数值线性代数与算法

第六章 解线性方程组的直接法




本章考虑如下 n 阶线性方程组

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{6.1}$$

的直接法, 其中 $\boldsymbol{A} = (a_{ij})$ 称为方程组的系数矩阵, $\boldsymbol{b} = (b_1, b_2, \cdots, b_n)^{\mathrm{T}}$ 称为方程组的右端项, 向量 $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)^{\mathrm{T}}$ 为所求的解. 若系数矩阵 $\boldsymbol{A}$ 是非奇异的, 则线性方程组 (6.1) 存在唯一解.

所谓直接法, 是指在没有舍入误差的情况下经过有限次运算可求得方程组的精确解的方法. 本章主要介绍求解线性方程组 (6.1) 最基本的直接法—Gauss 消去法和 LU 分解法以及舍入误差分析等.

Back

Close

§6.1 Gauss 消去法

§6.1.1 顺序 Gauss 消去法

Gauss 消去法的基本思想是: 首先使用初等行变换将方程组转化为一个同解的上三角形方程组 (称为消元), 再通过回代法求解该三角形方程组 (称为回代). 按行原先的位置进行消元的 Gauss 消去法称为顺序 Gauss 消去法.

例 6.1 用顺序 Gauss 消去法解线性方程组

$$\begin{cases} x_1 + x_2 + x_3 + x_4 = 6, \\ -x_1 + 2x_2 - 3x_3 + x_4 = -2, \\ 3x_1 - 3x_2 + 6x_3 - 2x_4 = 7, \\ -4x_1 + 5x_2 + 2x_3 - 3x_4 = 7. \end{cases}$$

解 (1) 消元过程:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 6 \\ -1 & 2 & -3 & 1 & -2 \\ 3 & -3 & 6 & -2 & 7 \\ -4 & 5 & 2 & -3 & 7 \end{bmatrix} \xrightarrow[r_4+4r_1]{\substack{r_2+r_1 \\ r_3-3r_1}} \begin{bmatrix} 1 & 1 & 1 & 1 & 6 \\ 0 & 3 & -2 & 2 & 4 \\ 0 & -6 & 3 & -5 & -11 \\ 0 & 9 & 6 & 1 & 31 \end{bmatrix} \xrightarrow[r_4-3r_2]{r_3+2r_2} \begin{bmatrix} 1 & 1 & 1 & 1 & 6 \\ 0 & 3 & -2 & 2 & 4 \\ 0 & 0 & -1 & -1 & -3 \\ 0 & 0 & 12 & -5 & 19 \end{bmatrix} \xrightarrow{r_4+12r_3} \begin{bmatrix} 1 & 1 & 1 & 1 & 6 \\ 0 & 3 & -2 & 2 & 4 \\ 0 & 0 & -1 & -1 & -3 \\ 0 & 0 & 0 & -17 & -17 \end{bmatrix}.$$

(2) 回代过程:

$$\begin{cases} x_1+x_2+x_3+x_4=6, \\ 3x_2-2x_3+2x_4=4, \\ -x_3-x_4=-3, \\ -17x_4=-17. \end{cases} \Rightarrow \begin{cases} x_4=1, \\ x_3=3-x_4=2, \\ x_2=(4+2x_3-2x_4)/3=2, \\ x_1=6-x_2-x_3-x_4=1. \end{cases}$$

故原方程组的解是: $x_1=1,\ x_2=2,\ x_3=2,\ x_4=1.$

对于一般线性方程组, 使用顺序 Gauss 消去法求解

$$\begin{cases} a_{11}^{(1)}x_1+a_{12}^{(1)}x_2+\cdots+a_{1n}^{(1)}x_n=b_1^{(1)}, \\ a_{21}^{(1)}x_1+a_{22}^{(1)}x_2+\cdots+a_{2n}^{(1)}x_n=b_2^{(1)}, \\ \vdots \\ a_{n1}^{(1)}x_1+a_{n2}^{(1)}x_2+\cdots+a_{1n}^{(1)}x_n=b_n^{(1)}. \end{cases} \tag{6.2}$$

(1) 消元过程:

$$\begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & a_{1n}^{(1)} & b_1^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} & a_{23}^{(1)} & \cdots & a_{2n}^{(1)} & b_2^{(1)} \\ a_{31}^{(1)} & a_{32}^{(1)} & a_{33}^{(1)} & \cdots & a_{3n}^{(1)} & b_3^{(1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1}^{(1)} & a_{n2}^{(1)} & a_{n3}^{(1)} & \cdots & a_{nn}^{(1)} & b_n^{(1)} \end{bmatrix} \rightarrow \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & a_{1n}^{(1)} & b_1^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2n}^{(2)} & b_2^{(2)} \\ 0 & a_{32}^{(2)} & a_{33}^{(2)} & \cdots & a_{3n}^{(2)} & b_3^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & a_{n2}^{(2)} & a_{n3}^{(2)} & \cdots & a_{nn}^{(2)} & b_n^{(2)} \end{bmatrix}$$

$$\rightarrow \cdots \rightarrow \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & a_{1n}^{(1)} & b_1^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2n}^{(2)} & b_2^{(2)} \\ 0 & 0 & a_{33}^{(3)} & \cdots & a_{3n}^{(3)} & b_3^{(3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & a_{nn}^{(n)} & b_n^{(n)} \end{bmatrix},$$

Back

Close

式中:

$$a_{ij}^{(2)} = a_{ij}^{(1)} - m_{i1}a_{1j}^{(1)}, \quad b_i^{(2)} = b_i^{(1)} - m_{i1}b_1^{(1)}, \quad m_{i1} = \frac{a_{i1}^{(1)}}{a_{11}^{(1)}}, \quad i, j = 2, \cdots, n.$$

一般地, 有

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - m_{ik}a_{kj}^{(k)}, \quad b_i^{(k+1)} = b_i^{(k)} - m_{ik}b_k^{(k)}, \quad m_{ik} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}},$$

$$i, j = k+1, \cdots, n; \ k = 1, \cdots, n-1. \tag{6.3}$$

Back

Close

(2) 回代过程:

$$\begin{cases} a_{11}^{(1)}x_1 + a_{12}^{(1)}x_2 + \cdots + a_{1n}^{(1)}x_n = b_1^{(1)}, \\ \qquad\quad a_{22}^{(2)}x_2 + \cdots + a_{2n}^{(2)}x_n = b_2^{(2)}, \\ \qquad\qquad\qquad\qquad\qquad\quad \vdots \\ \qquad\qquad\qquad\qquad\quad a_{nn}^{(n)}x_n = b_n^{(n)}, \end{cases}$$

$$\Longrightarrow \begin{cases} x_n = b_n^{(n)}/a_{nn}^{(n)}, \\ x_k = \left(b_k^{(k)} - \sum\limits_{j=k+1}^{n} a_{kj}^{(k)}x_j\right)/a_{kk}^{(k)}, \quad k = n-1, \cdots, 2, 1. \end{cases} \tag{6.4}$$

在此基础上, 得到顺序 Gauss 消去法的具体算法步骤.

算法 6.1 (顺序 Gauss 消去法)

步1, 输入系数矩阵 $\boldsymbol{A}$, 右端项 $\boldsymbol{b}$, 置 $k := 1$.

步2, 消元: 对 $k = 1, \cdots, n-1$, 计算

$$m_{ik} = a_{ik}^{(k)}/a_{kk}^{(k)}, \quad a_{ik}^{(k+1)} = 0,$$

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - m_{ik}a_{kj}^{(k)}, \quad b_i^{(k+1)} = b_i^{(k)} - m_{ik}b_k^{(k)}.$$
$$(i = k+1, \cdots, n; \quad j = k+1, \cdots, n.)$$

步3, 回代:

$$x_n = b_n^{(n)}/a_{nn}^{(n)},$$
$$x_k = \left(b_k^{(k)} - \sum_{j=k+1}^{n} a_{kj}^{(k)}x_j\right)/a_{kk}^{(k)}, \ k = n-1, \cdots, 1.$$

可以统计顺序 Gauss 消去法的计算量. 由于加减法的计算量可忽略不计, 此处只统计乘除法次数.

消元过程: 第 $k\,(k = 1, \cdots, n-1)$ 步消元, 有

$$(n-k)(n-k+1) + (n-k) = (n-k)(n-k+2)$$

次乘除法, 共

$$N_1 = \sum_{k=1}^{n-1}(n-k)(n-k+2) = \sum_{i=1}^{n-1}(i^2+2i)$$
$$= \frac{n(n-1)(2n-1)}{6} + n(n-1) = \frac{n(n-1)(2n+5)}{6}$$

次乘除法.

回代过程: 计算 $x_k\,(k=n,\cdots,2,1)$ 时, 有 $n-k+1$ 次乘除法, 共

$$N_2 = \sum_{k=1}^{n}(n-k+1) = \sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

次乘除法.

消元和回代过程共计

$$N_1 + N_2 = \frac{n(n-1)(2n+5)}{6} + \frac{n(n+1)}{2} = \frac{n^3}{3} + n^2 - \frac{n}{3}$$

次乘除法.




Back
Close

可见消元过程的计算量为 $O(n^3)$, 而回代过程的计算量为 $O(n^2)$, 因此顺序 Gauss 消去法的计算量主要在消元过程部分.

算法 6.1 要求对所有的 $k = 1, \cdots, n$, $a_{kk}^{(k)} \neq 0$, 此时称顺序 Gauss 消去法是可行的. 一般将 $a_{kk}^{(k)}$ 称为第 k 步消元的主元. 下面的定理给出了顺序 Gauss 消去法可行的一个充要条件.

定理 6.1 证明: 顺序 Gauss 消去法可行的充分必要条件是系数矩阵 $\boldsymbol{A}$ 的所有顺序主子式 $\boldsymbol{D}_k \neq 0$, $k = 1, 2, \cdots, n$.

证明 必要性. 若顺序 Gauss 消去法是可行的, 即 $a_{kk}^{(k)} \neq 0$, 则可进行消去法的 $k-1$ 步 $(k \leqslant n)$. 由于 $\boldsymbol{A}^{(k)}$ 是由 $\boldsymbol{A}$ 逐行实行初等变换 (某数乘以某一行加到另一行) 得到的, 这些运算不改变相应顺序主子式的值, 故有

$$\boldsymbol{D}_k = \begin{vmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1k}^{(1)} \\ & a_{22}^{(2)} & \cdots & a_{2k}^{(2)} \\ & & \ddots & \vdots \\ & & & a_{kk}^{(k)} \end{vmatrix} = a_{11}^{(1)} a_{22}^{(2)} \cdots a_{kk}^{(k)} \neq 0, \quad k = 1, 2, \cdots, n.$$

充分性. 用归纳法证明. 当 $k = 1$ 时显然成立. 设命题对 $k-1$ 成立. 现设 $\boldsymbol{D}_1 \neq 0, \cdots, \boldsymbol{D}_{k-1} \neq 0, \boldsymbol{D}_k \neq 0$. 由归纳法假设有 $a_{11}^{(1)} \neq 0, \cdots, a_{k-1,k-1}^{(k-1)} \neq 0$. 因此, 消去法可以进行第 $k-1$ 步, $\boldsymbol{A}$ 约化为

$$\boldsymbol{A}^{(k)} = \begin{bmatrix} \boldsymbol{A}_{11}^{(k-1)} & \boldsymbol{A}_{12}^{(k-1)} \\ & \boldsymbol{A}_{22}^{(k)} \end{bmatrix},$$

式中: $\boldsymbol{A}_{11}^{(k-1)}$ 是对角元为 $a_{11}^{(1)}, \cdots, a_{k-1,k-1}^{(k-1)}$ 的上三角矩阵. 因 $\boldsymbol{A}^{(k)}$ 是通过行初等变换由 $\boldsymbol{A}$ 逐步得到的, 故 $\boldsymbol{A}$ 的 k 阶顺序主子式与

Back

Close

$\boldsymbol{A}^{(k)}$ 的 k 阶顺序主子式相等, 即

$$\boldsymbol{D}_k = \det \begin{bmatrix} \boldsymbol{A}_{11}^{(k-1)} & \alpha_{12}^{(k-1)} \\ & a_{kk}^{(k)} \end{bmatrix} = a_{11}^{(1)} \cdots a_{k-1,k-1}^{(k-1)} a_{kk}^{(k)},$$

式中: $\alpha_{12}^{(k-1)}$ 为 $\boldsymbol{A}_{12}^{(k-1)}$ 的第 1 列. 故由 $\boldsymbol{D}_k \neq 0$ 及归纳法假设可推出 $a_{kk}^{(k)} \neq 0$. 证毕. □

根据算法 6.1, 可编制 MATLAB 程序如下:

```
%顺序Gauss消去法程序-msgauss.m
function x=msgauss(A,b,flag)
%输入:A为系数矩阵,b为右端项,若flag=0(默认),
%      则不显示中间过程,否则显示中间过程
%输出:x为解向量
if nargin<3, flag=0; end
```

```
n=length(b);
for k=1:(n-1)  %消元过程
   m=A(k+1:n,k)/A(k,k);
   A(k+1:n,k+1:n)=A(k+1:n,k+1:n)-m*A(k,k+1:n);
   b(k+1:n)=b(k+1:n)-m*b(k);
   A(k+1:n,k)=zeros(n-k,1);
   if flag~=0, Ab=[A,b], end
end
x=zeros(n,1);  %回代过程
x(n)=b(n)/A(n,n);
for k=n-1:-1:1
   x(k)=(b(k)-A(k,k+1:n)*x(k+1:n))/A(k,k);
end
```

例 6.2 利用程序 msgauss.m 计算下列方程组的解

$$\begin{cases} x_1 + x_2 + x_3 + x_4 = 6, \\ -x_1 + 2x_2 - 3x_3 + x_4 = -2, \\ 3x_1 - 3x_2 + 6x_3 - 2x_4 = 7, \\ -4x_1 + 5x_2 + 2x_3 - 3x_4 = 7. \end{cases}$$

解 在 MATLAB 命令窗口输入:

```
>> A=[1 1 1 1;-1 2 -3 1;3 -3 6 -2;-4 5 2 -3];
>> b=[6 -2 7 7]';
>> x=msgauss(A,b)
```

可得计算结果 (略).

§6.1.2 列主元 Gauss 消去法

一般来说, 顺序 Gauss 消去法的计算过程是不可靠的, 一旦出现 $a_{kk}^{(k)} = 0$, 计算就无法进行下去. 另外, 即使对所有 $k = 1, 2, \cdots, n$, $a_{kk}^{(k)} \neq 0$, 也不能保证计算过程是数值稳定的.

例 6.3 设有线性方程组

$$\begin{cases} 0.0001x_1 + 1.0x_2 = 1.0, \\ 1.0x_1 + 1.0x_2 = 2.0. \end{cases}$$

其精确解为

$$x_1 = \frac{10000}{9999} \approx 1.00010, \quad x_2 = \frac{9998}{9999} \approx 0.99990.$$

现在假定用尾数为 4 位十进制字长的浮点数来求解.

解 (1) 消元过程: 根据 4 位浮点数运算规则 $1.0 - 10000.0 = (0.00001 - 0.1)10^5 = (0.0000 - 0.1)10^5 = -10000.0$ (舍入). 同理,

$$2.0 - 10000.0 = -10000.0.$$

$$\begin{bmatrix} 0.0001 & 1.0 & 1.0 \\ 1.0 & 1.0 & 2.0 \end{bmatrix} \xrightarrow{r_2 - 10^4 r_1} \begin{bmatrix} 0.0001 & 1.0 & 1.0 \\ 0 & 1.0 - 10000.0 & 2.0 - 10000.0 \end{bmatrix}$$

$$\xrightarrow{\text{舍入}} \begin{bmatrix} 0.0001 & 1.0 & 1.0 \\ 0 & -10000.0 & -10000.0 \end{bmatrix}.$$

(2) 回代过程:

$$\begin{cases} 0.0001x_1 + 1.0x_2 = 1.0, \\ \qquad -10000.0x_2 = -10000.0. \end{cases} \Longrightarrow \begin{cases} x_2 = 1.0, \\ x_1 = 0.0. \end{cases}$$

代入原方程组验算, 发现结果严重失真.

