使用 CG 法等方法求解病态方程组的数值报告

22035029 郭淼

1、求解线性方程组$A_1x=b_1$

分析系数矩阵：

计算发现矩阵 A1 不是对称矩阵，故在使用 cg 法和 pcg 法时采用求解与原方程组等价的方程组 A'Ax=A'b。

cond(A1)= 8.0488e+13，cond(A1'*A1)= 8.1133e+18，A1 以及 A1'*A1 均是病态矩阵，故在收敛过程中可能会出现困难。

分别使用共轭梯度法（CG）、预处理共轭梯度法（PCG）和广义最小残差法（GMRES）来求解这个线性方程组。

程序说明： demo1.m 为主程序，CG.m、PCG.m 和 myGMRES.m 分别为共轭梯度法（CG）、预处理共轭梯度法（PCG）和广义最小残差法（GMRES）的算法实现。

1） 共轭梯度法：

设置收敛条件为 norm(r)<=epsilon, epsilon 取 1e-4, 最终需要迭代 74 步得到结果 x_cg，结果的残差$\|A1'*b1-A1'*A1*x_cg\|_2$=9.8962e-5。通过计算每一步迭代之后的残差范数发现，迭代到第 5 步时残差范数已经达到 9.4810e-4，说明之后的迭代效果甚微，若适当调整收敛条件可以大大减少所需迭代次数。

2） 预处理共轭梯度法：

设置收敛条件为 norm(r)<=epsilon, epsilon 取 1e-4, 最终需要迭代 24 步得到结果 x_pcg，结果的残差$\|A1'*b1-A1'*A1*x_pcg\|_2$=9.6737e-5。同样地，通过计算每一步迭代之后的残差范数发现，迭代到第 3 步时残差范数已经达到 4.3760e-4，说明之后的迭代效果甚微，若适当调整收敛条件可以大大减少所需迭代次数。

3） 广义最小残差法：

结果为 x_gmres，对应的残差$\|b1-A1*x_gmres\|_2$=1.9701e-14。

2、求解线性方程组$A_2x=b_2$

分析系数矩阵：

与第一个矩阵类似，计算发现矩阵 A2 不是对称矩阵，故在使用 cg 法和 pcg 法时采用求解与原方程组等价的方程组 A'Ax=A'b。

cond(A2)= 3.4818e+15，cond(A2'*A2)= 1.3499e+19，A2 以及 A2'*A2 均是病态矩阵，其条件数甚至比第一个矩阵更大。

分别使用共轭梯度法（CG）、预处理共轭梯度法（PCG）和广义最小残差法（GMRES）来求解这个线性方程组。

程序说明：demo2.m 为主程序，CG.m、PCG.m 和 myGMRES.m 分别为共轭梯度法（CG）、预处理共轭梯度法（PCG）和广义最小残差法（GMRES）的算法实现。

1）**共轭梯度法：**

设置收敛条件为 norm(r)<=epsilon，epsilon 取 1e-4，最终没有达到收敛条件，迭代到了最大步数 100 步得到结果 x2_cg，结果的残差$||A2'*b2-A2'*A2*x2_cg||_2$=0.0016。通过计算每一步迭代之后的残差范数发现，前 100 步残差下降非常缓慢，将最大迭代次数改为 1000 次之后，发现在第 102 步残差范数降至 8.9930e-4。

2） 预处理共轭梯度法：

设置收敛条件为 norm(r)<=epsilon，epsilon 取 1e-4，最终也没有达到收敛条件，迭代到 100 步得到结果 x2_pcg，结果的残差$|| A2'*b2-A2'*A2*x2_pcg||_2$=4.6040e-4。同样地，通过计算每一步迭代之后的残差范数发现，迭代到第 54 步时残差范数达到 9.0806e-4，相较于共轭梯度法收敛速度加快了。

3） 广义最小残差法：

结果为 x2_gmres，对应的残差$||b2-A2*x2_gmres||_2$=9.6711e-14。

3、 实验总结

从对两个线性方程组的求解情况来看，GMRES 方法的效果最优，解的误差最小，同时其花费时间也是最长的。再比较共轭梯度法和预处理共轭梯度法，预处理共轭梯度法相较于共轭梯度法来说收敛的速度更快，如果更换预处理算子可以取得更好的结果，但保证两者迭代充分的情况下，两个算法获得解的误差相近，这是由于随着迭代次数不断增加，误差不断累积，无法继续保证算法中搜索方向的正交性，因此会出现无法继续收敛的情况。

