线性方程组求解——Jacobi 迭代法

湘潭大学, 数学与计算科学学院, 21 级王艺博

一、Jacobi 迭代法

$n=3$ 时

$$A=\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}, \quad b=\begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix},$$

Jacobi 公式为

$$x_1^{(k+1)}=\frac{b_1-a_{12}x_2^{(k)}-a_{13}x_3^{(k)}}{a_{11}}$$
$$x_2^{(k+1)}=\frac{b_2-a_{21}x_1^{(k)}-a_{23}x_3^{(k)}}{a_{22}}$$
$$x_3^{(k+1)}=\frac{b_3-a_{31}x_1^{(k)}-a_{32}x_2^{(k)}}{a_{33}}$$

由公式可以看出，每一次迭代的各个分量都是独立计算的，这也是为什么 Jacobi 迭代可以用于并行计算。

或等价的，将 A 分解为 $A=D-L-U$，其中

$$D=diag(a_{11},a_{22},a_{33}),$$

$$L=-\begin{bmatrix} 0 & 0 & 0 \\ a_{21} & 0 & 0 \\ a_{31} & a_{32} & 0 \end{bmatrix}, \quad U=-\begin{bmatrix} 0 & a_{12} & a_{13} \\ 0 & 0 & a_{23} \\ 0 & 0 & 0 \end{bmatrix}.$$

$$(D-L-U)x=b$$
$$Dx=b+(L+U)x$$
$$x=D^{-1}(b+(L+U)x)$$

得 Jacobi 公式

$$x^{(k+1)}=D^{-1}(b+(L+U)x^{(k)})$$

写成矩阵的形式

$$\begin{bmatrix} x_1^{(k+1)} \\ x_2^{(k+1)} \\ x_3^{(k+1)} \end{bmatrix} = \begin{bmatrix} \frac{1}{a_{11}} & & \\ & \frac{1}{a_{22}} & \\ & & \frac{1}{a_{33}} \end{bmatrix} \left(\begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} - \begin{bmatrix} & a_{12} & a_{13} \\ a_{21} & & a_{23} \\ a_{31} & a_{32} & \end{bmatrix} \begin{bmatrix} x_1^{(k)} \\ x_2^{(k)} \\ x_3^{(k)} \end{bmatrix} \right)$$

下面是其一般形式下的算法

二、算法

♡ Jacobi 迭代法

主要思路 输入: $A, b, x^{(0)}$, 输出 x

$$\begin{aligned} x^{(0)} &= \text{ initial vector} \\ x^{(k+1)} &= D^{-1}(b + (L+U)x^{(k)}) \end{aligned}$$

添加一些限制

- 容许误差 e_tol,
- 最大迭代步 N.

当残差 <e_tol 或迭代步数 $\geq N$ 时，都会停止迭代，输出结果

实现步骤

- 步骤 1: $k = 0, x = x^{(0)}$;
- 步骤 2: 计算残差 $r = \|b - Ax\|$,
  - 如果残差 r >e_tol 且 $k < N$，转步骤 3;
  - 否则, 转步骤 5;
- 步骤 3: 更新解向量

$$x = D^{-1}(b + (L+U)x^{(0)})$$

- 步骤 4： $x0 = x, k = k + 1$，转步骤 2;
- 步骤 5: 输出 x.

算法 1 线性方程组的 Jacobi 迭代法

输入：系数矩阵 A, 右端向量 b, 初值 $x0$, 容许误差 e_tol, 最大迭代步 N;
输出：数值解 x
$k = 0, x = x0;$
计算残差 $r = \|b - Ax\|;$
while $r >$e_tol & $k < N$ **do**
  $x = D^{-1}(b + (L + U)x0)$
  $x0 = x;$
  $r = \|b - Ax\|;$
  $k = k + 1;$
end while
if $k > N$ **then**
  输出: 算法超出最大迭代次数;
else
  输出: 迭代次数 k;
end if
返回 x;

三、北太天元源程序

```
function [x,k,r] = myJacobi(A,b,x0,e_tol,N)
% Jacobi迭代法解线性方程组
% Input: A, b(列向量), x0(初始值)
%       e_tol: error tolerant
%       N: 限制迭代次数小于 N 次
% Output: x , k(迭代次数),r:残差
%  Version:         1.0
%  last modified:   01/27/2024
   n = length(b);k = 0; x=zeros(n,N); % 记录每一次迭代的结果, 方便后续作误差分析
   x(:,1)=x0;
   L = -tril(A,-1); U = -triu(A,1); D = diag(diag(A));
   r = norm(b - A*x,2);
   while r > e_tol && k < N
      x(:,k+2) = inv(D)*(b+(L+U)*x(:,k+1));
      r = norm(b - A*x(:,k+2),2); % 残差
      k = k+1;
   end
   x = x(:,2:k+1); % x取迭代时的结果
   if k>N
      fprintf('迭代超出最大迭代次数');
   else
      fprintf('迭代次数=%i\n',k);
   end
end
```

