武汉大学数学与统计学院

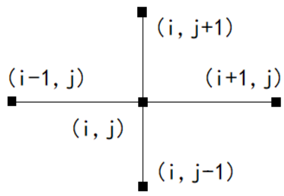
数值分析实验报告

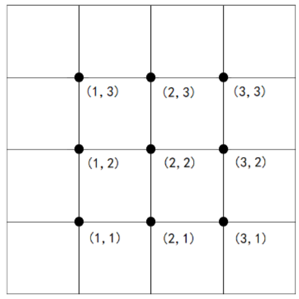
实验名称	《数值分析（1）》期末实验				实验时间	2018 年　12 月　31 日	
姓名	江金阳	班级	17 信计班	学号	2017301000090	成绩	

一、实验内容

求解 Dirichlet 问题 $\begin{cases} -\varepsilon\Delta u + a\dfrac{\partial u}{\partial x} = f(x,y) \\ u|_{\partial\Omega} = 0 \end{cases}$ $(x,y) \in \Omega = [0,1]^2$

其中 $f(x,y) = \pi(2\pi\varepsilon\sin\pi x + a\cos\pi x)\sin\pi y$





区域离散，x、y 方向 N 等分，$h = \frac{1}{N}$;

方程离散：$-(\Delta u)_{ij} \approx \frac{-u_{i,j-1}-u_{i-1,j}+4u_{i,j}-u_{i+1,j}-u_{i,j+1}}{h^2}$

$\left(\frac{\partial u}{\partial x}\right)_{i,j} \approx \frac{u_{i+1,j}-u_{i,j}}{h}$　$(a<0)$

在 (i,j) 偏微方程近似为：

$$-\varepsilon u_{i,j-1} - \varepsilon u_{i-1,j} + (4\varepsilon - ah)u_{i,j} - (\varepsilon - ah)u_{i+1,j} - \varepsilon u_{i,j+1} = h^2 f_{i,j} \quad (1 \le i,\ j \le N-1)$$

简单起见，考察 $N = 4$，

$$\begin{bmatrix}
4\varepsilon-ah & -\varepsilon+ah & & -\varepsilon & & & & & \\
-\varepsilon & 4\varepsilon-ah & -\varepsilon+ah & & -\varepsilon & & & & \\
 & -\varepsilon & 4\varepsilon-ah & & & -\varepsilon & & & \\
-\varepsilon & & & 4\varepsilon-ah & -\varepsilon+ah & & -\varepsilon & & \\
 & -\varepsilon & & -\varepsilon & 4\varepsilon-ah & -\varepsilon+ah & & -\varepsilon & \\
 & & -\varepsilon & & -\varepsilon & 4\varepsilon-ah & & & -\varepsilon \\
 & & & -\varepsilon & & & 4\varepsilon-ah & -\varepsilon+ah & \\
 & & & & -\varepsilon & & -\varepsilon & 4\varepsilon-ah & -\varepsilon+ah \\
 & & & & & -\varepsilon & & -\varepsilon & 4\varepsilon-ah
\end{bmatrix}
\begin{bmatrix} u_{11} \\ u_{21} \\ u_{31} \\ u_{12} \\ u_{22} \\ u_{32} \\ u_{13} \\ u_{23} \\ u_{33} \end{bmatrix}
= h^2 \begin{bmatrix} f_{11} \\ f_{21} \\ f_{31} \\ f_{12} \\ f_{22} \\ f_{32} \\ f_{13} \\ f_{23} \\ f_{33} \end{bmatrix}$$

记为 $Au = f$，一般地 $A = \begin{bmatrix} S & -\varepsilon I & & & & \\ -\varepsilon I & S & -\varepsilon I & & & \\ & -\varepsilon I & S & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & S & -\varepsilon I \\ & & & & -\varepsilon I & S \end{bmatrix}$,