分析结果失真的原因发现, 由于第 1 列的主元素 0.0001 绝对值过于小, 当它在消元过程中作分母时把中间过程数据放大 10000 倍, 使中间结果“吃”掉了原始数据, 从而造成数值不稳定.

针对以上问题, 考虑选用绝对值大的数作为主元素.

(1) 消元过程:

$$\begin{bmatrix} 0.0001 & 1.0 & 1.0 \\ 1.0 & 1.0 & 2.0 \end{bmatrix} \xrightarrow{r_1 \leftrightarrow r_2} \begin{bmatrix} 1.0 & 1.0 & 2.0 \\ 0.0001 & 1.0 & 1.0 \end{bmatrix}$$

$$\xrightarrow{r_2 - 0.0001 r_1} \begin{bmatrix} 1.0 & 1.0 & 2.0 \\ 0 & 1.0 - 0.0001 & 1.0 - 0.0002 \end{bmatrix}$$

$$\xrightarrow{\text{舍入}} \begin{bmatrix} 1.0 & 1.0 & 2.0 \\ 0 & 1.0 & 1.0 \end{bmatrix}.$$

这里, 舍入过程 $1.0 - 0.0001 = (0.1 - 0.00001)10^1$ (舍入), 同理 $1.0 - 0.0002 = 1.0$.

(2) 回代过程:

$$\begin{cases} 1.0x_1 + 1.0x_2 = 2.0, \\ \qquad\quad 1.0x_2 = 1.0. \end{cases} \Longrightarrow \begin{cases} x_2 = 1.0, \\ x_1 = 1.0. \end{cases}$$

代入原方程组验算, 发现结果基本合理.

例 6.3 说明了选主元素的重要性. 下面阐述列主元 Gauss 消去法的基本思想. 记 $\boldsymbol{A}^{(1)} = \boldsymbol{A}$, 在消元过程的第 1 步, 取第 1 列中绝对值最大的元素 $a_{r_1 1}^{(1)}$, 即

$$a_{r_1 1}^{(1)} = \max_{1 \leqslant i \leqslant n} |a_{i1}^{(1)}|$$

作为主元素. 若 $r_1 > 1$, 交换第 r_1 行和第 1 行.

一般地, 在消元过程的第 k 步, 取

$$a_{r_k k}^{(k)} = \max_{k \leqslant i \leqslant n} |a_{ik}^{(k)}| \tag{6.5}$$

作为主元素. 若 $r_k > k$, 交换第 r_k 行和第 k 行.

列主元 Gauss 消去法的算法具体步骤如下.

算法 6.2 (列主元 Gauss 消去法)

步1, 输入系数矩阵 $\boldsymbol{A}$, 右端项 $\boldsymbol{b}$, 置 $k := 1$.

步2, 对 $k = 1, \cdots, n-1$ 进行如下操作:

① 选列主元, 确定 r_k, 使

$$a_{r_k k}^{(k)} = \max_{k \leqslant i \leqslant n} |a_{ik}^{(k)}|,$$

若 $a_{r_k k}^{(k)} = 0$, 则停止计算, 否则, 进行下一步.

② 若 $r_k > k$, 交换 $(A^{(k)}, b^{(k)})$ 的第 k, r_k 两行.

③ 消元: 对 $i, j = k+1, \cdots, n$, 计算

$$m_{ik} = a_{ik}^{(k)} / a_{kk}^{(k)}, \quad a_{ik}^{(k+1)} = 0,$$

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - m_{ik} a_{kj}^{(k)}, \quad b_i^{(k+1)} = b_i^{(k)} - m_{ik} b_k^{(k)}.$$

步3, 回代:

$$x_n = b_n^{(n)}/a_{nn}^{(n)},$$
$$x_k = \left(b_k^{(k)} - \sum_{j=k+1}^{n} a_{kj}^{(k)} x_j\right)/a_{kk}^{(k)}, \quad k = n-1, \cdots, 1.$$

根据算法 6.2, **编制** MATLAB **程序如下**:

```
%列主元Gauss消去法程序-mgauss2.m
function x=mgauss2(A,b,flag)
%输入:A为系数矩阵,b为右端项,若flag=0(默认),
%      则不显示中间过程,否则显示中间过程
%输出:x为解向量
if nargin<3, flag=0; end
n=length(b);
for k=1:(n-1)  %选主元
    [ap,p]=max(abs(A(k:n,k)));p=p+k-1;
```

```
    if p>k
       A([k p],:)=A([p k],:);b([k p],:)=b([p k],:);
    end
    m=A(k+1:n,k)/A(k,k);  %消元
    A(k+1:n,k+1:n)=A(k+1:n,k+1:n)-m*A(k,k+1:n);
    b(k+1:n)=b(k+1:n)-m*b(k);A(k+1:n,k)=zeros(n-k,1);
    if flag~=0, Ab=[A,b], end
end
x=zeros(n,1); x(n)=b(n)/A(n,n); %回代
for k=n-1:-1:1
    x(k)=(b(k)-A(k,k+1:n)*x(k+1:n))/A(k,k);
end
```

例 6.4 利用程序 mgauss2.m 计算下列线性方程组的解

$$\begin{bmatrix} 2 & -1 & 4 & -3 & 1 \\ -1 & 1 & 2 & 1 & 3 \\ 4 & 2 & 3 & 3 & -1 \\ -3 & 1 & 3 & 2 & 4 \\ 1 & 3 & -1 & 4 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 11 \\ 14 \\ 4 \\ 16 \\ 18 \end{bmatrix}.$$

解 在 MATLAB **命令窗口输入**:

```
>> A=[2 -1 4 -3 1;-1 1 2 1 3;4 2 3 3 -1;-3 1 3 2 4;1 3 -1 4 4
>> b=[11 14 4 16 18]';
>> x=mpgauss(A,b); x=x'
```

得计算结果:

```
x =
    1.0000   2.0000   1.0000   -1.0000    4.0000
```

§6.2 LU 分解法

把一个 n 阶矩阵分解成两个三角形矩阵的乘积称为矩阵的三角分解. 本节介绍矩阵的 LU 分解 $\boldsymbol{A}=\boldsymbol{LU}$, 其中 $\boldsymbol{L}$ 是单位下三角矩阵, $\boldsymbol{U}$ 是上三角矩阵. 这种形式的分解对于求解方程组 (6.1) 是十分有用的. 事实上, 若 $\boldsymbol{A}=\boldsymbol{LU}$ 是一个 LU 分解, 此时线性方程组

$$\boldsymbol{Ax}=\boldsymbol{b}\Longrightarrow\boldsymbol{LUx}=\boldsymbol{b}\Longrightarrow\begin{cases}\boldsymbol{Ly}=\boldsymbol{b}\\ \boldsymbol{Ux}=\boldsymbol{y}\end{cases}\tag{6.6}$$

转化为 $\boldsymbol{Ly}=\boldsymbol{b}$ 及 $\boldsymbol{Ux}=\boldsymbol{y}$ 两个三角形方程组. 由于三角形方程组很容易通过向前消去法或回代方法求解, 且只有 $O(n^2)$ 的计算量, 故研究矩阵的 LU 分解十分有意义.

§6.2.1 顺序 LU 分解法

首先讨论矩阵的 LU 分解. 设 $\boldsymbol{A} = \boldsymbol{LU}$, 其中 $\boldsymbol{L}$ 为一个单位下三角矩阵, $\boldsymbol{U}$ 为一个上三角矩阵, 即

$$\boldsymbol{L} = \begin{bmatrix} 1 & & & \\ l_{21} & 1 & & \\ \vdots & \vdots & \ddots & \\ l_{n1} & l_{n2} & \cdots & 1 \end{bmatrix}, \quad \boldsymbol{U} = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ & u_{22} & \cdots & u_{2n} \\ & & \ddots & \vdots \\ & & & u_{nn} \end{bmatrix}. \tag{6.7}$$

下面推导三角形矩阵 $\boldsymbol{L}$ 和 $\boldsymbol{U}$ 的元素的计算公式. 由等式 $\boldsymbol{A} = \boldsymbol{LU}$, 得

$$a_{ij} = \begin{bmatrix} l_{i1}, & \cdots, & l_{i,i-1}, & 1, & 0, & \cdots, & 0 \end{bmatrix} \begin{bmatrix} u_{1j} \\ \vdots \\ u_{j-1,j} \\ u_{jj} \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{6.8}$$

当 $j \geqslant i$ 时, 有

$$a_{ij} = l_{i1}u_{1j} + \cdots + l_{i,\,i-1}u_{i-1,\,j} + u_{ij},$$

于是

$$u_{ij} = a_{ij} - \sum_{r=1}^{i-1} l_{ir}u_{rj};$$

当 $j < i$ 时, 有

$$a_{ij} = l_{i1}u_{1j} + \cdots + l_{i,\,j-1}u_{j-1,\,j} + l_{ij}u_{jj},$$

于是

$$l_{ij} = \left(a_{ij} - \sum_{r=1}^{j-1} l_{ir}u_{rj}\right)/u_{jj}.$$

即

$$u_{1j} = a_{1j}, \quad j = 1, \cdots, n; \quad l_{i1} = a_{i1}/u_{11}, \quad i = 2, \cdots, n; \tag{6.9}$$

$$u_{ij} = a_{ij} - \sum_{r=1}^{i-1} l_{ir}u_{rj}, \quad i = 2, \cdots, n; \ j = i, \cdots, n; \tag{6.10}$$

$$l_{ij} = \left(a_{ij} - \sum_{r=1}^{j-1} l_{ir}u_{rj}\right)/u_{jj}, \quad i = 3, \cdots, n; \ j = 2, \cdots, i-1. \tag{6.11}$$

为了便于编程计算, 将式 (6.9) 第 1 式中的下标 j 换成 i, 式 (6.10) 中的下标 i 换成 k, 下标 j 换成 i, 式 (6.11) 中的下标 j 换成 k, 则有

$$u_{1i} = a_{1i}, \quad i = 1, \cdots, n; \quad l_{i1} = a_{i1}/u_{11}, \quad i = 2, \cdots, n; \tag{6.12}$$

$$u_{ki} = a_{ki} - \sum_{r=1}^{k-1} l_{kr} u_{ri}, \quad k = 2, \cdots, n; \ i = k, \cdots, n; \tag{6.13}$$

$$l_{ik} = \left(a_{ik} - \sum_{r=1}^{k-1} l_{ir} u_{rk} \right) / u_{kk}, \quad k = 2, \cdots, n-1; \ i = k+1, \cdots, n. \tag{6.14}$$

上述这种按矩阵 $\boldsymbol{A}$ 元素的自然顺序进行分解的方法称为顺序 LU 分解法. 下面是用顺序 LU 分解求解线性方程组的算法步骤.

算法 6.3 (顺序 LU 分解法)

步1, 输入系数矩阵 $\boldsymbol{A}$, 右端项 $\boldsymbol{b}$.

步2, LU 分解:

$$u_{1i} = a_{1i}, \quad i = 1, \cdots, n;$$

$$l_{i1} = a_{i1}/u_{11}, \quad i = 2, \cdots, n;$$

对 $k = 2, \cdots, n$, 计算

$$u_{ki} = a_{ki} - \sum_{r=1}^{k-1} l_{kr} u_{ri}, \quad i = k, \cdots, n;$$

$$l_{ik} = \left(a_{ik} - \sum_{r=1}^{k-1} l_{ir} u_{rk}\right) \Big/ u_{kk}, \quad i = k+1, \cdots, n.$$

步3, 用向前消去法解下三角方程组 $\boldsymbol{Ly} = \boldsymbol{b}$:

$$y_1 = b_1; \; y_k = b_k - \sum_{i=1}^{k-1} l_{ki} y_i, \quad k = 2, \cdots, n.$$

步4, 用回代法解上三角方程组 $\boldsymbol{Ux} = \boldsymbol{y}$:

$$x_n = y_n/u_{nn}; \; x_k = \left(y_k - \sum_{i=k+1}^{n} u_{ki} x_i\right) \Big/ u_{kk}, \quad k = n - 1, \cdots, 1.$$

Back

Close

注 6.1 可以看出, 利用 LU 分解, 分开了系数矩阵的计算和对右端项的计算. 正是这一特点, 使得 LU 分解法特别适用于求解系数矩阵相同而右端项不同的一系列方程组, 而控制论等领域中刚好存在这样的实际问题.

下面考虑顺序 LU 分解的程序实现. 注意到 LU 分解后, 原系数矩阵 $\boldsymbol{A}$ 的数据不再需要保留, 因此, 为了节省存储空间, 可在 MATLAB 程序中将分解后的单位下三角矩阵 $\boldsymbol{L}$ 和上三角矩阵 $\boldsymbol{U}$ 分别存放在系数矩阵 $\boldsymbol{A}$ 的严格下三角和上三角部分 (单位下三角矩阵的对角线元素 1 不需存储), 而不再为其开辟额外的存储单元.

算法 6.3 的 MATLAB 程序如下:

```
%顺序LU分解法程序-mslu.m
function [x,A]=mslu(A,b)
%输入:A为系数矩阵,b为右端向量
```

```
%输出:x为解向量,L和U分别存放在A的严格下三角和上三角部分
n=length(b);
for k=1:n  %顺序LU分解
   A(k:n,k)=A(k:n,k)-A(k:n,1:k-1)*A(1:k-1,k);
   A(k+1:n,k)=A(k+1:n,k)/A(k,k);  %乘子向量
   A(k,k+1:n)=A(k,k+1:n)-A(k,1:k-1)*A(1:k-1,k+1:n);
end
y=zeros(n,1);
for k=1:n, %解下三角矩阵Ly=b
   y(k)=b(k)-A(k,1:k-1)*y(1:k-1);
end
x=zeros(n,1);
for k=n:-1:1, %解上三角方程组Ux=y
```

```
    x(k)=(y(k)-A(k,k+1:n)*x(k+1:n))/A(k,k);
end
```

例 6.5 利用程序 mslu.m 计算例 6.4 中的线性方程组的解.

解 在 MATLAB 命令窗口输入:

```
>> A=[2 -1 4 -3 1;-1 1 2 1 3;4 2 3 3 -1;
      -3 1 3 2 4;1 3 -1 4 4];
>> b=[11 14 4 16 18]';
>> [x,A]=mslu(A,b)
```

可得计算结果 (略).

