

**《数值分析》实验报告(1)**

**——线性代数方程组求解综合实验**

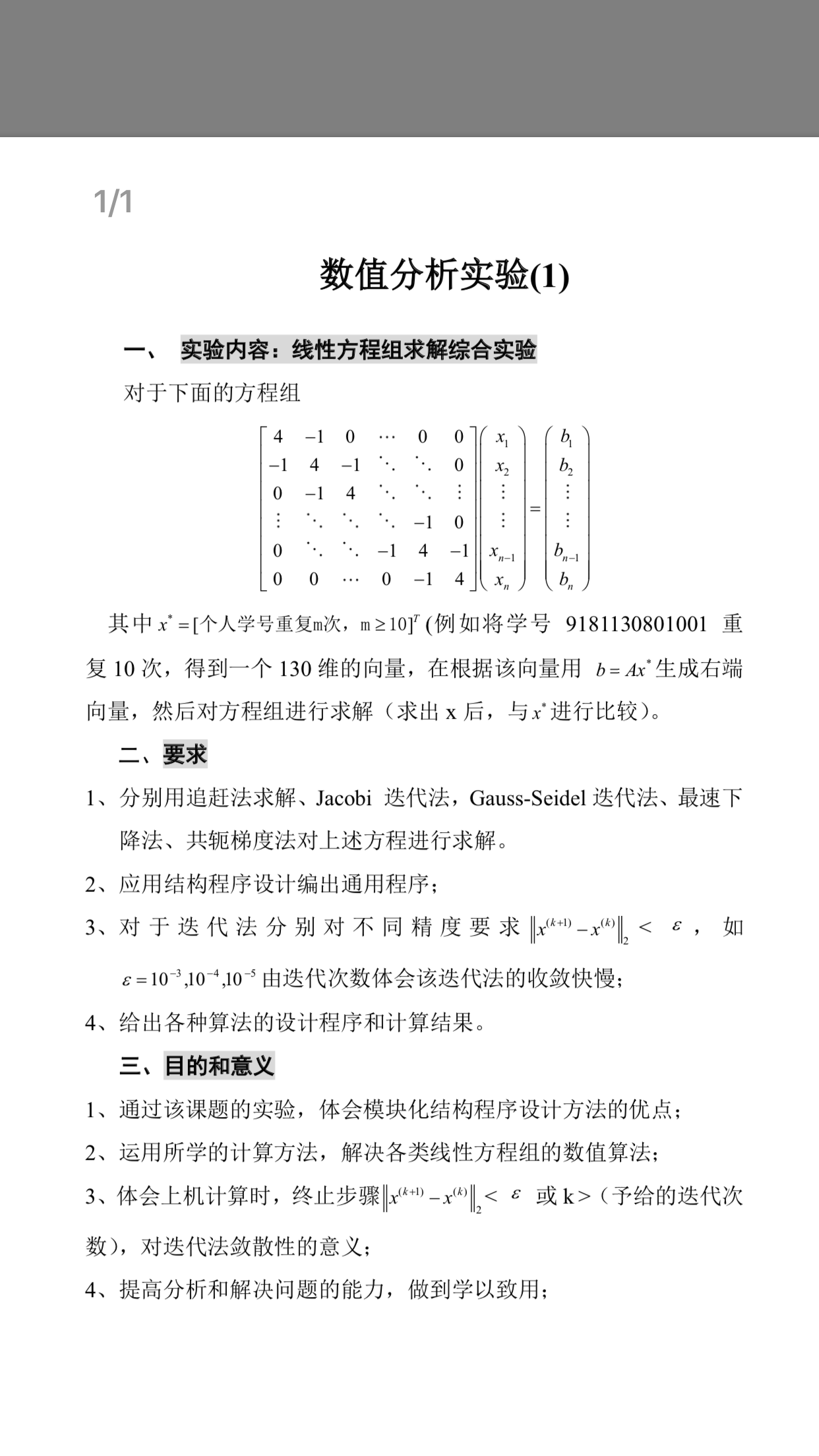
姓 名

学 号

专 业

班 级

**二Ｏ 20 年 5 月 21 日**



**代码均放置在附件中**

**更详细的数值结果也在附件中**

**均取m=10：**

1. **追赶法**

**代码：**

**1.**

%A是系数矩阵

%B是方程组的右端向量

clc

clear

s = 13;