由 $n \times n$ 块矩阵构成，每个块（子）矩阵为 $n \times n$，$n = N - 1$

取 $\varepsilon = 1, a = -1$ 分别用求解线性方程组的数值方法（Jacobi 迭代法、G-S 迭代法、SOR 迭代法、共轭梯度法）求解 Dirichlet 问题方程的近似解（收敛准则为残差小于 1e-6），并比较各类方法的迭代步数，绘制各方法的残差曲线，图示数值解 $u(x,y)$。进一步，改变参数 $\varepsilon = 0.1, 0.01$ 和 $a = -10, -100$ 进行求解，考察参数对迭代法的影响。

二、相关背景知识介绍

1. Jacobi 迭代法

基本过程：对非奇异且主对角元素不全为零的矩阵 A，令$\mathrm{A}=\mathrm{D}-\mathrm{L}-\mathrm{U}$，迭代矩阵为

$$\mathrm{B}=\mathrm{D}^{-1}(\mathrm{L}+\mathrm{U}) \quad \mathrm{g}=\mathrm{D}^{-1}\mathrm{b}$$

再给定初始向量x_0，作格式为$\mathrm{x}_k=\mathrm{B}\mathrm{x}_{k+1}+\mathrm{g}$的迭代。

其分量形式为 ： $x_i^{(k+1)}=\frac{1}{a_{ii}}(-\sum_{j=1 j\neq 1}^{n}a_{ij}x_j^{(k)}+b_i)$，依据此分量形式可不显式形成迭代矩阵，节省迭代所需的内存和时间。

2. Gauss-Seidel 迭代法

基本过程：对非奇异且主对角元素不全为零的矩阵 A，令$\mathrm{A}=\mathrm{D}-\mathrm{L}-\mathrm{U}$，迭代矩阵为

$$\mathrm{B}=(\mathrm{D}-\mathrm{L})^{-1}\mathrm{U} \quad \mathrm{g}=(\mathrm{D}-\mathrm{L})^{-1}\mathrm{b}$$

再给定初始向量x_0，作格式为$\mathrm{x}_k=\mathrm{B}\mathrm{x}_{k+1}+\mathrm{g}$的迭代。

其分量形式为 ： $x_i^{(k+1)}=\frac{1}{a_{ii}}(\sum_{j=1}^{i-1}a_{ij}x_j^{(k+1)}-\sum_{j=i+1}^{n}a_{ij}\,x_j^{(k)}+b_i)$

3. 超松弛迭代法

基本过程：设已知第 k 次迭代向量$x^{(k)}$，及第 k+1 次迭代向量的前 i-1 个分量$x_j^{(k+1)}$（$\mathrm{j}=1,2,\ldots \mathrm{i}-1$），现在研究如何求向量$x^{(k+1)}$的第 i 个分量 $x_i^{(k+1)}$。

首先，有高斯—赛德尔迭代法求出一个值，记为

$$\tilde{x}_i^{(k+1)}=\frac{1}{a_{ii}}(b_i-\sum_{j=1}^{i-1}a_{ij}\,x_j^{(k+1)}-\sum_{j=i+1}^{n}a_{ij}x_j^{(k)}) \qquad (\mathrm{i}=1,2,\ldots \mathrm{n})$$

再将第 k 次迭代向量的第 i 个分量 $x_j^{(k)}$ 与 $\tilde{x}_i^{(k+1)}$ 进行加权平均，得 $x_i^{(k+1)}$，即：

$$x_i^{(k+1)}=(1-\omega)x_i^{(k)}+\omega\tilde{x}_i^{(k+1)}=x_i^{(k)}+\omega(\tilde{x}_i^{(k+1)}-x_i^{(k)})$$

