# 作业三

敖旭扬　　PB18071477　　**2021 年 1 月 7 日**

第一题 解：

**(a)** 由 Gauss 顺序消元法的步骤和过程，第一次迭代后 $A$ 的第一列完成消去时，有

$$a_{ij}^{(1)} = a_{ij} - \frac{a_{i1}}{a_{11}} a_{1j}, \quad i, j = 2, 3, \cdots, n \tag{1}$$

由于 $A$ 是对称矩阵且满足 $a_{11} \neq 0$，所以有

$$a_{ij} = a_{ji}, \quad i, j = 1, 2, 3, \cdots, n \tag{2}$$

所以

$$a_{ij}^{(1)} = a_{ij} - \frac{a_{i1}}{a_{11}} a_{1j} = a_{ji} - \frac{a_{1i}}{a_{11}} a_{j1} = a_{ji} - \frac{a_{j1}}{a_{11}} a_{1i} = a_{ji}^{(1)}, \quad i, j = 2, 3, \cdots, n \tag{3}$$

从而 $A^{(1)}$ 是对称的。

**(b)** 计算一个正定 (positive definite) 矩阵 LU 分解的算法如下：

```matlab
%% 正定矩阵(Positive Definite Matrix)LU分解算法
function [L, U] = LU_PDM(A)
    [~, n] = size(A);
    L = zeros(n, n);
    U = zeros(n, n);
    for i = 1:n
        for j = i + 1:n
            for k = j:n
                A(j, k) = A(j,k)-A(i,k)*A(j,i)/A(i,i);
                A(k, j) = A(j,k);
            end
        end
        for k = i:n
            U(i, k) = A(i, k);
            L(k, i) = A(i, k)/U(i, i);
        end
    end
end
```

(c) 使用 Cholesky 分解解方程组 $Ax = b$ 的 MATLAB 程序显示如下：

```
clear,clc,clf
A = [4,-2,4,2; -2,10,-2,-7; 4,-2,8,4; 2,-7,4,7];
b = [8; 2; 16; 6];
x = deEquations(A,b)
%% 运行的输出结果
% x =
%
%      1
%      2
%      1
%      2

%% 用Cholesky分解解方程组Ax=b
function [X]=deEquations(A,b)
N=length(A);
L=cholesky(A);
Lt=L';
X=zeros(N, 1);
Y=zeros(N, 1);
for j=1:N
    Y(j)=(b(j)-L(j,1:j-1)*Y(1:j-1))/L(j,j);
end
for k=N:-1:1
    X(k)=(Y(k)-Lt(k,k+1:N)*X(k+1:N))/Lt(k,k);
end
end

function [L]=cholesky(A)
N=length(A);
for i=1:N
    A(i,i)=sqrt(A(i,i)-A(i,1:i-1)*A(i,1:i-1)');
    if A(i,i)==0
        ME=MException('Zero Element', 'A(%d,%d) = 0',i,i);
        throw(ME);
```

```
    end
    for j=i+1:N
        A(j,i)=(A(j,i)-A(j,1:i-1)*A(i,1:i-1)')/A(i,i);
    end
end
L = tril(A); %取下三角部分
end
```

根据上述程序在命令行输出的结果（以注释形式写在了上面的代码中），原题所求的解为：

$$x = \begin{bmatrix} 1 \\ 2 \\ 1 \\ 2 \end{bmatrix}$$

第二题 解：

**(a)** 证明：

Richardson 迭代方法的迭代关系式为：

$$x^{(k+1)} = \omega I(\frac{1}{\omega}I - A)x^{(k)} + \omega Ib = (I - \omega A)x^{(k)} + \omega b \tag{4}$$

