}}{a_{nn}} & \frac{a_{n2}}{a_{nn}} & \cdots & 1\end{pmatrix}\begin{pmatrix}x_1 \\ x_2 \\ \vdots \\ x_n\end{pmatrix}=\begin{pmatrix}\frac{b_1}{a_{11}} \\ \frac{b_2}{a_{22}} \\ \vdots \\ \frac{b_n}{a_{nn}}\end{pmatrix}$$

$$\to\begin{pmatrix}1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1\end{pmatrix}\begin{pmatrix}x_1 \\ x_2 \\ \vdots \\ x_n\end{pmatrix}+\begin{pmatrix}0 & \frac{a_{12}}{a_{11}} & \cdots & \frac{a_{1n}}{a_{11}} \\ \frac{a_{21}}{a_{22}} & 0 & \cdots & \frac{a_{2n}}{a_{22}} \\ \vdots & & \ddots & \\ \frac{a_{n1}}{a_{nn}} & \frac{a_{n2}}{a_{nn}} & \cdots & 0\end{pmatrix}\begin{pmatrix}x_1 \\ x_2 \\ \vdots \\ x_n\end{pmatrix}=\begin{pmatrix}\frac{b_1}{a_{11}} \\ \frac{b_2}{a_{22}} \\ \vdots \\ \frac{b_n}{a_{nn}}\end{pmatrix}$$

$$\to\begin{pmatrix}x_1 \\ x_2 \\ \vdots \\ x_n\end{pmatrix}=-\begin{pmatrix}0 & \frac{a_{12}}{a_{11}} & \cdots & \frac{a_{1n}}{a_{11}} \\ \frac{a_{21}}{a_{22}} & 0 & \cdots & \frac{a_{2n}}{a_{22}} \\ \vdots & & \ddots & \\ \frac{a_{n1}}{a_{nn}} & \frac{a_{n2}}{a_{nn}} & \cdots & 0\end{pmatrix}\begin{pmatrix}x_1 \\ x_2 \\ \vdots \\ x_n\end{pmatrix}+\begin{pmatrix}\frac{b_1}{a_{11}} \\ \frac{b_2}{a_{22}} \\ \vdots \\ \frac{b_n}{a_{nn}}\end{pmatrix}$$

即得到 Jacobi 迭代的分量形式：

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

用矩阵形式表示：

$$\begin{aligned}&Ax=b\\&\Rightarrow D^{-1}Ax=D^{-1}b\\&\Rightarrow [I+(D^{-1}A-I)]x=D^{-1}b\\&\Rightarrow x=(I-D^{-1}A)x+D^{-1}b\end{aligned}$$

(2)Gauss-Seidel 迭代法

在 Jacobi 基础上，设想方法收敛，第(K+1)次的分量比第(K)次的分量更接近于真实解，为了加快收敛速度，在计算 $x^{(K+1)}$ 的第 i 个分量时，所用的 $x^{(K)}$ 的前 i-1 个分量换成新算好的值，即用 $x_1^{(K+1)}, x_2^{(K+1)}, \cdots, x_{i-1}^{(K+1)}, x_i^{(K)}, \cdots, x_n^{(K)}$ 来计算 $x_i^{(K+1)}$.