于是得到 SOR 迭代公式

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

当ω=1 时，上式即为 G-S 法；当 0<ω<1 时，上式称为低松弛方法，当某些方程组用 G-S 法不收敛时，可以用低松弛方法获得收敛；当ω>1 时，上式称为超松弛方法，可以用来提高收敛速度。
将上式变换后写成矩阵的形式，得：$(D-\omega L)x^{(k+1)}=[(1-\omega)D+\omega U]x^{(k)}+\omega b$
于是得 SOR 迭代的矩阵表示为：$x^{(k+1)}=B_\omega X^{(k)}+f_\omega$
其中：$B_\omega=(D-\omega L)^{-1}[(1-\omega)D+\omega U]$ ， $f_\omega=\omega(D-\omega L)^{-1}b$

4. 共轭梯度法

基本过程：对于对称正定矩阵 A，给定初始x_0，计算$\mathrm{r}_0=\mathrm{b}-\mathrm{A}\mathrm{x}_0=\mathrm{p}_0$，

对$\mathrm{k}=0,1,2\ldots$依次计算： $\alpha_k=\frac{r_k^T r_k}{p_k^T A p_k}$ $\qquad \mathrm{x}_{k+1}=x_k+\alpha_k p_k$

$$r_{k+1}=r_k-\alpha_k A p_k$$

$$\beta_k = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k} \qquad p_{k+1} = r_{k+1} + \beta_k p_k$$

在确定下山方向p_k时，第一步仍选用最速下降法负梯度方向为下山方向，第二步以后下山方向不再r_k，而是在过点由向量r_k和p_{k-1}所张成的二维平面

$$\pi_2 = \{x|x = x_k + \xi r_k + \eta p_{k-1}, \xi, \eta \in R\}$$

内找出使函数φ下降最快的方向作为新的下山方向p_k。

三、代码

1. 主函数

```
maxstep = 2e4;
e = 1;
a = -1;
N = 50;
h = 1/N;
n = N-1;
tol = 1e-6;

d = 4*e - a*h;
up = -e + a*h;
tmp = ones(n-1,1);
E = -e * eye(n);
S = d * eye(n) + up * diag(tmp,1) - e * diag(tmp,-1);
I = eye(n);
tmp2 = diag(tmp,1) + diag(tmp,-1);
A = kron(I,S)+kron(tmp2,E);

F = ones(n);
for i = 1:n-1;
    for j = 1:n;
        x = i/N;
        y = j/N;
        F(i,j)=pi * (2*pi*e*sin(pi*x) + a*cos(pi*x)) * sin(pi*y);
    end
end
F = reshape(F,n^2,1);
F = h^2*F;

[u0,n0,resjacobi] = Jacobi(A,F,tol,maxstep);
[u1,n1,resGS] = GS(A,F,tol,maxstep);
[u20,n20,resSOR0] = SOR0(A,F,tol,maxstep);
[u21,n21,resSOR1] = SOR1(A,F,tol,maxstep);
[u3,n3,resCG] = CG(A'*A,A'*F,tol,maxstep);

%plot:res
count = 1: length(A)/3;
rjacobi = resjacobi(count);
```

```
rGS = resGS(count);
rSOR0 = resSOR0(count);
rSOR1 = resSOR1(count);
rCG = resCG(count);
figure,plot(count,rjacobi ,count,rGS, count,rSOR0, count,rSOR1, count,rCG)
legend('rjacobi','rGS','rSOR0','rSOR1','rCG');
grid on

%plot:solution of equation
u0=reshape(u0,n,n);
t = ( 1 : n )' * h;
figure, surf ( t, t, u0);
```

2. Jacobi 迭代法

```
function[U,n0,resjacobi]=Jacobi(A,F,tol,maxstep)
tic;
n0 = 1;
U = zeros(length(A),1);
resjacobi = zeros( length(A),1 );
resjacobi(1) = norm(F-A*U,inf);
d = diag(A);

while (resjacobi(n0) > tol)
    n0 = n0 + 1;
    res = F - A * U;
    U = U + res ./ d;
    resjacobi(n0) = norm(F-A*U,inf);
    if n0 > maxstep
        break
    end
end
display(n0);
toc;
end
```