x1 = [];

for i=1:s

fprintf('请分别依次单位输入%d位学号:\n',s);

x = input('请输入：');

x1(end+1) = x;

end

x1 %完整学号

m = 10; %学号重复次数(m>=10)

n = s\*m;

A1 = ones(n,n);

A2 = ones(n-1,n-1);

A = diag(diag(4\*A1))+ diag(diag(-1\*A2),1)...

+ diag(diag(-1\*A2),-1); %系数矩阵

x0 = repmat(x1,1,m); %学号重复m次得到的x0(x\*)

B = A\*x0.'; %利用b=Ax0生成的B

x = run(A,B,n);

**2.**

function [x] = run(A,B,n)

a=[0;diag(tril(A,-1),-1)];%下对角线

b=diag(A);%中对角线

c=[diag(triu(A,1),1);0];%上对角线

l=zeros(size(a,1),1);%求L

u=zeros(size(b,1),1);%求U

x=zeros(n,1);

y=zeros(n,1);

u(1)=b(1);

for i=2:n

l(i)=a(i)/u(i-1);

u(i)=b(i)-l(i)\*c(i-1);

end

y(1)=B(1);

for i=2:n

y(i)=B(i)-l(i)\*y(i-1);

end

x(n)=y(n)/u(n);

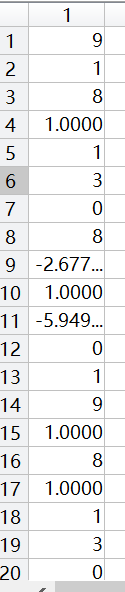
for i=n-1:-1:1

x(i)=(y(i)-c(i)\*x(i+1))/u(i);

end

end

**实验结果：**



1. **Jacobi迭代法**

**代码：**

**1.**

clc

clear

s = 13;

x1 = [];

for i=1:s

fprintf('请分别依次单位输入%d位学号:\n',s);

x = input('请输入：');

x1(end+1) = x;

end

x1 %完整学号

m = 10; %学号重复次数(m>=10)

n = s\*m;

A1 = ones(n,n);

A2 = ones(n-1,n-1);

A = diag(diag(4\*A1))+ diag(diag(-1\*A2),1)...

+ diag(diag(-1\*A2),-1); %系数矩阵

x0 = repmat(x1,1,m); %学号重复m次得到的x0(x\*)

B = A\*x0.'; %利用b=Ax0生成的B

x = jacobi(A,B);

**2.**

function r = jacobi(A,B,varargin)

sizeA=size(A);

sizev=size(varargin);

if sizev(2) == 0

rol = 0.001; %精度epsilon

n = 1000;

x = zeros(sizeA(1),1);

elseif sizev(2) == 1

rol = varargin{1};

n = 1000;

x = zeros(sizeA(1),1);

elseif sizev(2) == 2

rol = varargin{1};

n = varargin{2};

x = zeros(sizeA(1),1000);

elseif sizev(2) == 3

rol = varargin{1};

n = varargin{2};

x = varargin{3};

else

error('输入参数过多');

end

for i = 2:n

fprintf('迭代%d次\n',i-1)

for j = 1:sizeA(2)

sum1=0;

for k = 1:sizeA(1)

if j == k

continue;

end

sum1 = sum1 - x(k,i-1)\*A(j,k)/A(j,j);

end

x(j,i)=B(j)/A(j,j)+sum1;

end

if any(abs(x(:,i)-x(:,i-1))>rol) == 0

break;

end

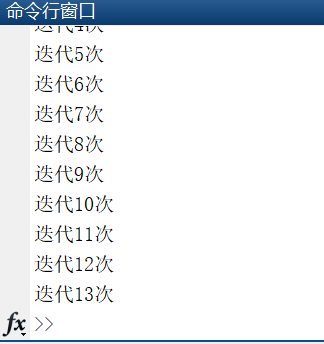
end

r = x;

