

数值分析课程设计

****

**题 目 三对角线性代数方程组的共轭梯度法**

学生姓名

学 号 2018

学 院 数学与统计学院

专 业信息与计算科学（嵌入式）

**二Ｏ20 年 6 月 20日**

目 录

[1 设计背景 7](#_Toc43687491)

[1.1问题描述 7](#_Toc43687492)

[1.2 程序设计 7](#_Toc43687493)

[2 算法实现 8](#_Toc43687494)

[**3.1 共轭梯度法在不同精度要求下的迭代次数** 9](#_Toc43687509)

[**3.2共轭梯度法、Jacobi迭代法、Gauss-Seidel迭代法及最速下降法，在同一精度要求下的迭代次数** 12](#_Toc43687530)

[4 总结 14](#_Toc43687559)

[5 附录 15](#_Toc43687560)

[5.1 [3.1部分]代码 15](#_Toc43687561)

[5.2 [3.2部分]代码 17](#_Toc43687562)

[6 参考文献 20](#_Toc43687563)

三对角线性代数方程组的共轭梯度法

# 1 设计背景

1.1问题描述

对于如下三对角线性方程组

将其改写成更加直观的矩阵

设，再根据向量用生成右端（列）向量，然后使用共轭梯度法对方程组进行求解。

求出后，再与进行比较，得到，即该次迭代实验中的迭代精度要求。

最终输出共轭梯度法

1. 在不同精度要求下的迭代次数并比较，体会其收敛快慢；
2. 在同一精度要求下，比较共轭梯度法、Jacobi迭代法、Gauss-Seidel迭代法及最速下降法，求解所需的迭代次数，体会其收敛快慢。

1.2 程序设计

1. 设计程序，输入位数字，重复次，得到一个维的向量，（将）记为；
2. 输入维的三对角线性方程组的系数矩阵如题目所示；
3. 利用向量用生成右端（列）向量；
4. 在要求的迭代精度下，使用共轭梯度法求解得到，并输出达到目标精度所需要进行的迭代次数。
5. 最后通过比较各迭代法的迭代次数体会该迭代法的收敛快慢。

# 2 算法实现

利用diag函数等，输入三对角线性代数方程组的系数矩阵，及利用向量用生成右端（列）向量；

共轭梯度法部分，算法步骤如下：

1. 给出初始点及；

1. ；
2. 对于，有

* 1. ；

* 1. ；

* 1. ，

或;

* 1. 1. 若或，则输出，取作为的解；
     2. 否则，，.

# 3 迭代结果及分析

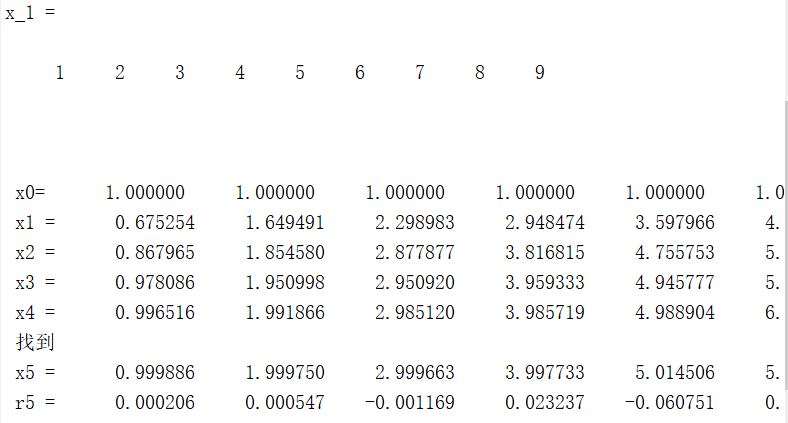
本次实验中，取，输入的9位数字始终为123456789。由此得到；迭代精度分别取值.

