数值试验一
矩形区域上柏松问题数值解

董建宇　2019511017

3 月 29 日

1 本次数值试验目的

在计算中体验三种迭代过程：Jacobi 迭代法，Gauss-Seidel 迭代法，逐次超松弛迭代法的不同，感受使用不同迭代法为达到相同精度时所需要的迭代次数，以及当 h 取不同值时 Jacobi 迭代误差限与迭代次数关于 h 变化的关系。

2 问题的提出

考虑在矩形区域的柏松问题:

$$\begin{cases} -\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)=f(x,y), 0<x,y<1, \\ u(0,y)=u(1,y)=u(x,0)=u(x,1)=0 \end{cases}$$

取 $f(x,y)=2\pi^2\sin(\pi x)\sin(\pi y)$，则原方程精确解为

$$u^*(x,y)=\sin(\pi x)\sin(\pi y)$$

令 $h=\frac{1}{N}$，N 为正整数，$x_i=ih, y_j=jh, u_{i,j}\approx u(x_i,y_j), f_{ij}=f(x_i,y_j)$.

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{x=x_i,y=y_i}=\frac{u_{i+1,j}-2u_{i,j}+u_{i-1,j}}{h^2},\quad \left.\frac{\partial^2 u}{\partial y^2}\right|_{x=x_i,y=y_i}=\frac{u_{i,j+1}-2u_{i,j}+u_{i,j-1}}{h^2}$$

则差分格式为

$$\begin{cases} -u_{i-1,j}-u_{i,j-1}+4u_{i,j}-u_{i+1,j}-u_{i,j+1}=h^2 f_{i,j} \\ u_{i,0}=u_{0,j}=ui,N=uN,j=0,i,j=1,2,\ldots,N-1. \end{cases}$$

记

$$u_j^h=\begin{pmatrix} u_{1,j} \\ u_{2,j} \\ . \\ . \\ . \\ u_{N-1,j} \end{pmatrix},\ f_j^h=\begin{pmatrix} f_{1,j} \\ f_{2,j} \\ . \\ . \\ . \\ f_{N-1,j} \end{pmatrix},\ u^h=\begin{pmatrix} u_1^h \\ u_2^h \\ . \\ . \\ . \\ u_{N-1}^h \end{pmatrix},\ f^h=\begin{pmatrix} f_1^h \\ f_2^h \\ . \\ . \\ . \\ f_{N-1}^h \end{pmatrix}$$

进而可以写为线性代数方程组:

$$L_h u^h=h^2 f^h$$

其中

$$C=\begin{pmatrix} 0 & 1 & 0 & 0 & ... & 0 & 0 \\ 1 & 0 & 1 & 0 & ... & 0 & 0 \\ 0 & 1 & 0 & 1 & ... & 0 & 0 \\ 0 & 0 & 1 & 0 & ... & 0 & 0 \\ . & . & . & . & & . & . \\ 0 & 0 & 0 & 0 & ... & 0 & 1 \\ 0 & 0 & 0 & 0 & ... & 1 & 0 \end{pmatrix}_{(N-1)\times(N-1)} \qquad L_h=\begin{pmatrix} 4I-C & -I & & & \\ -I & 4I-C & -I & & \\ & .. & .. & .. & \\ & & -I & 4I-C & -I \\ & & & -I & 4I-C \end{pmatrix}$$

I 为 N-1 阶单位矩阵。

3 数值试验结果

3.1

3.1.1 Jacobi 迭代法

当 h=0.1 时，取 $\epsilon=10^{-8}$，利用 Jacobi 迭代法求出近似解所需迭代次数为 389 次，近似解与精确解误差限为 0.00826523.

3.1.2 Gauss-Seidel 迭代法

当 h=0.1 时，取 $\epsilon = 10^{-8}$，利用 Gauss-Seidel 迭代法求出近似解所需迭代次数为 243 次，近似解与精确解误差限为 0.00826532.

3.1.3 SOR 迭代法

当 h=0.1 时，取 $\epsilon = 10^{-8}$，利用 Gauss-Seidel 迭代法，当 w=1.2 时迭代次数为 108, 近似解与精确解误差限为 0.00826535；当 w=1.3 时迭代次数为 86, 近似解与精确解误差限为 0.00826537；当 w=1.9 时迭代次数为 171, 近似解与精确解误差限为 0.00826542；当 w=0.9 时迭代次数为 197, 近似解与精确解误差限为 0.00826530。

3.2

3.2.1 h=0.1

当 h=0.1 时，取 $\epsilon = 10^{-8}$，利用 Jacobi 迭代法求出近似解所需迭代次数为 309 次，近似解与精确解误差限为 0.00826523.

3.2.2 h=0.05

当 h=0.05 时，取 $\epsilon = 10^{-8}$，利用 Jacobi 迭代法求出近似解所需迭代次数为 1134 次，近似解与精确解误差限为 0.00205791.

3.2.3 h=0.02

当 h=0.02 时，取 $\epsilon = 10^{-8}$，利用 Jacobi 迭代法求出近似解所需迭代次数为 6174 次，近似解与精确解误差限为 0.00032399.

