1 实验课题

n阶希尔伯特矩阵$\boldsymbol{H}$的元素为$h_{ij}=1.0/(i+j-1)$, 假设$\boldsymbol{x}$是元素均为1.0的n维向量, 计算$\boldsymbol{b}=\boldsymbol{H}\boldsymbol{x}$, 完成以下实验:

1. 利用SOR迭代法求解$\boldsymbol{H}\hat{\boldsymbol{x}}=\boldsymbol{b}$,计算$\|\hat{\boldsymbol{x}}-\boldsymbol{x}\|_\infty$.
2. 利用共轭梯度(CG)法求解$\boldsymbol{H}\hat{\boldsymbol{x}}=\boldsymbol{b}$, 计算$\|\hat{\boldsymbol{x}}-\boldsymbol{x}\|_\infty$.

2 几点说明

和前四次的实验不同, 这次实验的程序文件不止一个, 所以需要稍作说明.

- generateMatrixHilbert.m是生成希尔伯特矩阵的函数文件.
- rho.m是计算谱半径的函数文件.
- sor.m是用SOR迭代法求解线性方程的函数文件.
- cg.m是用共轭梯度法求解线性方程的函数文件.
- displayResult.m是一个在控制台打印结果的小函数, 用于减少代码冗余.
- Lab05.m是主程序文件, 它创建变量并调用上面的函数完成计算和结果显示.

此外, 由于代码较长, 此次实验报告正文部分不再详细解释每段代码的含意, 而是展示计算结果. 当然, 我们依旧会在附录里附上完整的代码.

3 SOR迭代

首先, 对于阶数为2,3,4,5,6的希尔伯特矩阵, 我们可以先计算出它们的最优松弛因子ω_{opt}和迭代矩阵的谱半径ρ_{min}.计算结果如下表:

表3.1 SOR迭代法的最优松弛因子和迭代矩阵的谱半径

阶数n	ω_{opt}	ρ_{min}
2	1.33333	0.333333
3	1.62310	0.850815
4	1.75806	0.986261
5	1.82138	0.999149
6	1.85791	0.999956

观察**表3.1**, 我们可以知道, 当SOR迭代法被使用于求解希尔伯特矩阵时, 即使矩阵的阶数n不大, 并且使用最优松弛因子, 其其迭代矩阵的谱半径也非常接近于1, 也就是说收敛速度非常缓慢. 以$n=5$时为例, 我们使用最优松弛因子进行SOR迭代, 最后当迭代进行到10473次时, 所求解的最大绝对误差为0.001332.

收敛速度慢还会导致一个问题, 那就是提高一点点精度就要增加非常多的迭代次数. **表3.2**,**表3.3**和**表3.4**分别是ω取0.3,1.0和1.5时的迭代次数和最大绝对误差对照表.

表3.2　SOR迭代法当$\omega = 0.3$时的迭代次数和最大绝对误差

阶数n	迭代次数	e_{max}
10	8530	0.009025
20	9102	0.019048
40	11636	0.021042
60	15578	0.020771
80	14157	0.022182

表3.3　SOR迭代法当$\omega = 1.0$时的迭代次数和最大绝对误差

阶数n	迭代次数	e_{max}
10	10000	0.022660
20	10000	0.034198
40	16000	0.033042
60	16000	0.041134
80	15000	0.056167

表3.4　SOR迭代法当$\omega = 1.5$时的迭代次数和最大绝对误差

阶数n	迭代次数	e_{max}
10	10000	0.043229
20	10000	0.061247
40	16000	0.049080
60	16000	0.104971
80	15000	0.096149

表3.2到表3.4的结果表明SOR迭代法应用于希尔伯特线性方程组的求解, 可以保持一定的计算精度, 且松弛因子越大, 在同样的计算次数下计算越不精确.

4　CG迭代

CG迭代法相比SOR迭代法迭代次数小得多且在矩阵阶数非常高时任可以获得10^{-3}的计算精度. 我们分别生成阶数为40,80,200,500,1000的希尔伯特矩阵进行实验, 计算结果如表4.1.

表4.1　CG迭代法的迭代次数和最大绝对误差

阶数n	迭代次数	e_{max}
40	11	0.001320
80	14	0.001534
200	21	0.001066
500	22	0.001740
1000	26	0.002110

共轭梯度法有一个很好的性质, 那就是如果线性方程组的系数矩阵$\boldsymbol{A}$至多有k个互不相同的特征值, 则共轭梯度法至多k步就可以得到方程组$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$的精确解. 而在现实生活中, 由于计算机舍入误差的存在, 共轭梯度法常作为一种迭代方法被使用.

5 实验结论

本实验探索了使用SOR迭代法和CG迭代法求解希尔伯特矩阵为系数矩阵的线性代数方程组的可行性. 其中SOR迭代法迭代次数非常多, CG法相比之下迭代次数少得多, 但两种迭代法都不能获得非常高的计算精度, 这和希尔伯特矩阵的病态性质有关.

此外, 为了确认算法程序编写的正确性, 本实验的所有实验数据选取都和课本一致. 在绝大多数情况我们获得了和书上完全相同或十分接近的计算结果, 但在两处与课本的结果存在较大出入. 第一处出现在**表3.1**阶数为6的希尔伯特矩阵的ω_{opt}和ρ_{min}的计算结果. 第二处出现在**表3.4**阶数为80的希尔伯特矩阵, 且取$\omega = 1.5$ 时得到的e_{max}的计算结果. 当然我们认为自己编写的程序无误, 导致这种现象的原因可能和MATLAB以及MATHEMATICA本身的计算特性有关. 这里我们就不深究了.