MATLAB 代码：

demo1.m

```
%使用共轭梯度法（CG）、预处理共轭梯度法（PCG）和广义最小残差法（GMRES）来求解
A1x=b1
load('A1.mat')
load('b1.mat')
A=A1'*A1;
```

```
b=A1'*b1;
%由于矩阵 A1 不是对称矩阵，故在使用 cg 法和 pcg 法时采用求解与原方程组等价的方程组 A'Ax=A'b
[x_cg,i_cg,r_cg]=CG(A,b,1e-4);

[x_pcg,i_pcg,r_pcg]=PCG(A,b,1e-4);

[x_gmres,r_gmres]=myGMRES(A1,b1); %运行时间较长
```

demo2.m

```
%使用共轭梯度法（CG）、预处理共轭梯度法（PCG）和广义最小残差法（GMRES）来求解 A2x=b2
load('A2.mat')
load('b2.mat')
AA=A2'*A2;
bb=A2'*b2;
%由于矩阵 A2 不是对称矩阵，故在使用 cg 法和 pcg 法时采用求解与原方程组等价的方程组 A'Ax=A'b
[x2_cg,i2_cg,r2_cg]=CG(AA,bb,1e-4);

[x2_pcg,i2_pcg,r2_pcg]=PCG(AA,bb,1e-4);

[x2_gmres,r2_gmres]=myGMRES(A2,b2);   %运行时间较长
```

CG.m

```
function [x,i,y]=CG(A,b,epsilon)
%共轭梯度法求解线性方程组 Ax=b
%输入：系数矩阵 A，向量 b,收敛条件 epsilon
%输出：数值解 x，迭代步数 i，残差向量的二范数 y
[n,m]=size(A);
x0=zeros(m,1);
r0=b-A*x0;
u0=r0;
max=1000;
%y=zeros(max,1);
for i=1:max
```

```
    alpha=(r0'*r0)/(u0'*A*u0);
    x=x0+alpha*u0;
    r=r0-alpha*A*u0;
    %y(i)=norm(r);
    beta=(r'*r)/(r0'*r0);
    u=r+beta*u0;
    %判断收敛条件 1
%     if norm(x-x0)<=10^(-8)
%         break
%     end
    %判断收敛条件 2
    if norm(r)<=epsilon
        break
    end
    x0=x;
    r0=r;
    u0=u;
end
y=norm(b-A*x);
```

PCG.m

```
function [x,i,y]=PCG(A,b,epsilon)
%预处理共轭梯度法求解线性方程组 Ax=b
%输入：系数矩阵 A，向量 b,收敛条件 epsilon
%输出：数值解 x，迭代步数 i，残差向量的二范数 y
[n,m]=size(A);
M=zeros(n,n); %M 为 jacobi 预处理矩阵
for i=1:n
    M(i,i)=1/A(i,i);
end
x0=zeros(m,1);
r0=b-A*x0;
z0=M*r0;
u0=z0;
max=100;
```

```
%y=zeros(max,1);
for i=1:max
    alpha=(r0'*z0)/(u0'*A*u0);
    x=x0+alpha*u0;
    r=r0-alpha*A*u0;
    %y(i)=norm(r);
    z=M*r;
    beta=(z'*r)/(z0'*r0);
    u=z+beta*u0;
    %判断收敛条件 1
%       if norm(x-x0)<=10^(-8)
%           break
%       end
    %判断收敛条件 2
    if norm(r)<=epsilon
        break
    end
    x0=x;
    r0=r;
    z0=z;
    u0=u;
end
y=norm(b-A*x);
```

myGMRES.m

```
function [x,r]=myGMRES(A,b)
%使用 gmres 法求解线性方程组
%输入：系数矩阵 A，右端向量 b
%输出：求得的数值解 x，所得解的残差的二范数 r
[m, ~] = size(A);
x0=zeros(m,1);
H = zeros(m+1,m);
V = zeros(m,m+1);     %A*V=V*H
r0 = b-A*x0;
beta=norm(r0);
```

```
V(:,1) = r0./beta;
for j = 1:m
    w = A*V(:,j);
    for i = 1:j
        H(i,j) = w'*V(:,i);
        w = w - H(i,j)*V(:,i);
    end
    H(j+1,j) = norm(w);

    if abs(H(j+1,j)) < 1e-10
        sprintf('done without residual')
        break;
    else
        V(:,j+1) = w./H(j+1,j);
    end
end
%接下来用 qr 分解求解最小二乘问题得到 y：min||beta*e1-H*y||
e1=zeros(j+1,1);
e1(1)=beta;
[Q,R]=qr(H(1:j+1,1:j)); %R 为 m+1*m
R=R(1:j,1:j);
bb=Q'*e1;
y=backward(R,bb(1:j));
x=x0+V(:,1:j)*y;

r=norm(b-A*x);
```

backward.m

```
function b=backward(A,b)
%回代法求解上三角形方程组
%输入 A:一个上三角形方阵    b:右端向量
%输出：b 为解
[n,~]=size(A);
for j=n:-1:2
    b(j)=b(j)/A(j,j);
```

```
    b(1:j-1)=b(1:j-1)-b(j)*A(1:j-1,j);
end
b(1)=b(1)/A(1,1);
```