数学仿真创新设计大赛

比较 Gauss-Seidel 迭代法和共轭梯度法

系别：信息与计算科学

班级：04—1 班

姓名：李晓敏

数力系数学专业实验室

题目：

比较 Gauss-Seidel 迭代法和共轭梯度法

摘要：

随着科技的发展，人类在实践中所处理的数据也越来越大。在所解的线性方程组的维数也越来越多了，尤其在实际工业生产中所得到的线性方程组的系数矩阵通常都是稀疏矩阵，那么通常应用比较方便的 Gauss 消元法等方法就会失去作用，因此新的求解方法（Gauss-Seidel 迭代法和共轭梯度法等）也应运而生。这里主要通过在算法理论和具体实现方面比较 Gauss-Seidel 迭代法和共轭梯度法的优劣，以进一步了解稀疏矩阵的求解方法。

With the development of science and technology,the data human being deal with in practice becomes more and more large.In the linear equations we solve,the dimension are becoming more and more,and it usually can be sparse matrix especially in real industrial production ,that is means that the Gauss elimination of unknows which has been used conveniently may have no effect ,so the new solvement appears,such as Gauss-Seidel iterative method,conjugate gradient method and so on.Here we mainly compare the Gauss-Seidel iterative method and conjugate gradient method in its algorithm and calculation in the computer,in order to understand the solvement of the sparse matrix.

关键词：

Jacobi 迭代法（Jacobi iterative method）
Gauss-Seidel 迭代法（Gauss-Seidel iterative method）
最速下降法（most descent method）
共轭梯度法（conjugate gradient method）

随着计算技术的发展，计算机的存储量日益增大，计算速度也迅速提高，直接法（如 Gauss 消去法、平方根法等）在计算机上可以求解的线性方程组的规模也越来越大，但直接法大多数均需对系数矩阵 A 进行分解，因而一般不能保持 A 的稀疏性。而实际应用中，特别是偏微分方程的数值求解时，常常遇到的恰恰就是大型稀疏性方程的求解问题。因此寻求能够保证稀疏性的有效性算法就成为数值线性代数中一个非常重要的研究课题。

目前发展起来的求解稀疏线性方程组的方法主要有两类：

一类是稀疏直接法。稀疏直接法是直接法与某些稀疏矩阵技巧有机结合的产物，它充分利用所给线性方程组的系数矩阵的零元素的分布特点（即矩阵的结构），采用灵活的主元选取策略，使得分解出的矩阵因子尽可能地保持原有的稀疏性。

另一类是迭代法。迭代法是按照某种规划构造一个向量序列$\{x_k\}$，使其极限向量x_*是方程组$Ax=b$的精确解。因此，对迭代法来说一般有下面几个问题：

（1） 如何构造迭代序列？

（2） 构造的迭代序列是否收敛？在什么情况下收敛？

（3） 如果收敛，收敛的速度如何？我们应该给予量的刻画，用以比较各种迭代方法收敛的快慢。

（4） 因为计算总是有限次的，所以总要讨论近似解的误差估计和迭代过程的中断处理等问题，这又和舍入误差的分析有关。

一个方法是否有效要看得到具有某个精确度的近似解而付出的代价如何，通常以运算和存储量的要求为标志。在这个标准下，直接法在很多情况下比迭代法好，但是对于大型的稀疏方程来说，应用比较多的包括以 Gauss-Seidel 迭代法为代表的迭代法求解方法和用于求解无约束最优化问题的共轭梯度法。这里我们主要通过比较 Gauss-Seidel 迭代法和共轭梯度法，讨论一下二者在解决线性方程组的优与劣。

下面我们介绍一下 Gauss-Seidel 迭代法的思想：

首先，看一下 Jacobi 迭代法的推导过程。

考虑非奇异线性方程组$Ax=b$，令$A=D-L-U$，其中$D=diag(a_{11},a_{22},a_{33},\cdots,a_{nn})$

$$L=\begin{bmatrix} 0 & & & & \\ -a_{21} & 0 & & & \\ -a_{31} & -a_{32} & 0 & & \\ \vdots & \vdots & \vdots & \ddots & \\ -a_{n1} & -a_{n2} & \cdots & -a_{n,n-1} & 0 \end{bmatrix},\quad U=\begin{bmatrix} 0 & -a_{12} & -a_{13} & \cdots & -a_{1n} \\ & 0 & -a_{23} & \cdots & -a_{2n} \\ & & 0 & \cdots & \vdots \\ & & & \ddots & -a_{n-1,n} \\ & & & & 0 \end{bmatrix}$$