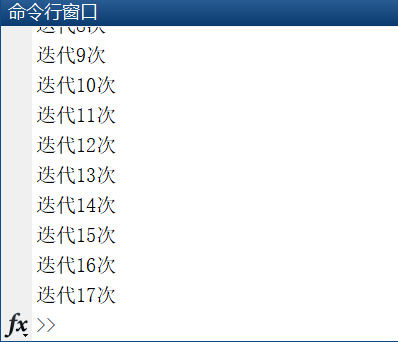
end

**实验结果：**

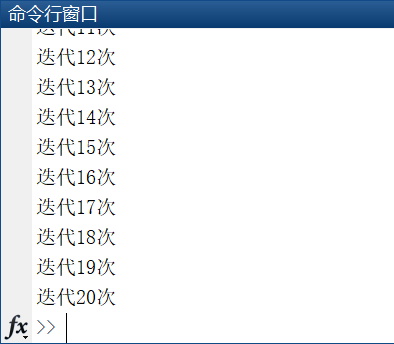
当要求精度为时，经过13次迭代解得x



当要求精度为时，经过17次迭代解得x



当要求精度为时，经过20次迭代解得x



1. **Gauss-Seidel迭代法**

**代码：**

**1.**

%A是系数矩阵

%B是方程组的右端向量

clc

clear

s = 13;

x1 = [];

for i=1:s

fprintf('请分别依次单位输入%d位学号:\n',s);

x = input('请输入：');

x1(end+1) = x;

end

x1 %完整学号

m = 10; %学号重复次数(m>=10)

n = s\*m;

A1 = ones(n,n);

A2 = ones(n-1,n-1);

A = diag(diag(4\*A1))+ diag(diag(-1\*A2),1)...

+ diag(diag(-1\*A2),-1); %系数矩阵

x0 = repmat(x1,1,m); %学号重复m次得到的x0(x\*)

B = A\*x0.'; %利用b=Ax0生成的B

x = Gauss\_Seidel(A,B);

**2.**

function r = Gauss\_Seidel(A,B,varargin)

sizeA=size(A);

sizev=size(varargin);

if sizev(2) == 0

rol = 0.00001;

n = 1000;

x = zeros(sizeA(1),1);

elseif sizev(2) == 1

rol = varargin{1};

n = 1000;

x = zeros(sizeA(1),1);

elseif sizev(2) == 2

rol = varargin{1};

n = varargin{2};

x = zeros(sizeA(1),1000);

elseif sizev(2) == 3

rol = varargin{1};

n = varargin{2};

x = varargin{3};

else

error('输入参数过多');

end

for i = 2:n

fprintf('迭代第%d次\n',i-1)

for j = 1:sizeA(2)

sum1=0;

for k = 1:j

if j == k

continue;

end

sum1 = sum1 - x(k,i)\*A(j,k)/A(j,j);

end

for k = j+1:sizeA(1)

sum1 = sum1 - x(k,i-1)\*A(j,k)/A(j,j);

end

x(j,i)=B(j)/A(j,j)+sum1;

end

if any(abs(x(:,i)-x(:,i-1))>rol) == 0

break;

end

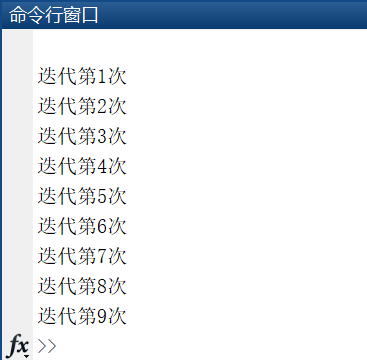
end

r = x;