§6.2.2 列主元 LU 分解法

顺序 LU 分解法在本质上与顺序 Gauss 消去法是一致的. 由算法 6.1 可知, 顺序 Gauss 消去法的第 1 步消元相当于用矩阵

$$\boldsymbol{M}_1 = \begin{bmatrix} 1 & & & & \\ -m_{21} & 1 & & & \\ -m_{31} & & 1 & & \\ \vdots & & & \ddots & \\ -m_{n1} & & & & 1 \end{bmatrix}$$

左乘 $[\boldsymbol{A}^{(1)}, \boldsymbol{b}^{(1)}]$, 这里 m_{i1} 由式 (6.3) 所定义, 即

$$[\boldsymbol{A}^{(2)}, \boldsymbol{b}^{(2)}] = \boldsymbol{M}_1[\boldsymbol{A}^{(1)}, \boldsymbol{b}^{(1)}].$$

第 2 步消元相当于用矩阵

$$\boldsymbol{M}_2 = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & -m_{32} & 1 & \\ & \vdots & & \ddots \\ & -m_{n2} & & 1 \end{bmatrix}$$

左乘 $[\boldsymbol{A}^{(2)}, \boldsymbol{b}^{(2)}]$, 即

$$[\boldsymbol{A}^{(3)}, \boldsymbol{b}^{(3)}] = \boldsymbol{M}_2[\boldsymbol{A}^{(2)}, \boldsymbol{b}^{(2)}] = \boldsymbol{M}_2\boldsymbol{M}_1[\boldsymbol{A}^{(1)}, \boldsymbol{b}^{(1)}].$$

一般地, 第 k 步相当于用矩阵

$$\boldsymbol{M}_k = \begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & -m_{k+1,k} & 1 & \\ & & \vdots & & \ddots \\ & & -m_{nk} & & 1 \end{bmatrix}$$

左乘 $[\boldsymbol{A}^{(k)}, \boldsymbol{b}^{(k)}]$, 即

$$\begin{aligned} \left[\boldsymbol{A}^{(k+1)}, \boldsymbol{b}^{(k+1)}\right] &= \boldsymbol{M}_k \left[\boldsymbol{A}^{(k)}, \boldsymbol{b}^{(k)}\right] = \cdots \\ &= \boldsymbol{M}_k \cdots \boldsymbol{M}_2 \boldsymbol{M}_1 \left[\boldsymbol{A}^{(1)}, \boldsymbol{b}^{(1)}\right], \quad k = 1, \cdots, n-1. \end{aligned}$$

由顺序 Gauss 消去法可知, 经过 $n-1$ 步消元后, 系数矩阵 $\boldsymbol{A}$ 被化成了上三角矩阵, 即 $\boldsymbol{A}^{(n)} = \boldsymbol{U}$, 从而

$$\boldsymbol{U} = \boldsymbol{A}^{(n)} = \boldsymbol{M}_{n-1} \boldsymbol{A}^{(n-1)} = \boldsymbol{M}_{n-1} \cdots \boldsymbol{M}_2 \boldsymbol{M}_1 \boldsymbol{A},$$




Back
Close

于是

$$\boldsymbol{A} = \boldsymbol{M}_1^{-1}\boldsymbol{M}_2^{-1}\cdots\boldsymbol{M}_{n-1}^{-1}\boldsymbol{U} = \boldsymbol{L}\boldsymbol{U},$$

式中:

$$\boldsymbol{L} = \boldsymbol{M}_1^{-1}\boldsymbol{M}_2^{-1}\cdots\boldsymbol{M}_{n-1}^{-1} = \begin{bmatrix} 1 & & & & \\ m_{21} & 1 & & & \\ \vdots & & \ddots & & \\ m_{n-1,1} & m_{n-1\,2} & \cdots & 1 & \\ m_{n1} & m_{n2} & \cdots & m_{n,n-1} & 1 \end{bmatrix}$$

为单位下三角矩阵. 由此可见, 顺序 Gauss 消去法实际上就是将方程组的系数矩阵分解成单位下三角矩阵与上三角矩阵的乘积. 对比算法 6.1 和算法 6.3, 不难看出, 顺序 Gauss 消去法的消元过程相当于 LU 分解过程和 $\boldsymbol{L}\boldsymbol{y} = \boldsymbol{b}$ 的求解, 而回代过程则相当于解线性方程组 $\boldsymbol{U}\boldsymbol{x} = \boldsymbol{y}$.

Back

Close

定理 6.2 *若 n 阶方阵 $\boldsymbol{A}$ 的所有顺序主子式都不等于零, 则 $\boldsymbol{A}$ 存在唯一的 LU 分解 $\boldsymbol{A}=\boldsymbol{LU}$.*

证明 因顺序 LU 分解本质上等同于顺序 Gauss 消去法, 故存在性由定理 6.1 立即可得. 下面证明唯一性. 事实上, 若 $\boldsymbol{A}$ 存在两种不同的三角分解:

$$\boldsymbol{A}=\boldsymbol{LU}=\boldsymbol{L}_1\boldsymbol{U}_1,$$

式中: $\boldsymbol{L}$ 和 $\boldsymbol{L}_1$ 均为单位下三角矩阵; $\boldsymbol{U}$ 和 $\boldsymbol{U}_1$ 均为上三角矩阵. 因 $\boldsymbol{A}$ 是非奇异的, 故 $\boldsymbol{U}$ 和 $\boldsymbol{U}_1$ 也是非奇异的. 于是由上式, 得

$$\boldsymbol{L}_1^{-1}\boldsymbol{L}=\boldsymbol{U}_1\boldsymbol{U}^{-1}.$$

注意到上式的左边是单位下三角矩阵, 而右边则是上三角矩阵, 故必有

$$\boldsymbol{L}_1^{-1}\boldsymbol{L}=\boldsymbol{U}_1\boldsymbol{U}^{-1}=\boldsymbol{I}\ (\text{单位阵}),$$

即 $\boldsymbol{L}_1=\boldsymbol{L}, \boldsymbol{U}_1=\boldsymbol{U}$. 证毕. □

根据上面的分析, 既然顺序 LU 分解法本质上等同于顺序 Gauss 消去法, 因此, 为了提高计算的数值稳定性, 有必要考虑列主元 LU 分解技术. 这只需要在一般 LU 分解的第 k 步避免绝对值较小的 u_{kk} 作除数即可. 假设第 $k-1$ 步已经完成, 在进行第 k 步分解之前进行选主元的操作. 可以引入

$$s_i = a_{ik} - \sum_{r=1}^{k-1} l_{ir}u_{rk}, \quad i = k, k+1, \cdots, n,$$

且令

$$|s_{i_k}| = \max_{k \leqslant i \leqslant n} |s_i|.$$

然后用 s_{i_k} 作为 u_{kk} 并交换增广矩阵 $[\boldsymbol{A}, \boldsymbol{b}]$ 的第 k 行和第 i_k 行, 于是有 $|l_{ik}| \leqslant 1\,(i = k+1, \cdots, n)$, 再进行第 k 步分解. 算法如下:

算法 6.4 (列主元 LU 分解法)

步1, 输入系数矩阵 $\boldsymbol{A}$, 右端项 $\boldsymbol{b}$.

步2, 列主元 LU 分解:

对 $k=1,\cdots,n$,

① 计算 $s_i = a_{ik} - \sum_{r=1}^{k-1} l_{ir}u_{rk} \Longrightarrow a_{ik}, \quad i = k, k+1, \cdots, n.$

② 选主元 $|s_{i_k}| = \max_{k\leqslant i\leqslant n} |s_i|$, 并记录 i_k, $s_{i_k} \Longrightarrow u_{kk}$.

③ 交换 $[\boldsymbol{A}, \boldsymbol{b}]$ 的第 k 行和第 i_k 行元素.

④ 计算 $\boldsymbol{L}$ 的第 k 列元素: $l_{ik} = s_i/u_{kk} = a_{ik}/a_{kk} \Longrightarrow a_{ik}, \quad i = k+1, \cdots, n.$

⑤ 计算 $\boldsymbol{U}$ 的第 k 行元素: $u_{kj} = a_{kj} - \sum_{r=1}^{k-1} l_{kr}u_{rj} \Longrightarrow a_{kj}, \quad j = k+1, \cdots, n.$

步3, 用向前消去法解下三角方程组 $\boldsymbol{Ly} = \boldsymbol{b}$:

$$y_1 = b_1;\ y_k = b_k - \sum_{j=1}^{k-1} l_{kj}y_j, \quad k = 2, \cdots, n.$$

步4, 用回代法解上三角方程组 $\boldsymbol{Ux}=\boldsymbol{y}$:

$$x_n = y_n/u_{nn};; \; x_k = \Big(y_k - \sum_{j=k+1}^{n} u_{kj}x_j\Big)\Big/u_{kk}, \quad k = n-1, \cdots, 1.$$

下面给出列主元 LU 分解法的 MATLAB 程序:

```
%列主元LU分解法程序-mplu.m
function [x,A,P]=mplu(A,b)
%列主元LU分解PA=LU
%输入:A为系数矩阵,b为右端向量
%输出:x返回解向量,L和U分别存放在A的严格下三角和
%      上三角部分,P返回选主元时记录行交换的置换阵
n=length(b);
P=eye(n);  %P记录选择主元时候所进行的行变换
for k=1:n  %列主元LU分解
```

```
    A(k:n,k)=A(k:n,k)-A(k:n,1:k-1)*A(1:k-1,k);
    [s,m]=max(abs(A(k:n,k)));  %选列主元
    m=m+k-1;
    if m~=k
       A([k m],:)=A([m k],:);P([k m],:)=P([m k],:);
    end
    A(k+1:n,k)=A(k+1:n,k)/A(k,k);
    A(k,k+1:n)=A(k,k+1:n)-A(k,1:k-1)*A(1:k-1,k+1:n);
end
b=P*b; y=zeros(n,1);
for k=1:n, %解下三角矩阵Ly=b
    y(k)=b(k)-A(k,1:k-1)*y(1:k-1);
end
```

Back

Close

```
x=zeros(n,1);
for k=n:-1:1, %解上三角方程组Ux=y
    x(k)=(y(k)-A(k,k+1:n)*x(k+1:n))/A(k,k);
end
```

例 6.6 利用程序 mplu.m 计算下列线性方程组的解

$$\begin{bmatrix} 2 & -1 & 4 & -3 & 1 \\ -1 & 1 & 2 & 1 & 3 \\ 4 & 2 & 3 & 3 & -1 \\ -3 & 1 & 3 & 2 & 4 \\ 1 & 3 & -1 & 4 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 11 \\ 14 \\ 4 \\ 16 \\ 18 \end{bmatrix}.$$

解 在 MATLAB 命令窗口输入:

```
>> A=[2 -1 4 -3 1;-1 1 2 1 3;4 2 3 3 -1;
      -3 1 3 2 4;1 3 -1 4 4];
```

```
>> b=[11 14 4 16 18]';
>> [x,A,P]=mplu(A,b)
```

可得计算结果 (略).

§6.2.3 不完全 LU 分解

当 $\boldsymbol{A}$ 的非零元较少且按一定规则分布时 (称为非零元的稀疏模式), 其 LU 分解 (又称完全 LU 分解) 产生的单位下三角矩阵 $\boldsymbol{L}$ 和上三角矩阵 $\boldsymbol{U}$ 一般不能保持和 $\boldsymbol{A}$ 相同的稀疏模式. 本节讨论矩阵 $\boldsymbol{A}$ 的不完全 LU 分解, 即在保持稀疏模式的前提下进行 LU 分解, 这样分解后 $\boldsymbol{L}$ 和 $\boldsymbol{U}$ 的乘积是 $\boldsymbol{A}$ 的一个近似: $\boldsymbol{A} \approx \boldsymbol{LU}$. 这种分解得到的 $\boldsymbol{L}$ 和 $\boldsymbol{U}$ 的逆矩阵往往用来对方程组 $\boldsymbol{Ax} = \boldsymbol{b}$ 作预处理:

$$\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{x}} = \widetilde{\boldsymbol{b}}, \quad \widetilde{\boldsymbol{A}} = \boldsymbol{L}^{-1}\boldsymbol{A}\boldsymbol{U}^{-1}, \quad \widetilde{\boldsymbol{x}} = \boldsymbol{U}\boldsymbol{x}, \quad \widetilde{\boldsymbol{b}} = \boldsymbol{L}^{-1}\boldsymbol{b}.$$

使得预处理后方程组的系数矩阵有更好的条件数.

定义 6.1 设 $\boldsymbol{A}=(a_{ij})\in\mathbb{R}^{n\times n}$, 非对角元的位置和除对角线之外的其他非零元的位置分别记为下面的两个集合

$$\mathrm{ND}=\big\{(i,j):1\leqslant i,j\leqslant n\ \text{且}\ i\neq j\big\},\quad \mathrm{NZ}=\big\{(i,j)\in\mathrm{ND}:a_{ij}\neq 0\big\}.$$

设指标集 $\mathcal{F}$ 满足 $\mathrm{NZ}\subseteq\mathcal{F}\subseteq\mathrm{ND}$, $0\leqslant\omega\leqslant 1$. 若矩阵 $\boldsymbol{A}$ 有分解

$$\boldsymbol{A}=\boldsymbol{L}\boldsymbol{U}+\boldsymbol{R},$$

式中: $\boldsymbol{L}=(l_{ij})\in\mathbb{R}^{n\times n}$ 为单位下三角阵且满足

$$l_{ij}=0,\quad i>j,\quad (i,j)\notin\mathcal{F};\tag{6.15}$$

$\boldsymbol{U}=(u_{ij})\in\mathbb{R}^{n\times n}$ 为上三角阵且满足

$$u_{ij}=0,\quad i<j,\quad (i,j)\notin\mathcal{F};\tag{6.16}$$

矩阵 $\boldsymbol{R}=(r_{ij})\in\mathbb{R}^{n\times n}$ 满足

$$r_{ij}=0,\quad (i,j)\in\mathcal{F}, \tag{6.17}$$

$$r_{ii}=-\omega\sum_{j=1,\,j\neq i}^{n} r_{ij},\quad i=1,2,\cdots,n. \tag{6.18}$$

则称 $\boldsymbol{A}$ 有关于 $\mathcal{F}$ 和 ω 的松弛不完全 LU 分解, ω 称为松弛参数.

若 $\mathcal{F}=\mathrm{ND}$, 由式 (6.17) 表明 $\boldsymbol{R}$ 的非对角元为 0, 式 (6.18) 表明 $\boldsymbol{R}$ 的对角元也为 0, 所以 $\boldsymbol{R}=\boldsymbol{O}$. 同时式 (6.15) 和式 (6.16) 也不需要满足, 此时的松弛不完全 LU 分解即为完全 LU 分解. 若 $\mathcal{F}=\mathrm{NZ}$, 式 (6.17) 和式 (6.18) 表明 $\boldsymbol{L}$ 的下三角部分以及 $\boldsymbol{U}$ 的上三角部分和 $\boldsymbol{A}$ 的零元分布是一样的. 在分解中可以把集合

$$\mathrm{ND}\backslash\mathcal{F}=\left\{(i,j):1\leqslant i\neq j\leqslant n,(i,j)\notin\mathcal{F}\right\}$$

看成矩阵 $\boldsymbol{A}$ 的零模式, 而 $\boldsymbol{L}$ 和 $\boldsymbol{U}$ 保持了这种零模式. 上述松弛不完全 LU 分解中, 若参数 $\omega = 0$, 相应的分解称为不完全 LU 分解 (ILU). 若参数 $\omega = 1$, 相应的分解称为修正的不完全 LU 分解 (MILU).

下面给出矩阵 $\boldsymbol{A}$ 关于 $\mathcal{F}$ 和 ω 的松弛不完全 LU 分解, 该分解可以通过 $n-1$ 步 Gauss 消去法和 $n-1$ 次修正来实现, 称为修正的 Gauss 消去法. 具体步骤如下: 记 $\boldsymbol{A}^{(1)} = (a_{ij}^{(1)}) = \boldsymbol{A}$, 假定已经执行了 $k-1$ 步修正的 Gauss 消去法并产生了 $\boldsymbol{A}^{(k)} = (a_{ij}^{(k)})$, 则第 k 步修正的 Gauss 消去法如下.