令

$$G_\omega = I - \omega A \tag{5}$$

则

$$x^{(k+1)} = G_\omega x^{(k)} + \omega b \tag{6}$$

根据课本第五章开头处对迭代法收敛性的讨论可知，上述 Richardson 迭代方法收敛的充要条件为谱半径 $\rho(G_\omega) < 1$，又 $A$ 是正定矩阵，所以它的特征值均为正数，由已知，$G_\omega$ 的特征值为 $1 - \omega\lambda_i$，其中 $0 < \lambda_1 \le \lambda_i \le \lambda_n$，则

$$\rho(G_\omega) = max|1 - \omega\lambda_i| < 1 \tag{7}$$

则对任意 $\lambda_i$，需要满足

$$0 < \omega\lambda_i < 2 \tag{8}$$

即

$$\omega < 2/\lambda_n \tag{9}$$

(b) 由 (a) 得，

$$\rho(G_\omega) = \max|1-\omega\lambda_i| = \max(1-\omega\lambda_1, \omega\lambda_n - 1) \tag{10}$$

应该使得收敛速度尽量快，所以 $\rho(G_\omega)$ 应该尽量小，即取

$$\rho(G_\omega) = \min_{\omega}\max(1-\omega\lambda_1, \omega\lambda_n - 1) \tag{11}$$

所以 $\omega$ 的最佳值为

$$\omega_b = \arg\min_{\omega}\max(1-\omega\lambda_1, \omega\lambda_n - 1) = \frac{2}{\lambda_1 + \lambda_n} \tag{12}$$

且

$$\rho(G_\omega) = \begin{cases} 1-\omega\lambda_1 & \omega \le \omega_b \\ \dfrac{\lambda_n - \lambda_1}{\lambda_n + \lambda_1} & \omega = \omega_b \\ \omega\lambda_n - 1 & \omega \ge \omega_b \end{cases} \tag{13}$$

(c) 使用 Richardson 迭代方法解 $Ax = b$ 的 MATLAB 程序显示如下：

```matlab
clear, clc, clf
%% 构造A和b
D = orth(rand(5, 5));
B = diag([1, 2, 3, 4, 5]);
A = D \ B * D;
b = rand(5, 1);
xexact = A \ b; % x精确解

%% 使用多个omega迭代求解
lambda1 = 1;
lambdan = 5;
omegab=2/(lambda1+lambdan);
eponch=1001;
omegalist = linspace(0.001, 0.399, eponch);
count=zeros(eponch,1)';
for i = 1:eponch
    count(i)=Richardson(A,b,xexact,omegalist(i));
end

%% 输出结果
```

```
[mink,idx]=min(count);
omegalist(idx);
fp=fopen('p2c_out.txt','w');
fprintf (fp,'bestomega = %f\tk=%d\n',omegalist(idx),mink);
fprintf (fp,'omegab = %f\tk=%d\n',omegab,Richardson(A,b,xexact,omegab));
semilogy(omegalist, count, 'k.-');
xlabel('\omega');
ylabel('迭代次数');

%% 返回Richardson迭代方法的迭代次数
function [k]=Richardson(A,b,xexact,omega)
    G = eye(5) - omega * A;
    x = zeros(5, 1);
    k = 0;
    while norm(x - xexact) > 1e-13
        x = G * x + omega * b;
        k = k + 1;
    end
end
```