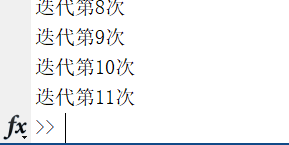
end

**实验结果：**

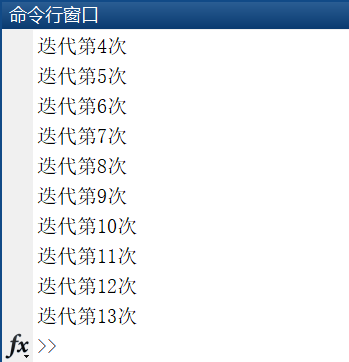
当要求精度为时，经过9次迭代解得x



当要求精度为时，经过11次迭代解得x



当要求精度为时，经过13次迭代解得x



1. **最速下降法**

**代码：**

**1.**

%A是系数矩阵

%B是方程组的右端向量

clc

clear

s = 13;

x1 = [];

for i=1:s

fprintf('请分别依次单位输入%d位学号:\n',s);

x = input('请输入：');

x1(end+1) = x;

end

x1 %完整学号

m = 10; %学号重复次数(m>=10)

n = s\*m;

A1 = ones(n,n);

A2 = ones(n-1,n-1);

A = diag(diag(4\*A1))+ diag(diag(-1\*A2),1)...

+ diag(diag(-1\*A2),-1); %系数矩阵

x0 = repmat(x1,1,m); %学号重复m次得到的x0(x\*)

B = A\*x0.'; %利用b=Ax0生成的B

x\_0 = ones(n,1); % 迭代初始值

rol = 0.001; % 误差

x = steepest(A,B,x\_0,rol,n);

**2.**

function [x,i] = steepest(A,B,x\_0,rol,n)

% n是线性方程组的维度

max1 = 1000;

r = B-A\*x\_0;

% 求内积

c = dot(r,r)/dot(A\*r,r);

x = x\_0+c\*r;

i = 1;

D = {};

while norm(x-x\_0)>=rol

x\_0 = x;

r = B-A\*x\_0;

c = dot(r,r)/dot(A\*r,r);

x = x\_0+c\*r;

fprintf('迭代第%d次\n',i);

D{i} = x;

xlswrite('x.xlsx',cell2mat(D),1); % 存放到x.xlsx的第一张sheet

i = i+1;

if(i>=max1)

disp('迭代次数超过',max1,'次，方程组可能不收敛');

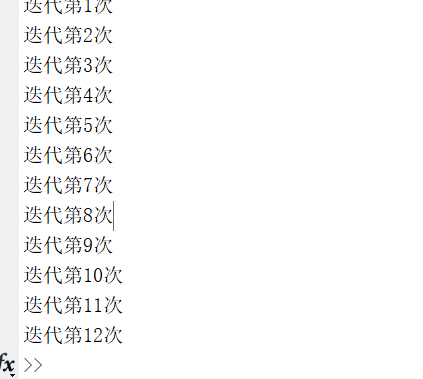
return;

end

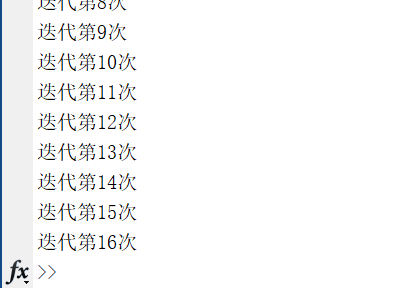
end

**实验结果：**

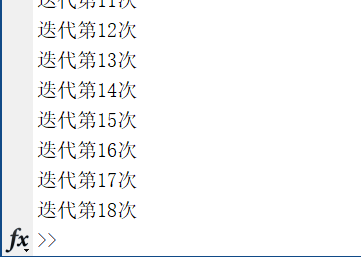
当要求精度为时，经过12次迭代解得x；



当要求精度为时，经过16次迭代解得x；



当要求精度为时，经过18次迭代解得x.