(1) Gauss 消去法. 把 $\boldsymbol{A}^{(k)}$ 的第 k 列中第 $k+1$ 个元素到第 n 个元素变为 0, 令

$$\widetilde{\boldsymbol{A}}^{(k)} = \boldsymbol{L}_k \boldsymbol{A}^{(k)} = \boldsymbol{A}^{(k)} - \boldsymbol{l}_k \boldsymbol{a}_k,$$

式中: $\boldsymbol{L}_k = \boldsymbol{I} - \boldsymbol{l}_k \boldsymbol{e}_k^{\mathrm{T}}$, 且




$$\boldsymbol{l}_k = (0, \cdots, 0, l_{k+1,k}, \cdots, l_{nk})^{\mathrm{T}}, \quad l_{ik} = a_{ik}^{(k-1)} / a_{kk}^{(k-1)}, \ i = k+1, \cdots, n,$$

$$\boldsymbol{a}_k = \boldsymbol{e}_k^{\mathrm{T}} \boldsymbol{A}^{(k)} = (0, \cdots, 0, a_{k,k}^{(k)}, \cdots, a_{kn}^{(k)})^{\mathrm{T}}.$$

(2) 修正 $\widetilde{\boldsymbol{A}}^{(k)} = (\widetilde{a}_{ij}^{(k)})$ 右下角的 $n-k$ 阶子矩阵. 将 $\widetilde{\boldsymbol{A}}^{(k-1)}$ 修正为

$$\boldsymbol{A}^{(k+1)} = \widetilde{\boldsymbol{A}}^{(k)} + \boldsymbol{R}_k,$$

式中: $\boldsymbol{R}_k = (r_{ij}^{(k)})$ 为

$$r_{ij}^{(k)} = \begin{cases} \widetilde{a}_{ij}^{(k-1)}, & k+1 \leqslant i, j \leqslant n, \ i \neq j, \ (i,j) \notin \mathcal{F}; \\ -\omega \sum\limits_{\substack{l=k+1 \\ l \neq i, \, (i,l) \notin \mathcal{F}}} \widetilde{a}_{il}^{(k-1)}, & k+1 \leqslant i, j \leqslant n, \ i = j; \\ 0, & \text{其他}. \end{cases}$$

第 $n-1$ 步修正的 Gauss 消去法后, 可得单位下三角矩阵

$$\boldsymbol{L}=(\boldsymbol{L}_{n-1}\boldsymbol{L}_{n-2}\cdots\boldsymbol{L}_1)^{-1}$$

和上三角阵 $\boldsymbol{U}=\boldsymbol{A}^{(n)}$. 令

$$\boldsymbol{R}=\sum_{k=1}^{n-1}\boldsymbol{R}_k,\quad \boldsymbol{R}_k=(r_{ij}^{(k)}),$$

则可得如下定理 (见文献 [13]).

定理 6.3 设 $a_{kk}^{(k)}\neq 0,\ k=1,2,\cdots,n,$ 则上述修正的 Gauss 消去法得到的 $\boldsymbol{A}=\boldsymbol{L}\boldsymbol{U}+\boldsymbol{R}$ 就是关于 $\mathcal{F}$ 和 ω 的松弛不完全 LU 分解.

下面给出修正 Gauss 消去法可以进行下去的充分条件.

定义 6.2 若矩阵 $\boldsymbol{A}$ 满足

(1) $a_{ii} > 0\,(i = 1, 2, \cdots, n-1)$, $a_{nn} \geqslant 0$.

(2) $a_{ij} \leqslant 0\,(i \neq j)$, $1 \leqslant i, j \leqslant n$.

(3) $n(i) = \max\{j : 1 \leqslant j \leqslant n,\ a_{ij} \neq 0\} > i,\ \ i = 1, 2, \cdots, n-1$.

则称 $\boldsymbol{A}$ 为 $\hat{M}$ 矩阵.

有下面的定理, 详细证明参考文献 [6].

定理 6.4 设 $\boldsymbol{A} = (a_{ij}) \in \mathbb{R}^{n\times n}$ 为弱严格对角占优的 $\hat{M}$ 矩阵, 对任意指标集 $\mathcal{F}$ $(\mathrm{NZ} \subset \mathcal{F} \subset \mathrm{ND})$ 和 ω $(0 \leqslant \omega \leqslant 1)$, 修正的 Gauss 消去法给出了 $\boldsymbol{A}$ 关于 $\mathcal{F}$ 和 ω 的松弛不完全 LU 分解.

定义非零模式

$$\widetilde{\mathcal{F}} = \mathcal{F} \cup \big\{(i, i) : i = 1, 2, \cdots, n\big\},$$

它是零模式的补集. 加入修正的 Gauss 消去法是为了确保每步的零模式和非零模式都保持不变. 所以每次修正消元实际上只是对非零模式 $\widehat{\mathcal{F}}$ 的元素进行了 Gauss 消元, 同时将对角元素进行相应的修正. 下面给出松弛不完全 LU 分解具体的算法, 其中 $\boldsymbol{L}$ 存放在 $\boldsymbol{A}$ 的严格下三角部分, $\boldsymbol{U}$ 存放在 $\boldsymbol{A}$ 的上三角部分.

算法 6.5 (松弛不完全 LU 分解) 给定矩阵 $\boldsymbol{A}$, 非零模式 $\widetilde{\mathcal{F}}$ 和松弛参数 ω.

for $k = 1 : n-1$

  for $i = k+1 : n$

    if $(i,k) \in \widetilde{\mathcal{F}}$ (非零模式)

      $a_{ik} = a_{ik}/a_{kk}$;

      for $j = k+1 : n$

        $a_{ij} = a_{ij} - a_{ik} \cdot a_{kj}$;

end

for $j = k+1 : n$

if $(i,j) \notin \widetilde{\mathcal{F}}$ (零模式)

$r_{ij} = r_{ij} + a_{ij};\ r_{ii} = r_{ii} - \omega * a_{ij};$

$a_{ii} = a_{ii} + \omega * a_{ij};\ a_{ij} = 0;$

end

end

end

end

end

下面给出 ILU 分解的 MATLAB 程序如下:

```
function [L,U,R]=milu(A,NF,omega)
```

```
%矩阵A的松弛不完全LU分解
%输入:矩阵A,非零模式NF,松弛参数omega
%输出:单位下三角阵L,上三角阵U,剩余矩阵R,满足A=LU+R
n=size(A,1); R=zeros(n);
for k=1:n-1
   for i=k+1:n
      if (ismember(i+k*sqrt(-1),NF)==1)
      %(i,k)属于集合NF
         A(i,k)=A(i,k)/A(k,k);
         for j=k+1:n
            A(i,j)=A(i,j)-A(i,k)*A(k,j);
         end
         for j=k+1:n
```

Back

Close

```
            if (ismember(i+j*sqrt(-1),NF)==0)
            %(i,j)不属于集合NF
            R(i,j)=R(i,j)+A(i,j);R(i,i)=R(i,i)-omega*A(i,j);
            A(i,i)=A(i,i)+omega*A(i,j);A(i,j)=0;
            end
          end
        end
    end
end
L=tril(A,-1)+eye(n); U=triu(A);
```

由于实际应用中一般不需要剩余矩阵 $\boldsymbol{R}$ 的信息, 为了节省存储量, 可以不对其计算及保存, 这只需在上述算法程序中将涉及计算 $\boldsymbol{R}$ 的有关语句去掉即可.

例 6.7 利用程序 milu.m 对矩阵 $\boldsymbol{A}$ 进行松弛不完全 LU 分解, 其中

$$\boldsymbol{A} = \begin{bmatrix} 4 & 0 & 0 & 1 & 0 \\ 0 & 3 & 2 & 0 & 1 \\ 8 & 0 & 2 & 0 & 0 \\ 0 & 9 & 0 & 5 & 0 \\ 0 & 0 & 2 & 0 & 6 \end{bmatrix},$$

$\mathcal{F} = \{(1,1),(1,4),(2,2),(2,3),(2,5),(3,1),(3,3),(4,2),(4,4),(5,3),(5,5)\}$, $\omega =$

解 编写 MATLAB 脚本文件 ex67.m, 在命令窗口运行之, 可得相应的计算结果 (略).

§6.3 对称正定方程组的直接法

前面讨论的 Gauss 消去法和 LU 分解法, 都是求解一般方程组的方法, 它们均不考虑方程组系数矩阵本身的特点. 但在实际应用中经常会遇到一些特殊类型的方程组, 其系数矩阵具有某种特殊性, 如对称正定矩阵、稀疏 (带状) 矩阵等. 对于这些方程组, 若还用原有的一般方法来求解, 势必造成存储空间和计算的浪费. 因此, 有必要构造适合特殊方程组的求解方法. 本节主要介绍解对称正定方程组的 Cholesky 分解法.

§6.3.1 Cholesky 分解法

当线性方程组的系数矩阵 $\boldsymbol{A}$ 是对称正定矩阵时, 可利用对称正定的特点使 LU 分解减少计算量, 从而节省存储空间. 由于对称正定矩阵的所有顺序主子式都大于零, 故由定理 6.2 可知 $\boldsymbol{A}$ 存在唯一的 LU 分解. 由于 $\boldsymbol{A}$ 是对称的, 即 $a_{ij} = a_{ji}$, $i, j = 1, 2, \cdots, n$.

由 LU 分解式 (6.12)~式 (6.14), 有

$$u_{1i} = a_{1i}, \quad i = 1, \cdots, n; \quad l_{i1} = \frac{a_{i1}}{a_{11}}, \quad i = 2, \cdots, n,$$

则

$$l_{i1} = \frac{a_{i1}}{a_{11}} = \frac{a_{1i}}{a_{11}} = \frac{u_{1i}}{u_{11}}, \quad i = 2, \cdots, n. \tag{6.19}$$

若已求得第 1 步到第 $k-1$ 步的 $\boldsymbol{L}$ 和 $\boldsymbol{U}$ 的元素有如下关系, 即

$$l_{ij} = \frac{u_{ji}}{u_{jj}}, \quad j = 1, \cdots, k-1; \ i = j+1, \cdots, n, \tag{6.20}$$

则对于第 k 步, 由式 (6.13)、式 (6.14) 和式 (6.20), 得

$$u_{ki} = a_{ki} - \sum_{r=1}^{k-1} l_{kr} u_{ri} = a_{ki} - \sum_{r=1}^{k-1} \frac{u_{rk} u_{ri}}{u_{rr}}, \quad i = k, \cdots, n;$$

$$l_{ik} = \left(a_{ik} - \sum_{r=1}^{k-1} l_{ir} u_{rk} \right) / u_{kk}$$

$$= \left(a_{ik} - \sum_{r=1}^{k-1} \frac{u_{rk} u_{ri}}{u_{rr}} \right) / u_{kk} = \frac{u_{ki}}{u_{kk}}, \quad i = k+1, \cdots, n.$$

Back

Close

由此, 得

$$l_{ik} = \frac{u_{ki}}{u_{kk}}, \quad k = 1, \cdots, n-1; \ i = k+1, \cdots, n. \tag{6.21}$$

这样, 利用式 (6.21) 计算 $\boldsymbol{L}$ 的元素可节省工作量, 计算量节省了将近一半, 而 $\boldsymbol{U}$ 的元素仍用式 (6.13) 计算:

$$u_{ki} = a_{ki} - \sum_{r=1}^{k-1} l_{kr} u_{ri}, \quad k = 2, \cdots, n; \ i = k, \cdots, n.$$

注 6.2 由式 (6.21), 得

$$u_{ki} = u_{kk} l_{ik}, \ k = 1, \cdots, n-1; \ i = k+1, \cdots, n,$$

此即

$$\boldsymbol{U} = \boldsymbol{D}\boldsymbol{L}^{\mathrm{T}},$$

式中: $\boldsymbol{D}$ 为以 $u_{kk}\,(k = 1, \cdots, n)$ 为对角元的对角矩阵. 这样, 就把

Back

Close

对称正定矩阵 $\boldsymbol{A}$ 分解成了

$$\boldsymbol{A} = \boldsymbol{L}\boldsymbol{U} = \boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^{\mathrm{T}}$$

的形式. 这种分解方法称为 Cholesky 分解法.

下面建立用 Cholesky 分解法求解对称正定方程组的算法步骤.

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \Longrightarrow \begin{cases} \boldsymbol{A} = \boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^{\mathrm{T}}, \\ \boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^{\mathrm{T}}\boldsymbol{x} = \boldsymbol{b}, \end{cases} \Longrightarrow \begin{cases} \boldsymbol{L}\boldsymbol{y} = \boldsymbol{b}, \\ \boldsymbol{D}\boldsymbol{z} = \boldsymbol{y}, \\ \boldsymbol{L}^{\mathrm{T}}\boldsymbol{x} = \boldsymbol{z}. \end{cases} \tag{6.22}$$

算法 6.6 (Cholesky 分解法)

步 1, 输入对称正定矩阵 $\boldsymbol{A}$ 和右端向量 $\boldsymbol{b}$.

步 2, Cholesky 分解:

$$u_{1i} = a_{1i}, \quad i = 1, \cdots, n;$$

$$l_{i1} = u_{1i}/u_{11}, \quad i = 2, \cdots, n.$$

对 $k=2,\cdots,n$, 计算

$$u_{ki}=a_{ki}-\sum_{r=1}^{k-1}l_{kr}u_{ri},\quad i=k,\cdots,n;$$

$$l_{ik}=u_{ki}/u_{kk},\quad i=k+1,\cdots,n.$$

步 3, 用向前消去法解下三角方程组 $\boldsymbol{Ly}=\boldsymbol{b}$:

$$y_1=b_1,$$

对 $k=2,\cdots,n$, 计算 $y_k=b_k-\sum_{i=1}^{k-1}l_{ki}y_i$.

步 4, 解对角形方程组 $\boldsymbol{Dz}=\boldsymbol{y}$:

对 $k=1,\cdots,n$, 计算 $z_k=y_k/d_k$.

步 5, 用回代法解上三角方程组 $\boldsymbol{L}^{\mathrm{T}}\boldsymbol{x}=\boldsymbol{z}$:

$$x_n=z_n,$$

对 $k=n-1,\cdots,1$, 计算 $x_k=z_k-\sum_{i=k+1}^{n}l_{ik}x_i$.

根据算法 6.6, **编制** MATLAB **程序如下**:

Back

Close

```
%Cholesky分解法程序-mschol.m
function [x,L,D]=mschol(A,b)
%用Cholesky分解法解对称正定方程组Ax=b
%输入:系数矩阵A,右端项b
%输出:解向量x,单位下三角阵L,对角阵D
n=size(A,1); D=zeros(1,n); L=eye(n,n);
U(1,:)=A(1,:); L(2:n,1)=U(1,2:n)/U(1,1);  %Cholesky分解
for k=2:n
   U(k,k:n)=A(k,k:n)-L(k,1:k-1)*U(1:k-1,k:n);
   L(k+1:n,k)=U(k,k+1:n)/U(k,k);
end
%求解下三角方程组Ly=b(向前消去法)
y=zeros(n,1); y(1)=b(1);
```

```
for k=2:n,
    y(k)=b(k)-L(k,1:k-1)*y([1:k-1]);
end
%求解对角方程组Dz=y
D=diag(diag(U));
for k=1:n,
    z(k)=y(k)/D(k,k);
end
%求解上三角方程组L'x=z(回代法)
x=zeros(n,1); U=L'; x(n)=z(n);
for k=(n-1):-1:1,
    x(k)=z(k)-U(k,k+1:n)*x(k+1:n);
end
```

例 6.8 利用程序 mschol.m 计算下列对称正定方程组的解

$$\begin{bmatrix} 1 & 1 & -1 \\ 1 & 2 & -3 \\ -1 & -3 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ -3 \\ 2 \end{bmatrix}.$$

解 在 MATLAB 命令窗口输入:

```
>> A=[1,1,-1; 1,2,-3; -1,-3,3];
>> b=[0,-3,2]';
>> [x,L,D]=mschol(A,b)
```

可得计算结果 (略).

§6.3.2 不完全 Cholesky 分解

在预处理共轭梯度法中, 往往需要对称正定矩阵 $\boldsymbol{A}$ 的不完全 Cholesky 分解 $\boldsymbol{A} \approx \boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^{\mathrm{T}}$ 来获得预处理子 $\boldsymbol{M} = \boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^{\mathrm{T}}$. 由于 $\boldsymbol{A}$




Back
Close

的对称性, 此时, 假定指标集 $\mathcal{F}$ 也是对称正定的, 即若 $(i,j)\in\mathcal{F}$, 则必有 $(j,i)\in\mathcal{F}$. 相对于非对称情形的松弛 ILU 分解, 可以考虑 $\boldsymbol{A}$ 关于 $\mathcal{F}$ 和 ω 的松弛不完全 Cholesky 分解 (简称 RIC 分解):

$$\boldsymbol{A}=\boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^{\mathrm{T}}+\boldsymbol{R}, \tag{6.23}$$

式中: $\boldsymbol{L}$ 为单位下三角矩阵且满足式 (6.15); $\boldsymbol{R}$ 为剩余矩阵并满足式 (6.17) 和式 (6.18); $\boldsymbol{D}$ 为对角矩阵.