3. Gauss-Seidel 迭代法

```
function[U,n1,resGS]=GS(A,F,tol,maxstep)
tic;
D = diag(diag(A));
Low = - tril(A,-1);
Up = - triu(A,1);
B = ( D-Low ) \ Up;
g = ( D-Low ) \ F;
U0 = zeros(length(A),1);
U = B*U0 + g;
n1 = 1;
resGS=zeros(length(A),1);
```

```
resGS(1)=norm(F-A*U,inf);
while (norm(U-U0,inf)>tol)
    U0 = U;
    U = B*U0+g;
    n1 = n1+1;
    resGS(n1) = norm(F-A*U,inf);
    if n1>maxstep
        break
    end
end
display(n1);
toc;
end
```

4. SOR 迭代法（预估 ω）

```
function[U,n20,resSOR]=SOR0(A,F,tol,maxstep)
tic;
w = 1.5;
n20 = 1;
U = zeros(length(A),1);
resSOR = zeros(length(A),1);
resSOR(1)= norm(F-A*U,inf);

while (resSOR(n20) > tol)
    n20 = n20 + 1;
    for i = 1:length(A);
        U(i) = U(i)+(F(i)-A(i,:)*U)*w/A(i,i);
    end
    resSOR(n20) = norm(F-A*U,inf);
     if n20>maxstep
        break
    end
end
display(n20);
toc;
end
```

5. SOR 迭代法（最佳松弛因子）

```
function[U,n21,resSOR]=SOR1(A,F,tol,maxstep)
tic;
D = diag( diag( A ) );
Low = - tril( A ,-1 );
Up = - triu( A ,1 );
B=D\(Low+Up);
row=max(abs(eig(B)));
w=2/(1+sqrt(1-row^2));
clear D Low Up B row;
```

```
n21 = 1;
U = zeros(length(A),1);
resSOR = zeros(length(A),1);
resSOR(1)= norm(F-A*U,inf);

while (resSOR(n21) > tol)
    n21 = n21 + 1;
    for i = 1:length(A);
        U(i) = U(i)+(F(i)-A(i,:)*U)*w/A(i,i);
    end
    resSOR(n21)=norm(F-A*U,inf);
     if n21>maxstep
        break
    end
end
display(n21);
toc;
end
```

6. 共轭梯度法

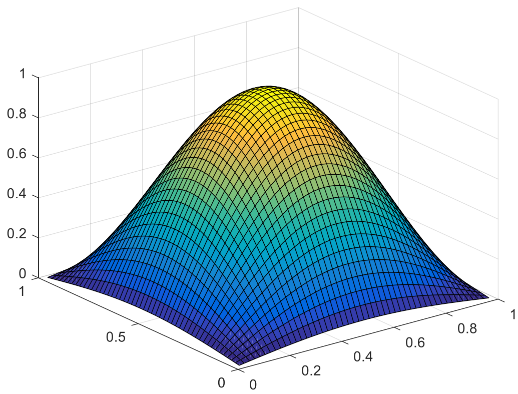
```
function[U,n3,resCG]=CG(A,F,tol,maxstep)
tic;
U = zeros(length(A),1);
res0 = F-A*U;
n3 = 1;
p = res0;
resCG = zeros(length(A),1);
resCG(1) = norm(F-A*U,inf);

while (resCG(n3)>tol)
    n3=n3+1;
    Ap = A*p;
    alpha = res0'*res0/(p'*Ap);
    U = U + alpha*p;
    res1 = res0 - alpha*Ap;
    beta = res1'*res1/(res0'*res0);
    p = res1 + beta*p;
    res0 = res1;
    resCG(n3) = norm(F-A*U,inf);
    if n3>maxstep
        break
    end
end
display(n3);
toc;
end
```

