

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

考虑在矩形区域的泊松问题:

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

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, $x_i = ih, y_j = jh, u_{i,j} \approx u(x_i, y_j), f_{ij} = f(x_i, y_j)$.

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

则差分格式为

$$\begin{cases} -u_{i-1,j} - u_{i,j-1} + 4u_{i,j} - u_{i+1,j} - u_{i,j+1} = h^2 f_{i,j} \\ u_{i,0} = u_{0,j} = u_i, N = u_N, j = 0, i, j = 1, 2, \dots, N-1. \end{cases}$$

记

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

进而可以写为线性代数方程组:

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

其中

$$C = \begin{pmatrix} 0 & 1 & 0 & 0 & \dots & 0 & 0 \\ 1 & 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & 1 & \dots & 0 & 0 \\ 0 & 0 & 1 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 0 & 1 \\ 0 & 0 & 0 & 0 & \dots & 1 & 0 \end{pmatrix}_{(N-1) \times (N-1)}$$

$$L_h = \begin{pmatrix} 4I - C & -I & & & \\ -I & 4I - C & -I & & \\ & \ddots & \ddots & \ddots & \\ & & -I & 4I - C & -I \\ & & & -I & 4I - C \end{pmatrix}$$

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 and 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 $h=0.1;0.05;0.02$

```
1 h=input('Please input h\n');%输入h
2 N=1/h;
3 ep=1.0e-8;%误差限
4 C=[];
5 for i=1:N-1
6     for j=1:N-1
7         if (i-j==1||j-i==1)
8             C(i,j)=1;
9         else
10            C(i,j)=0;
11        end
12    end
13 end
14 %计算矩阵C.
15
16 Lh=[];
17 for l=1:(N-1)^2%
18     for k=1:(N-1)^2
19         if (l-k==1||k-l==1)
20             Lh(l,k)=-1;
21         elseif (l==k)
22             Lh(l,k)=4;
23         elseif (l-k==N-1||k-l==N-1)
24             Lh(l,k)=-1;
25         else
26             Lh(i,j)=0;
27         end
28     end
29 end
30 for i=1:(N-1)^2
31     Lh(i,i)=4;
32 end
33 for i=N:(N-1):(N-2)*N
34     Lh(i,i-1)=0;
35 end
36 for i=(N-1):(N-1):(N-2)*N
37     Lh(i,i+1)=0;
38 end
39 %以上计算出Lh.
40 A=Lh;
41
42 f=[];
43 u0=[];
44 for j=1:N-1
45     for i=1:N-1
46         f((j-1)*(N-1)+i,1)=2.*pi.^2.*sin(i.*h.*pi).*s
47         u0((j-1)*(N-1)+i,1)=sin(i*h*pi)*sin(j*h*pi);
48     end
49 end
50 b=h.^2.*f;
51 %计算出常数项f.
52
53 x0=[];
54 for i=1:(N-1)^2
55     x0(i,1)=0;
56 end
57 %初始迭代矩阵.
58
59 x=[];
60 for i=1:(N-1)^2
61     sum=0;
62     for j=1:(N-1)^2
63         if j~=i
64             sum=sum+A(i,j)*x0(j);
65         end
66     end
67     x(i,1)=(b(i,1)-sum)/A(i,i);
68 end
69 num=1;
70 while max(abs(x0-x))>=ep
71     x0=x;
```

```

72 -     for i=1:(N-1)^2
73 -         sum=0;
74 -         for j=1:(N-1)^2
75 -             if j~i
76 -                 sum=sum+A(i,j)*x0(j);
77 -             end
78 -         end
79 -         x(i)=(b(i)-sum)/A(i,i);
80 -     end
81 -     num=num+1;
82 - end
83 - num
84 - max(abs(x-u0))
85
86
87
88