易知, 此时 $\boldsymbol{R}$ 也是对称的, 而且也可以采用前面介绍的修正 Gauss 消去法实现分解 (6.23), 即先用修正 Gauss 消去法求 $\boldsymbol{A}$ 的 RILU 分解

$$\boldsymbol{A}=\boldsymbol{L}\boldsymbol{U}+\boldsymbol{R},$$

然后取 $\boldsymbol{D}=\mathrm{diag}(u_{11},u_{22},\cdots,u_{nn})$ 为 $\boldsymbol{U}$ 的对角元素所构成的对角矩阵. 这样便得到了式 (6.23) 中的 $\boldsymbol{L}$, $\boldsymbol{D}$ 和 $\boldsymbol{R}$.

下面的定理给出了对称正定矩阵 $\boldsymbol{A}$ 关于 $\mathcal{F}$ 和 ω 的松弛不完全 Cholesky 分解 (6.23) 中 $\boldsymbol{LDL}^{\mathrm{T}}$ 正定的一个充分条件.

可以在算法 6.5 的基础上, 利用 $\boldsymbol{A}$ 的对称性给出求 $\boldsymbol{A}$ 的 RIC 分解的算法. 也可从 Cholesky 分解出发稍加修正求得. 下面是不完全 Cholesky 分解的一个简易可行的算法.

算法 6.7 (不完全 Cholesky 分解法) 给定对称正定矩阵 $\boldsymbol{A}$. 下面的算法计算 $\boldsymbol{A}$ 的不完全 Cholesky 分解: $\boldsymbol{A} \approx \boldsymbol{LL}^{\mathrm{T}}$.

$n = \mathrm{size}(\boldsymbol{A}, 1)$;

for $k = 1 : n$

$$l_{kk} = \left(a_{kk} - \sum_{r=1}^{k-1} l_{kr}^2\right)^{1/2};$$

for $i = k+1 : n$

if $(a_{ik} = 0)$

$l_{ik} = 0$;

else

$$l_{ik} = \left(a_{ik} - \sum_{r=1}^{k-1} l_{ir} l_{kr}\right) \Big/ l_{kk};$$

end

end

end

显然, 由算法 6.7 得到的不完全分解 $\boldsymbol{M} = \boldsymbol{L}\boldsymbol{L}^{\mathrm{T}}$ 与 $\boldsymbol{A}$ 有相同的稀疏性, 但这一算法并不总是稳定的 (当 l_{kk} 很小的时候, $k = 1, 2, \cdots, n$).

算法 6.7 的 MATLAB 程序如下:

```
%不完全Cholesky分解程序-michol.m
function [L]=michol(A)
%输入:对称正定矩阵A
%输出:下三角矩阵L,满足A=LL'+R
n=size(A,1); L=zeros(n,n);
```

```
L(1,1)=sqrt(A(1,1)); L(2,1)=A(2,1)/L(1,1);
for k=2:n
    s=0;
    for(p=1:k-1),s=s+L(k,p)^2;end
    L(k,k)=sqrt(A(k,k)-s);
    for i=k+1:n
        if (A(i,k)~=0)
            s=0;
            for (p=1:k-1),s=s+L(i,p)*L(k,p); end
            L(i,k)=(A(i,k)-s)/L(k,k);
        end
    end
end
```

例 6.9 利用程序 michol.m 对矩阵 $\boldsymbol{A}$ 进行松弛不完全 Cholesky 分解, 其中

$$\boldsymbol{A}=\begin{bmatrix} 8 & -2 & -1 & 0 & 0 & 1 \\ -2 & 8 & -2 & 0 & 3 & 0 \\ -1 & -2 & 8 & 0 & 0 & 0 \\ 0 & 0 & 0 & 8 & -2 & -1 \\ 0 & 3 & 0 & -2 & 8 & -2 \\ 1 & 0 & 0 & -1 & -2 & 8 \end{bmatrix}.$$

解 编写 MATLAB 脚本文件 ex69.m, 在命令窗口运行之, 可得相应的计算结果 (略).




Back
Close

§6.4 带状线性方程组的直接法

§6.4.1 三对角方程组

在科学与工程计算中, 经常遇到求解三对角方程组的问题. 例如, 三次样条插值计算, 以及用有限差分法求解二阶常系数线性常微分方程的边值问题和热传导问题, 经常需要求解三对角方程组(即系数矩阵为三对角矩阵的方程组). 三对角矩阵属于所谓的“带状矩阵”, 在大多数应用中, 带状矩阵是严格对角占优的或正定的. 下面给出带状矩阵的定义.

定义 6.3 n 阶矩阵称为带状矩阵, 如果存在正整数 $p, q\,(1 < p, q < n)$, 当 $j \geqslant i+p$ 或 $i \geqslant j+q$ 时, 有 $a_{ij} = 0$, 并称 $w = p+q-1$ 为该带状矩阵的“带宽”. n 阶矩阵称为带状矩阵, 如果存在正整数 $p, q\,(1 < p, q < n)$, 当 $i+p \leqslant j$ 或 $j+q \leqslant i$ 时, 有 $a_{ij} = 0$, 并称 $w = p+q-1$ 为该带状矩阵的“带宽”.

三对角方程组的一般形式是

$$\boldsymbol{A}\boldsymbol{x} := \begin{bmatrix} b_1 & c_1 & & & \\ a_2 & b_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & a_{n-1} & b_{n-1} & c_{n-1} \\ & & & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_{n-1} \\ f_n \end{bmatrix} := \boldsymbol{f}. \tag{6.24}$$

显然三对角矩阵的带宽为 3. 本节介绍求解方程组 (6.24) 的追赶法和变参数追赶法.

1. 追赶法

将顺序 LU 分解法应用于三对角方程组得到所谓的“追赶法”. 事实上, 一方面, 可将三对角矩阵 $\boldsymbol{A}$ 分解为

$$\boldsymbol{A} = \boldsymbol{L}\boldsymbol{U}, \tag{6.25}$$

式中:

$$\boldsymbol{L}=\begin{bmatrix} l_1 & & & & \\ a_2 & l_2 & & & \\ & \ddots & \ddots & & \\ & & a_{n-1} & l_{n-1} & \\ & & & a_n & l_n \end{bmatrix},\quad \boldsymbol{U}=\begin{bmatrix} 1 & u_1 & & & \\ & 1 & u_2 & & \\ & & \ddots & \ddots & \\ & & & 1 & u_{n-1} \\ & & & & 1 \end{bmatrix}.$$

比较式 (6.25) 两端的对应元素, 得

$$b_1=l_1,\ c_{k-1}=l_{k-1}u_{k-1},\ b_k=a_ku_{k-1}+l_k,\ k=2,\cdots,n.$$

于是有

$$l_1=b_1,\ u_{k-1}=\frac{c_{k-1}}{l_{k-1}},\ l_k=b_k-a_ku_{k-1},\ k=2,\cdots,n. \tag{6.26}$$

另一方面, 方程组 (6.24) 等价于求解 $\boldsymbol{Ly}=\boldsymbol{f}$ 和 $\boldsymbol{Ux}=\boldsymbol{y}$, 其中 $\boldsymbol{y}=(y_1,y_2,\cdots,y_n)^{\mathrm{T}}$. 分别比较 $\boldsymbol{Ly}=\boldsymbol{f}$ 和 $\boldsymbol{Ux}=\boldsymbol{y}$ 两端的对

应元素, 得

$$\begin{cases} l_1 y_1 = f_1, \quad a_k y_{k-1} + l_k y_k = f_k, \quad k = 2, 3, \cdots, n, \\ x_k + u_k x_{k+1} = y_k, \quad k = 1, 2, \cdots, n-1, \quad x_n = y_n. \end{cases} \tag{6.27}$$

结合式 (6.26) 和式 (6.27) 可得下面的算法.

算法 6.8 (追赶法) 计算三对角方程组 $\boldsymbol{Ax} = \boldsymbol{f}$ 的解.

$l_1 = b_1;\ y_1 = \dfrac{f_1}{l_1};$

for $k = 2 : n$

$u_{k-1} = \dfrac{c_{k-1}}{l_{k-1}};$

$l_k = b_k - a_k u_{k-1};$

$y_k = \dfrac{f_k - a_k y_{k-1}}{l_k};$

end

$x_n = y_n$;

for $k = n-1 : -1 : 1$

$x_k = y_k - u_k x_{k+1}$;

end

算法 6.8 被称为“追赶法”的原因: 一是第 1 步关于指标 k 由小到大计算 l_k, u_k 和 y_k 这三个量, 这是“向前追”的过程; 二是第 2 步关于指标 k 从大到小计算方程组的解 x_k, 此即“往回赶”的过程. 追赶法只有 $5n-4$ 次乘除法运算和 $3n-3$ 次加减法运算, 且当系数矩阵对角占优时数值稳定, 是解三对角方程组的优秀算法. 编程计算时, 可将 l_k, u_k 依次存放在 b_k, c_k 的位置, 而将 y_k 和 x_k 先后存放在 f_k 的位置, 因此整个计算过程只需 $4n$ 个存储单元.

追赶法的 MATLAB 程序如下:

```
%追赶法程序-mchase.m
```

```
function [f]=mchase(a,b,c,f)
%用追赶法解三对角方程组Ax=f
%输入:a为A的下对角线,b为A的主对角线,
%       c为A的次上对角线,f为右端向量
%输出:解向量f(LU分解中的l(k),u(k)存放在b(k),
%       c(k)的位置, y(k)和x(k)先后存放在f(k)的位置)
n=length(b); f(1)=f(1)/b(1);
for k=2:n
    c(k-1)=c(k-1)/b(k-1); b(k)=b(k)-a(k)*c(k-1);
    f(k)=(f(k)-a(k)*f(k-1))/b(k);
end
for k=n-1:-1:1
    f(k)=f(k)-c(k)*f(k+1);
```

Back

Close

end

定理 6.5 若方程组 (6.24) 的系数矩阵的元素满足条件 $|b_1| > |c_1| > 0,\ |b_n| > |a_n| > 0,\ |b_i| > |a_i| + |c_i|,\quad i = 2, \cdots, n-1,$ 则追赶法是可行的.

证明 由式 (6.26) 和式 (6.27) 可知, 只需证明 $l_k \neq 0 (k = 1, 2, \cdots, n)$ 即可. 显然 $l_1 = b_1 \neq 0$, 且 $|l_1| = |b_1| > |c_1|$. 设 $|l_{k-1}| > |c_{k-1}|$, 则

$$
\begin{aligned}
|l_k| = |b_k - a_k u_{k-1}| &= \left| b_k - \frac{a_k}{l_{k-1}} c_{k-1} \right| \\
&\geqslant |b_k| - |a_k| \cdot \left| \frac{c_{k-1}}{l_{k-1}} \right| > |b_k| - |a_k| \\
&> \begin{cases} |c_k|, & k < n, \\ 0, & k = n, \end{cases}
\end{aligned}
$$

即 $l_k \neq 0,\ k = 2, \cdots, n$. 从而, 追赶法是可行的. 证毕. □

注 6.3 满足定理 6.5 条件的三对角矩阵即为严格对角占优的, 例 6.5 说明对于严格对角占优的三对角矩阵, 追赶法总是可行的.

2. 变参数追赶法

需要指出的是, 追赶法的本质是没有选取主元的 Gauss 消去法, 因此对于一般的三对角矩阵, 不一定存在如式 (6.25) 的三角分解. 或者三角分解可以进行下去, 但计算过程不是数值稳定的, 因而最终得到的解并不可靠. 这就促使考虑对追赶法进行改进和修正.

将三对角矩阵 $\boldsymbol{A}$ 分解为

$$\boldsymbol{A} = \boldsymbol{D}\widetilde{\boldsymbol{L}}\widetilde{\boldsymbol{U}}, \tag{6.28}$$

式中: $\boldsymbol{D} = \mathrm{diag}(d_1, d_2, \cdots, d_n)$,




$$\widetilde{\boldsymbol{L}} = \begin{bmatrix} l_1 & 1 & & & \\ & l_2 & 1 & & \\ & & \ddots & \ddots & \\ & & & l_n & 1 \end{bmatrix}_{n\times(n+1)}, \quad \widetilde{\boldsymbol{U}} = \begin{bmatrix} u_1 & & & & \\ 1 & u_2 & & & \\ & 1 & \ddots & & \\ & & \ddots & u_n & \\ & & & 1 & \end{bmatrix}_{(n+1)\times n}.$$

比较式 (6.25) 两端的元素, 得

$$\begin{cases} b_k = d_k(l_k u_k + 1), \ k = 1, 2, \cdots, n; \\ a_k = d_k l_k, \ c_{k-1} = d_{k-1} u_k, \ k = 2, \cdots, n. \end{cases}$$

选取 l_1, u_1 使得 $l_1 u_1 + 1 \neq 0$, 则有

$$\begin{cases} d_1 = \dfrac{b_1}{l_1 u_1 + 1}, \\ u_k = \dfrac{c_{k-1}}{d_{k-1}}, \ d_k = b_k - a_k u_k, \ l_k = \dfrac{a_k}{d_k}, \\ \qquad (k = 2, 3, \cdots, n). \end{cases}$$

注意到, 求解 $\boldsymbol{Ax}=\boldsymbol{f}$ 等价于求解 $\widetilde{\boldsymbol{L}}\boldsymbol{y}=\boldsymbol{D}^{-1}\boldsymbol{f}:=\boldsymbol{g}$ 和 $\widetilde{\boldsymbol{U}}\boldsymbol{x}=\boldsymbol{y}$, 其中

$$\boldsymbol{y}=(y_1,y_2,\cdots,y_{n+1})^{\mathrm{T}},\quad \boldsymbol{g}=\left(\frac{f_1}{d_1},\frac{f_2}{d_2},\cdots,\frac{f_n}{d_n}\right)^{\mathrm{T}}.$$

比较 $\widetilde{\boldsymbol{L}}\boldsymbol{y}=\boldsymbol{g}$ 和 $\widetilde{\boldsymbol{U}}\boldsymbol{x}=\boldsymbol{y}$ 两端的元素, 得

$$l_k y_k+y_{k+1}=g_k,\ k=1,2,\cdots,n; \tag{6.29}$$

$$\begin{cases} u_1x_1=y_1, \\ x_k+u_{k+1}x_{k+1}=y_{k+1},\ k=1,2,\cdots,n-1, \\ x_n=y_{n+1}. \end{cases} \tag{6.30}$$

由式 (6.29) 和式 (6.30) 可以发现, 只要计算出 x_1, 其他的可以依次计算出来. 下面推导计算 x_1 的公式.

由式 (6.30), 得

$$x_1 = y_2 - u_2x_2 = y_2 - u_2(y_3 - u_3x_3) = y_2 + (-u_2)y_3 + (-u_2)(-u_3)x_3$$
$$= \cdots = y_2 + (-u_2)y_3 + (-u_2)(-u_3)y_4$$
$$+ \cdots + (-u_2)(-u_3)\cdots(-u_{n-1})y_n + (-u_2)(-u_3)\cdots(-u_n)y_{n+1}. \quad (6.31)$$

由式 (6.29), 得

$$y_{k+1} = g_k - l_ky_k = g_k - l_k(g_{k-1} - l_{k-1}y_{k-1})$$
$$= g_k + (-l_k)g_{k-1} + (-l_k)(-l_{k-1})y_{k-1}$$
$$= \cdots$$
$$= g_k + (-l_k)g_{k-1} + (-l_k)(-l_{k-1})g_{k-2}$$
$$+ \cdots + (-l_k)(-l_{k-1})\cdots(-l_2)g_1$$
$$+ (-l_k)(-l_{k-1})\cdots(-l_1)(u_1x_1),\ k = 1, 2, \cdots, n. \quad (6.32)$$




将式 (6.32) 代入式 (6.31) 并整理, 得

$$
\begin{aligned}
&[1+u_1l_1+(u_1u_2)(l_2l_1)+\cdots+(u_1\cdots u_n)(l_n\cdots l_1)]x_1\\
&=[1+u_2l_2+\cdots+(u_2\cdots u_n)(l_n\cdots l_2)]g_1\\
&\quad+(-u_2)[1+u_3l_3+\cdots+(u_3\cdots u_n)(l_n\cdots l_3)]g_2\\
&\quad+\cdots+[(-u_2)\cdots(-u_{n-1})](1+u_nl_n)g_{n-1}+[(-u_2)\cdots(-u_n)]g_n.
\end{aligned}
$$

令

$$
\begin{aligned}
s_k=&1+u_kl_k+\cdots+(u_k\cdots u_n)(l_n\cdots l_k),\ k=1,2,\cdots,n,\\
t_1=&s_2g_1+(-u_2)s_3g_2+\cdots+[(-u_2)\cdots(-u_{n-1})]s_ng_{n-1}\\
&+[(-u_2)\cdots(-u_n)]g_n.
\end{aligned}
$$

则

$$
x_1=\frac{t_1}{s_1},
$$

且有递推公式

$$s_n = 1 + u_n l_n, \quad s_k = 1 + u_k s_{k+1} l_k, \quad k = n-1, n-2, \cdots, 1,$$

$$t_n = g_n, \quad t_k = s_{k+1} g_k - u_{k+1} t_{k+1}, \quad k = n-1, n-2, \cdots, 1.$$

综合上述, 可得下面的变参数追赶法.

