

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：松弛法		
实验编号：No. 6	实验日期：2015.12.16	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

练习利用松弛法解线性方程组；

二、实验内容

用松弛法解

$$\begin{cases} 4x_1 - x_2 = 1 \\ -x_1 + 4x_2 - x_3 = 4 \\ -x_2 + 4x_3 = -3 \end{cases}$$

分别取 $\omega = 1.03, \omega = 1, \omega = 1.1$ 。要求当 $\|x^{(k)} - x^{(k-1)}\| < 5 \times 10^{-6}$ 时迭代终止，并对每个 ω 值确定迭代次数（初值 $x^{(0)} = (0, 0, 0)^T$ ）。

三、实验环境

PC 计算机；

MATLAB R2014a；

四.实验方法：松弛法

方法简述：

设 $Ax = b$ ，令 $B = I - A$

$$\therefore (I - B)x = b \Rightarrow x = Bx + b$$

$$\therefore x^{(k+1)} = Bx^{(k)} + b$$

$$\text{令 } r^{(k)} = b - Ax^{(k)}, \text{ 则 } x^{(k+1)} = Bx^{(k)} + Ax^{(k)} + r^{(k)} = x^{(k)} + r^{(k)}.$$

松弛法的基本思想便是在校正项前乘一个参数 ω ，通过调节 ω 来调控算法的收敛性。以下用该思想来改进解 $Ax = b$ 的 Gauss-Seidel 方法。

(1) SOR 分量形式:

可以对 Gauss-Seidel 的分量形式做如下变换

$$\begin{aligned}x_i^{(k+1)} &= \frac{1}{a_{ii}}(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^n a_{ij}x_j^{(k)}) \\&= \frac{1}{a_{ii}}(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} + a_{ii}x_i^{(k)} - \sum_{j=i}^n a_{ij}x_j^{(k)}) \\&= x_i^{(k)} + \frac{1}{a_{ii}}(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i}^n a_{ij}x_j^{(k)})\end{aligned}$$

在校正项上乘一个因子 ω ，就成了松弛法的分量形式:

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

当 $\omega > 1$, 称为逐次超松弛迭代法;

当 $\omega < 1$, 称为低松弛迭代法;

当 $\omega = 1$, 即为 Gauss-Seidel 迭代法;

(2) SOR 矩阵形式:

可以对 (*) 式改写为矩阵形式:

$$x^{(k+1)} = x^{(k)} + \omega D^{-1}[b - Lx^{(k+1)} - (D+U)x^{(k)}]$$

即

$$\begin{aligned}x^{(k+1)} &= (D + \omega L)^{-1}[(1-\omega)D - \omega U]x^{(k)} + \omega(D + \omega L)^{-1}b \\&= H_{\omega}x^{(k)} + \omega(D + \omega L)^{-1}b\end{aligned}$$