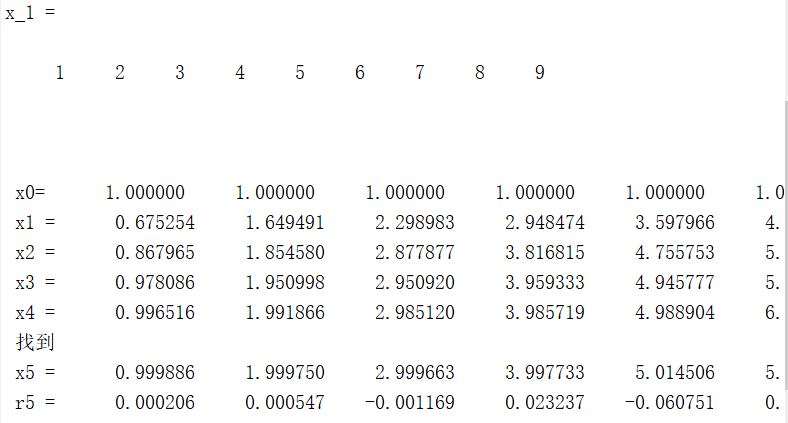
***P.S：在MATLAB程序中将记作了.***

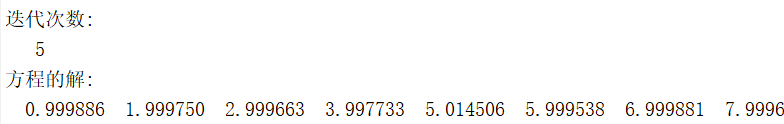
**3.1 共轭梯度法在不同精度要求下的迭代次数**

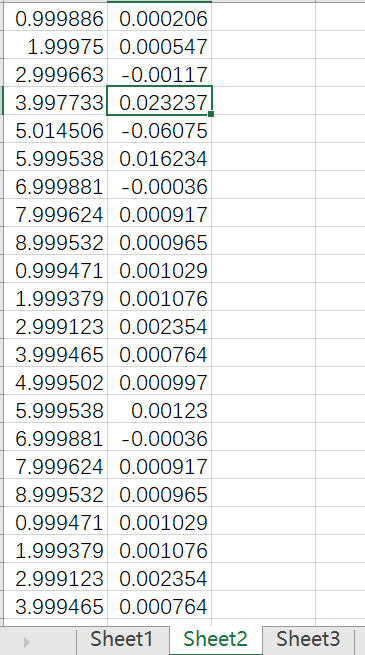
**迭代结果：**

* 1. **当迭代精度取时，经过5次迭代可求得满足迭代精度的解；**

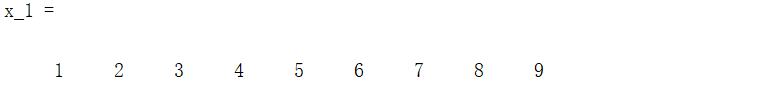


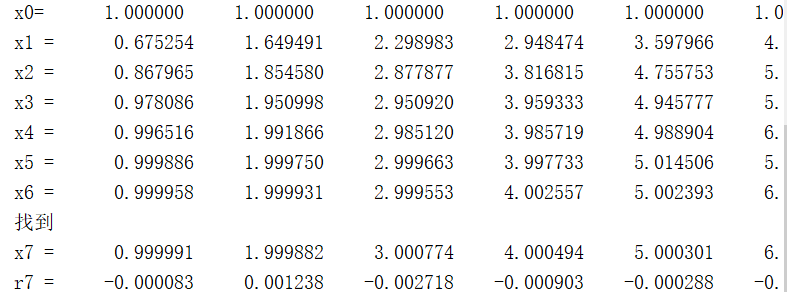


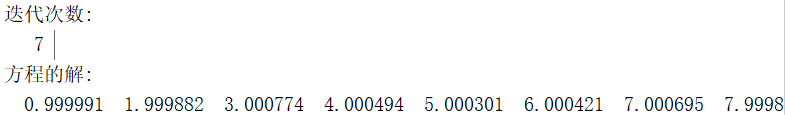


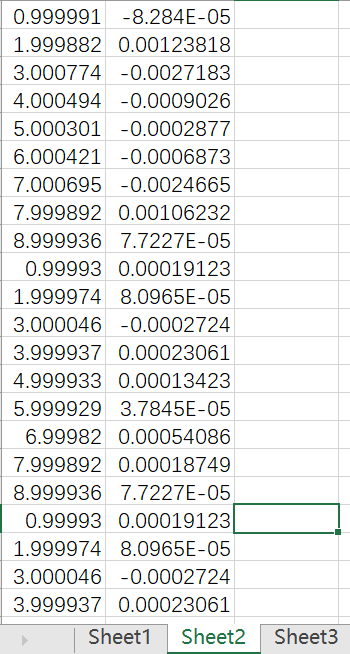


* 1. **当迭代精度取时，经过7次迭代可求得满足迭代精度的解；**

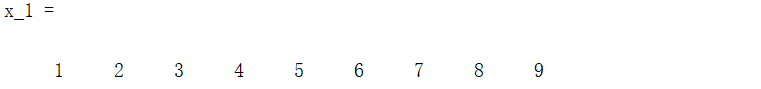


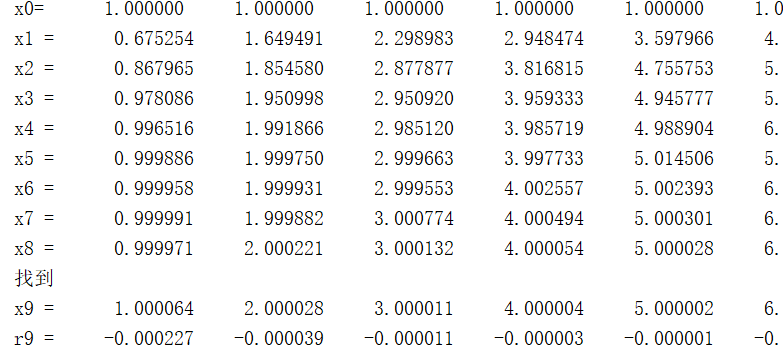


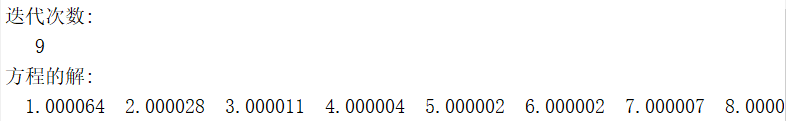


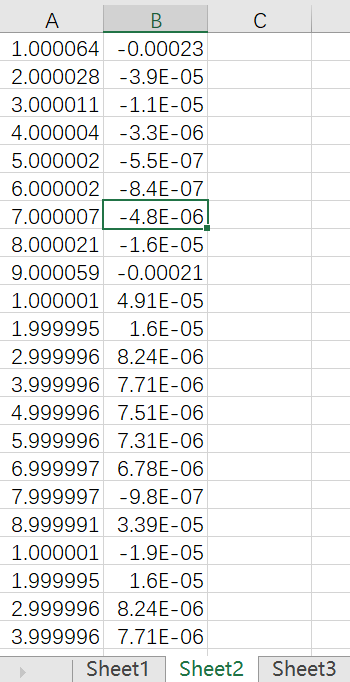


* 1. **当迭代精度取时，经过9次迭代可求得满足迭代精度的解；**