算法 6.9 (变参数追赶法) 计算三对角方程组 $\boldsymbol{Ax} = \boldsymbol{f}$ 的解. 选取 l_1 和 u_1 使得 $l_1 u_1 + 1 \neq 0$.

$$d_1 = \frac{b_1}{l_1 u_1 + 1}; \; g_1 = \frac{f_1}{d_1};$$

for $k = 2 : n$

$$u_k = \frac{c_{k-1}}{d_{k-1}}; \quad d_k = b_k - a_k u_k; \quad l_k = \frac{a_k}{d_k}; \; g_k = \frac{f_k}{d_k};$$

end

$$s_n = 1 + u_n l_n; \; t_n = g_n;$$

for $k = n-1 : -1 : 1$

$$s_k = 1 + u_k s_{k+1} l_k; \quad t_k = s_{k+1} g_k - u_{k+1} t_{k+1};$$

end

$$x_1 = \frac{t_1}{s_1}; \; y_1 = u_1 x_1;$$

for $k = 1 : n$

$$y_{k+1} = g_k - l_k y_k;$$

end

$$x_n = y_{n+1};$$

for $k = n-1 : -1 : 2$

$$x_k = y_{k+1} - u_{k+1} x_{k+1};$$

end

算法 6.9 需 $10n-8$ 次乘除运算, $5n-5$ 次加减运算. 由于算法中的 l_1 和 u_1 的选取只需使得 $l_1u_1+1\neq 0$, 可以认为它们是两个可变的参数, 故称为“变参数追赶法”.

变参数追赶法的 MATLAB 程序如下:

```
%变参数追赶法程序-mchase_var.m
function [x]=mchase_var(a,b,c,f,l1,u1)
%用变参数追赶法解三对角方程组Ax=f
%输入:a为A的下对角线,b为A的主对角线,c为A的次
%      上对角线,f为右端向量;l1,u1满足l1*u1+1不为0.
%输出:解向量x
n=length(b);x=zeros(n,1);l=zeros(n,1);
u=zeros(n,1);l(1)=l1; u(1)=u1;
d(1)=b(1)/(l(1)*u(1)+1); g(1)=f(1)/d(1);
```

```
for k=2:n
    u(k)=c(k-1)/d(k-1);d(k)=b(k)-a(k)*u(k);
    l(k)=a(k)/d(k);g(k)=f(k)/d(k);
end
s(n)=1+u(n)*l(n); t(n)=g(n);
for k=n-1:-1:1
    s(k)=1+u(k)*s(k+1)*l(k);
    t(k)=s(k+1)*g(k)-u(k+1)*t(k+1);
end
x(1)=t(1)/s(1);y(1)=u(1)*x(1);
for k=1:n
    y(k+1)=g(k)-l(k)*y(k);
end
```

```
x(n)=y(n+1);
for k=n-1:-1:2
    x(k)=y(k+1)-u(k+1)*x(k+1);
end
```

例 6.10 设

$$\boldsymbol{A}=\begin{bmatrix} 4 & -1 & & & \\ -2 & 4 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -2 & 4 & -1 \\ & & & -2 & 4 \end{bmatrix}, \quad \boldsymbol{f}=\begin{bmatrix} 3 \\ 1 \\ \vdots \\ 1 \\ 2 \end{bmatrix}.$$

用追赶法和变参数追赶法 (取 $l_1 = u_1 = 1$) 求解三对角方程组 $\boldsymbol{Ax} = \boldsymbol{f}$, 并计算实际误差 $\|\boldsymbol{f} - \boldsymbol{Ax}\|_2$.

Back

Close

解 利用程序 mchase.m 和 mchase_var.m, 编写 MATLAB 脚本文件 ex610.m, 并对不同的维数 n, 在命令窗口运行该程序, 得到数值结果如表 6.1 所示.

表 6.1 追赶法和变参数追赶法的数值结果

维数	追赶法		变参数追赶法	
	误差	计算时间	误差	计算时间
1024	7.1650e-15	0.0056	1.2212e-15	0.0103
2048	1.0091e-14	0.0058	1.2212e-15	0.0106
4096	1.4241e-14	0.0064	1.2212e-15	0.0109
8192	2.0118e-14	0.0066	1.2212e-15	0.0128




Back
Close

例 6.11 设

$$\boldsymbol{A}=\begin{bmatrix} 2 & 3 & & & \\ 6 & 9 & 3 & & \\ & \ddots & \ddots & \ddots & \\ & & 6 & 9 & 3 \\ & & & 6 & 9 \end{bmatrix}, \quad \boldsymbol{f}=\begin{bmatrix} 5 \\ 10 \\ \vdots \\ 10 \\ 12 \end{bmatrix}.$$

试用追赶法和变参数追赶法 (取 $l_1 = u_1 = 1$) 求解三对角方程组 $\boldsymbol{Ax} = \boldsymbol{f}$, 并计算实际误差 $\|\boldsymbol{f} - \boldsymbol{Ax}\|_2$.

解 对于此例, 追赶法已经失效, 但变参数追赶法仍然非常有效, 数值结果 (运行 MATLAB 脚本文件 ex611.m) 如表 6.2 所示.

表 6.2 变参数追赶法的数值结果

维数	追赶法	变参数追赶法	
		误差	计算时间
1024	失效	5.7293e-14	0.0101
4096	失效	1.1391e-13	0.0111

§6.4.2 块三对角方程组

用有限差分法五点格式离散二维 Poisson 方程, 得到的代数方程组 $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{f}$ 的系数矩阵是一个块三对角矩阵, 即

$$\begin{bmatrix} \boldsymbol{B}_1 & \boldsymbol{C}_1 & & & \\ \boldsymbol{A}_2 & \boldsymbol{B}_2 & \boldsymbol{C}_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \boldsymbol{A}_{m-1} & \boldsymbol{B}_{m-1} & \boldsymbol{C}_{m-1} \\ & & & \boldsymbol{A}_m & \boldsymbol{B}_m \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_{m-1} \\ \boldsymbol{x}_m \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}_1 \\ \boldsymbol{f}_2 \\ \vdots \\ \boldsymbol{f}_{m-1} \\ \boldsymbol{f}_m \end{bmatrix}, \quad (6.33)$$

式中: $\boldsymbol{A}_k, \boldsymbol{B}_k, \boldsymbol{C}_k$ 均为 r 阶方阵; $\boldsymbol{x}_k, \boldsymbol{f}_k$ 为 r 维列向量. 当 $\boldsymbol{A}$ 可逆时, 方程组有唯一解 $\boldsymbol{x}^*$. 本节介绍求解方程组 (6.33) 的块追赶法和双参数法.

1. 块追赶法

将三对角方程矩阵 $\boldsymbol{A}$ 分解为 $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{U}$, 其中

$$\boldsymbol{L} = \begin{bmatrix} \boldsymbol{L}_1 & & & \\ \boldsymbol{A}_2 & \boldsymbol{L}_2 & & \\ & \ddots & \ddots & \\ & & \boldsymbol{A}_{m-1} & \boldsymbol{L}_{m-1} \\ & & & \boldsymbol{A}_m \quad \boldsymbol{L}_m \end{bmatrix}, \quad \boldsymbol{U} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{U}_1 & & \\ & \boldsymbol{I} & \boldsymbol{U}_2 & \\ & & \ddots & \ddots \\ & & & \boldsymbol{I} \quad \boldsymbol{U}_{m-1} \\ & & & \boldsymbol{I} \end{bmatrix}.$$

比较 $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{U}$ 两端的对应的子矩阵, 得

$$\boldsymbol{B}_1 = \boldsymbol{L}_1, \ \boldsymbol{C}_{k-1} = \boldsymbol{L}_{k-1}\boldsymbol{U}_{k-1}, \ \boldsymbol{B}_k = \boldsymbol{A}_k\boldsymbol{U}_{k-1} + \boldsymbol{L}_k, \ k = 2, \cdots, n.$$

于是有

$$\boldsymbol{L}_1 = \boldsymbol{B}_1,\ \boldsymbol{U}_{k-1} = \boldsymbol{L}_{k-1}^{-1}\boldsymbol{C}_{k-1},\ \boldsymbol{L}_k = \boldsymbol{B}_k - \boldsymbol{A}_k\boldsymbol{U}_{k-1},\ k = 2, \cdots, n. \tag{6.34}$$

求解三对角方程组 $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{f}$ 等价于求解

$$\boldsymbol{L}\boldsymbol{y} = \boldsymbol{f} \quad \text{和} \quad \boldsymbol{U}\boldsymbol{x} = \boldsymbol{y},$$

式中: $\boldsymbol{y} = \left[\boldsymbol{y}_1^{\mathrm{T}}, \boldsymbol{y}_2^{\mathrm{T}}, \cdots, \boldsymbol{y}_m^{\mathrm{T}}\right]^{\mathrm{T}}$. 分别比较 $\boldsymbol{L}\boldsymbol{y} = \boldsymbol{f}$ 和 $\boldsymbol{U}\boldsymbol{x} = \boldsymbol{y}$ 两端的对应元素, 得

$$\begin{cases} \boldsymbol{L}_1\boldsymbol{y}_1 = \boldsymbol{f}_1, \quad \boldsymbol{A}_k\boldsymbol{y}_{k-1} + \boldsymbol{L}_k\boldsymbol{y}_k = \boldsymbol{f}_k, \quad k = 2, 3, \cdots, m, \\ \boldsymbol{x}_k + \boldsymbol{U}_k\boldsymbol{x}_{k+1} = \boldsymbol{y}_k, \quad k = 1, 2, \cdots, m-1, \quad \boldsymbol{x}_m = \boldsymbol{y}_m. \end{cases} \tag{6.35}$$

结合式 (6.34) 和式 (6.35) 可得下面的算法.

算法 6.10 (块追赶法) 计算块三对角方程组 $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{f}$ 的解.

$\boldsymbol{L}_1 = \boldsymbol{B}_1;\ \boldsymbol{y}_1 = \boldsymbol{L}_1^{-1}\boldsymbol{f}_1;$

for $k = 2 : m$

$\boldsymbol{U}_{k-1} = \boldsymbol{L}_{k-1}^{-1}\boldsymbol{C}_{k-1};\ \boldsymbol{L}_k = \boldsymbol{B}_k - \boldsymbol{A}_k\boldsymbol{U}_{k-1};\ \boldsymbol{y}_k = \boldsymbol{L}_k^{-1}(\boldsymbol{f}_k - \boldsymbol{A}_k\boldsymbol{y}_{k-1});$

end

$\boldsymbol{x}_m = \boldsymbol{y}_m;$

for $k = m - 1 : -1 : 1$

$\boldsymbol{x}_k = \boldsymbol{y}_k - \boldsymbol{U}_k\boldsymbol{x}_{k+1};$

end

使用块追赶法求解三对角方程组 $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{f}$ 需要的乘除法次数约为

$$mr^2\left(\frac{10}{3}r + 3\right) - r^2(3r + 2).$$

编程计算时, 可将 $\boldsymbol{L}_k$ 和 $\boldsymbol{U}_k$ 依次存放在 $\boldsymbol{B}_k$ 和 $\boldsymbol{C}_k$ 的位置, 而将 $\boldsymbol{y}_k$ 和 $\boldsymbol{x}_k$ 先后存放在 $\boldsymbol{f}_k$ 的位置.

块追赶法的 MATLAB 程序如下:

```
%块追赶法程序-mchase_block.m
function [fi]=mchase_block(Ai,Bi,Ci,fi,m)
%用块追赶法解块三对角方程组Ax=f
%输入:Ai为A的次下对角块,Bi为A的主对角块,
%      Ci为A的次上对角块,fi为右端向量,m为分块数
%输出:解向量fi,其中LU分解的L{k},U{k}存放在Bi{k},
%      Ci{k}的位置,y{k}和x{k}先后存放在fi{k}的位置
fi{1}=Bi{1}\fi{1};
for k=2:m
    Ci{k-1}=Bi{k-1}\Ci{k-1};Bi{k}=Bi{k}-Ai{k}*Ci{k-1};
```

```
    fi{k}=Bi{k}\(fi{k}-Ai{k}*fi{k-1});
end
for k=m-1:-1:1
    fi{k}=fi{k}-Ci{k}*fi{k+1};
end
```

2. 双参数法

注意到块三角对方程组 (6.33) 的前 $m-1$ 个子方程为

$$\begin{cases} \boldsymbol{B}_1\boldsymbol{x}_1 + \boldsymbol{C}_1\boldsymbol{x}_2 = \boldsymbol{f}_1, \\ \boldsymbol{A}_2\boldsymbol{x}_1 + \boldsymbol{B}_2\boldsymbol{x}_2 + \boldsymbol{C}_2\boldsymbol{x}_3 = \boldsymbol{f}_2, \\ \qquad\qquad \vdots \\ \boldsymbol{A}_{m-1}\boldsymbol{x}_{m-2} + \boldsymbol{B}_{m-1}\boldsymbol{x}_{m-1} + \boldsymbol{C}_{m-1}\boldsymbol{x}_m = \boldsymbol{f}_{m-1}. \end{cases}$$

将上式中的 $\boldsymbol{x}_2, \boldsymbol{x}_3, \cdots, \boldsymbol{x}_m$ 都用 $\boldsymbol{x}_1$ 表示出来, 即

$$\begin{aligned}
\boldsymbol{x}_2 &= \boldsymbol{C}_1^{-1}(\boldsymbol{f}_1 - \boldsymbol{B}_1\boldsymbol{x}_1) := \boldsymbol{s}_2 + \boldsymbol{T}_2\boldsymbol{x}_1, \\
\boldsymbol{x}_3 &= \boldsymbol{C}_2^{-1}(\boldsymbol{f}_2 - \boldsymbol{A}_2\boldsymbol{x}_1 - \boldsymbol{B}_2\boldsymbol{x}_2) \\
&= \boldsymbol{C}_2^{-1}(\boldsymbol{f}_2 - \boldsymbol{B}_2\boldsymbol{s}_2) - \boldsymbol{C}_2^{-1}(\boldsymbol{A}_2 + \boldsymbol{B}_2\boldsymbol{T}_2)\boldsymbol{x}_1 := \boldsymbol{s}_3 + \boldsymbol{T}_3\boldsymbol{x}_1, \\
&\vdots \\
\boldsymbol{x}_m &= \boldsymbol{C}_{m-1}^{-1}(\boldsymbol{f}_{m-1} - \boldsymbol{A}_{m-1}\boldsymbol{x}_{m-2} - \boldsymbol{B}_{m-1}\boldsymbol{x}_{m-1}) := \boldsymbol{s}_m + \boldsymbol{T}_m\boldsymbol{x}_1.
\end{aligned}$$