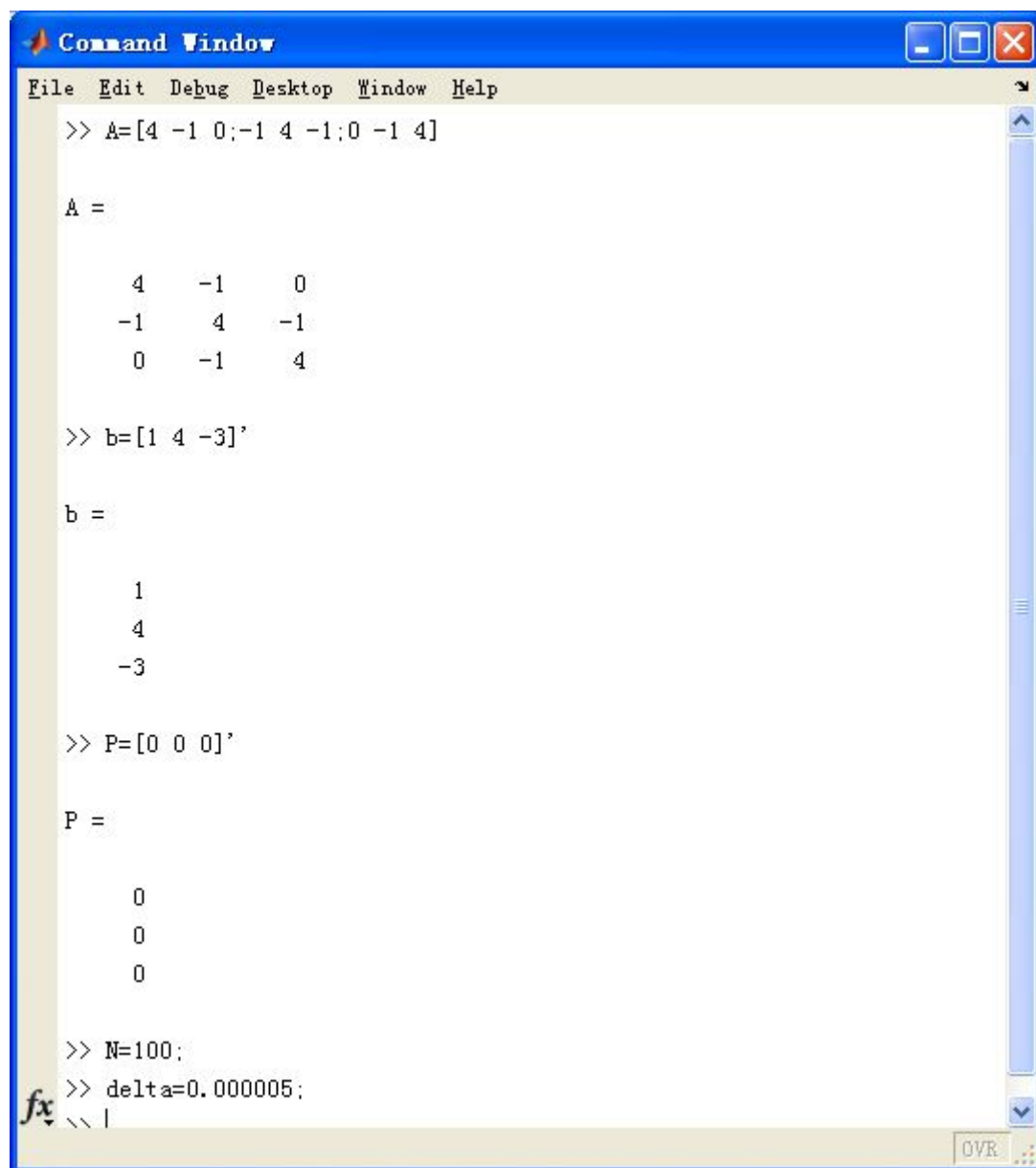
H_{ω} 称为 SOR 的迭代矩阵。

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, 初值 P, 最大迭代次数 N, 精确度 delta 如下:



A screenshot of the MATLAB Command Window. The window has a blue title bar with the text "Command Window" and standard window controls. Below the title bar is a menu bar with "File", "Edit", "Debug", "Desktop", "Window", and "Help". The main area contains the following MATLAB commands and their outputs:

```
>> A=[4 -1 0;-1 4 -1;0 -1 4]

A =

     4     -1     0
    -1     4     -1
     0     -1     4

>> b=[1 4 -3]

b =

     1
     4
    -3

>> P=[0 0 0]

P =

     0
     0
     0

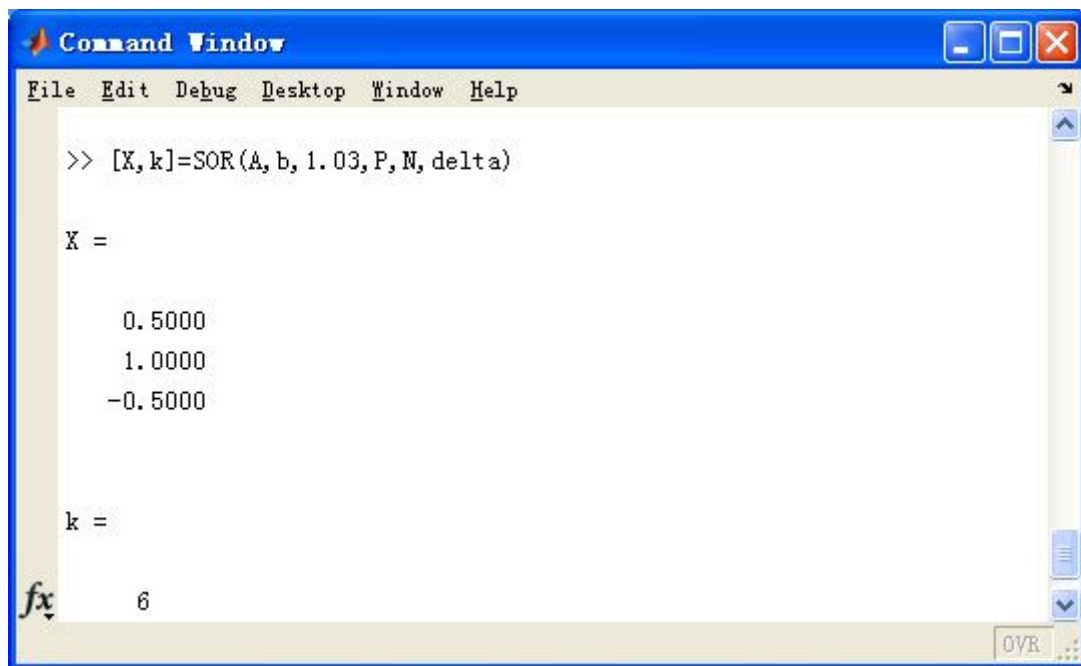
>> N=100;
>> delta=0.000005;
fx \ \ |
```

At the bottom right of the window, there is a small button labeled "OVR" and a scroll bar.

运行结果

①SOR 分量形式运行结果如下：

$$\omega = 1.03:$$



A screenshot of the MATLAB Command Window. The title bar is blue with the text "Command Window" and standard window controls. The menu bar includes "File", "Edit", "Debug", "Desktop", "Window", and "Help". The command prompt shows the execution of `>> [X,k]=SOR(A,b,1.03,P,N,delta)`. The output displays the solution vector `X =` as a column vector with values 0.5000, 1.0000, and -0.5000. Below this, the iteration count `k =` is shown as 6. At the bottom left, there is a small icon and the text "fx". At the bottom right, there is a button labeled "OVR".

```
>> [X,k]=SOR(A,b,1.03,P,N,delta)

X =

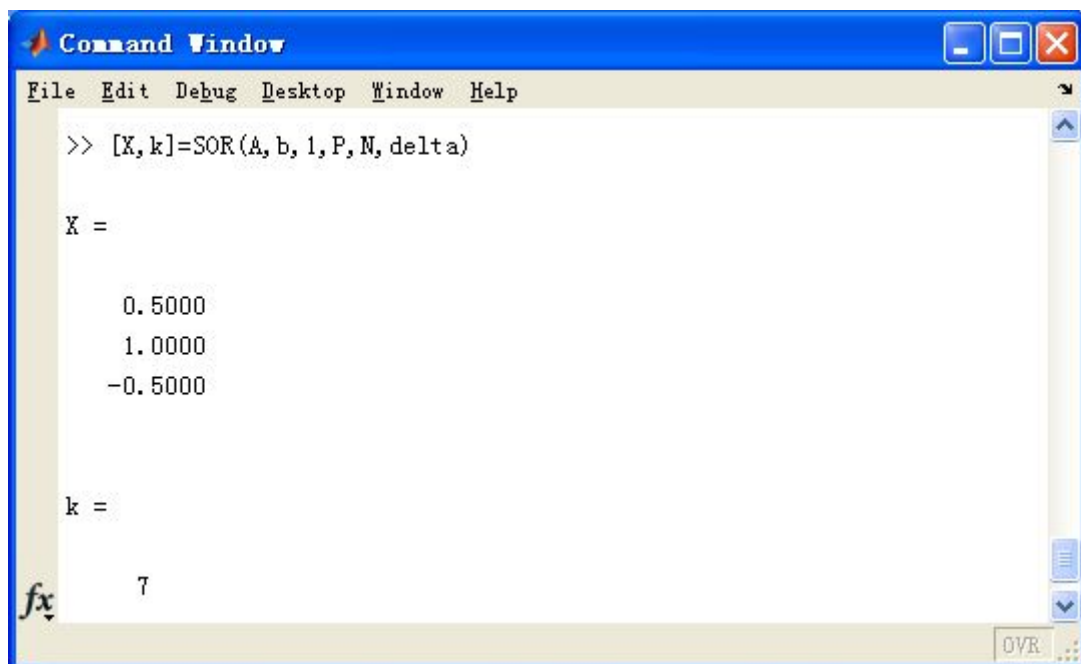
    0.5000
    1.0000
   -0.5000

k =

     6
```