Gauss-Seidel 迭代的分量形式

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

Gauss-Seidel 迭代的矩阵形式为：

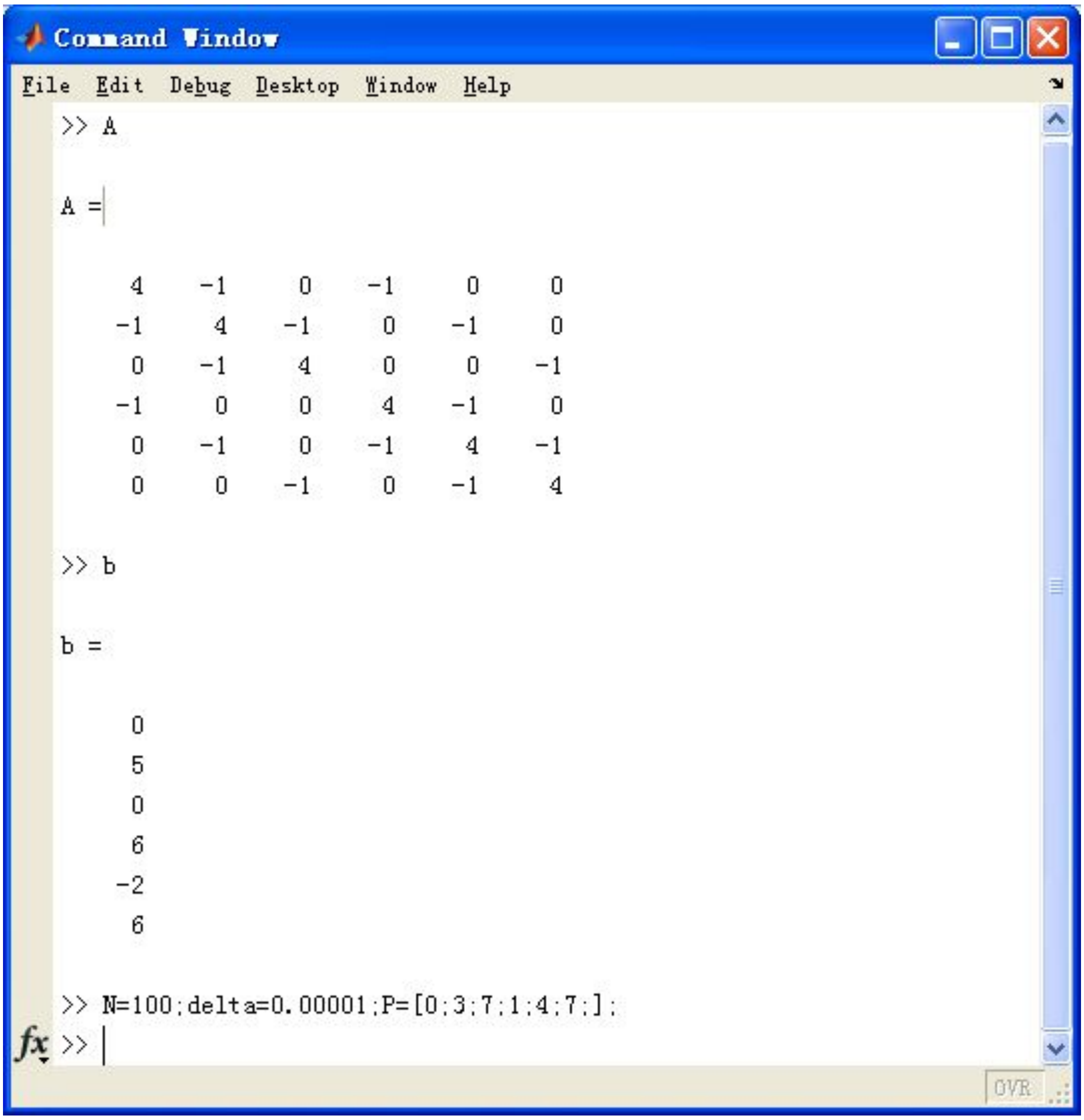
$$x^{(K+1)} = -(D+L)^{-1} U x^{(K)} + (D+L)^{-1} b$$

