

## 第二讲 线性方程组直接解法

- 1 Gauss 消去法和 LU 分解
- 2 特殊方程组的求解
- 3 扰动分析
- 4 误差分析
- 5 解的改进和条件数估计

# 线性方程组的求解方法

- **直接法**: LU 分解, Cholesky 分解, ...
- **迭代法**: 古典迭代法, Krylov 子空间迭代法

本章介绍直接法, 即 **Gauss 消去法** 或 **PLU 分解**

**直接法优点**: 稳定可靠  $\rightarrow$  在工程计算中很受欢迎

**直接法缺点**: 运算量大  $O(n^3) \rightarrow$  不适合大规模稀疏线性方程组  
(针对特殊结构矩阵的快速方法除外)

# 1 Gauss 消去法和 LU 分解

## 1.1 LU 分解

## 1.2 LU 分解的实现

## 1.3 IKJ 型 LU 分解

## 1.4 待定系数法计算 LU 分解

## 1.5 三角方程求解

## 1.6 选主元 LU 分解

## 1.7 矩阵求逆

## 1.1 LU 分解

考虑线性方程组

$$Ax = b \quad (2.1)$$

其中  $A \in \mathbb{R}^{n \times n}$  非奇异,  $b \in \mathbb{R}^n$  为给定的右端项.

Gauss 消去法本质上就是对系数矩阵  $A$  进行 LU 分解:

$$A = LU \quad (2.2)$$

其中  $L$  是单位下三角矩阵,  $U$  为非奇异上三角矩阵.

分解 (2.2) 就称为 LU 分解

$$Ax = b \iff \begin{cases} Ly = b, \\ Ux = y. \end{cases} \implies \text{只需求解两个三角方程组}$$

---

### 算法 1.1 Gauss 消去法

---

- 1: 将  $A$  进行 LU 分解:
  - 2:  $A = LU$ , 其中  $L$  为单位下三角矩阵,  $U$  为非奇异上三角矩阵;
  - 3: 向前回代: 求解  $Ly = b$ , 即得  $y = L^{-1}b$ ;
  - 4: 向后回代: 求解  $Ux = y$ , 即得  $x = U^{-1}y = (LU)^{-1}b = A^{-1}b$ .
-

† 需要指出的是:  $A$  非奇异, 则解存在唯一, 但并不一定存在 LU 分解!

**定理 (LU 分解的存在性和唯一性)** 设  $A \in \mathbb{R}^{n \times n}$ . 则存在唯一的单位下三角矩阵  $L$  和非奇异上三角矩阵  $U$ , 使得  $A = LU$  的充要条件是  $A$  的所有顺序主子矩阵  $A_k = A(1:k, 1:k)$  都非奇异,  $k = 1, 2, \dots, n$ . (板书)

## 1.2 LU 分解的实现 — 矩阵初等变换

给定一个矩阵

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & \ddots & \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \in \mathbb{R}^{n \times n}.$$

- 第一步: 假定  $a_{11} \neq 0$ , 构造矩阵

$$L_1 = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ l_{21} & 1 & 0 & \cdots & 0 \\ l_{31} & 0 & 1 & \cdots & 0 \\ \vdots & & & \ddots & \\ l_{n1} & 0 & 0 & \cdots & 1 \end{bmatrix}, \quad \text{其中} \quad l_{i1} = \frac{a_{i1}}{a_{11}}, i = 2, 3, \dots, n.$$

易知  $L_1$  的逆为

$$L_1^{-1} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ -l_{21} & 1 & 0 & \cdots & 0 \\ -l_{31} & 0 & 1 & \cdots & 0 \\ \vdots & & & \ddots & \\ -l_{n1} & 0 & 0 & \cdots & 1 \end{bmatrix}.$$

用  $L_1^{-1}$  左乘  $A$ , 并将所得到的矩阵记为  $A^{(1)}$ , 则

$$A^{(1)} = L_1^{-1} A \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & a_{22}^{(1)} & \cdots & a_{2n}^{(1)} \\ \vdots & \vdots & \ddots & \\ 0 & a_{n2}^{(1)} & \cdots & a_{nn}^{(1)} \end{bmatrix}.$$

即左乘  $L_1^{-1}$  后,  $A$  的第一列中除第一个元素外其它都变为 0.



- 第二步: 将上面的操作作用在  $A^{(1)}$  的子矩阵  $A^{(1)}(2:n, 2:n)$  上, 将其第一列除第一个元素外都变为 0: 假定  $a_{22}^{(1)} \neq 0$ , 构造矩阵

$$L_2 = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & l_{32} & 1 & \cdots & 0 \\ \vdots & \vdots & & \ddots & \\ 0 & l_{n2} & 0 & \cdots & 1 \end{bmatrix}, \quad \text{其中} \quad l_{i2} = \frac{a_{i2}^{(1)}}{a_{22}^{(1)}}, i = 3, 4, \dots, n.$$