1. **共轭梯度法**

**代码：**

**1.**

%A是系数矩阵

%B是方程组的右端向量

clc

clear

s = 13;

x\_1 = [];

for i=1:s

fprintf('请分别依次单位输入%d位学号:\n',s);

x = input('请输入：');

x\_1(end+1) = x;

end

x\_1 %完整学号

m = 10; %学号重复次数(m>=10)

n = s\*m;

A1 = ones(n,n);

A2 = ones(n-1,n-1);

A = diag(diag(4\*A1))+ diag(diag(-1\*A2),1)...

+ diag(diag(-1\*A2),-1); %系数矩阵

x0\_ = repmat(x\_1,1,m); %学号重复m次得到的x0(x\*)

B = A\*x0\_.'; %利用b=Ax0生成的B

x\_0 = ones(n,1); % 迭代初始值

rol = 0.0001; % 误差

Conjugate(A,B,x\_0,rol);

**2.**

function Conjugate(A,B,x\_0,rol)

max\_i=1000; %迭代次数上限

fprintf('\n');

[y,i]= f1(A,B,x\_0,max\_i,rol);

fprintf('\n');

fprintf('迭代次数:\n %d \n',i);

fprintf('方程的解: \n');

fprintf('%10.6f',y);

xlswrite('data.xlsx',y,3); % 存放到data.xlsx的第3张sheet

end

**3.**

function [x,i] = f1(A,B,x\_0,max\_i,rol)

x=x\_0;

fprintf('\n x0= ');

fprintf(' %10.6f',x\_0);

r=B-A\*x;

d=r;

D={};

I={};

for k=0:max\_i

alpha=(r'\*r)/(d'\*A\*d);

xx=x+alpha\*d;

rr=B-A\*xx;

if (norm(rr,2)/norm(B,2))<= rol

fprintf('\n 找到');

i = k+1;

x=xx;

r=rr;

fprintf('\n x%d = ',k+1);

fprintf(' %10.6f',x);

I{1}=x;

fprintf('\n r%d = ',k+1);

fprintf(' %10.6f',r);

I{2}=r;

xlswrite('data.xlsx',cell2mat(I),2); % 存放到data.xlsx的第2张sheet

return

end

beta=(rr'\*rr)/(r'\*r);

d=rr+beta\*d;

x=xx;

r=rr;

fprintf('\n x%d = ',k+1);

fprintf(' %10.6f',x);

D{k+1}=x;