```

当 $h=0.01$ 时, 代码如下:

```

1 - h=0.01;
2 - N=1/h;
3 - ep=1.0e-8;
4 - x0=[];
5 - for i=1:(N-1)^2
6 -     x0(i,1)=0;
7 - end
8 - x=[];
9 - for i=1:(N-1)^2
10 -    x(i,1)=1;
11 - end
12 - %初始迭代矩阵.
13
14 - f=[];
15 - u0=[];
16 - for j=1:N-1
17 -     for i=1:N-1
18 -         f((j-1)*(N-1)+i,1)=2.*pi.^2.*sin(i.*h.*pi).*sin(j.*h.*pi);
19 -         u0((j-1)*(N-1)+i,1)=sin(i*h*pi)*sin(j*h*pi);
20 -     end
21 - end
22 - b=h^2*f;
23 - %计算出常数项f.
24
25 - Lh=[];
26 - for l=1:(N-1)^2
27 -     for k=1:(N-1)^2
28 -         if (l-k==1 || k-l==1)
29 -             Lh(l,k)=-1;
30 -         elseif (l==k)
31 -             Lh(l,k)=4;
32 -         elseif (l-k==N-1 || k-l==N-1)
33 -             Lh(l,k)=-1;
34 -         else
35 -             Lh(i,j)=0;
36 -         end
37 -     end
38 - end
39 - for i=1:(N-1)^2
40 -     Lh(i,i)=4;
41 - end
42 - for i=N:(N-1):(N-2).*N
43 -     Lh(i,i-1)=0;
44 - end
45 - for i=(N-1):(N-1):(N-2).*N
46 -     Lh(i,i+1)=0;
47 - end
48 - %以上计算出Lh.
49
50 - for i=1:(N-1)^2
51 -     for j=1:(N-1)^2
52 -         if i==j
53 -             D(i,j)=Lh(i,j);
54 -         else
55 -             D(i,j)=0;
56 -         end
57 -     end
58 - end
59 - L=zeros((N-1)^2,(N-1)^2);
60 - U=zeros((N-1)^2,(N-1)^2);
61 - for i=2:(N-1)^2
62 -     for j=1:i-1
63 -         L(i,j)=Lh(i,j);
64 -     end
65 - end
66 - for j=2:(N-1)^2
67 -     for i=1:j-1
68 -         U(i,j)=Lh(i,j);
69 -     end
70 - end
71 - B=-D\'(L+U);
72 - d=D\'b;

```

```

73 num=1;
74 while max(abs(x-x0))>=ep
75     x=x0;
76     x0=B*x+d;
77     num=num+1;
78 end
79 num
80 max(abs(x-x0))
81
82

```

5.2 Gauss-Seidel 迭代法代码

```

令 h=0.1
1 h=input('Please input h\n');%输入n
2 N=1/h;
3 ep=1.0e-8;%误差限
4 C=[];
5 for i=1:N-1
6     for j=1:N-1
7         if (i-j==1 || j-i==1)
8             C(i,j)=1;
9         else
10            C(i,j)=0;
11        end
12    end
13 end
14 %计算矩阵C.
15
16 Lh=[];
17 for l=1:(N-1)^2
18     for k=1:(N-1)^2
19         if (l-k==1 || k-l==1)
20             Lh(l,k)=-1;
21         elseif (l==k)
22             Lh(l,k)=4;
23         elseif (l-k==N-1 || k-l==N-1)
24             Lh(l,k)=-1;
25         else
26             Lh(i,j)=0;
27         end
28     end
29 end
30 for i=N:(N-1):(N-2)*N
31     Lh(i,i-1)=0;
32 end
33 for i=(N-1):(N-1):(N-2)*N
34     Lh(i,i+1)=0;
35 end
36 Lh(9,9)=4;
37 %以上计算出Lh.
38 A=Lh;
39
40 f=[];
41 u0=[];
42 for j=1:N-1
43     for i=1:N-1
44         f((j-1)*(N-1)+i,1)=2.*pi.^2.*sin(i.*h.*pi).*s
45         u0((j-1)*(N-1)+i,1)=sin(i*h*pi)*sin(j*h*pi);
46     end
47 end
48 b=h.^2.*f;
49 %计算出常数项f.
50
51 x0=[];
52 for i=1:(N-1)^2
53     x0(i,1)=0;
54 end
55 %初始迭代矩阵.
56
57 x=[];
58 for i=1:(N-1)^2
59     sum=0;
60     for j=1:(N-1)^2
61         if j~=i
62             sum=sum+A(i,j)*x0(j);
63         end
64     end
65     x(i,1)=(b(i,1)-sum)/A(i,i);
66 end
67 while max(abs(x0-x))>=ep & k<=1000
68     x=x0;
69     for i=1:(N-1)^2
70         sum=0;
71         for j=1:(N-1)^2
72             if j~=i