五、实验过程

1 实验步骤

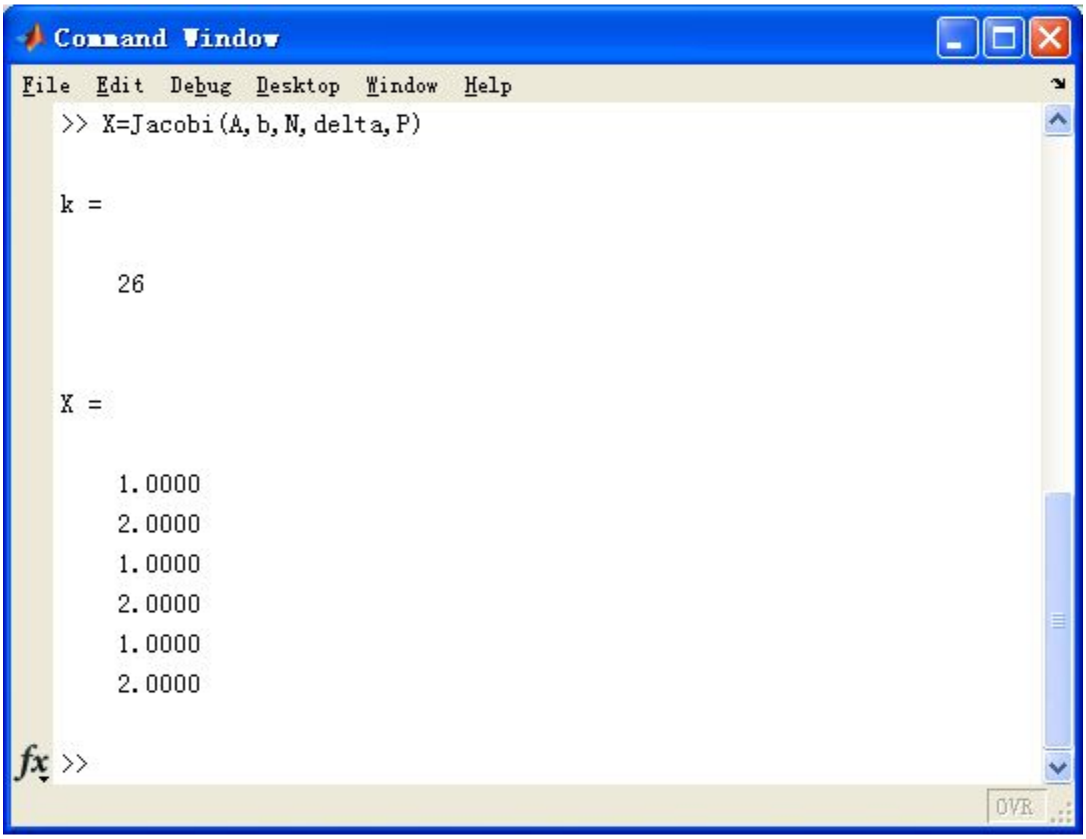
①编程：用 MATLAB 语言编出源程序。
②开机,键入所编程序源代码.
③调试程序, 修改错误至能正确运行.
④运行程序并输出计算结果.
⑤尝试改进