**分析：**

容易看出，共轭梯度法的迭代次数较少，收敛速度较快，在的精度范围内都具有较好的收敛情况。

另外，随着迭代精度的数值逐渐变小，精度提高，而迭代次数的增加并不明显。在的精度范围内，精度每提高，仅增加2次迭代次数。

因此，对于求解三对角线性代数方程组问题，共轭梯度法是不错的选择。

**3.2共轭梯度法、Jacobi迭代法、Gauss-Seidel迭代法及最速下降法，在同一精度要求下的迭代次数**

**迭代结果：**

* 1. **当迭代精度取时，**

|  |  |
| --- | --- |
| **共轭梯度法** |  |
| **Jacobi迭代法** |  |
| **Gauss-Seidel迭代法** |  |
| **最速下降法** |  |

* 1. **当迭代精度取时，**

|  |  |
| --- | --- |
| **共轭梯度法** |  |
| **Jacobi迭代法** |  |
| **Gauss-Seidel迭代法** |  |
| **最速下降法** |  |

* 1. **当迭代精度取时，**

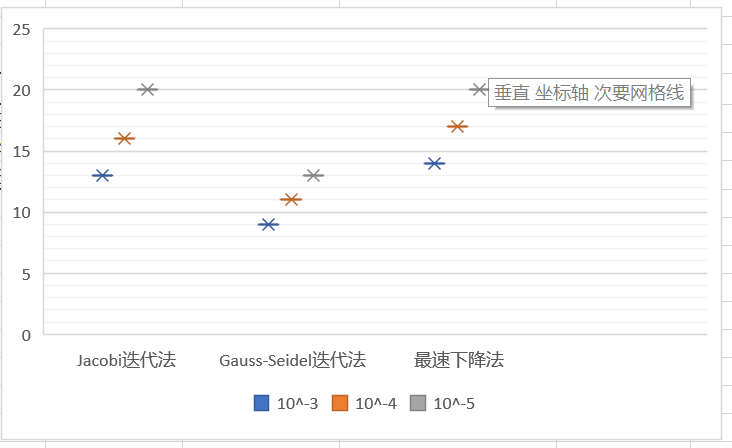
|  |  |
| --- | --- |
| **共轭梯度法** |  |
| **Jacobi迭代法** |  |
| **Gauss-Seidel迭代法** |  |
| **最速下降法** |  |