将上述代码保存为 `myJacobi.m` 文件。

四、数值算例

例 1 利用 *Jacobi* 迭代法求解线性方程组

$$\begin{bmatrix} 10 & -1 & 2 & 0 \\ -1 & 11 & -1 & 3 \\ 2 & -1 & 10 & -1 \\ 0 & 3 & -1 & 8 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 6 \\ 25 \\ -11 \\ 15 \end{bmatrix}$$

取初值为

$$x^{(0)} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

容许误差及最大迭代步分别为 `e_tol=1.0e-08, N=100`。

调用函数 `[x,k,r] = myJacobi(A,b,x0,e_tol,N)`，进行实现

容易验证，线性方程组的真解为

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

在计算过程中，我们同时将每一迭代步求得的解向量 $x^{(k)}$，并计算残差 $r^{(k)} = \|b - Ax^{(k)}\|$ 和误差向量 $e^{(k)} = |x - x^{(k)}|$，如图1-3 所示。从图1可以看出，随着迭代步的增加，数值解向量的每一个分量均收敛到真解。图2显示方程的残差 $r^{(k)}$ 快速收敛到 0，且开始几个迭代步残差下降的非常快。类似的，图3显示误差 $e^{(k)}$ 快速收敛到 0。

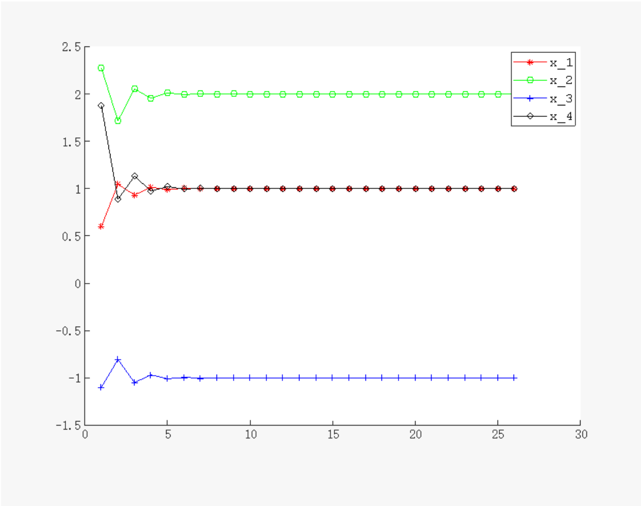

图 1　数值解

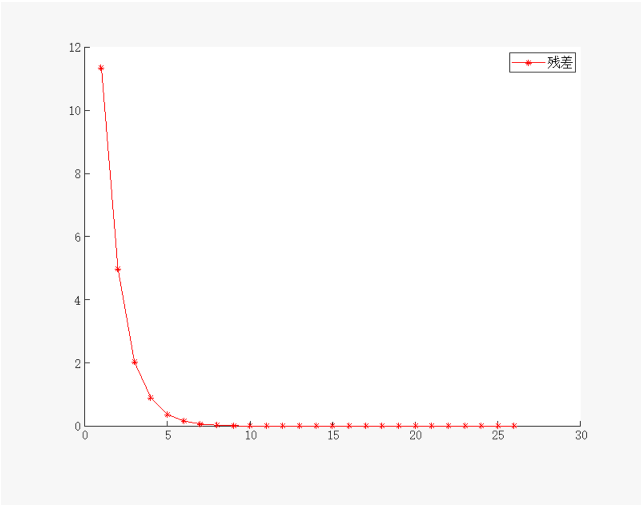

图 2　残差变化

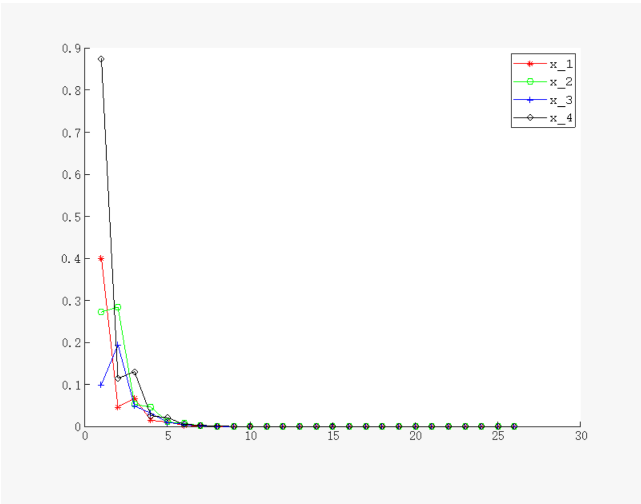

图 3　误差

例 2

$$\begin{bmatrix} 3 & -1 & 0 & 0 & 0 & \frac{1}{2} \\ -1 & 3 & -1 & 0 & \frac{1}{2} & 0 \\ 0 & -1 & 3 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & 0 \\ 0 & \frac{1}{2} & 0 & -1 & 3 & -1 \\ \frac{1}{2} & 0 & 0 & 0 & -1 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} \frac{5}{2} \\ \frac{3}{2} \\ 1 \\ 1 \\ \frac{3}{2} \\ \frac{5}{2} \end{bmatrix}.$$