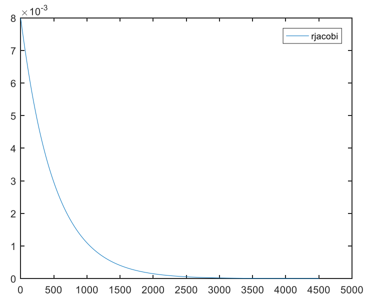
四、数值结果

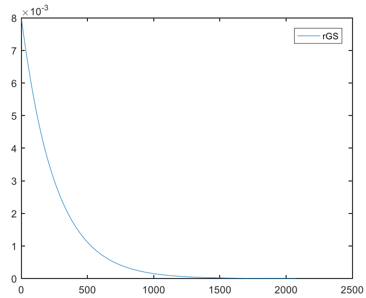
1. 取$\varepsilon=1, a=-1$时

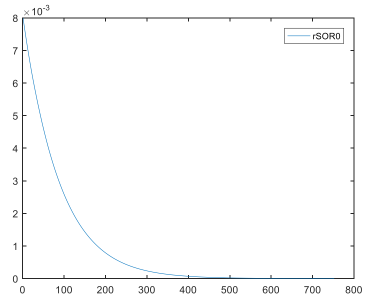
数值方法	执行时间/s	迭代步数
Jacobi	36.571566	4503
Gauss-Seidel	17.494056	2080
SOR（预估ω）	36.563143	752
SOR（计算最佳ω）	12.425167	113
共轭梯度法	6.469778	827

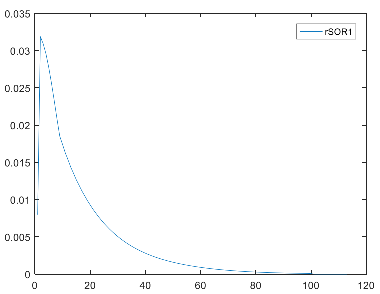


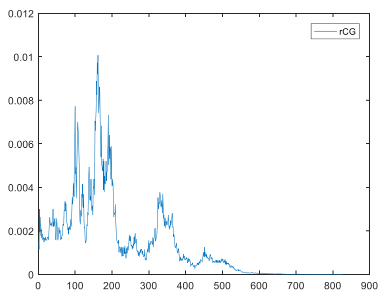
(由于各方法都收敛且计算出同一个 u，故只选取了其中一张图示解)