用  $L_2^{-1}$  左乘  $A^{(1)}$ , 并将所得到的矩阵记为  $A^{(2)}$ , 则

$$A^{(2)} = L_2^{-1} A = L_2^{-1} L_1^{-1} A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & \cdots & a_{2n}^{(1)} \\ 0 & 0 & a_{33}^{(2)} & \cdots & a_{3n}^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \\ 0 & 0 & a_{n3}^{(2)} & \cdots & a_{nn}^{(2)} \end{bmatrix}.$$

- 依此类推, 假定  $a_{kk}^{(k-1)} \neq 0$  ( $k = 3, 4, \dots, n-1$ ), 则我们可以构造一系列的矩阵  $L_3, L_4, \dots, L_{n-1}$ , 使得

$$L_{n-1}^{-1} \cdots L_2^{-1} L_1^{-1} A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & \cdots & a_{2n}^{(1)} \\ 0 & 0 & a_{33}^{(2)} & \cdots & a_{3n}^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \\ 0 & 0 & 0 & \cdots & a_{nn}^{(n-1)} \end{bmatrix} \triangleq U \rightarrow \text{上三角}$$

于是可得  $A = LU$  其中

$$L = L_1 L_2 \cdots L_{n-1} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ l_{21} & 1 & 0 & \cdots & 0 \\ l_{31} & l_{32} & 1 & \cdots & 0 \\ \vdots & \vdots & & \ddots & \\ l_{n1} & l_{n2} & l_{n3} & \cdots & 1 \end{bmatrix}$$

---

## 算法 1.2 LU 分解

---

```
1: for  $k = 1$  to  $n - 1$  do
2:   for  $i = k + 1$  to  $n$  do
3:      $l_{ik} = a_{ik}/a_{kk}$     % 计算  $L$  的第  $k$  列
4:   end for
5:   for  $j = k$  to  $n$  do
6:      $u_{kj} = a_{kj}$     % 计算  $U$  的第  $k$  行
7:   end for
8:   for  $i = k + 1$  to  $n$  do
9:     for  $j = k + 1$  to  $n$  do
10:       $a_{ij} = a_{ij} - l_{ik}u_{kj}$     % 更新  $A(k + 1 : n, k + 1 : n)$ 
11:    end for
12:   end for
13: end for
```

---

## Gauss 消去法的运算量

由算法 1.2 可知, LU 分解的运算量 (加减乘除) 为

$$\sum_{i=1}^{n-1} \left( \sum_{j=i+1}^n 1 + \sum_{j=i+1}^n \sum_{k=i+1}^n 2 \right) = \sum_{i=1}^{n-1} (n-i+2(n-i)^2) = \frac{2}{3}n^3 + O(n^2).$$

加上回代过程的运算量  $O(n^2)$ , 总运算量为

$$\frac{2}{3}n^3 + O(n^2)$$

† 评价算法的一个主要指标是**执行时间**,但这依赖于计算机硬件和编程技巧等,因此直接给出算法执行时间是不太现实的. 所以我们通常是统计算法中**算术运算 (加减乘除)** 的次数.

† 在数值算法中,大多仅仅涉及加减乘除和开方运算. 一般地,加减运算次数与乘法运算次数具有相同的量级,而除法运算和开方运算次数具有更低的量级.

† 为了尽可能地减少运算量,在实际计算中,数,向量和矩阵做乘法运算时的先后执行次序为: 先计算数与向量的乘法,然后计算矩阵与向量的乘法,最后才计算矩阵与矩阵的乘法.

## 矩阵 $L$ 和 $U$ 的存储

当  $A$  的第  $i$  列被用于计算  $L$  的第  $i$  列后, 在后面的计算中不再被使用.  
同样地,  $A$  的第  $i$  行被用于计算  $U$  的第  $i$  行后, 在后面计算中也不再使用.

为了节省存储空间, 在计算过程中将  $L$  的第  $i$  列存放在  $A$  的第  $i$  列, 将  $U$  的第  $i$  行存放在  $A$  的第  $i$  行, 这样就不需要另外分配空间存储  $L$  和  $U$ .

计算结束后,  $A$  的上三角部分为  $U$ , 其绝对下三角部分为  $L$  的绝对下三角部分 ( $L$  的对角线全部为 1, 不需要存储).

---

### 算法 1.3 LU 分解

---

```
1: for  $k = 1$  to  $n - 1$  do  
2:   for  $i = k + 1$  to  $n$  do  
3:      $a_{ik} = a_{ik} / a_{kk}$   
4:     for  $j = k + 1$  to  $n$  do  
5:        $a_{ij} = a_{ij} - a_{ik}a_{kj}$   
6:     end for  
7:   end for  
8: end for
```

---

† 根据指标的循环次序, 算法 1.3 也称为 KIJ 型 LU 分解. 实际计算中一般不建议使用: 对指标  $k$  的每次循环, 都需要更新  $A$  的第  $k + 1$  至第  $n$  行, 这种反复读取数据的做法会使得计算效率大大降低.

对于按行存储的数据结构, 一般采用后面介绍的 IKJ 型 LU 分解.