3.2.4 h=0.01

当 h=0.01 时，取 $\epsilon = 10^{-8}$，利用 Jacobi 迭代法求出近似解所需迭代次数为 21897 次，近似解与精确解误差限为 10^{-8}.

4 对数值结果的分析和总结

分析 3.1 内容，容易发现当 h 取值相同时，三种不同迭代法在使得近似解与精确解的误差相似条件下，SOR 迭代法需要的迭代次数小于 Gauss-Seidel 迭代法小于 Jacobi 迭代法。即 SOR 迭代法迭代效率最高，而 Jacobi 迭代法迭代效率较低。

分析 3.2 内容可知，当 h 取值越小时，Jacobi 矩阵迭代所需次数越多，近似成平方比例增长，误差限成平方反比关系。即当 h 变为原来的 n 分之 1 时，迭代次数近似变为原来的 n^2 倍，误差限近似为初始的 $\frac{1}{n^2}$。

总结：为使得迭代更快收敛至要求精度，可以选择使用 SOR 迭代法；若使用 Jacobi 迭代法可以适当减小 h 的取值使得精度增加，但不可过小，因为 h 过小会导致迭代次数成平方关系增加。

5 程序

5.1 Jacobi 迭代代码

分别令 h=0.1;0.05;0.02

```
h=input('Please input h\n');%输入n
N=1/h;
ep=1.0e-8;%误差限
C=[];
for i=1:N-1
    for j=1:N-1
        if (i-j==1||j-i==1)
            C(i,j)=1;
        else
            C(i,j)=0;
        end
    end
end
%计算矩阵C.

Lh=[];
for l=1:(N-1)^2%
    for k=1:(N-1)^2
        if (l-k==1||k-l==1)
            Lh(l,k)=-1;
        elseif (l==k)
            Lh(l,k)=4;
        elseif (l-k==N-1||k-l==N-1)
            Lh(l,k)=-1;
        else
            Lh(i,j)=0;
        end
    end
end
for i=1:(N-1)^2
    Lh(i,i)=4;
end
for i=N:(N-1):(N-2).*N
    Lh(i,i-1)=0;
end
for i=(N-1):(N-1):(N-2).*N
    Lh(i,i+1)=0;
end
%以上计算出Lh.
A=Lh;

f=[];
u0=[];
for j=1:N-1
    for i=1:N-1
        f((j-1).*(N-1)+i,1)=2.*pi.^2.*sin(i.*h.*pi).*s
        u0((j-1).*(N-1)+i,1)=sin(i*h*pi)*sin(j*h*pi);
    end
end
b=h.^2.*f;
%计算出常数项f.

x0=[];
for i=1:(N-1)^2
    x0(i,1)=0;
end
%初始迭代矩阵.

x=[];
for i=1:(N-1)^2
      sum=0;
      for j=1:(N-1)^2
          if j~=i
              sum=sum+A(i,j)*x0(j);
          end
      end
      x(i,1)=(b(i,1)-sum)/A(i,i);
end
num=1;
while max(abs(x0-x))>=ep
    x0=x;
```

```
    for i=1:(N-1)^2
        sum=0;
        for j=1:(N-1)^2
            if j~=i
                sum=sum+A(i,j)*x0(j);
            end
        end
        x(i)=(b(i)-sum)/A(i,i);
    end
    num=num+1;
end
num
max(abs(x-u0))
```

当 h=0.01 时，代码如下：

```
h=0.01;
N=1/h;
ep=1.0e-8;
x0=[];
for i=1:(N-1)^2
    x0(i,1)=0;
end
x=[];
for i=1:(N-1)^2
    x(i,1)=1;
end
%初始迭代矩阵.

f=[];
u0=[];
for j=1:N-1
    for i=1:N-1
        f((j-1).*(N-1)+i,1)=2.*pi.^2.*sin(i.*h.*pi).*sin(j.*h.*pi);
        u0((j-1).*(N-1)+i,1)=sin(i*h*pi)*sin(j*h*pi);
    end
end
b=h^2*f;
%计算出常数项f.

Lh=[];
for l=1:(N-1)^2
    for k=1:(N-1)^2
        if (l-k==1||k-l==1)
            Lh(l,k)=-1;
        elseif (l==k)
            Lh(l,k)=4;
        elseif (l-k==N-1||k-l==N-1)
            Lh(l,k)=-1;
        else
            Lh(i,j)=0;
        end
    end
end
for i=1:(N-1)^2
    Lh(i,i)=4;
end
for i=N:(N-1):(N-2).*N
    Lh(i,i-1)=0;
end
for i=(N-1):(N-1):(N-2).*N
    Lh(i,i+1)=0;
end
%以上计算出Lh.

for i=1:(N-1)^2
    for j=1:(N-1)^2
        if i==j
            D(i,j)=Lh(i,j);
        else
            D(i,j)=0;
        end
    end
end
L=zeros((N-1)^2,(N-1)^2);
U=zeros((N-1)^2,(N-1)^2);
for i=2:(N-1)^2
    for j=1:i-1
        L(i,j)=Lh(i,j);
    end
end
for j=2:(N-1)^2
    for i=1:j-1
        U(i,j)=Lh(i,j);
    end
end
B=-D\(L+U);
d=D\b;
```

```
num=1;
while max(abs(x-x0))>=ep
    x=x0;
    x0=B*x+d;
    num=num+1;
end
num
max(abs(x-x0))
```