输入 A，b，最大迭代次数 N，精确度 delta，初值 P 如下：



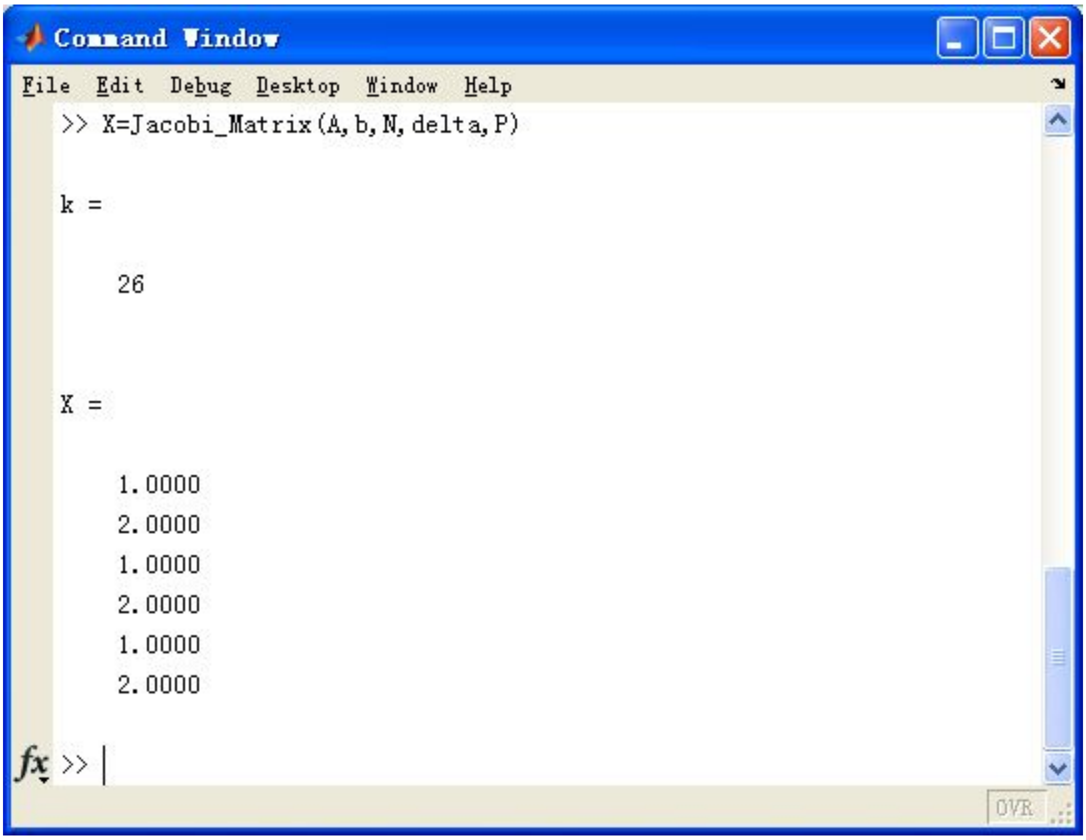
运行结果

①Jacobi 方法分量形式运行结果如下：



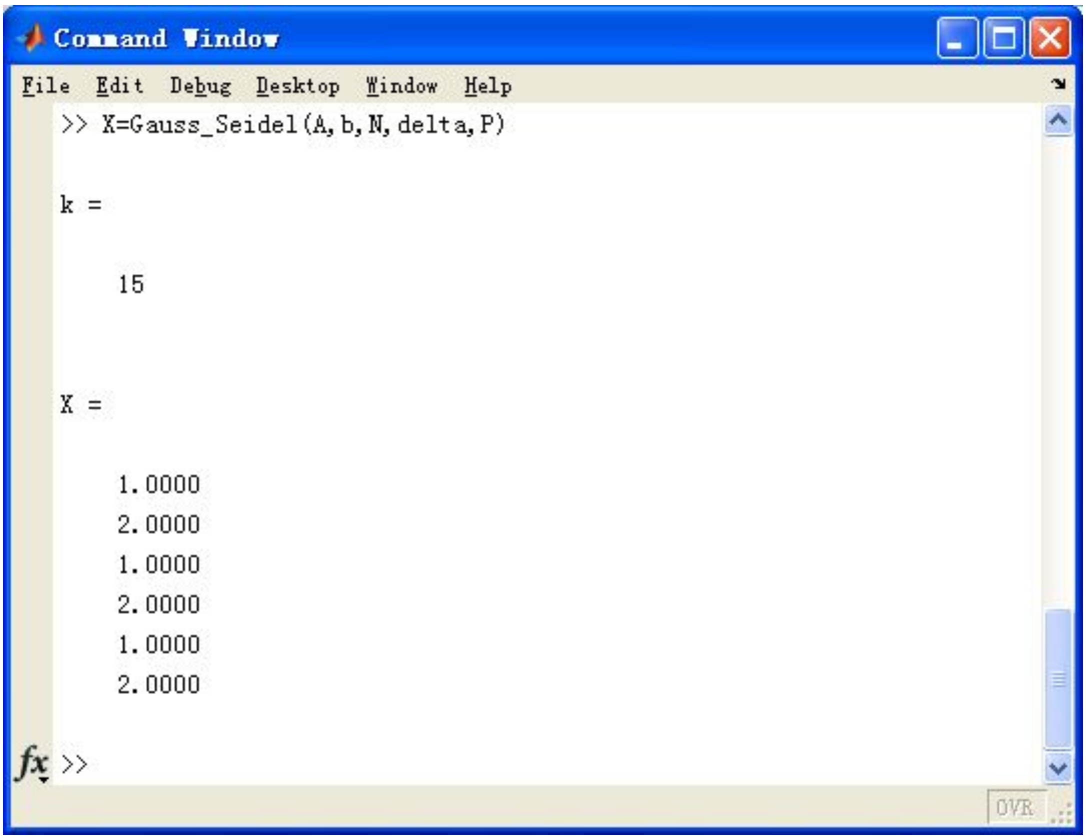
即迭代次数 26 次，解为 X=(1.0000, 2.0000, 1.0000, 2.0000, 1.0000, 2.0000)’