即迭代次数 6 次，解为 $X = (0.5000, 1.0000, -0.5000)'$ ；

$\omega = 1$:



A screenshot of the MATLAB Command Window. The title bar is blue with the text "Command Window" and standard window controls. The menu bar includes "File", "Edit", "Debug", "Desktop", "Window", and "Help". The command prompt shows the execution of `>> [X,k]=SOR(A,b,1,P,N,delta)`. The output displays the solution vector `X =` as a column vector with values 0.5000, 1.0000, and -0.5000. Below this, the iteration count `k =` is shown as 7. At the bottom left, there is a small icon and the text "fx". At the bottom right, there is a button labeled "OVR".

```
>> [X,k]=SOR(A,b,1,P,N,delta)

X =

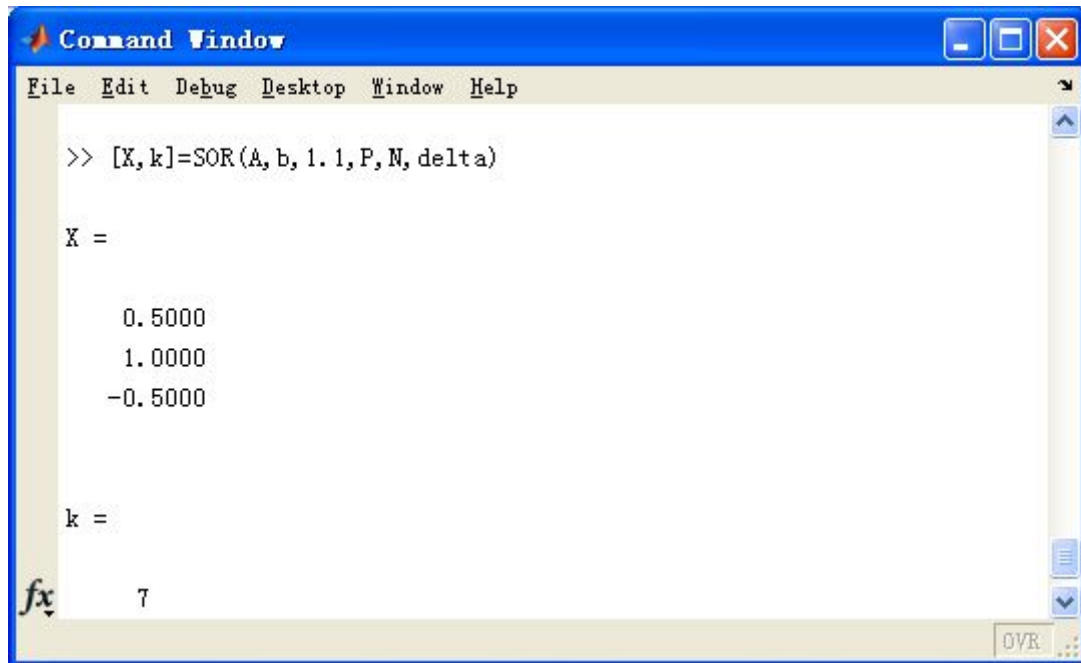
    0.5000
    1.0000
   -0.5000

k =

     7
```