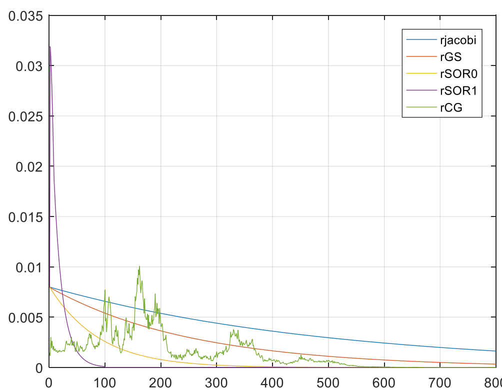






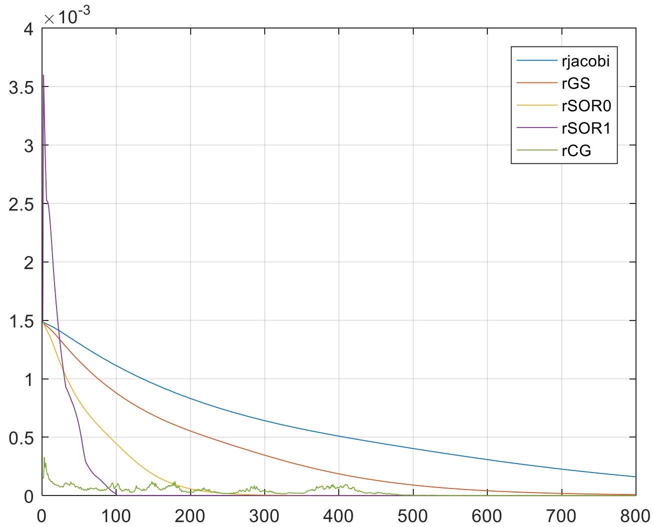


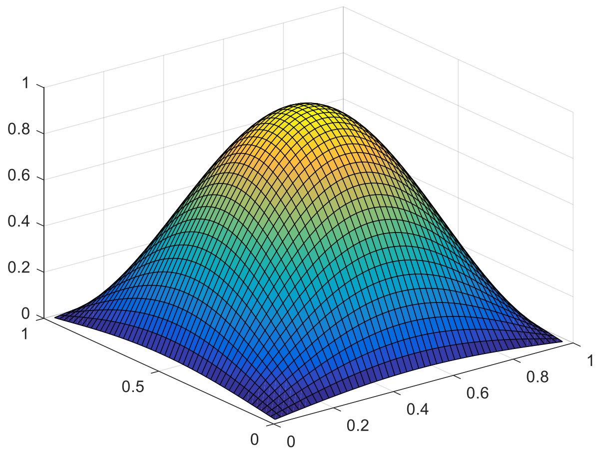




2. 取$\varepsilon=0.1, a=-1$时

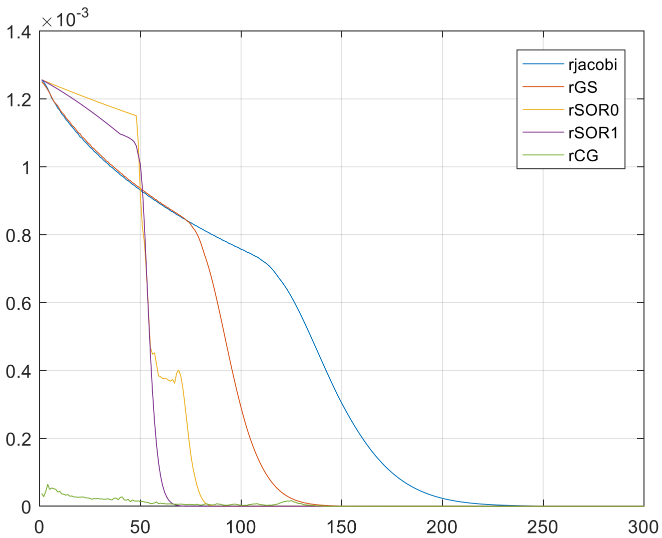
数值方法	执行时间/s	迭代步数
Jacobi	15.217335	2074
Gauss-Seidel	9.351581	1241
SOR（预估ω）	17.612458	364
SOR（计算最佳ω）	11.148416	103
共轭梯度法	4.470588	563