**分析：**

以上3个表格比较直观的表现出了4种迭代方法随着迭代精度的提高，迭代次数的增加情况。为更加直观，作出箱形图如下**图1：**

显然，在这4种迭代方法中，共轭梯度法的迭代次数是最少的，同时，其迭代次数的增速也是最慢的，与Gauss-Seidel迭代法的迭代次数增速持平。

**图1** 4种迭代方法随着迭代精度的提高，迭代次数的增加情况



因此，在的迭代精度范围内，对于三对角线性代数方程组的求解问题，共轭梯度法、Jacobi迭代法、Gauss-Seidel迭代法及最速下降法这4种迭代法之中，共轭梯度法是最佳的选择。

# 4 总结

本次课程设计中对于共轭梯度法设计了合理的实验并完成，由此对于共轭梯度法游轮更深刻的了解和认识。

本次课程设计

1. 探讨了在的迭代精度范围内，对于三对角线性代数方程组的求解问题，利用共轭梯度法求解的可行性及可能的效果。
2. 同时，探讨了在的迭代精度范围内，同一迭代精度要求下，共轭梯度法、Jacobi迭代法、Gauss-Seidel迭代法及最速下降法这4种迭代法，对于三对角线性代数方程组的求解，哪者迭代收敛的速度更快，因此在更高精度要求的迭代求解问题中，将有可能有更好的迭代效果。最终的优胜算法为梯度共轭法。

因此，在的迭代精度范围内，对于三对角线性代数方程组的求解问题，共轭梯度法、Jacobi迭代法、Gauss-Seidel迭代法及最速下降法这4种迭代法之中，共轭梯度法是最佳的选择。

此外，对于三对角线性代数方程组的求解问题，当对于所求解有一个较高的迭代精度要求时，共轭梯度法较另三种迭代法将可能会有更好的迭代效果。

# 5 附录

5.1 [3.1部分]代码

Data.m

%A是系数矩阵

%B是方程组的右端向量

clc

clear

s = 9;

m = 10; %数字重复次数(m>=10)

x\_1 = [];

for i=1:s

fprintf('请分别依次单位输入%d位数字:\n',s);

x = input('请输入：');

x\_1(end+1) = x;

end

x\_1 %完整数字

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); %数字重复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

eps = 0.001; % 迭代精度

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

f1.m

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

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))<= eps

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

Conjugate.m

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

max\_i=1000; %迭代次数上限

fprintf('\n');

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

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

5.2 [3.2部分]代码

Conjugate.m

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

max\_i=1000; %迭代次数上限

fprintf('\n');

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

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

jacobi.m

function r = jacobi(A,B,varargin)

sizeA=size(A);

sizev=size(varargin);

if sizev(2) == 0

eps = 0.00001; %精度epsilon

n = 1000;

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

elseif sizev(2) == 1

eps = varargin{1};

n = 1000;

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

elseif sizev(2) == 2

eps = varargin{1};

n = varargin{2};

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

elseif sizev(2) == 3

eps = 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))>eps) == 0

break;

end

end

r = x;

end

Gauss\_Seidel.m

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

sizeA=size(A);

sizev=size(varargin);

if sizev(2) == 0

eps = 0.001;

n = 1000;

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

elseif sizev(2) == 1

eps = varargin{1};

n = 1000;

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

elseif sizev(2) == 2

eps = varargin{1};

n = varargin{2};

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

elseif sizev(2) == 3

eps = 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))>eps) == 0

break;

end

end

r = x;

end

steepest.m

function [x,i] = steepest(A,B,x\_0,eps,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)>=eps

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

# 6 参考文献

1. MATLAB R2014官方手册
2. 近代优化方法，徐成贤，陈志平，李乃成编著.科学出版社，2002.
3. 最优化方法（第二版）.孙文瑜，徐成贤，朱德通编著.高等教育出版社，2010.
4. 数值分析，李庆扬，王能超，易大义编.清华大学出版社