5.2 Gauss-Seidel 迭代法代码

令 h=0.1

```
h=input('Please input h\n');%输入n
N=1/h;
ep=1.0e-8;%误差限
C=[];
for i=1:N-1
    for j=1:N-1
        if (i-j==1||j-i==1)
            C(i,j)=1;
        else
            C(i,j)=0;
        end
    end
end
%计算矩阵C.

Lh=[];
for l=1:(N-1)^2
    for k=1:(N-1)^2
        if (l-k==1||k-l==1)
            Lh(l,k)=-1;
        elseif (l==k)
            Lh(l,k)=4;
        elseif (l-k==N-1||k-l==N-1)
            Lh(l,k)=-1;
        else
            Lh(i,j)=0;
        end
    end
end
for i=N:(N-1):(N-2).*N
    Lh(i,i-1)=0;
end
for i=(N-1):(N-1):(N-2).*N
    Lh(i,i+1)=0;
end
Lh(9,9)=4;
%以上计算出Lh.
A=Lh;

f=[];
u0=[];
for j=1:N-1
    for i=1:N-1
        f((j-1).*(N-1)+i,1)=2.*pi.^2.*sin(i.*h.*pi).*s
        u0((j-1).*(N-1)+i,1)=sin(i*h*pi)*sin(j*h*pi);
    end
end
b=h.^2.*f;
%计算出常数项f.

x0=[];
for i=1:(N-1)^2
    x0(i,1)=0;
end
%初始迭代矩阵.

x=[];
for i=1:(N-1)^2
       sum=0;
       for j=1:(N-1)^2
           if j~=i
               sum=sum+A(i,j)*x0(j);
           end
       end
       x(i,1)=(b(i,1)-sum)/A(i,i);
    end
while max(abs(x0-x))>=ep & k<=1000
    x=x0;
    for i=1:(N-1)^2
       sum=0;
       for j=1:(N-1)^2
           if j~=i
```

```
                sum=sum+A(i,j)*x0(j);
            end
        end
        x0(i)=(b(i)-sum)/A(i,i);
    end
    k=k+1;
end
k
max(abs(x-u0))
```

5.3 SOR 迭代法

令 h=0.1, w 分别等于 1.2, 1.3, 1.9, 0.9.

```
h=input('Please input h\n');%输入n
w=input('Please input w\n');
N=1/h;
ep=1.0e-8;%误差限
C=[];
for i=1:N-1
    for j=1:N-1
        if (i-j==1||j-i==1)
            C(i,j)=1;
        else
            C(i,j)=0;
        end
    end
end
%计算矩阵C.

Lh=(zeros((n-1)^2,(n-1)^2));
for l=1:(N-1)^2
    for k=1:(N-1)^2
        if (l-k==1||k-l==1)
            Lh(l,k)=-1;
        elseif (l==k)
            Lh(l,k)=4;
        elseif (l-k==N-1||k-l==N-1)
            Lh(l,k)=-1;
        else
            Lh(i,j)=0;
        end
    end
end
for i=N:(N-1):(N-2)*N
    Lh(i,i-1)=0;
end
for i=(N-1):(N-1):(N-2)*N
    Lh(i,i+1)=0;
end
Lh(9,9)=4;
%以上计算出Lh.
A=Lh;

f=[];
u0=[];
for j=1:N-1
    for i=1:N-1
        f((j-1).*(N-1)+i,1)=2.*pi.^2.*sin(i.*h.*pi).*
        u0((j-1).*(N-1)+i,1)=sin(i*h*pi)*sin(j*h*pi);
    end
end
b=h^2.*f;
%计算出常数项f.

x0=[];
for i=1:(N-1)^2
    x0(i,1)=0;
end
%初始迭代矩阵.
x=[];
for i=1:(N-1)^2
    x(i,1)=1;
end
% x=[];
% for i=1:(N-1)^2
%        sum=0;
%        for j=1:(N-1)^2
%            if j~=i
%                sum=sum+A(i,j)*x0(j);
%            end
%        end
%        x(i,1)=(b(i,1)-sum)/A(i,i);
% end
num=0;
while max(abs(x-x0))>=ep
```

```
    x=x0;
    for i=1:(N-1)^2
       sum=0;
       for j=1:(N-1)^2
           if j~=i
               sum=sum+A(i,j)*x0(j);
           end
       end
       x0(i)=(1-w)*x0(i)+w*(b(i)-sum)/A(i,i);
    end
    num=num+1;
end
num
max(abs(x-u0))
```