**实验报告填写说明**

**（实验项目名称、实验项目类型必须与实验教学大纲保持一致）**

**1．实验环境**：

详细书写实验使用的硬件和软件环境。

**2．实验目的**：

根据实验教学大纲，写出实验的要求和目的。

**3．实验原理：**

简要说明本实验项目所涉及的理论知识。

**4．实验内容**：

这是实验报告极其重要的容。对于验证性验，要写清楚操作方法，需要经过哪几个步骤来实现其操作。对于设计性和综合性实验，还应写出设计思路和设计方法。对于创新性实验，还应注明其创新点。

**5．实验结论：**

根据实验过程中得到的结果，做出结论。

**6．实验总结：**

本次实验的收获、体会和建议。

**7．指导教师评语及成绩：**

指导教师依据学生的实际报告内容，给出本次实验报告的评价和成绩。

实验报告

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **1 实验环境**  硬件：Core i7-8750H  软件：MATLAB 2020a  **2 实验目的**  掌握程序语言的基本知识，能够编写简单程序  熟练掌握用Jacobi和Gauss-Seidel迭代法求线性方程组的问题  **3实验原理**  Jacobi迭代法和Gauss-Seidel迭代法原理，基本步骤，迭代格式  Jacobi迭代法公式简单，每迭代一次只需计算一次矩阵和向量乘法。在用计算机计算时，需要两组工作单元，以存储及。  Jacobi迭代法迭代格式    由Jacobi方法迭代公式可知，迭代的每一步计算过程，都是用的全部分量来计算的所以分量，显然在计算第i个分量时，已经计算出的最新分量没有被利用。从直观上看，最新计算出的分量可能比旧的分量要好些。因此，对这些最新计算出来的第k+1次近似的分量加以利用，就得到了所谓解方程组的Gauss-Seidel迭代法（简称G-S方法）：  ，    Gauss-Seidel迭代法迭代格式      **4实验内容**  方案一：  用Jacobi迭代法和Gauss-Seidel迭代法求解线性方程组    思考：如果选取不同初值，对迭代法解会有影响吗？  方案二：  用Jacobi迭代法和Gauss-Seidel迭代法求解线性方程组，要求精确到小数点后两位。（教材P86 第15（1）    思考：考虑两种迭代法的收敛性？观察两种迭代法的收敛性与收敛速度？观察比较并分析原因。  方案三：  用Jacobi迭代法和Gauss-Seidel迭代法求解线性方程组，要求精确到小数点后两位。    思考：观察两种迭代法的收敛性？观察比较并分析原因。如果是发散的，思考如何解决问题？  **5实验结论**  **方案一**  由于matlab数值的位数的影响，会导致迭代计算后的结果的精度高于常规，所以在单求迭代次数时会有些高，为正常状况，不影响实验分析结果。  方案一实验结果如下：  Jacobi迭代法 初值为(0,0,0)时 迭代次数为34  Gauss 迭代法 初值为(0,0,0)时 迭代次数为18  Jacobi迭代法 初值为(1,1,1)时 迭代次数为32  Gauss 迭代法 初值为(1,1,1)时 迭代次数为17  显然，Gauss迭代法在同等初值的情况下，迭代次数明显小于Jacobi迭代法，而在初值与结果更贴近时，会减少两种方法的迭代次数  **方案二**  方案二实验结果如下：  方案二 Jacobi 初值为[0,0,0] 精度到小数点后两位 迭代次数为7 迭代解为  x =  2.4260 3.5740 1.9260  方案二 Gauss 初值为[0,0,0] 精度到小数点后两位 迭代次数为3 迭代解为  x =  2.4257 3.5729 1.9260  与迭代解相对比，可见Jacobi迭代法相较下迭代的收敛性较低，而且迭代的次数也比Guass迭代法的高，收敛速度慢。  通过分析两个方法步骤，得出结论：Jacobi迭代法求解方程组系数没有用到本轮迭代已经解出的前若干系数，而Guass迭代法用到了本轮迭代已经求解出的前若干系数。  **方案三**  方案三实验结果如下：  方案三 Jacobi 初值为[0,0,0] 精度到小数点后两位 迭代次数为Inf 迭代解为  x =  '发散无结果'  方案三 Gauss 初值为[0,0,0] 精度到小数点后两位 迭代次数为Inf 迭代解为  x =  '发散无结果'  俩方法均为发散，通过计算发现俩方法得出的谱半径均大于1，故无法收敛。但当使用solve函数解决时，可以看到如下情景：  >> syms x1 x2 x3;  >> [x1 x2 x3] = solve(x1+5\*x2-3\*x3==2, 5\*x1-2\*x2+x3==4, 2\*x1+x2-5\*x3==-11)  x1 = 1 x2 = 2 x3 = 3  可以看出精确解确实存在，所以显然这两种方法都有其局限性。    **6实验总结（收获、体会和建议）**  普遍情况下，Gauss迭代法的收敛速度和收敛性都优于Jacobi迭代法，但在某种情况下，可能会出现Gauss迭代法的结果发散，而Jacobi迭代法收敛的情况，所以Jacobi迭代法也有其作用，也有可能出现均发散的情况，此时就要寻找新的方法解决问题。  **7指导教师评语及成绩** | | | | | |
| **评 语** | **评语等级** | | | | |
| **优** | **良** | **中** | **及格** | **不及格** |
| **实验方案设计的合理程度** |  |  |  |  |  |
| **实验结论的记录情况** |  |  |  |  |  |
| **实验总结情况** |  |  |  |  |  |
| **实验报告是否按时完成,书写是否规范（文字叙述，层次结构）** |  |  |  |  |  |
| **成 绩：**    **指导教师签名：**    **批阅日期：** | | | | | |