即迭代次数 7 次，解为 $X = (0.5000, 1.0000, -0.5000)'$ ；

$\omega = 1.1$:



```
Command Window
File Edit Debug Desktop Window Help
>> [X,k]=SOR(A,b,1.1,P,N,delta)

X =

    0.5000
    1.0000
   -0.5000

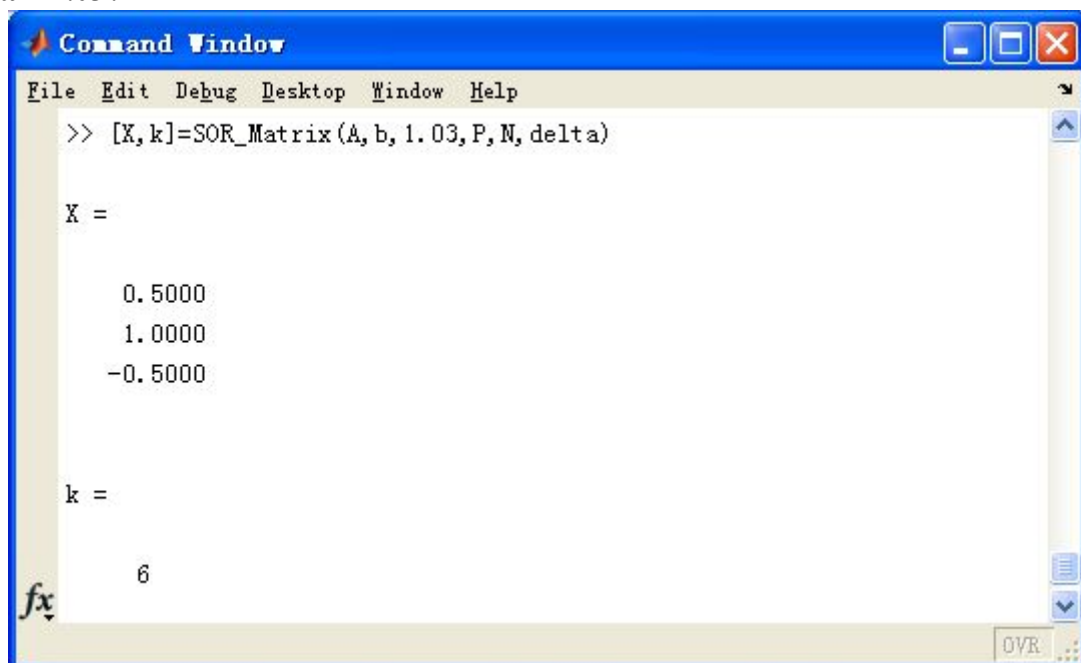
k =

     7
```

即迭代次数 7 次，解为 $X=(0.5000, 1.0000, -0.5000)'$ ；

②SOR 方法矩阵形式运行结果如下：

$\omega=1.03$ ：



```
Command Window
File Edit Debug Desktop Window Help
>> [X,k]=SOR_Matrix(A,b,1.03,P,N,delta)

X =

    0.5000
    1.0000
   -0.5000

k =

     6
```

即迭代次数 6 次，解为 $X=(0.5000, 1.0000, -0.5000)'$ ；

$\omega = 1$:



A screenshot of the MATLAB Command Window. The title bar is blue with the text 'Command Window'. Below the title bar is a menu bar with 'File', 'Edit', 'Debug', 'Desktop', 'Window', and 'Help'. The main area is white and contains the following text: '>> [X,k]=SOR_Matrix(A,b,1,P,N,delta)', 'X =', a 3x1 column vector with values 0.5000, 1.0000, and -0.5000, 'k =', and the number 7. At the bottom left is a 'fx' icon, and at the bottom right is an 'OVR' icon.

```
>> [X,k]=SOR_Matrix(A,b,1,P,N,delta)

X =

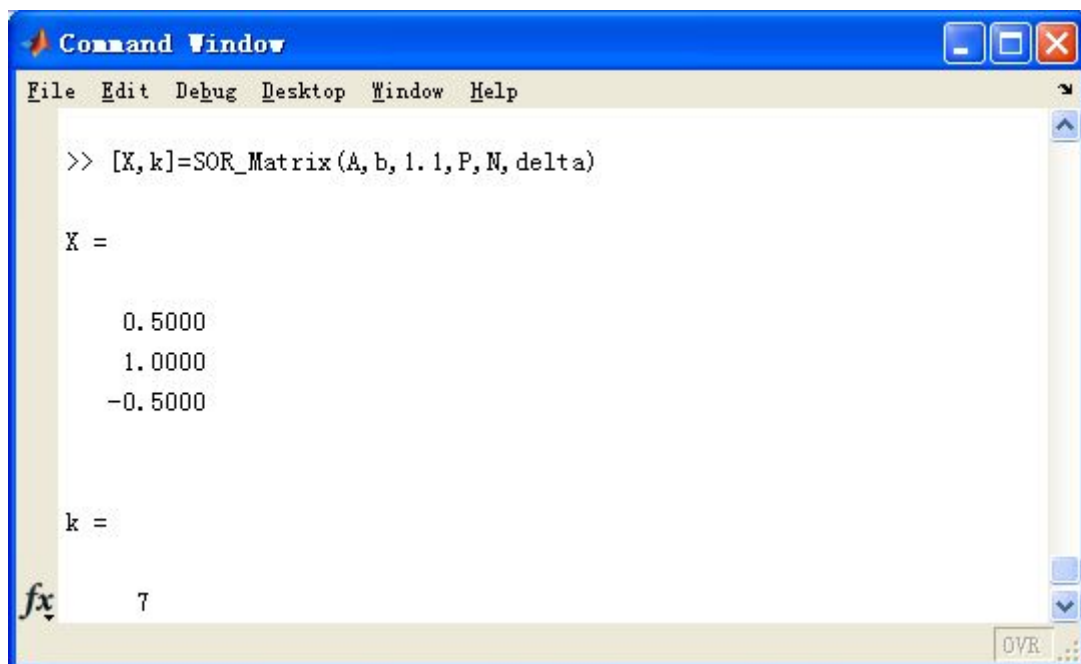
    0.5000
    1.0000
   -0.5000

k =

     7
```

即迭代次数 7 次，解为 $X = (0.5000, 1.0000, -0.5000)'$ ；

$\omega = 1.1$:



A screenshot of the MATLAB Command Window, similar to the one above. The title bar is blue with the text 'Command Window'. Below the title bar is a menu bar with 'File', 'Edit', 'Debug', 'Desktop', 'Window', and 'Help'. The main area is white and contains the following text: '>> [X,k]=SOR_Matrix(A,b,1.1,P,N,delta)', 'X =', a 3x1 column vector with values 0.5000, 1.0000, and -0.5000, 'k =', and the number 7. At the bottom left is a 'fx' icon, and at the bottom right is an 'OVR' icon.

```
>> [X,k]=SOR_Matrix(A,b,1.1,P,N,delta)

X =

    0.5000
    1.0000
   -0.5000

k =

     7
```

即迭代次数 7 次，解为 $X = (0.5000, 1.0000, -0.5000)'$ ；

2 关键代码及其解释

①SOR 法分量形式

```

while (k<=N)
    for i=1:n

        %计算新的迭代值

        X(i)= X(i)+w*( b(i)-A(i,1:i-1)* X(1:i-1)- A(i,i:n) *
P(i: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