记 $\boldsymbol{s}_1 = \boldsymbol{0}, \boldsymbol{T}_1 = \boldsymbol{I}_r$ (r 阶单位阵), 则 $\boldsymbol{x}_1 = \boldsymbol{s}_1 + \boldsymbol{T}_1\boldsymbol{x}_1$. 于是形式上有

$$\boldsymbol{x}_k = \boldsymbol{s}_k + \boldsymbol{T}_k\boldsymbol{x}_1, \quad k = 1, 2, \cdots, m. \tag{6.36}$$

下面推导参数序列 $\{\boldsymbol{s}_k\}$ 和 $\{\boldsymbol{T}_k\}$ 的递推计算公式. 注意到由第 k 个子方程 $\boldsymbol{A}_k\boldsymbol{x}_{k-1}+\boldsymbol{B}_k\boldsymbol{x}_k+\boldsymbol{C}_k\boldsymbol{x}_{k+1}=\boldsymbol{f}_k$ 可得

$$\begin{aligned}\boldsymbol{x}_{k+1}&=\boldsymbol{C}_k^{-1}(\boldsymbol{f}_k-\boldsymbol{A}_k\boldsymbol{x}_{k-1}-\boldsymbol{B}_k\boldsymbol{x}_k)\\&=\boldsymbol{C}_k^{-1}(\boldsymbol{f}_k-\boldsymbol{A}_k\boldsymbol{s}_{k-1}-\boldsymbol{B}_k\boldsymbol{s}_k)-\boldsymbol{C}_k^{-1}(\boldsymbol{A}_k\boldsymbol{T}_{k-1}+\boldsymbol{B}_k\boldsymbol{T}_k)\boldsymbol{x}_1.\end{aligned}$$

将上式与式 (6.36) 比较, 得

$$\begin{aligned}&\boldsymbol{s}_1=\boldsymbol{0},\quad \boldsymbol{s}_2=\boldsymbol{C}_1^{-1}\boldsymbol{f}_1,\\&\boldsymbol{s}_{k+1}=\boldsymbol{C}_k^{-1}(\boldsymbol{f}_k-\boldsymbol{A}_k\boldsymbol{s}_{k-1}-\boldsymbol{B}_k\boldsymbol{s}_k),\ k=2,\cdots,m-1,\\&\boldsymbol{T}_1=\boldsymbol{I}_r,\quad \boldsymbol{T}_2=-\boldsymbol{C}_1^{-1}\boldsymbol{B}_1,\\&\boldsymbol{T}_{k+1}=-\boldsymbol{C}_k^{-1}(\boldsymbol{A}_k\boldsymbol{T}_{k-1}+\boldsymbol{B}_k\boldsymbol{T}_k),\ k=2,\cdots,m-1.\end{aligned}$$

将式 (6.36) 代入式 (6.33) 的第 m 个子方程, 得

$$(\boldsymbol{A}_m\boldsymbol{T}_{m-1}+\boldsymbol{B}_m\boldsymbol{T}_m)\boldsymbol{x}_1=\boldsymbol{f}_m-\boldsymbol{A}_m\boldsymbol{s}_{m-1}-\boldsymbol{B}_m\boldsymbol{s}_m. \tag{6.37}$$




若 r 阶方程组 (6.37) 有解 $\boldsymbol{x}_1$, 则可由式 (6.36) 确定方程组 $\boldsymbol{Ax}=\boldsymbol{f}$ 的唯一解. 称这种方法为解块三对角方程组的双参数法.

因为 $\boldsymbol{A}$ 可逆, 故方程组 $\boldsymbol{Ax}=\boldsymbol{f}$ 有唯一解, 从而 $\boldsymbol{x}_1$ 存在. 假设方程组 (6.37) 有无穷多组解, 则对每一个解 $\boldsymbol{x}_1$, 由式 (6.36) 都可求得方程组 $\boldsymbol{Ax}=\boldsymbol{f}$ 的一组解, 这与方程组 $\boldsymbol{Ax}=\boldsymbol{f}$ 有唯一解矛盾. 因此 $\boldsymbol{x}_1$ 存在且唯一, 从而方程组 (6.37) 的系数矩阵 $\boldsymbol{A}_m\boldsymbol{T}_{m-1}+\boldsymbol{B}_m\boldsymbol{T}_m$ 非奇异, 且有

$$\boldsymbol{x}_1=(\boldsymbol{A}_m\boldsymbol{T}_{m-1}+\boldsymbol{B}_m\boldsymbol{T}_m)^{-1}(\boldsymbol{f}_m-\boldsymbol{A}_m\boldsymbol{s}_{m-1}-\boldsymbol{B}_m\boldsymbol{s}_m),$$

代入 (6.36) 即可求得方程组 $\boldsymbol{Ax}=\boldsymbol{f}$ 的唯一解. 算法如下.

算法 6.11 (双参数法) 计算块三对角方程组 $\boldsymbol{Ax}=\boldsymbol{f}$ 的解.

$\boldsymbol{s}_1=\boldsymbol{0}$; $\boldsymbol{s}_2=\boldsymbol{C}_1^{-1}\boldsymbol{f}_1$; $\boldsymbol{T}_1=\boldsymbol{I}_r$; $\boldsymbol{T}_2=-\boldsymbol{C}_1^{-1}\boldsymbol{B}_1$;

for $k=2:m-1$

$$\boldsymbol{s}_{k+1} = \boldsymbol{C}_k^{-1}(\boldsymbol{f}_k - \boldsymbol{A}_k\boldsymbol{s}_{k-1} - \boldsymbol{B}_k\boldsymbol{s}_k);$$

$$\boldsymbol{T}_{k+1} = -\boldsymbol{C}_k^{-1}(\boldsymbol{A}_k\boldsymbol{T}_{k-1} + \boldsymbol{B}_k\boldsymbol{T}_k);$$

end

$$\boldsymbol{x}_1 = (\boldsymbol{A}_m\boldsymbol{T}_{m-1} + \boldsymbol{B}_m\boldsymbol{T}_m)^{-1}(\boldsymbol{f}_m - \boldsymbol{A}_m\boldsymbol{s}_{m-1} - \boldsymbol{B}_m\boldsymbol{s}_m);$$

for $k = 2 : m$

$$\boldsymbol{x}_k = \boldsymbol{s}_k + \boldsymbol{T}_k\boldsymbol{x}_1;$$

end

使用双参数法求解三对角方程组 $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{f}$ 需要的乘除法次数约为

$$mr^2\left(\frac{13}{3}r + 4\right) - r^2(4r + 3).$$

双参数法的 MATLAB 程序如下:

```
%双参数法程序-mdouble_par.m
```

```
function [x]=mdouble_par(Ai,Bi,Ci,fi,m)
%用双参数法解三对角方程组Ax=f
%输入:Ai为A的次下对角块,Bi为A的主对角块,
%     Ci为A的次上对角块,fi为右端向量
%输出:解向量x
x=cell(m,1); s=cell(m,1); T=cell(m,1);
s{1}=zeros(3,1); s{2}=Ci{1}\fi{1};
T{1}=eye(3); T{2}=-Ci{1}\Bi{1};
for k=2:m-1
    s{k+1}=Ci{k}\(fi{k}-Ai{k}*s{k-1}-Bi{k}*s{k});
    T{k+1}=-Ci{k}\(Ai{k}*T{k-1}+Bi{k}*T{k});
end
p=fi{m}-Ai{m}*s{m-1}-Bi{m}*s{m}';
```




Back
Close

```
x{1}=(Ai{m}*T{m-1}+Bi{m}*T{m})\p;
for k=2:m
    x{k}=s{k}+T{k}*x{1};
end
```

例 6.12 选取子矩阵分别为

$$\boldsymbol{B}_k=\begin{bmatrix}4&-1&0\\-1&4&-1\\0&-1&4\end{bmatrix},\quad \boldsymbol{A}_k=\boldsymbol{C}_k^{\mathrm{T}}=\begin{bmatrix}13&0&0\\0&11&0\\1&0&12\end{bmatrix},\ k=1,2,\cdots,m.$$

分别对 $m=10^3,5\times10^3,10^4,5\times10^4,10^5,5\times10^5$ 使用块追赶法和双参数法求解块三对角矩阵 $\boldsymbol{A}\boldsymbol{x}=\boldsymbol{f}$, 其中 $\boldsymbol{f}_k=(1,0,1)^{\mathrm{T}}$. 并将所得数值解 $\boldsymbol{x}$ 代入 $\boldsymbol{A}\boldsymbol{x}=\boldsymbol{f}$, 各子方程的误差为

$$\delta_1=\|\boldsymbol{B}_1\boldsymbol{x}_1+\boldsymbol{C}_1\boldsymbol{x}_2-\boldsymbol{f}_1\|_2,$$
$$\delta_k=\|\boldsymbol{A}_k\boldsymbol{x}_{k-1}+\boldsymbol{B}_k\boldsymbol{x}_k+\boldsymbol{C}_k\boldsymbol{x}_{k+1}-\boldsymbol{f}_k\|_2,\quad k=2,\cdots,m-1,$$




Back
Close

$$\delta_m = \|\boldsymbol{A}_m\boldsymbol{x}_{m-1} + \boldsymbol{B}_m\boldsymbol{x}_m - \boldsymbol{f}_m\|_2.$$

两种方法的计算时间 (PC Intel Core i5-3470 CPU 3.2GHz, MATLAB R2015b) 和各子方程的误差最大值 $\max\{\delta_k\}_{k=1}^m$ 如表 6.3 所示.

表 6.3 块追赶法和双参数法的数值结果

维数	块追赶法		双参数法	
	误差	计算时间	误差	计算时间
1×10^3	1.9956e-11	0.0319	5.5943e-16	0.0235
5×10^3	1.8646e-11	0.1277	7.0217e-16	0.0821
1×10^4	1.2669e-10	0.2503	4.9772e-16	0.1551
5×10^4	5.1618e-11	1.2301	8.8991e-16	0.7532
1×10^5	1.2903e-09	2.4382	8.9509e-16	1.4845
5×10^5	8.6523e-10	12.1268	6.2942e-16	7.4758

从表 6.3 可以看出, 对于此例, 双参数法要比块追赶法有效得多.

例 6.13 选取子矩阵分别为

$$\boldsymbol{B}_k = \begin{bmatrix} 2 & -1 & 0 \\ -2 & 1 & 0 \\ 0 & 0 & 3 \end{bmatrix}, \quad \boldsymbol{A}_k = \boldsymbol{C}_k = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix}, \ k = 1, 2, \cdots, m.$$

分别对 $m = 10^3, 5\times10^3, 10^4, 5\times10^4, 10^5, 5\times10^5$ 使用块追赶法和双参数法求解块三对角矩阵 $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{f}$, 其中 $\boldsymbol{f}_k = (1, 2, 1)^{\mathrm{T}}$. 两种方法的计算时间 (PC Intel Core i5-3470 CPU 3.2GHz, MATLAB R2015b) 和各子方程的误差最大值 $\max\{\delta_k\}_{k=1}^m$ 如表 6.4 所示.

表 6.4 双参数法的数值结果

维数	块追赶法	双参数法	
		误差	计算时间
1×10^3	失效	4.4409e-16	0.0240
5×10^3	失效	4.4409e-16	0.0848
1×10^4	失效	5.5511e-16	0.1622
5×10^4	失效	4.4409e-16	0.7594
1×10^5	失效	6.6613e-16	1.5165
5×10^5	失效	5.5511e-16	7.4960

从表 6.4 可以看出, 当块追赶法失效时, 双参数法依然十分有效. 当然, 也可以构造出块追赶法有效而双参数法失效的算例. 由此可见, 块追赶法和双参数法可以互为补充.

§6.5 直接法的舍入误差分析

本节对用直接法求解方程组得到的解进行舍入误差分析. 下面先介绍矩阵的条件数, 它是判断矩阵病态与否的一种度量.

§6.5.1 矩阵的条件数

定义 6.4 设 $\boldsymbol{A}$ 为非奇异矩阵, 称 $\kappa(\boldsymbol{A}) = \|\boldsymbol{A}^{-1}\| \cdot \|\boldsymbol{A}\|$ 为矩阵 $\boldsymbol{A}$ 的条件数, 这里 $\|\cdot\|$ 是任意的算子范数.

从定义 6.4 知道, $\kappa(\boldsymbol{A}) = \kappa(\boldsymbol{A}^{-1})$. 常用的条件数有

(1) 无穷范数条件数

$$\kappa(\boldsymbol{A})_\infty = \|\boldsymbol{A}^{-1}\|_\infty \cdot \|\boldsymbol{A}\|_\infty.$$

(2) 谱条件数

$$\kappa(\boldsymbol{A})_2 = \|\boldsymbol{A}^{-1}\|_2 \cdot \|\boldsymbol{A}\|_2 = \sqrt{\frac{\lambda_{\max}(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A})}{\lambda_{\min}(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A})}}.$$

当 $\boldsymbol{A}$ 为对称矩阵时, $\kappa(\boldsymbol{A})_2 = |\lambda_1|/|\lambda_n|$, 其中 λ_1 和 λ_n 分别是 $\boldsymbol{A}$ 的绝对值最大和绝对值最小的特征值.

容易验证, 矩阵的条件数有如下性质:

(1) 任意非零矩阵的条件数 $\kappa(\boldsymbol{A}) \geqslant 1$.

(2) 若 $c \neq 0$ 为常数, 则 $\kappa(c\boldsymbol{A}) = \kappa(\boldsymbol{A})$.

(3) 若 $\boldsymbol{A}$ 为正交矩阵, 则 $\kappa(\boldsymbol{A})_2 = 1$.

(4) 若 $\boldsymbol{A}$ 为非奇异矩阵, $\boldsymbol{Q}$ 为正交矩阵, 则

$$\kappa(\boldsymbol{Q}\boldsymbol{A})_2 = \kappa(\boldsymbol{A}\boldsymbol{Q})_2 = \kappa(\boldsymbol{A})_2.$$

§6.5.2 矩阵条件数的估算

本节介绍一种非常实用的方法来估计矩阵无穷范数条件数 $\kappa(\boldsymbol{A})_\infty$. 显然

$$\|\boldsymbol{A}\|_\infty = \max_{1\leqslant i\leqslant n} \sum_{j=1}^{n} |a_{ij}|$$

可以直接计算, 主要的困难在于计算 $\|\boldsymbol{A}^{-1}\|_\infty$.

下面介绍的算法是由 Cline 等[24]给出的, 它在列主元 LU 分解 $\boldsymbol{PA} = \boldsymbol{LU}$ 的基础上, 再增加 $O(n^2)$ 的工作量, 就可以给出条件数的量级上的估计, 关键的是这个过程中不需要求 $\boldsymbol{A}^{-1}$.

设 $\boldsymbol{y}$ 为方程组 $\boldsymbol{Ay} = \boldsymbol{d}$ 的解, 则

$$\|\boldsymbol{A}^{-1}\|_\infty \geqslant \frac{\|\boldsymbol{y}\|_\infty}{\|\boldsymbol{d}\|_\infty}.$$

上式表明, 线性方程组的解和右端项在无穷范数意义下的比是 $\|\boldsymbol{A}^{-1}\|_\infty$ 的一个下界. 如果能选取向量 $\boldsymbol{d}$ 使得 $\|\boldsymbol{y}\|_\infty/\|\boldsymbol{d}\|_\infty$ 尽量大, 那么这个比值将会是 $\|\boldsymbol{A}^{-1}\|_\infty$ 的一个很好的估计. 下面讨论如何选取 $\boldsymbol{d}$. 设 $\boldsymbol{A}$ 的奇异值分解为

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}},$$

式中: $\boldsymbol{U}$ 和 $\boldsymbol{V}$ 为正交阵; $\boldsymbol{\Sigma} = \mathrm{diag}(\sigma_1, \sigma_2, \cdots, \sigma_n)$ 为由 $\boldsymbol{A}$ 的奇异值构成的对角矩阵.