上述程序某一次执行输出的结果为：

```
bestomega = 0.333330    k=70
omegab = 0.333333    k=70
```

同时输出的收敛迭代次数随 $\omega$ 变化的 semilogy 图为：

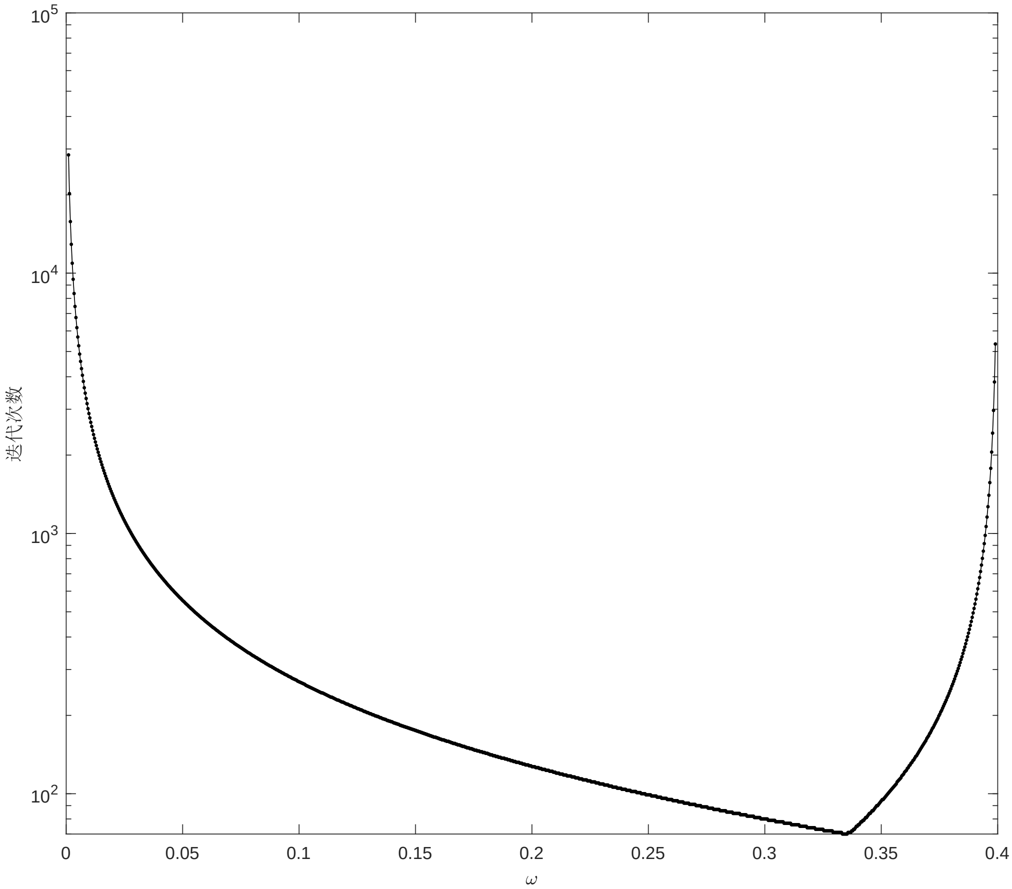


图 1: 寻找 Richardson 迭代方法的最佳 $\omega$

这一结果验证了最佳 $\omega$ 是 $\omega_b$，它使得收敛速度最快。

第三题 解：

**(a)** $I(f)=\int_a^b f(x)\mathrm{d}x$ 关于积分节点 $\{x_1,x_2,\cdots,x_n\}$ 的 Gauß 积分的数值积分公式为

$$I_n(f)=\sum_{i=1}^{n}\alpha_i f(x_i) \tag{14}$$

$n=6$ 时，代数精度不超过 $2n-1=11$，所以依次取线性不相关的 $f(x)=1,x^2,\cdots,x^{10}$，即 $f_k(x)=x^{2k},\quad k=0,1,\cdots,5$，根据

$$I_n(f)=I(f) \tag{15}$$

取 $a=-1,b=1$ 得

$$\sum_{i=1}^{6}\alpha_i x_i^{2k}=\int_{-1}^{1}x^{2k}\mathrm{d}x=\frac{2}{2k+1}\quad,\quad k=0,1,\cdots,5 \tag{16}$$

又根据 Gauß 的节点和积分权重关于原点对称，即

$$\alpha_i = \alpha_{i+3}, \quad x_i = -x_{i+3}, \quad i = 1, 2, 3 \tag{17}$$

所以式 (16) 变为

$$\sum_{i=1}^{6} \alpha_i x_i^{2k} = 2\sum_{i=1}^{3} \alpha_i x_i^{2k} = \frac{2}{2k+1} \quad , \quad k = 0, 1, \cdots, 5 \tag{18}$$

即

$$\sum_{i=1}^{3} \alpha_i x_i^{2k} = \frac{1}{2k+1} \quad , \quad k = 0, 1, \cdots, 5 \tag{19}$$