```

②SOR 法矩阵形式:

```

%迭代矩阵
Hw=inv (D+w*L) * ( (1-w) *D-w*U) ;

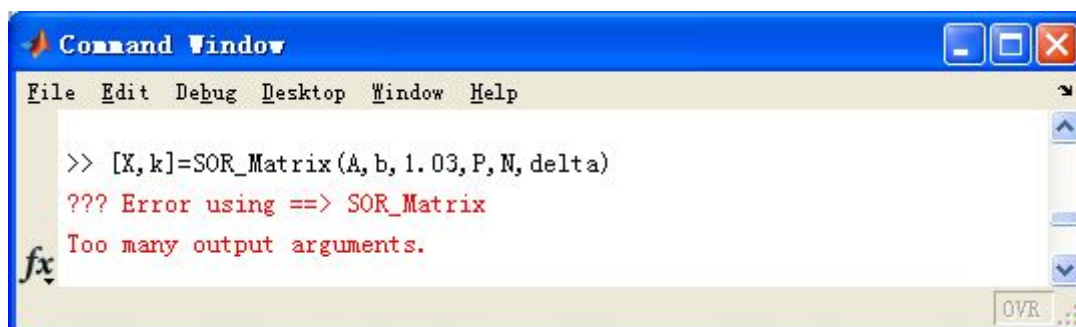
%k 记录迭代次数
k=1;
while (k<=N)
    %计算新的迭代值
    X=Hw*P+w*inv (D+w*L) *b;

    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

```

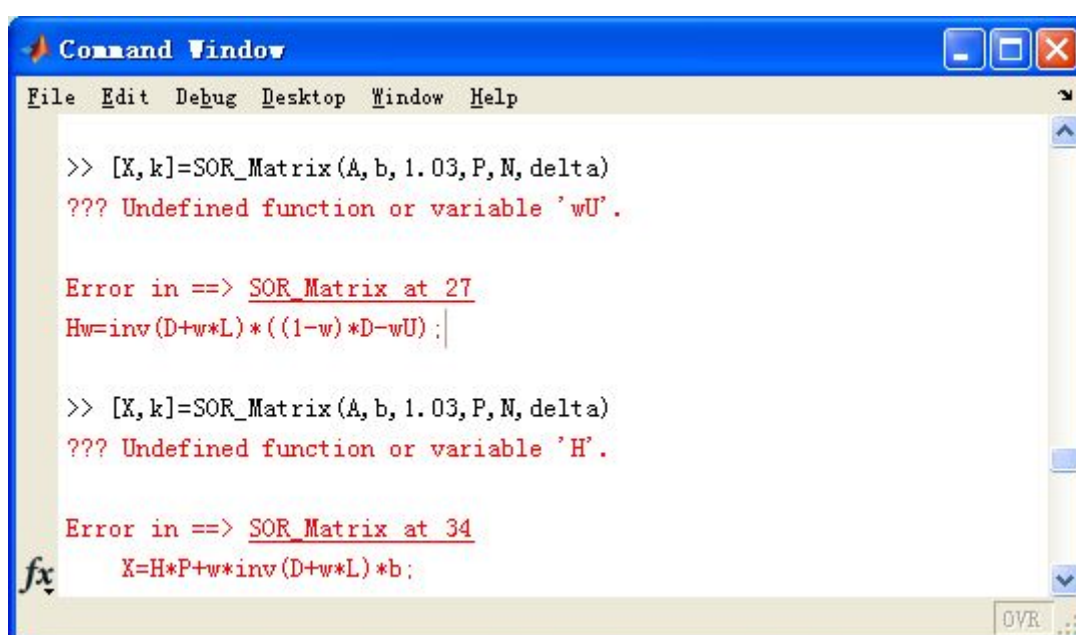
3 调试过程

①未调整好函数的输出个数，导致引用函数出错



```
Command Window
File Edit Debug Desktop Window Help
>> [X,k]=SOR_Matrix(A,b,1.03,P,N,delta)
??? Error using ==> SOR_Matrix
Too many output arguments.
```