使用 *Gauss* 消去法与 *Jacobi* 迭代法分别对其进行求解，并观察有何差异。

试着得到在更高阶数下的结果。(x 的真解为 $[1,1,1,1,1,1]^T$)

稀疏矩阵的构造:

```
function [A,b,x_sp] = setup_Sparse1(n)
  % 定义一个 n 阶的稀疏矩阵， 真解全为1
  % Input: n
  % Output: A,b
  %
  %  Version:         1.0
  %  last modified:   09/28/2023
  A = zeros(n,n);b = ones(n,1) * 1.5;
  b([1,n],1) = 2.5; b([n/2,n/2 + 1],1) = 1;
  x_sp = ones(n,1); % 真解
  for i = 1:1:n
      A(i,n+1-i) = 1/2;
      A(i,i) = 3;
  end
  for i =1:1:n-1
      A(i,i+1)= -1;A(i+1,i)= -1;
  end
end
```

将上述代码保存为 setup_Sparse1.m 文件。

容许误差及最大迭代步分别为 e_tol=1.0e-08，N=100。

经过实现后，结果如下:

阶数为 6 时，得到:

- Jacobi 迭代次数为 33 次，Jacobi 耗时 0.254356 秒。$r = 8.383869485405770e - 09$
- Gauss 列主元耗时 0.246983 秒。$r = 0$

阶数为 50 时，得到:

- Jacobi 迭代次数为 84 次，Jacobi 耗时 0.252936 秒。$r = 8.506205291756777e - 09$
- Gauss 列主元耗时 0.499854 秒。$r = 5.197930934883577e - 15$

阶数为 100 时，得到:

- Jacobi 迭代次数为 84 次，Jacobi 耗时 1.017717 秒。$r = 9.969971572640032e - 09$
- Gauss 列主元耗时 1.121281 秒。$r = 7.077617359078848e - 15$

阶数为 500 时，得到:

- Jacobi 迭代次数为 84 次，Jacobi 耗时 20.812204 秒。$r = 9.964771950043455e - 09$
- Gauss 列主元耗时 21.329216 秒。$r = 8.862333997095065e - 15$

阶数为 1000 时，得到:

- Jacobi 迭代次数为 84 次，Jacobi 耗时 34.252031 秒。$r = 9.964771950894769e - 09$
- Gauss 列主元耗时 89.985333 秒。$r = 1.072960336432300e - 14$

可以看出，随着稀疏矩阵规模的不断增大，迭代法相比直接法的优势越来越明显。直接法即能够在有限步内完成计算，随着阶数增大，(gauss 消去法运算量 $O(n^3)$)，所需步数，指数级增加。而 Jacobi 迭代法，只需把精度控制在所需范围内即算完成任务，能够节约很多时间。

例 1 和例 2 的代码如下

```
%% Jacobi test
% time :    1/28/2023
clc;clear all,format long;
addpath("../base","../Gauss消去法解线性方程组","../Jacobi迭代法解线性方程组") % 加载函数文件
N = 100; e_tol = 1e-8;
%% example 1
A=[10 -1 2 0; -1 11 -1 3; 2 -1 10 -1; 0 3 -1 8];
b=[6; 25; -11; 15];
x0=[0; 0; 0; 0];
t1 =tic;
   [x11,k1] = myJacobi(A,b,x0,e_tol,N)
toc(t1);
t2 = tic;
   [x12] = gsem_column(A,b)
toc(t2);
% 作图查看误差变化
   x_exact=[1;2;-1;1]; %真解
   n = length(b);
   error=zeros(n,k1);% 每个分量的误差
      error = abs(x_exact - x11)
   res =zeros(1,k1); % 残差
   for i=1:1:k1
      res(i) = norm(b-A*x11(:,i),2);
   end
   % 数值解
      figure(1);
```

```
    plot(1:k1,x11(1,:),'-*r',1:k1,x11(2,:),'-og', 1:k1,x11(3,:),'-+b',1:k1,x11(4,:),'-dk');
    legend('x_1','x_2','x_3','x_4');
  % 残差变化
    figure(2);
    plot(1:k1,res,'-*r');
    legend('残差');
  % 误差
    figure(3);
    plot(1:k1,error(1,:),'-*r',1:k1,error(2,:),'-og',
        1:k1,error(3,:),'-+b',1:k1,error(4,:),'-dk');
    legend('x_1','x_2','x_3','x_4');


%% 测试 消去法 与 迭代法 在处理稀疏矩阵问题上的差距
n = 6; %
[A,b,x]=setup_Sparse1(n);
x0 = zeros(n,1);

t1 =tic;
[x11,k1,r1] = myJacobi(A,b,x0,e_tol,N);
toc(t1);

t2 = tic;
[x12] = gsem_column(A,b);
toc(t2);
r2 = norm(b-A*x12);
```