②Jacobi 方法矩阵形式运行结果如下：



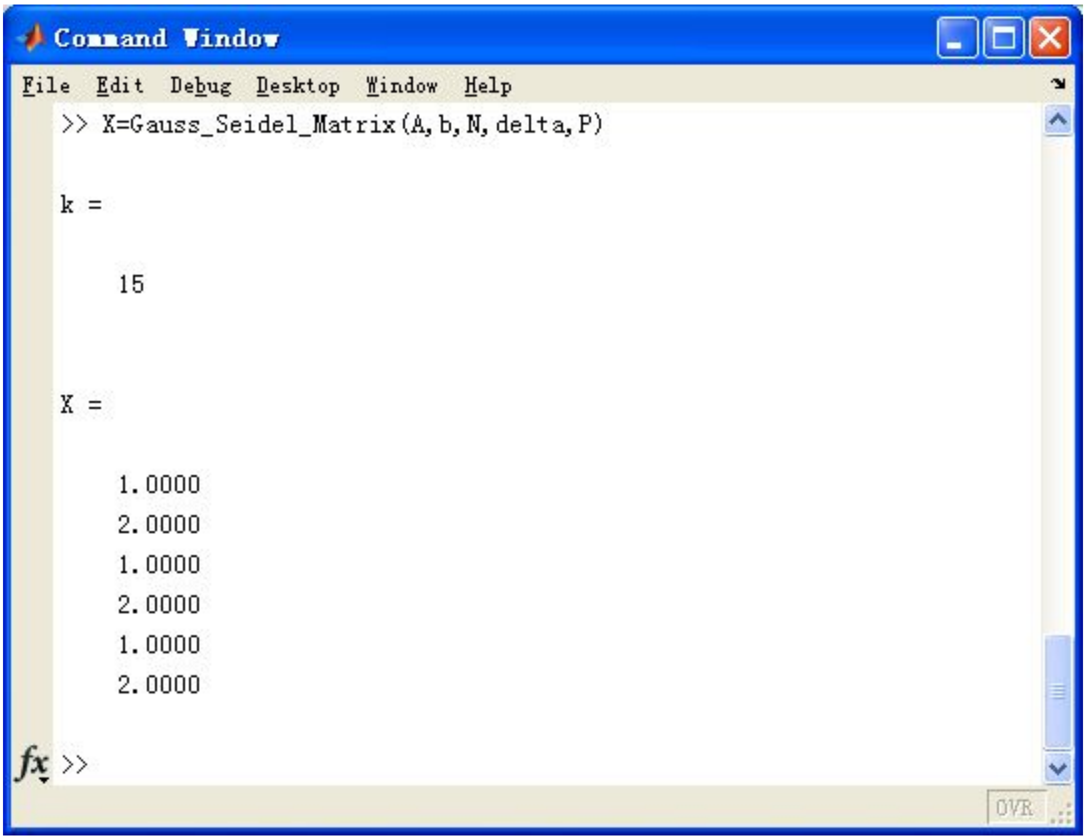
即迭代次数 26 次，解为 X=(1.0000, 2.0000, 1.0000, 2.0000, 1.0000, 2.0000)’

③Gauss-Seidel 方法分量形式运行结果如下：



即迭代次数 15 次，解为 X=(1.0000, 2.0000, 1.0000, 2.0000, 1.0000, 2.0000)’

③Gauss-Seidel 方法矩阵形式运行结果如下：



即迭代次数 15 次，解为 X=(1.0000, 2.0000, 1.0000, 2.0000, 1.0000, 2.0000)’

2 关键代码及其解释

①Jacobi 方法(分量形式)

```
while (k<=N)
   for i=1:n
      %计算新的值
      X(i)=( b(i)- A(i,[1:i-1,i+1:n]) * P([1:i-1,i+1:n])  ) / A(i,i);
   end

   %检验精读，若精读符合要求，退出循环
   if (max(abs(X-P) )<=delta)
       break;
   end
   %将新值保存在 P 中
   P=X;
   %迭代次数加 1
   k=k+1;
end
```

②Jacobi 方法(矩阵形式):

```
while (k<=N)
   X=(I-INVD*A)*P+INVD*b;

   if (max(abs(X-P) )<=delta)
       break;
   end
   %将新值保存在 P 中
   P=X;
   %迭代次数加 1
   k=k+1;
end
```