②函数变量名前后不一致



```
Command Window
File Edit Debug Desktop Window Help
>> [X,k]=SOR_Matrix(A,b,1.03,P,N,delta)
??? Undefined function or variable 'wU'.

Error in ==> SOR_Matrix at 27
Hw=inv(D+w*L)*((1-w)*D-wU);

>> [X,k]=SOR_Matrix(A,b,1.03,P,N,delta)
??? Undefined function or variable 'H'.

Error in ==> SOR_Matrix at 34
X=H*P+w*inv(D+w*L)*b;
```

六、实验总结

1. 遇到的问题及解决过程

- ①使用分量形式方法， $x_1^{(K+1)}, x_2^{(K+1)}, \dots, x_{i-1}^{(K+1)}, x_i^{(K)}, \dots, x_n^{(K)}$ 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；

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；
- ②MATLAB 的函数调用中，调用的格式必须与函数定义的格式相同，包括输出参数、输入参数的类型和个数。

③MATLAB 可以直接进行矩阵运算,这使得许多 \sum 运算,在 C 语言中需要二重循环才能表达,而在 MATLAB 中可以用一个语句完成

3. 体会和收获。

①Gauss-Seidel 是一种优秀的解线性方程组的迭代法,但是人们想在迭代过程中通过自己的控制来加速收敛,松弛法便满足了这样的要求;

②对于不同的 ω , 松弛法的收敛速度是不一样的,以 Gauss-Seide 迭代法为例当 $\omega > 1$, 称为逐次超松弛迭代法; 当 $\omega < 1$, 称为低松弛迭代法; 当 $\omega = 1$, 即为 Gauss-Seidel 迭代法;

③ 对于松弛法,我们希望找到 ω , 使得 $\rho(H_\omega)$ 最小, 这样的 ω 成为“最佳松弛因子”, 然而目前只有少数特殊类型的矩阵, 才有确定的最佳松弛因子的理论公式。对于一般的矩阵, 寻找最佳松弛因子还是一个正在探讨中的问题。

④在程序实现上, SOR 法与之前的 Gauss-Seidel 迭代法代码格式基本相同, 只需改动关键处加上 ω 即可;

七、程序源代码：

①SOR 法分量形式

SOR.m

```
function [X,k]=SOR(A,b,w,P,N,delta)
%Input  -A n*n 非奇异矩阵
%        -b n*1 矩阵
%        -w 松弛因子
%        -P n*1 矩阵，初始值
%        -N 最大迭代次数
%        -delta 精确度
%Output -X n*1 使用 SOR 方法得到的方程组 AX=b 的解
%        -k 迭代次数

if (w<=0 || w>=2)
    error('松弛因子 w 不合法.')
end

%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

        %计算新的迭代值

        X(i)= X(i)+w*( b(i)-A(i,1:i-1)* X(1:i-1)- A(i,i:n) *
P(i:n) )/A(i,i);
    end

    %检验精度，若精度符合要求，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
```

②SOR 法矩阵形式:

SOR_Matrix.m

```
function [X,k]=SOR_Matrix(A,b,w,P,N,delta)

%Input  -A n*n 非奇异矩阵
%        -b n*1 矩阵
%        -w 松弛因子
%        -P n*1 矩阵, 初始值
%        -N 最大迭代次数
%        -delta 精确度
%Output -X n*1 使用 SOR 矩阵形式方法得到的方程组 AX=b 的解
%        -k 迭代次数
if (w<=0 || w>=2)
    error('松弛因子 w 不合法. ')
end

%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));

%迭代矩阵
Hw=inv(D+w*L)*( (1-w)*D-w*U);

%k 记录迭代次数
k=1;
while (k<=N)
    %计算新的迭代值
    X=Hw*X+w*inv(D+w*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
```

八、教师评语