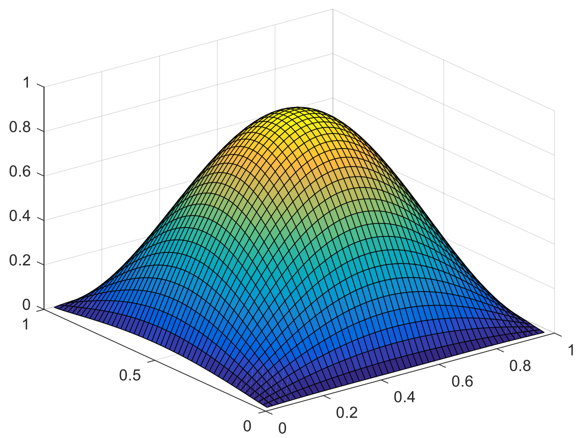




3. 取$\varepsilon = 0.01, a = -1$时

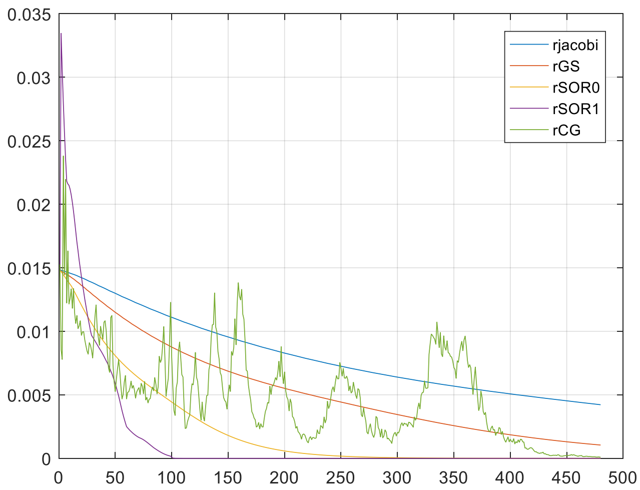
数值方法	执行时间/s	迭代步数
Jacobi	2.146680	246
Gauss-Seidel	2.005071	170
SOR（预估ω）	4.550937	86
SOR（计算最佳ω）	9.900468	71
计算最佳ω	6.666028	
共轭梯度法	1.036709	141

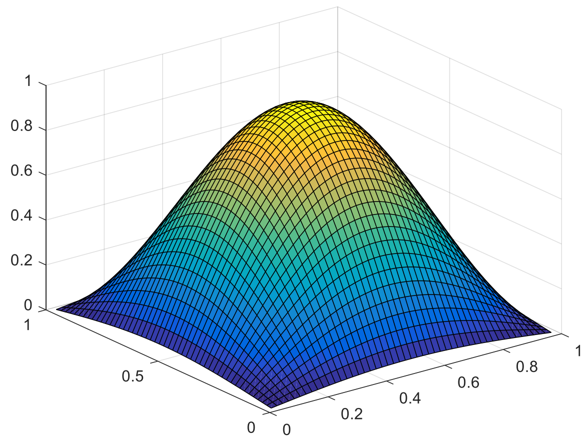




4. 取$\varepsilon = 1, a = -10$时

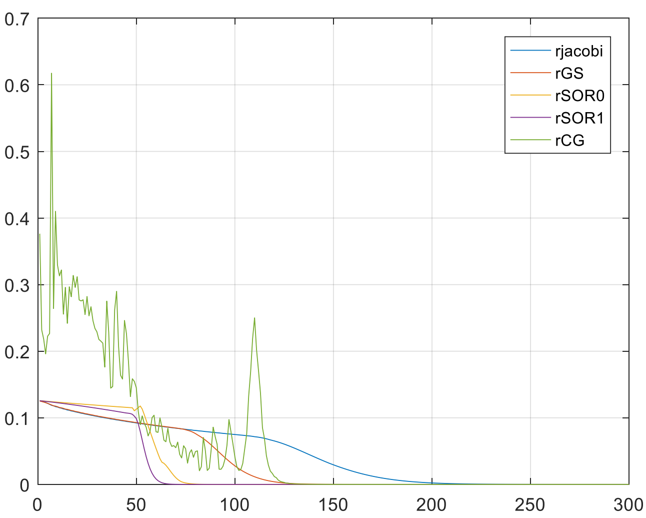
数值方法	执行时间/s	迭代步数
Jacobi	19.769182	2628
Gauss-Seidel	9.607717	1240
SOR（预估ω）	23.429071	455
SOR（计算最佳ω）	11.647128	109
共轭梯度法	5.257491	685