**附录1：源 程 序**

|  |
| --- |
| **main.m**  clc;clear  % 方案一  A = [10,-1,-2;  -1,10,-2;  -1,-1,5];  b = [7.2,8.3,4.2]';  k1 = Jacobi\_k(A,b,[0,0,0]',[1.1,1.2,1.3]');  fprintf('方案一 Jacobi 初值为[0,0,0] 迭代次数为%d\n',k1)  k1 = Gauss\_k(A,b,[0,0,0]',[1.1,1.2,1.3]');  fprintf('方案一 Gauss 初值为[0,0,0] 迭代次数为%d\n',k1)  k1 = Jacobi\_k(A,b,[1,1,1]',[1.1,1.2,1.3]');  fprintf('方案一 Jacobi 初值为[1,1,1] 迭代次数为%d\n',k1)  k1 = Gauss\_k(A,b,[1,1,1]',[1.1,1.2,1.3]');  fprintf('方案一 Gauss 初值为[1,1,1] 迭代次数为%d\n',k1)  % 方案二  A = [27,6,-1;  6,15,2;  1,1,54];  b = [85,72,110]';  [x\_J2,k2] = Jacobi(A,b,[0,0,0]',1.0e-2);  fprintf('方案二 Jacobi 初值为[0,0,0] 精度到小数点后两位 迭代次数为%d 迭代解为\n',k2)  x = x\_J2'  [x\_G2,k2] = Gauss(A,b,[0,0,0]',1.0e-2);  fprintf('方案二 Gauss 初值为[0,0,0] 精度到小数点后两位 迭代次数为%d 迭代解为\n',k2)  x = x\_G2'  % 方案三  A = [1,5,-3;  5,-2,1;  2,1,-5];  b = [2,4,-11]';  [x\_J3,k3] = Jacobi(A,b,[0,0,0]',1.0e-2);  fprintf('方案三 Jacobi 初值为[0,0,0] 精度到小数点后两位 迭代次数为%d 迭代解为\n',k3)  x = x\_J3'  [x\_G3,k3] = Gauss(A,b,[0,0,0]',1.0e-2);  fprintf('方案三 Gauss 初值为[0,0,0] 精度到小数点后两位 迭代次数为%d 迭代解为\n',k3)  x = x\_G3'  % 结果发散  **Jacobi.m**  function [x,n] = Jacobi(A,b,x0,eps)  % Jacobi迭代法，nargin在新版本中不支持  % if nargin == 3  % eps = 1.0e-6;  % elseif nargin < 3  % error  % return  % end  D = diag(diag(A)); % 对角矩阵  L = -tril(A,-1); % 下三角阵  U = -triu(A,1); % 上三角阵  B = D\(L+U);  R = max(abs(eig(B)));% 谱半径  f = D\b;  x = B\*x0+f;  n = 1; % 迭代次数  while norm(x-x0) >= eps  x0 = x;  x = B\*x0+f;  n = n+1;  end  if R > 1  n = inf;  x = ('发散无结果')';  end  **Jacobi\_k.m**  function n = Jacobi\_k(A,b,x0,x\_k)  % Jacobi迭代法，已知解析解，求迭代次数  % nargin在新版本中不支持  % if nargin == 3  % eps = 1.0e-6;  % elseif nargin < 3  % error  % return  % end  D = diag(diag(A)); % 对角矩阵  L = -tril(A,-1); % 下三角阵  U = -triu(A,1); % 上三角阵  B = D\(L+U);  R = max(abs(eig(B)));% 谱半径  f = D\b;  x = B\*x0+f;  n = 1; % 迭代次数  while x ~= x\_k  x0 = x;  x = B\*x0+f;  n = n+1;  end  if R > 1  n = inf;  x = ('发散无结果')';  end  **Guass.m**  function [x,n] = Gauss(A,b,x0,eps)  % Gauss迭代法，nargin在新版本中不支持  % if nargin == 3  % eps = 1.0e-6;  % elseif nargin < 3  % error  % return  % end  D = diag(diag(A)); % 对角矩阵  L = -tril(A,-1); % 下三角阵  U = -triu(A,1); % 上三角阵  G = (D-L)\U;  f = (D-L)\b;  R = max(abs(eig(G)));% 谱半径  x = G\*x0+f;  n = 1; % 迭代次数  while norm(x-x0) >= eps  x0 = x;  x = G\*x0+f;  n = n+1;  end  if R > 1  n = inf;  x = ('发散无结果')';  end  **Guass\_k.m**  function n = Gauss\_k(A,b,x0,x\_k)  % Gauss迭代法，已知解析解，求迭代次数  % nargin在新版本中不支持  % if nargin == 3  % eps = 1.0e-6;  % elseif nargin < 3  % error  % return  % end  D = diag(diag(A)); % 对角矩阵  L = -tril(A,-1); % 下三角阵  U = -triu(A,1); % 上三角阵  G = (D-L)\U;  f = (D-L)\b;  R = max(abs(eig(G)));% 谱半径  x = G\*x0+f;  n = 1; % 迭代次数  while x ~= x\_k  x0 = x;  x = G\*x0+f;  n = n+1;  end  if R > 1  n = inf;  x = ('发散无结果')';  end |