那么 $Ax=b$ 可以写成 $x=Bx+g$，其中 $B=D^{-1}(L+U), g=D^{-1}b$。

若给定初始向量 $x_0=(x_1^{(0)},x_2^{(0)},\cdots,x_n^{(0)})^T$，代入 $x=Bx+g$ 右端，计算得向量 x_1，即：

$$x_1=Bx_0+g \quad ， \quad x_2=Bx_1+g \quad ， \cdots \quad ， \quad x_{k+1}=Bx_k+g$$

当 $\| x_{k+1}-x_k \|<\varepsilon$ 时，表明 x_{k+1} 已经在指定精度范围内接近精确解 x_*，故可以将 x_{k+1} 作为线性方程组的近似解。

Gauss-Seidel 迭代法则是在 Jacobi 迭代法的基础上加以改进得来的，具体推导过程如下：

将 Jacobi 迭代矩阵 $B=D^{-1}(L+U)=D^{-1}L+D^{-1}U$，代入 $x_{k+1}=Bx_k+g$ 中，得如下表达式： $x_{k+1}=D^{-1}Lx_k+D^{-1}Ux_k+g$，对该迭代公式进行改进，得：

$$x_{k+1}=D^{-1}Lx_{k+1}+D^{-1}Ux_k+g \quad ，$$

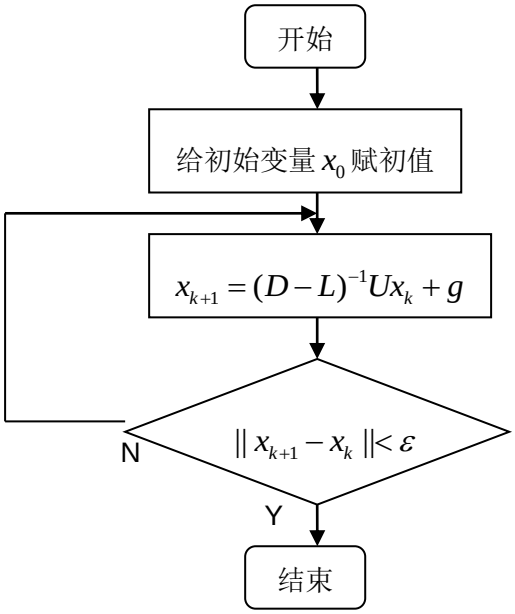
写成迭代分量表达式，得：

$$\begin{cases} x_1^{(k+1)} = & & b_{12}x_2^{(k)} + b_{13}x_3^{(k)} + \cdots + b_{1n}x_n^{(k)} + g_1 \\ x_2^{(k+1)} = b_{21}x_1^{(k+1)} & & + b_{23}x_3^{(k)} + \cdots + b_{2n}x_n^{(k)} + g_2 \\ x_3^{(k+1)} = b_{31}x_1^{(k+1)} + b_{32}x_2^{(k)} & & + \cdots + b_{3n}x_n^{(k)} + g_3 \\ \vdots \quad\quad \vdots \quad\quad\quad \vdots & & \quad\quad\quad\quad\quad\quad\quad\quad \vdots \quad\quad \vdots \\ x_n^{(k+1)} = b_{n1}x_1^{(k+1)} + b_{n2}x_2^{(k)} + b_{n3}x_3^{(k)} + \cdots & & \quad\quad\quad\quad\quad\quad\quad\quad + g_n \end{cases}$$