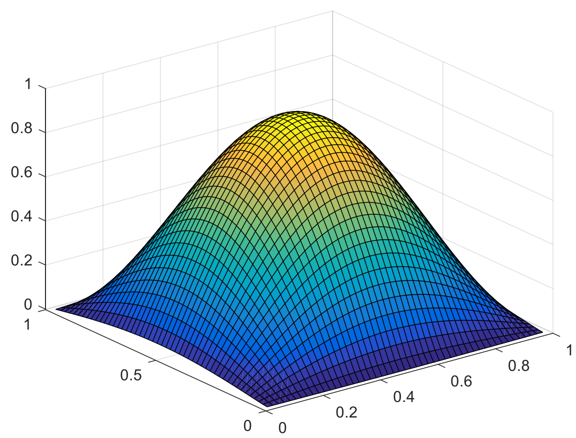




5. 取$\varepsilon = 1, a = -100$时

数值方法	执行时间/s	迭代步数
Jacobi	2.338611	303
Gauss-Seidel	1.573012	169
SOR（预估ω）	4.627654	93
SOR（计算最佳ω）	11.250084	79
计算最佳ω	6.677588	
共轭梯度法	2.332058	306





五、结果分析

1. 对参数ε和a的分析

从上述数值结果对比可知，对此 Dirichlet 问题中的参数ε和a，解线性方程组$Au = f$的难度随着ε的减小而降低，随着a的减小而降低。

2. 对四种数值方法的分析

对条件一般的ε和a，迭代的速度大致符合理论值，Jacobi 迭代法的收敛速度最慢，Gauss-Seidel 迭代法比 Jacobi 迭代法的收敛速度快大约一倍，SOR 迭代法在预估松弛因子ω时可能效果不够理想，而在计算最佳松弛因子ω时，收敛速度明显快于 Jacobi 迭代法和 Gauss-Seidel 迭代法，在迭代步数上甚至比共轭梯度法更具优势，但由于共轭梯度法每步的运算量更小，四种方法的执行时长排序仍为 CG＜SOR＜Gauss-Seidel＜Jacobi。

对使得解线性方程组$Au = f$的难度比较低的ε和a，四种方法中 SOR 由于预估ω的不理想或计算ω浪费的时间过长成为最不适用的方法，Jacobi 和 Gauss-Seidel 迭代法的收敛速度仍然保持着一倍的差距，但两者相对于 CG 法在执行时间上的差距均有缩短，甚至在极为容易的条件下会优于 CG 法。

3. 对 SOR 迭代法的补充分析

在实验中对 SOR 迭代法的松弛因子ω作了预估和计算最佳值两种处理，各组结果对比可知在方程条件一般时，计算最佳值可以有效缩短执行步数和时间；但在方程条件相当好时，预估ω反而会明显快于计算最佳ω，通过数值结果可知两者在进行 SOR 迭代时速度相近，但后者计算ω时浪费了较长的时间。再对比其他迭代法，得出 SOR 迭代法只应在方程条件一般时使用，并采用计算最佳松弛因子的方案。

六、计算过程碰到的问题，解决办法及收获或体会

1. 碰到的问题及解决办法

1) CG 法解线性方程组$Au = f$时刚开始没有注意到 A 对称正定的要求，导致 CG 法运行失败，后改用 CGLS 法对方程进行处理：$A^TAu = A^Tf$得以解决，其中 A 被转化为对称正定的A^TA。

2) 在绘制残差曲线时，四种方法储存每步残差的向量长度可能会不同，当读取的值的数量（count）超过向量长度时会报错，需要在一次试验后根据迭代步数对 count 的上限进行调整，重新绘图。

2. 体会

1） 设计算法时，应该尽量避免显式地形成迭代矩阵（Jacobi、SOR），采用分量计算可以有效节省迭代所需的内存和时间，否则就违背了弃用直接法改用迭代法降低内存需求的初衷。

2） 对于此 Dirichlet 问题，我给出的建议是在参数条件相当好时使用 G-S 迭代法，在参数条件不够好时使用共轭梯度法或计算最佳松弛银子的 SOR 迭代法解线性方程组。

3） 通过本次期末大作业，我对解线性方程组的四种数值方法（Jacobi 迭代法、G-S 迭代法、SOR 迭代法、共轭梯度法）的特点有了更清楚的认识，了解了理论中细微的点是如何在实验中表现出来的，同时也积累了应用 MATLAB 解决实际问题的宝贵经验。

教师评语	指导教师： 年 月 日