写成方程组则表示为

$$\begin{cases} \sum\limits_{i=1}^{3} \alpha_i = 1 \\ \sum\limits_{i=1}^{3} \alpha_i x_i^2 = \dfrac{1}{3} \\ \sum\limits_{i=1}^{3} \alpha_i x_i^4 = \dfrac{1}{5} \\ \sum\limits_{i=1}^{3} \alpha_i x_i^6 = \dfrac{1}{7} \\ \sum\limits_{i=1}^{3} \alpha_i x_i^8 = \dfrac{1}{9} \\ \sum\limits_{i=1}^{3} \alpha_i x_i^{10} = \dfrac{1}{11} \end{cases} \tag{20}$$

**(b)** 方程组 (20) 中一共有 $\langle \alpha_1, \alpha_2, \alpha_3, x_1, x_2, x_3 \rangle$ 共 6 个变量，且有

$$\frac{\partial\left(\sum\limits_{i=1}^{3} \alpha_i x_i^{2k}\right)}{\partial \alpha_j} = x_j^{2k}$$

$$\frac{\partial\left(\sum\limits_{i=1}^{3} \alpha_i x_i^{2k}\right)}{\partial x_j} = 2k\alpha_j x_j^{2k-1} \quad (k \neq 0)$$

$$\frac{\partial\left(\sum\limits_{i=1}^{3} \alpha_i\right)}{\partial x_j} = 0$$

其中 $j = 1, 2, 3; \quad k = 0, 1, 2, 3, 4, 5$

所以非线性方程组 (20) 的 Jacobian 的表达式为

$$
\boldsymbol{J} = \frac{\partial\left(\sum_{i=1}^{3}\alpha_i, \sum_{i=1}^{3}\alpha_i x_i^2, \sum_{i=1}^{3}\alpha_i x_i^4, \sum_{i=1}^{3}\alpha_i x_i^6, \sum_{i=1}^{3}\alpha_i x_i^8, \sum_{i=1}^{3}\alpha_i x_i^{10}\right)}{\partial(\alpha_1, \alpha_2, \alpha_3, x_1, x_2, x_3)}
= \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ x_1^2 & x_2^2 & x_3^2 & 2\alpha_1 x_1 & 2\alpha_2 x_2 & 2\alpha_3 x_3 \\ x_1^4 & x_2^4 & x_3^4 & 4\alpha_1 x_1^3 & 4\alpha_2 x_2^3 & 4\alpha_3 x_3^3 \\ x_1^6 & x_2^6 & x_3^6 & 6\alpha_1 x_1^5 & 6\alpha_2 x_2^5 & 6\alpha_3 x_3^5 \\ x_1^8 & x_2^8 & x_3^8 & 8\alpha_1 x_1^7 & 8\alpha_2 x_2^7 & 8\alpha_3 x_3^7 \\ x_1^{10} & x_2^{10} & x_3^{10} & 10\alpha_1 x_1^9 & 10\alpha_2 x_2^9 & 10\alpha_3 x_3^9 \end{bmatrix} \tag{21}
$$

(c) 求解 $n = 6$ 情况下的 Gauß 积分的积分节点和积分权重的 MATLAB 程序显示如下：

```
clear, clc, clf
x = linspace(-1, 1, 6);
w = zeros(3, 1)';

for i = 1:3
    syms t;
    Fi = @(t) alpha_fun(t, x, i);
    % 使用课本130页下方的\alpha公式初始化w
    w(i) = int(Fi, t, -1, 1);
end

x = x(1:3);
dwdx = ones(6);

while (max(dwdx) > 1e-10)
    J = Jacobian(w, x);
    b = cal_b(w, x);
    dwdx = J \ b;
    dwdx = dwdx';
    w = w - dwdx(1:3);
    x = x - dwdx(4:6);
end
%% 输出w和x
```

```
w
x

function f = alpha_fun(t, x, i)
    n = length(x);
    fac1 = t - x;
    fac2 = x(i) - x;
    f = prod(fac1((1:n) ~= i))/prod(fac2((1:n) ~= i));
end
%% 第三题(b)中非线性方程组的Jacobian的表达式
function [J] = Jacobian(w,x)
    J = [1, 1, 1,0, 0, 0;
        x.^2,x .* w .* 2;
        x.^4,(x.^3) .* w .* 4;
        x.^6,(x.^5) .* w .* 6;
        x.^8,(x.^7) .* w .* 8;
        x.^10,(x.^9) .* w .* 10
        ];
end

function [b] = cal_b(w, x)
    b = zeros(6, 1);
    for i = 1:6
        b(i) = sum(w .* (x.^(2*i-2)))-1/(2*i-1);
    end
end
```