对迭代公式 $x_{k+1}=D^{-1}Lx_{k+1}+D^{-1}Ux_k+g$，进行变形，得：

$$x_{k+1}=(D-L)^{-1}Ux_k+g$$

根据上述算法，我们可以绘制 Gauss-Seidel 迭代法的流程图，如下：

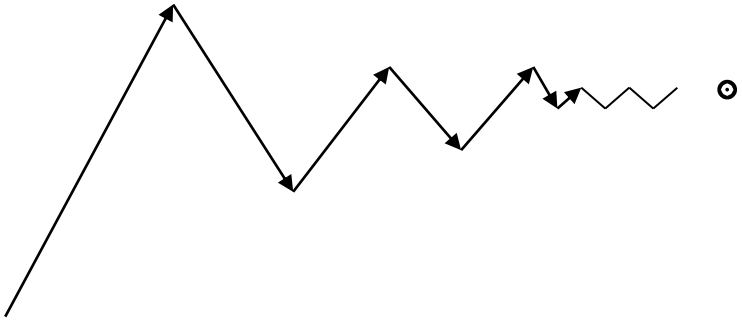


具体 MATLAB 程序实现代码(1)如下：

```
%Gauss-Seidel 迭代法
x=[];
%请输入初始向量 X0 于 X(:,1)中
x(:,1)=[4 -2 6 1]';
%请输入方程组系数矩阵 A
A=[16 4 8 4
    4 10 8 4
    8 8 12 10
    4 4 10 12];
%请输入常数向量 b
b=[32 26 38 30]';
L=-tril(A,-1)
U=-triu(A,1)
D=A+L+U
x(:,2)=inv(D-L)*U*x(:,1)+inv(D-L)*b
n=2;
while(x(:,n)-x(:,n-1)>0.00001|x(:,n-1)-x(:,n)>0.00001)
    x(:,n+1)=inv(D-L)*U*x(:,n)+inv(D-L)*b;
    n=n+1;
end
x=x(:,n)
```

接下来，我们来看一下共轭梯度法的思想：

首先，还是先了解一下最速下降法，算法思想如下简图：



考虑线性方程组 $Ax=b$ 的求解问题，其中 A 是给定的 n 阶对称正定矩阵，b 是给定 n 维向量，x 是待求的 n 维向量。为此，我们定义二次函数

$$\varphi(x)=x^TAx-2b^Tx=\sum_{i,j=1}^{n}a_{ij}x_ix_j-2\sum_{i=1}^{n}b_ix_i \quad ,$$

则其梯度为：

$$\frac{\partial\varphi(x)}{\partial x_i}=2(a_{i1}x_1+a_{i2}x_2+\cdots+a_{in}x_n)-2b_i \quad ,$$

即：$\nabla\varphi(x)=2(Ax-b)$，

要使 $\varphi(x)$ 减小的速度达到最大，只需要让 $\varphi(x)$ 沿着方向 $d=-\nabla\varphi(x)$ 即可。

因此，可以构造迭代公式为：

$x_{k+1}=x_k+\lambda_k d_k$，$d_k$ 为方向 d 的分量，

然后通过求 $\varphi(x_k+\lambda_k d_k)$ 的最小值即可确定

$$\lambda_k=\frac{-\nabla\varphi(x_k)^Td_k}{d_k{}^TAd_k},$$

给定初值 x_0，然后通过有限次迭代，当满足停机条件 $\|x_{k+1}-x_k\|<\varepsilon$ 即可求得方程的解 x_* 。

而共轭梯度法是在最速下降法的基础上通过改进得来的，在上述最速下降法中，通过第一次迭代，求得 x_1、d_1、λ_1，在第二次迭代时，方向不再沿着负梯度

方向$-\nabla\varphi(x_k)$，而是由迭代公式

$$d_{k+1}=-\nabla\varphi(x_{k+1})+\beta_k d_k$$

确定，其中

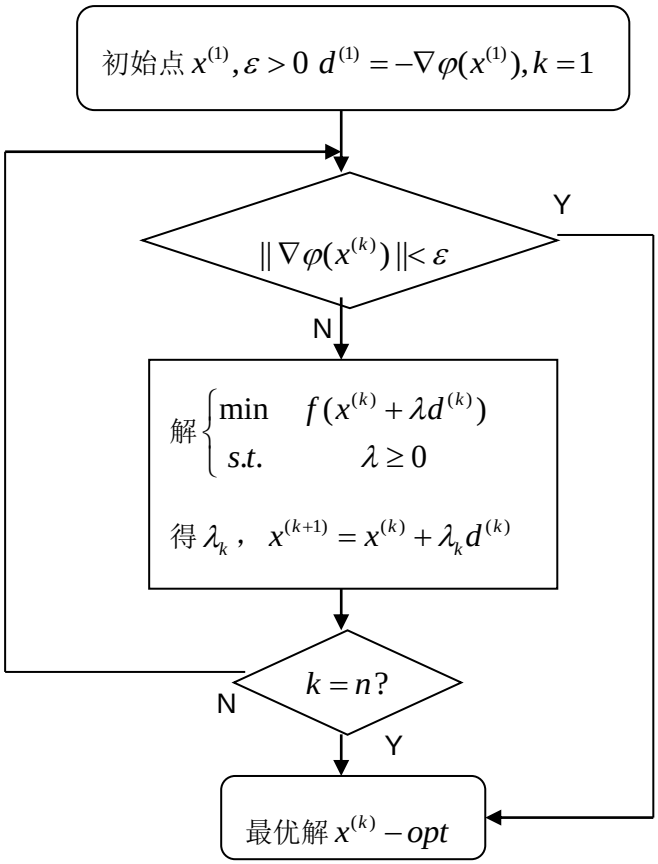
$$\beta_k=\frac{\xi_0}{\eta_0}\text{ 由 }\varphi(x_k+\xi(-\nabla\varphi(x_k))+\eta d_{k-1})$$