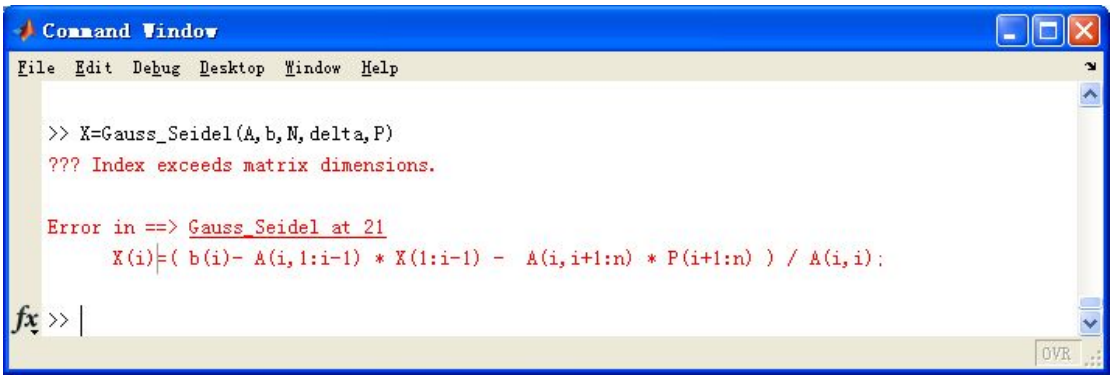
③Gauss-Seidel(分量形式):

```
while (k<=N)
   for i=1:n
      X(i)=( b(i)- A(i,1:i-1) * X(1:i-1) - A(i,i+1:n) * P(i+1:n) ) /
A(i,i);
   end

   %检验精读，若精读符合要求，退出循环
   if (max(abs(X-P) )<=delta)
       break;
   end
   %将新值保存在 P 中
   P=X;
   %迭代次数加 1
   k=k+1;
End
```

④Gauss-Seidel(矩阵形式):

```
while (k<=N)
   X=-inv(D+L)*U*P+inv(D+L)*b;

   if (max(abs(X-P) )<=delta)
       break;
   end
   %将新值保存在 P 中
   P=X;
   %迭代次数加 1
   k=k+1;
end
```

3 调试过程



①矩阵维度使用出错，未将同样维度的矩阵输入正确

六、实验总结

1．遇到的问题及解决过程

①Gauss-Seidel 的分量形式方法，$x_1^{(K+1)},x_2^{(K+1)},\cdots,x_{i-1}^{(K+1)},x_i^{(K)},\cdots,x_n^{(K)}$来计算$x_i^{(K+1)}$一开始难以解决；

②由分量形式推到出矩阵形式碰到问题，引入 D,L,U 矩阵后得到解决；

③不知如何表示出一个矩阵的上三角、下三角、对角线矩阵，通过查阅文档后顺利表达出来 L=tril(A,-1);U=triu(A,1);D=diag(diag(A));

2．产生的错误及原因分析

①MATLAB 计算中，会检验矩阵的维度来确定运算是否合理，故在前期要仔细检测输入数据的正确性；

②diag()函数有如下两种用法：

X = diag(v,k)

其中 v 是一个含有 n 个元素的向量，该调用格式可以构造一个 n+abs(k)阶的方阵 X。并把 v 作为方阵 X 的第 k 条对角线（k 大于 0，表示主对角线上方的第 k 条对角线，k 小于 0 表示主对角线下侧的第 k 条对角线，k 等于 0 表示主对线）。

v = diag(X,k)

以向量形式返回矩阵 X 中第 k 条对角线上的元素。

③通过 D=diag(diag(A))便可得到 A 的对角线矩阵。

3．体会和收获。

①通过查阅相关文档知，Jacobi 迭代公式、Gauss-Seidel 迭代公式是否收敛的充要条件是谱半径ρ<1，故在计算之前可先计算谱半径的值来判断是否

收敛;

②对于同一个方程组，jacobi 迭代公式、Gauss-Seidel 迭代公式的收敛性可能不同，Jacobi 方法收敛并不能保证 Gauss-Seidel 方法收敛，反之也如此。但当二者均收敛时，Gauss-Seidel 方法比 Jacobi 方法收敛速度快。

③对角线元素 D=diag(diag(A));

七、程序源代码:

①Jacobi 方法(分量形式)