xlswrite('data.xlsx',cell2mat(D),1); % 存放到data.xlsx的第1张sheet

end

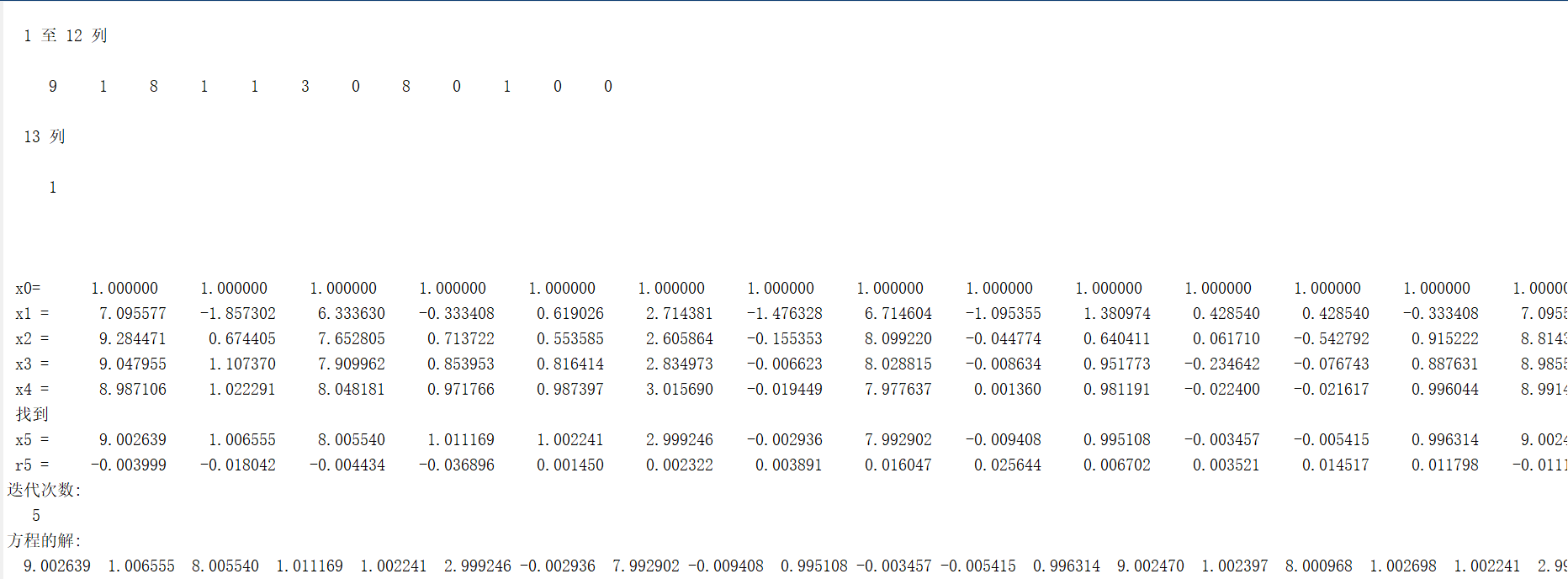
i = max\_i;

return

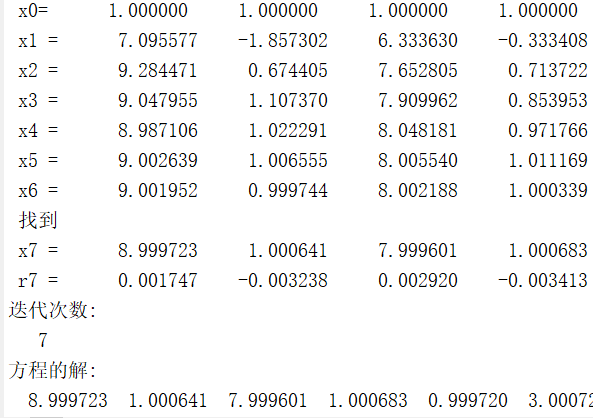
end

**实验结果：**

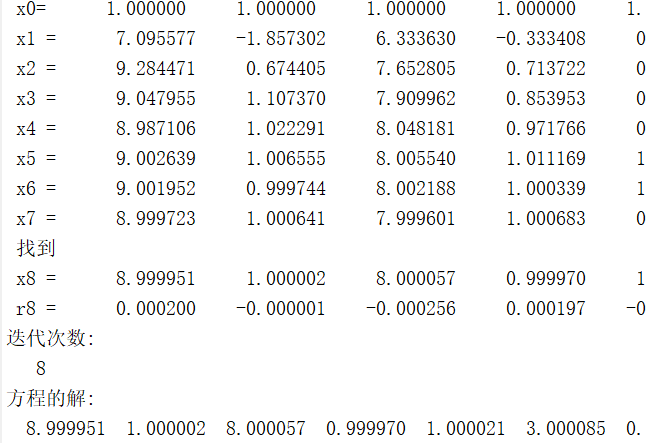
当要求精度为时，经过5次迭代解得x；



当要求精度为时，经过7次迭代解得x；



当要求精度为时，经过8次迭代解得x；



**实验心得：**

本次实验深刻地了解了这5种算法之间的区别，通过比较容易看出，共轭梯度法迭代收敛更快更佳。