6 附录

Lab05.m

```
%Lab05
clear; clc;

Emax = zeros(1,5);
Count = zeros(1,5);

disp('*********************************************************');
disp('*                         SOR                           *');
disp('*********************************************************');
N = [2,3,4,5,6];
W = zeros(1,5);
R = zeros(1,5);
w = linspace(0,2,100000);


for i = 1:5
    H = generateMatrixHilbert(N(i));
    r = rho(H,w);
    W(i) = w(find(r==min(r),1));
    R(i) = min(r);
end
disp([W;R]);

H = generateMatrixHilbert(N(4));
result = sor(H,H*ones(5,1),W(4),10^-6,16000);
[m,n] = size(result);
count = m - 1;
emax = max(abs(result(m,2:n)-ones(1,n-1)));
fprintf('%d\n',N(4));
fprintf('%f\n',count);
fprintf('%f\n\n',emax);

N = [10,20,40,60,80];

for i = 1:5
    H = generateMatrixHilbert(N(i));
```

```
    x = ones(N(i),1);
    b = H * x;
    result = sor(H,b,0.3,10^-6,16000);
    [m,n] = size(result);
    Count(i) = m - 1;
    Emax(i) = max(abs(result(m,2:n)-ones(1,n-1)));
end
displayResult(N,Count,Emax);

CountMax = [10000,10000,16000,16000,15000];
for i = 1:5
    H = generateMatrixHilbert(N(i));
    x = ones(N(i),1);
    b = H * x;
    result = sor(H,b,1.0,10^-6,CountMax(i));
    [m,n] = size(result);
    Count(i) = m - 1;
    Emax(i) = max(abs(result(m,2:n)-ones(1,n-1)));
end
displayResult(N,Count,Emax);

for i = 1:5
    H = generateMatrixHilbert(N(i));
    x = ones(N(i),1);
    b = H * x;
    result = sor(H,b,1.5,10^-6,CountMax(i));
    [m,n] = size(result);
    Count(i) = m - 1;
    Emax(i) = max(abs(result(m,2:n)-ones(1,n-1)));
end
displayResult(N,Count,Emax);

disp('*********************************************************');
disp('*                          CG                           *');
disp('*********************************************************');
N = [40,80,200,500,1000];

for i = 1:5
    H = generateMatrixHilbert(N(i));
    x = ones(N(i),1);
    b = H * x;
    result = cg(H,b,10^-8,1000);
    [m,n] = size(result);
    Count(i) = m - 1;
    Emax(i) = max(abs(result(m,2:n)-ones(1,n-1)));
end
displayResult(N,Count,Emax);
```

generateMatrixHilbert.m

```
function A = generateMatrixHilbert(n)
A = zeros(n,n);
for i = 1:n
   for j = 1:n
       A(i,j) = 1/(i+j-1);
   end
end
```

rho.m

```
function rho = rho(A,w)
format long;
m = size(w,2);
n = size(A,1);
rho = zeros(1,m);
Diag = zeros(n);
L = zeros(n);
U = zeros(n);
E = eye(n);

for i = 1:n
   Diag(i,i) = A(i,i);
end

for i = 1:n
   for j = 1:n
       if j < i
           L(i,j) = -A(i,j);
       end
       if j > i
           U(i,j) = -A(i,j);
       end
   end
end


for i = 1:m
    rho(i) = max(abs(eig(E/(Diag - w(i)*L)*((1 - w(i))*Diag + ...
        w(i)*U))));
end
```

sor.m

```
function result = sor(A,b,w,eps,kmax)
n = size(A,1);
x0 = zeros(1,n);
x1 = x0;
err = 1;
k = 0;
result = [0, x0];
while (err>eps && k<kmax)
    for i = 1:n
        sum = 0;
        for j = 1:i-1
            sum = sum + A(i,j)*x1(j);
        end
        for j = i+1:n
            sum = sum + A(i,j)*x0(j);
        end
        x1(i) = ((b(i)) - sum) / A(i,i);
        x1(i) = w*x1(i) + (1-w)*x0(i);
    end
    err = max(abs(x1-x0));
    x0 = x1;
    k = k + 1;
    result = [result;[k,x1]];
```

```
end
```

cg.m

```
function result = cg(A,b,eps,kmax)
n = size(A,1);
x = zeros(1,n);
r = (b - A * x')';
rho0 = r * r';
rho1 = rho0;
k = 0;
result = [0,x];

while (sqrt(rho1)>eps && k<kmax)
    k = k + 1;
    if (k==1)
       p = r;
    else
       beta = rho1/rho0;
       p = r + beta * p;
    end
    w = A * p';
    alpha = rho1/(p * w);
    x = x + alpha * p;
    r = r - alpha * w';
    rho0 = rho1;
    rho1 = r * r';
    result = [result;[k,x]];
end
```

displayResult.m

```
function displayResult(X,Y,Z)
n = size(X,2);
for i = 1:n
    fprintf('%8d\t',X(i));
end
fprintf('\n');
for i = 1:n
    fprintf('%8d\t',Y(i));
end
fprintf('\n');
for i = 1:n
    fprintf('%f\t',Z(i));
end
fprintf('\n\n');
```