```

```

73 -         sum=sum+A(i,j)*x0(j);
74 -     end
75 - end
76 - x0(i)=(b(i)-sum)/A(i,i);
77 - end
78 - k=k+1;
79 - end
80 - k
81 - max(abs(x-x0))

```

5.3 SOR 迭代法

令 $h=0.1$, w 分别等于 1.2, 1.3, 1.9, 0.9.

```

1 - h=input('Please input h\n');%输入h
2 - w=input('Please input w\n');%输入w
3 - N=1/h;
4 - ep=1.0e-8;%误差限
5 - C=[];
6 - for i=1:N-1
7 -     for j=1:N-1
8 -         if (i-j==1||j-i==1)
9 -             C(i,j)=1;
10 -        else
11 -            C(i,j)=0;
12 -        end
13 -    end
14 - end
15 - %计算矩阵C.
16 -
17 - Lh=zeros((n-1)^2,(n-1)^2);
18 - for l=1:(N-1)^2
19 -     for k=1:(N-1)^2
20 -         if (l-k==1||k-l==1)
21 -             Lh(l,k)=-1;
22 -         elseif (l==k)
23 -             Lh(l,k)=4;
24 -         elseif (l-k==N-1||k-l==N-1)
25 -             Lh(l,k)=-1;
26 -         else
27 -             Lh(l,k)=0;
28 -         end
29 -     end
30 - end
31 - for i=N:(N-1):(N-2)*N
32 -     Lh(i,i-1)=0;
33 - end
34 - for i=(N-1):(N-1):(N-2)*N
35 -     Lh(i,i+1)=0;
36 - end
37 - Lh(9,9)=4;
38 - %以上计算出Lh.
39 - A=Lh;
40 -
41 - f=[];
42 - u0=[];
43 - for j=1:N-1
44 -     for i=1:N-1
45 -         f((j-1)*(N-1)+i,1)=2.*pi.^2.*sin(i.*h.*pi).*
46 -         u0((j-1)*(N-1)+i,1)=sin(i.*h.*pi).*sin(j.*h.*pi);
47 -     end
48 - end
49 - b=h^2.*f;
50 - %计算出常数项f.
51 -
52 - x0=[];
53 - for i=1:(N-1)^2
54 -     x0(i,1)=0;
55 - end
56 - %初始迭代矩阵.
57 - x=[];
58 - for i=1:(N-1)^2
59 -     x(i,1)=1;
60 - end
61 - % x=[];
62 - % for i=1:(N-1)^2
63 - %     sum=0;
64 - %     for j=1:(N-1)^2
65 - %         if j~=i
66 - %             sum=sum+A(i,j)*x0(j);
67 - %         end
68 - %     end
69 - %     x(i,1)=(b(i,1)-sum)/A(i,i);
70 - % end
71 - num=0;
72 - while max(abs(x-x0))>=ep

```

```

73 - x=x0;
74 - for i=1:(N-1)^2
75 -     sum=0;
76 -     for j=1:(N-1)^2
77 -         if j~=i
78 -             sum=sum+A(i,j)*x0(j);
79 -         end
80 -     end
81 -     x0(i)=(1-w)*x0(i)+w*(b(i)-sum)/A(i,i);
82 - end
83 - num=num+1;
84 - end
85 - num
86 - max(abs(x-u0))
87
88
89

```