不失一般性, 设 $\sigma_1 \geqslant \sigma_2 \geqslant \cdots \geqslant \sigma_n > 0$. 设 $\boldsymbol{d}$ 为方程组 $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{d} = \boldsymbol{b}$ 的解, $\boldsymbol{b} \in \mathbb{R}^n$. 令 $\boldsymbol{V}^{\mathrm{T}}\boldsymbol{b} = (\alpha_1, \alpha_2, \cdots, \alpha_n)^{\mathrm{T}}$, 则有

$$\|\boldsymbol{d}\|_2^2 = \|\boldsymbol{A}^{-\mathrm{T}}\boldsymbol{b}\|_2^2 = \|\boldsymbol{U}\boldsymbol{\Sigma}^{-1}\boldsymbol{V}^{\mathrm{T}}\boldsymbol{b}\|_2^2 = \|\boldsymbol{\Sigma}^{-1}\boldsymbol{V}^{\mathrm{T}}\boldsymbol{b}\|_2^2 = \sum_{i=1}^{n}(\sigma_i^{-1}\alpha_i)^2, \tag{6.38}$$

及

$$\|\boldsymbol{y}\|_2^2 = \|\boldsymbol{A}^{-1}\boldsymbol{d}\|_2^2 = \|\boldsymbol{A}^{-1}(\boldsymbol{A}^{-\mathrm{T}}\boldsymbol{b})\|_2^2 = \|\boldsymbol{V}\boldsymbol{\Sigma}^{-2}\boldsymbol{V}^{\mathrm{T}}\boldsymbol{b}\|_2^2$$
$$= \|\boldsymbol{\Sigma}^{-2}\boldsymbol{V}^{\mathrm{T}}\boldsymbol{b}\|_2^2 = \sum_{i=1}^{n}(\sigma_i^{-2}\alpha_i)^2, \tag{6.39}$$

注意到

$$\sum_{i=1}^{n}(\sigma_i^{-2}\alpha_i)^2 \leqslant \frac{1}{\sigma_n^2}\sum_{i=1}^{n}(\sigma_i^{-1}\alpha_i)^2,$$

从而有

$$\frac{1}{\sigma_n} \geqslant \frac{\|\boldsymbol{y}\|_2}{\|\boldsymbol{d}\|_2}.$$

从上式可知, $1/\sigma_n$ 是 $\|\boldsymbol{y}\|_2/\|\boldsymbol{d}\|_2$ 的上界, 且当 α_n 越大时, $\|\boldsymbol{y}\|_2/\|\boldsymbol{d}\|_2$ 越接近 $1/\sigma_n\,(=\|\boldsymbol{A}^{-1}\|_2)$. 利用范数的等价性, $\|\boldsymbol{y}\|_\infty/\|\boldsymbol{d}\|_\infty$ 也会越大. 假设 $\boldsymbol{A}$ 有列主元 LU 分解 $\boldsymbol{PA} = \boldsymbol{LU}$. 令 $\widetilde{\boldsymbol{d}} = \boldsymbol{Pd}$, 那么有

$$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{d} = \boldsymbol{U}^{\mathrm{T}}\boldsymbol{L}^{\mathrm{T}}\widetilde{\boldsymbol{d}} = \boldsymbol{b},$$

而方程组 $\boldsymbol{Ay} = \boldsymbol{d}$ 等价于

$$\boldsymbol{PAy} = \boldsymbol{LUy} = \boldsymbol{Pd} = \widetilde{\boldsymbol{d}},$$

注意到 $\|\boldsymbol{Pd}\|_\infty = \|\boldsymbol{d}\|_\infty$, 则计算 $\|\boldsymbol{y}\|_\infty/\|\boldsymbol{d}\|_\infty$ 可以通过下面几步求得:

(1) 求解 $\boldsymbol{U}^{\mathrm{T}}\boldsymbol{x} = \boldsymbol{b}$ 和 $\boldsymbol{L}^{\mathrm{T}}\widetilde{\boldsymbol{d}} = \boldsymbol{x}$.

(2) 求解 $\boldsymbol{Lz} = \widetilde{\boldsymbol{d}}$ 和 $\boldsymbol{Uy} = \boldsymbol{z}$.

(3) 计算 $\|\boldsymbol{y}\|_\infty/\|\widetilde{\boldsymbol{d}}\|_\infty$.

现考虑向量 $\boldsymbol{b}$ 的选取. 由式 (6.38) 可知, 若 α_n 越大, $\|\boldsymbol{d}\|_2$ 也越大, 所以可以考虑选取向量 $\boldsymbol{b}$ 使得 $\boldsymbol{x}$ 的无穷范数尽可能大. 此外, 由于 $\boldsymbol{b}$ 乘上一个非零常数对 $\|\boldsymbol{y}\|_\infty/\|\boldsymbol{d}\|_\infty$ 的值没有影响, 不妨要求 $\boldsymbol{b}$ 的分量为 1 或 -1. 求解 $\boldsymbol{U}^{\mathrm{T}}\boldsymbol{x} = \boldsymbol{b}$ 的过程可通过下面算法给出:

$x_1 = b_1/u_{11}$; $\beta_1 = 0$;

for $k = 2 : n$

$$\beta_k = \sum_{i=1}^{k-1} u_{ik}x_i; \quad x_k = (b_k - \beta_k)/u_{kk};$$

end

为了使得 $\|\boldsymbol{x}\|_\infty$ 尽可能大, 取 $b_k = -\text{sign}(\beta_k)$, 则

$$|x_k| = \frac{|\text{sign}(\beta_k) + \beta_k|}{|u_{kk}|} = \frac{1 + |\beta_k|}{|u_{kk}|},$$

所以当 $|\beta_k|$ 越大时, $|x_k|$ 也会越大.

矩阵无穷范数条件数估算的 MATLAB 程序如下:

```
function [kappa]=cond_inf(A)
%输入:矩阵A
%输出:矩阵A无穷范数条件数的估计值
n=size(A,1);[L,U]=lu(A); %列主元LU分解PA=LU
%求解U'x=b
```


```
beta(1)=0; x=zeros(n,1);b(1)=1; x(1)=b(1)/U(1,1);
for k=2:n
    beta(k)=U(1:k-1,k)'*x(1:k-1);
    a=beta(k); b(k)=-sign(a);
    x(k)=(b(k)-beta(k))/U(k,k);
end
dw=L'\x; z=L\dw; y=U\z;
h2=norm(y,'inf')/norm(dw,'inf');
h1=norm(A,'inf');kappa=h1*h2;  %矩阵A的无穷范数条件数
```

例 6.14 利用程序 cond_inf.m 估算下面矩阵 $\boldsymbol{A}$ 的无穷范数条件数 $\kappa(A)_\infty$：

$$\boldsymbol{A} = \begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix}_{n\times n}.$$

解 取 $n = 128$, 在 MATLAB 命令窗口依次输入:

```
>> n=128;e=ones(n-1,1);A=2*eye(n)+diag(e,-1)+diag(e,1);
>> [kappa]=cond_inf(A)
```

即可得到数值结果 (略).

§6.5.3 舍入误差对解的影响

用直接法解线性方程组 $\boldsymbol{Ax} = \boldsymbol{b}$, 由于测量误差或舍入误差的存在, 导致 $\boldsymbol{A}$ 和 $\boldsymbol{b}$ 有误差 $\delta\boldsymbol{A}$ 和 $\delta\boldsymbol{b}$. $\delta\boldsymbol{A}$ 和 $\delta\boldsymbol{b}$ 同计算机运算和精度有关, 计算精度越高, $\|\delta\boldsymbol{A}\|$ 和 $\|\delta\boldsymbol{b}\|$ 必然越小. 下面分析 $\|\delta\boldsymbol{A}\|$ 和 $\|\delta\boldsymbol{b}\|$ 对解的扰动的影响.

定义 6.5 如果矩阵 $\boldsymbol{A}$ 或 $\boldsymbol{b}$ 的微小变化, 引起方程组 $\boldsymbol{Ax} = \boldsymbol{b}$ 的解的巨大变化, 则称方程组是病态的, 称矩阵 $\boldsymbol{A}$ 为病态矩阵. 否则, 称方程组是良态的, 称矩阵 $\boldsymbol{A}$ 为良态矩阵.

当 $\boldsymbol{A}$ 和 $\boldsymbol{b}$ 都有微小的变化时, 解的变化可以用下面定理描述.

定理 6.6 设 $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ 为非奇异矩阵, $\delta\boldsymbol{A} \in \mathbb{R}^{n\times n}$ 满足 $\|\delta\boldsymbol{A}\| \cdot \|\boldsymbol{A}^{-1}\| < 1$. 若 $\boldsymbol{x}$ 和 $\delta\boldsymbol{x}$ 满足

$$\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b},\quad (\boldsymbol{A}+\delta\boldsymbol{A})(\boldsymbol{x}+\delta\boldsymbol{x})=\boldsymbol{b}+\delta\boldsymbol{b}, \tag{6.40}$$

则

$$\frac{\|\delta\boldsymbol{x}\|}{\|\boldsymbol{x}\|}\leqslant\frac{\kappa(\boldsymbol{A})}{1-\kappa(\boldsymbol{A})\varepsilon_r(\boldsymbol{A})}\big(\varepsilon_r(\boldsymbol{A})+\varepsilon_r(\boldsymbol{b})\big), \tag{6.41}$$

式中: $\varepsilon_r(\boldsymbol{A})=\dfrac{\|\delta\boldsymbol{A}\|}{\|\boldsymbol{A}\|}$, $\varepsilon_r(\boldsymbol{b})=\dfrac{\|\delta\boldsymbol{b}\|}{\|\boldsymbol{b}\|}$, 这里的矩阵范数是由向量范数诱导出来的算子范数.

证明 由于 $\boldsymbol{A}+\delta\boldsymbol{A}=(\boldsymbol{I}+\delta\boldsymbol{A}\boldsymbol{A}^{-1})\boldsymbol{A}$ 和 $\|\delta\boldsymbol{A}\boldsymbol{A}^{-1}\|\leqslant\|\delta\boldsymbol{A}\|\cdot\|\boldsymbol{A}^{-1}\|<1$, 则 $\boldsymbol{A}+\delta\boldsymbol{A}$ 可逆且

$$\|(\boldsymbol{I}+\delta\boldsymbol{A}\boldsymbol{A}^{-1})^{-1}\|\leqslant\frac{1}{1-\|\boldsymbol{A}^{-1}\|\cdot\|\delta\boldsymbol{A}\|},$$

可知

$$\|(\boldsymbol{A}+\delta\boldsymbol{A})^{-1}\|=\|\boldsymbol{A}^{-1}(\boldsymbol{I}+\delta\boldsymbol{A}\boldsymbol{A}^{-1})^{-1}\|\leqslant\frac{\|\boldsymbol{A}^{-1}\|}{1-\|\boldsymbol{A}^{-1}\|\cdot\|\delta\boldsymbol{A}\|}.$$




Back
Close

此外, 由式 (6.40), 得

$$\begin{aligned}\delta\boldsymbol{x} &= (\boldsymbol{A}+\delta\boldsymbol{A})^{-1}(\boldsymbol{b}+\delta\boldsymbol{b})-\boldsymbol{x}\\ &= (\boldsymbol{A}+\delta\boldsymbol{A})^{-1}\boldsymbol{b}+(\boldsymbol{A}+\delta\boldsymbol{A})^{-1}\delta\boldsymbol{b}-\boldsymbol{A}^{-1}\boldsymbol{b}\\ &= \big((\boldsymbol{A}+\delta\boldsymbol{A})^{-1}-\boldsymbol{A}^{-1}\big)\boldsymbol{b}+(\boldsymbol{A}+\delta\boldsymbol{A})^{-1}\delta\boldsymbol{b}\\ &= -(\boldsymbol{A}+\delta\boldsymbol{A})^{-1}\delta\boldsymbol{A}\boldsymbol{A}^{-1}\boldsymbol{b}+(\boldsymbol{A}+\delta\boldsymbol{A})^{-1}\delta\boldsymbol{b}\\ &= -(\boldsymbol{A}+\delta\boldsymbol{A})^{-1}\delta\boldsymbol{A}\boldsymbol{x}+(\boldsymbol{A}+\delta\boldsymbol{A})^{-1}\delta\boldsymbol{b}.\end{aligned}$$

从而

$$\begin{aligned}\|\delta\boldsymbol{x}\| &\leqslant \|(\boldsymbol{A}+\delta\boldsymbol{A})^{-1}\|\cdot\|\delta\boldsymbol{A}\|\cdot\|\boldsymbol{x}\|+\|(\boldsymbol{A}+\delta\boldsymbol{A})^{-1}\|\cdot\|\delta\boldsymbol{b}\|\\ &\leqslant \frac{\|\boldsymbol{A}^{-1}\|}{1-\|\boldsymbol{A}^{-1}\|\cdot\|\delta\boldsymbol{A}\|}\big(\|\delta\boldsymbol{A}\|\cdot\|\boldsymbol{x}\|+\|\delta\boldsymbol{b}\|\big),\end{aligned}$$

两边除以 $\|\boldsymbol{x}\|$, 注意到 $\|\boldsymbol{b}\| = \|\boldsymbol{A}\boldsymbol{x}\| \leqslant \|\boldsymbol{A}\| \cdot \|\boldsymbol{x}\|$, 可得

$$\begin{aligned}
\frac{\|\delta \boldsymbol{x}\|}{\|\boldsymbol{x}\|} &\leqslant \frac{\|\boldsymbol{A}^{-1}\|}{1 - \|\boldsymbol{A}^{-1}\| \cdot \|\delta \boldsymbol{A}\|} \left(\|\delta \boldsymbol{A}\| + \frac{\|\delta \boldsymbol{b}\|}{\|\boldsymbol{x}\|} \right) \\
&= \frac{\|\boldsymbol{A}^{-1}\| \cdot \|\boldsymbol{A}\|}{1 - \|\boldsymbol{A}^{-1}\| \cdot \|\boldsymbol{A}\| \cdot \frac{\|\delta \boldsymbol{A}\|}{\|\boldsymbol{A}\|}} \left(\frac{\|\delta \boldsymbol{A}\|}{\|\boldsymbol{A}\|} + \frac{\|\delta \boldsymbol{b}\|}{\|\boldsymbol{A}\| \cdot \|\boldsymbol{x}\|} \right) \\
&\leqslant \frac{\|\boldsymbol{A}^{-1}\| \cdot \|\boldsymbol{A}\|}{1 - \|\boldsymbol{A}^{-1}\| \cdot \|\boldsymbol{A}\| \cdot \frac{\|\delta \boldsymbol{A}\|}{\|\boldsymbol{A}\|}} \left(\frac{\|\delta \boldsymbol{A}\|}{\|\boldsymbol{A}\|} + \frac{\|\delta \boldsymbol{b}\|}{\|\boldsymbol{b}\|} \right),
\end{aligned}$$

即得到式 (6.41). 证毕. □

根据定理 6.6, 当 $\kappa(\boldsymbol{A})$ 越大时, $\boldsymbol{A}$ 和 $\boldsymbol{b}$ 扰动之后得到的解的相对误差也越大, 相应的方程组也越病态.

对于病态的方程组, 为了得到较精确的近似解, 可以采用如下措施来减少舍入误差的影响: ① 采用高精度计算; ② 采用数值稳定性较好的算法; ③ 采用迭代改善计算解的办法.

设 x 是方程组 $Ax = b$ 的一个近似解, 如果它没有达到精度要求, 怎样产生一个更好的近似解呢? 记 $r = b - Ax$, 由于 x 为近似解, 所以 $r \neq 0$. 考虑方程组 $Az = r$, 如果 z 是 $Az = r$ 的精确解, 则 $A(x + z) = Ax + r = b$, 即 $x + z$ 为 $Ax = b$ 的一个精确解. 实际中 z 可能还是 $Az = r$ 的近似解. 当 $x + z$ 还是不够精确时, 把 $x + z$ 看作上述的 x, 重复上述过程. 设矩阵 A 有列主元 LU 分解 $PA = LU$, 下面给出对近似解的迭代改进:

(1) $r := b - Ax$; (2) 解方程组 $Ly = Pr$, 得到 y;

(3) 解方程组 $Uz = y$, 得到 z; (4) 令 $x := x + z$;

(5) 若 x 达到精度的要求, 停算; 否则, 转 (1).