取最小值时所确定的ξ_0，η_0求得，即：

$$\beta_k=\frac{-\nabla\varphi(x_{k+1})^T d_{k+1}}{d_k{}^T d_k}\text{。}$$

根据前面的迭代公式$x_{k+1}=x_k+\lambda_k d_k$，用上述的公式进行有限次的迭代，即可求得最优解$x_*$。

根据上述共轭迭代法，做算法流程图：



具体 MATLAB 程序实现代码(2)如下：

```
%共轭梯度法
%初始化 x0 向量
x0=[4 -2 6 1]';
%请输入系数矩阵 A
A=[16 4 8 4
    4 10 8 4
    8 8 12 10
    4 4 10 12];
n=size(A);
%请输入常数向量 b
b=[32 26 38 30]';
r=[];
r(:,1)=b-A*x0
p=[];
k=1;
while r(k)>0.00001
    k=k+1;
    if(k>n)
        break;
    end
    x(:,1)=x0;
    if(k==2)
        p(:,1)=r(:,1);
    else
        R(k-1)=r(:,k-1)'*r(:,k-1);
        R(k-2)=r(:,k-2)'*r(:,k-2);
        beta(k-2)=R(k-1)/R(k-2);
        p(:,k-1)=r(:,k-1)+beta(k-2)*p(:,k-2);
    end
    R(k-1)=r(:,k-1)'*r(:,k-1);
    P(k-1)=p(:,k-1)'*A*p(:,k-1);
    alph(k-1)=R(k-1)/P(k-1);
    x(:,k)=x(:,k-1)+alph(k-1)*p(:,k-1);
    r(:,k)=r(:,k-1)-alph(k-1)*A*p(:,k-1);
end
k
if(k<=n)
    X=x(:,k)
else
    X=x(:,n)
end
```

下面让我们根据上述算法求解线性方程组

$$\begin{cases} 16x_1 + 4x_2 + 8x_3 + 4x_4 = 32 \\ 4x_1 + 10x_2 + 8x_3 + 4x_4 = 26 \\ 8x_1 + 8x_2 + 12x_3 + 10x_4 = 38 \\ 4x_1 + 4x_2 + 10x_3 + 12x_4 = 30 \end{cases}$$

写成矩阵形式为：$Ax = b$ ，其中 $A = \begin{bmatrix} 16 & 4 & 8 & 4 \\ 4 & 10 & 8 & 4 \\ 8 & 8 & 12 & 10 \\ 4 & 4 & 10 & 12 \end{bmatrix}$ ， $b = \begin{bmatrix} 32 \\ 26 \\ 38 \\ 30 \end{bmatrix}$

我们取初始向量 $x_0 = (4, -2, 6, 1)$ ，

用 Gauss-Seidel 迭代法求解，在 MATLAB 软件上运行代码(1),得到运行结果为：

$$x = \begin{pmatrix} 0.9998 \\ 0.9995 \\ 1.0009 \\ 0.9994 \end{pmatrix}$$

用共轭梯度法求解，在 MATLAB 软件上运行代码(2),得到运行结果为：

$$x = \begin{pmatrix} 0.9998 \\ 0.9995 \\ 1.0009 \\ 0.9994 \end{pmatrix}$$

由此可见，在计算机上有限的精度范围内，二者的运行结果是一样的。但实际上，我们从这两个算法中就可以看出， Gauss-Seidel 迭代法是一种迭代算法，是经过多次的迭代,使计算结果达到所要求的精度就可以了,具有一定的误差性；而共轭梯度法虽然表面上看好像是迭代法，可实际上却不是，由其算法的特点，只要经过有限次的迭代就可以得到所要的精确的结果。因此，从理论上来说，共轭梯度法要比 Gauss-Seidel 迭代法更好。但是，由于这两种算法在计算机上应用比较广，所以在计算机上来说，二者是没有太大的区别的。

通过上面对 Gauss-Seidel 迭代法和共轭梯度法从理论算法上和实际计算的比较，我们可以发现这两种算法的各自的优缺点和应用领域的区别。尽管现在在有限的精度范围内，这两种算法的区别不是很大，但我相信随着计算机的发展，计算机的计算精度也会变得更加精确，那时候，二者的区别也就会体现出来了，那么其应用范围也会不同的。

参考文献：

1、运筹学与最优化方法—吴祈宗—机械工业出版社
2、数值线性代数—徐树方、高立、张平文—北京大学出版社