Jacobi.m

```
function X=Jacobi(A,b,N,delta,P)
%Input  -A n*n 非奇异矩阵
%       -b n*1 矩阵
%       -N 最大迭代次数
%       -delta 精确度
%       -P n*1 矩阵，初始值
%Output -X n*1 使用 Jacobi 方法(分量形式)得到的方程组 AX=b 的解

%求维数
n=length(b);

%初始化 X
X=zeros(n,1);

%k 记录迭代次数
k=1;
while (k<=N)
   for i=1:n
      %计算新的值
      X(i)=( b(i)- A(i,[1:i-1,i+1:n]) * P([1:i-1,i+1:n])  ) / A(i,i);
   end

   %检验精读，若精读符合要求，退出循环
   if (max(abs(X-P) )<=delta)
       break;
   end
   %将新值保存在 P 中
   P=X;
   %迭代次数加 1
   k=k+1;
end
%显示迭代次数
k
```

②Jacobi 方法(矩阵形式)：

Jacobi_Matrix.m

```
function X=Jacobi_Matrix(A,b,N,delta,P)
%Input  -A n*n 非奇异矩阵
%       -b n*1 矩阵
%       -N 最大迭代次数
%       -delta 精确度
%       -P n*1 矩阵，初始值
%Output -X n*1 使用 Jacobi 方法(矩阵形式)得到的方程组 AX=b 的解

%求维数
n=length(b);

%初始化 X
X=zeros(n,1);

%分别得到 A 对角线矩阵 D,D 的逆矩阵 INVD,单位矩阵 I
D=diag(diag(A));INVD=inv(D);
I=eye(n);



%k 记录迭代次数
k=1;
while (k<=N)
   X=(I-INVD*A)*P+INVD*b;

   if (max(abs(X-P) )<=delta)
       break;
   end
   %将新值保存在 P 中
   P=X;
   %迭代次数加 1
   k=k+1;
end

%显示迭代次数
k
```

③Gauss-Seidel(分量形式)：

Gauss_Seidel.m

```
function X=Gauss_Seidel(A,b,N,delta,P)
%Input  -A n*n 非奇异矩阵
%       -b n*1 矩阵
%       -N 最大迭代次数
%       -delta 精确度
%       -P n*1 矩阵，初始值
%Output -X n*1 使用 Gauss-Seidel(分量形式)方法得到的方程组 AX=b 的解

%求维数
n=length(b);

%初始化 X
X=zeros(n,1);

%k 记录迭代次数
k=1;
while (k<=N)
   for i=1:n
      %计算新的值,注意与 Jacobi 的 X(i)=( b(i)- A(i,[1:i-1,i+1:n]) *
P([1:i-1,i+1:n])
      %) / A(i,i)做比较
      X(i)=( b(i)- A(i,1:i-1) * X(1:i-1) -  A(i,i+1:n) * P(i+1:n) ) /
A(i,i);
   end

   %检验精读，若精读符合要求，退出循环
   if (max(abs(X-P) )<=delta)
       break;
   end
   %将新值保存在 P 中
   P=X;
   %迭代次数加 1
   k=k+1;
end

%显示迭代次数
k
```

④**Gauss-Seidel(矩阵形式)：**

Gauss_Seidel_Matrix.m

```
function X=Gauss_Seidel_Matrix(A,b,N,delta,P)
%Input  -A n*n 非奇异矩阵
%       -b n*1 矩阵
%       -N 最大迭代次数
%       -delta 精确度
%       -P n*1 矩阵，初始值
%Output -X n*1 使用 Gauss-Seidel 方法(矩阵形式)得到的方程组 AX=b 的解

%求维数
n=length(b);

%初始化 X
X=zeros(n,1);

%分别得到 A 的下三角矩阵 L,上三角矩阵 U,对角线矩阵 D
L=tril(A,-1);
U=triu(A,1);
D=diag(diag(A));

%k 记录迭代次数
k=1;
while (k<=N)
   X=-inv(D+L)*U*P+inv(D+L)*b;

   if (max(abs(X-P) )<=delta)
      break;
   end
   %将新值保存在 P 中
   P=X;
   %迭代次数加 1
   k=k+1;
end

%显示迭代次数
k
```

八、教师评语