上述程序在命令行的输出结果为

```
w =
   0.171324492379170   0.360761573048139
      0.467913934572691
x =
  -0.932469514203152  -0.661209386466265
     -0.238619186083197
```

所以积分节点和积分权重为

$$(x_1, x_2, x_3) = (-0.932469514203152, -0.661209386466265, -0.238619186083197)$$
$$(\alpha_1, \alpha_2, \alpha_3) = (0.171324492379170, 0.360761573048139, 0.467913934572691)$$

其中其他几项由式 (17) 得到。
这里初始权重的选取是根据课本（第三版）第 130 页下方的 $\alpha_i^{(n)}$ 公式计算而来（但是把其中的 $x_i^{(n)}$ 用 $x$ 初始值代替了）。

**(d)** Matlab 程序显示如下：

```
clear,clc,clf

[w,x]=gauss_w_x(5)

function [w,x]=gauss_w_x(n)
halfn=ceil(n/2);
w=zeros(halfn,1)';
x=che(1:n,n);
syms t;
for i=1:halfn
    Fi=@(t) alpha_fun(t,x,i);
    w(i)=int(Fi,t,-1,1);
end
x=x(1:floor(n/2));
dx=ones(halfn);
dw=ones(halfn);
J = Jacobian(x,w,n);
while(max([dx,dw]) > 1e-5)
    J = Jacobian(x,w,n);
    b = f(x,w,n);
    [dx,dw] =grad(J,b);
    x = x - dx;
    w = w - dw;
end
end
%% Chebyshev点
function [xj]=che(j,n)
```

```
    xj = cos(j*pi/n);
end
function [dx,dw] =grad(J,b)
n=size(J,1);
fn=floor(n/2);
dxdw=J\b;
dxdw=dxdw';
dx=dxdw(1:fn);
dw=dxdw(fn+1:n);
end
function f=alpha_fun(t,x,i)
n=length(x);
fac1=t-x;
fac2=x(i)-x;
f=prod(fac1((1:n) ~= i))/prod(fac2((1:n) ~= i));
end

function [J] = Jacobian(x ,w,n)
J=zeros(n,n);
m=length(w);
if mod(n,2)
    factor=ones(n,1)';
    factor(m)=1/2;
    xp=[x,0];
    m1zeros=zeros(m,1)';
    m2ones=ones(length(x),1)';
    J(1,:) = [m1zeros , m2ones].*factor;
    for i=2:n
        J(i,:) = [(xp .^ (2*i-3)) .* w .* (2*i-2) , x .^
           (2*i-2)].*factor;
    end
    J
else
    for i=1:n
        J(i,:) = [(x .^ (2*i-3)) .* w .* (2*i-2) , x .^
```

```
                (2*i-2)];
    end
end
end

function [b] = f(x,w,n)
b = zeros(n,1);
if mod(n,2)
    w=w(1:end-1);
    b(1)=sum(w) - 1;
    for i=2:n
        b(i)=sum(w .* (x .^ (2*i-2))) - 1/(2*i-1);
    end
else
    for i=1:n
        b(i)=sum(w .* (x .^ (2*i-2))) - 1/(2*i-1);
    end
end
end
```