## MATLAB 源代码 1 LU 分解的 Matlab 代码 (KIJ 型)

```
1  % Matlab code : LU 分解
2  function A = mylu(A)
3  n=size(A,1);
4  for k=1:n-1
5      if A(k,k) == 0
6          fprintf('Error: A(%d,%d)=0!\n', k, k);
7          return;
8      end
9      for i=k+1:n
10         A(i,k)=A(i,k)/A(k,k);
11         for j=k+1:n
12             A(i,j)=A(i,j)-A(i,k)*A(k,j);
13         end
14     end
15 end
```



为了充分利用 Matlab 的向量运算优势, 提高运算效率, 程序可改写为

### MATLAB 源代码 2 LU 分解 (KIJ 型)

```
1 function A = mylu(A)
2 n=size(A,1);
3 for k=1:n-1
4     if A(k,k) == 0
5         fprintf('Error: A(%d,%d)=0!\n', k, k);
6         return;
7     end
8     A(k+1:n,k)=A(k+1:n,k)/A(k,k);
9     A(k+1:n,k+1:n)=A(k+1:n,k+1:n)-A(k+1:n,k)*A(k,k+1:n);
10 end
```

## 1.3 IKJ 型 LU 分解

如果数据是按行存储的, 如 C/C++, 我们一般采用下面的 IKJ 型 LU 分解.

---

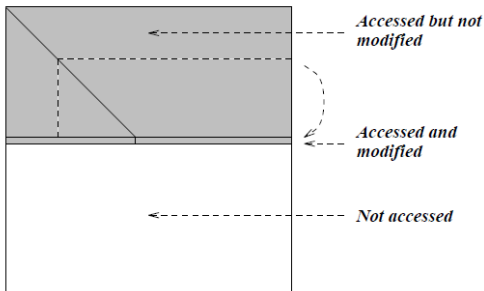
### 算法 1.4 LU 分解 (IKJ 型)

---

```
1: for  $i = 2$  to  $n$  do  
2:   for  $k = 1$  to  $i - 1$  do  
3:      $a_{ik} = a_{ik} / a_{kk}$   
4:     for  $j = k + 1$  to  $n$  do  
5:        $a_{ij} = a_{ij} - a_{ik} a_{kj}$   
6:     end for  
7:   end for  
8: end for
```

---

上述算法可以用下图来描述.



**思考** 如果数据按列存储, 如 FORTRAN/MATLAB, 如何设计算法?

## 1.4 待定系数法计算 LU 分解

设  $A = LU$ , 即

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & & & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} 1 & & & & \\ l_{21} & 1 & & & \\ l_{31} & l_{32} & 1 & & \\ \vdots & & & \ddots & \\ l_{n1} & l_{n2} & \cdots & l_{n,n-1} & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} & \cdots & u_{1n} \\ & u_{22} & u_{23} & \cdots & u_{2n} \\ & & u_{33} & \cdots & u_{3n} \\ & & & \ddots & \vdots \\ & & & & u_{nn} \end{bmatrix}$$

(1) 比较等式两边的**第一行**, 可得

$$u_{1j} = a_{1j}, \quad j = 1, 2, \dots, n.$$

再比较等式两边的**第一列**, 可得

$$a_{i1} = l_{i1}u_{11} \Rightarrow l_{i1} = a_{i1}/u_{11}, \quad i = 2, 3, \dots, n.$$

(2) 比较等式两边的**第二行**, 可得

$$a_{2j} = l_{21}u_{1j} \Rightarrow u_{2j} = a_{2j} - l_{21}u_{1j}, \quad j = 2, 3, \dots, n.$$

再比较等式两边的第二列, 可得

$$a_{i2} = l_{i1}u_{12} + l_{i2}u_{22} \Rightarrow l_{i1} = (a_{i2} - l_{i1}u_{12})/u_{22}, \quad i = 3, 4, \dots, n.$$

(3) 以此类推, 第  $k$  步时, 比较等式两边的第  $k$  行, 可得

$$u_{kj} = a_{kj} - (l_{k1}u_{1j} + \dots + l_{k,k-1}u_{k-1,j}), \quad j = k, k+1, \dots, n.$$

比较等式两边的第  $k$  列, 可得

$$l_{ik} = (a_{ik} - l_{i1}u_{1k} - \dots - l_{i,k-1}u_{k-1,k})/u_{kk}, \quad i = k+1, k+2, \dots, n.$$

直到第  $n$  步, 即可计算出  $L$  和  $U$  的所有元素.

同样, 我们可以利用  $A$  来存储  $L$  和  $U$ . 算法描述如下:

---

**算法 1.5** LU 分解 (待定系数法或 Doolittle 方法)

---

1: **for**  $k = 1$  to  $n$  **do**

$$2: \quad a_{kj} = a_{kj} - \sum_{i=1}^{k-1} a_{ki}a_{ij}, \quad j = k, k+1, \dots, n$$

$$3: \quad a_{ik} = \frac{1}{a_{kk}} \left( a_{ik} - \sum_{j=1}^{k-1} a_{ij}a_{jk} \right), \quad i = k+1, k+2, \dots, n$$

4: **end for**

---

### MATLAB 源代码 3 待定系数法 LU 分解

```
1 function A = mylu2(A)
2 [n,n]=size(A);
3 for k=1:n
4     A(k,k)=A(k,k)-A(k,1:k-1)*A(1:k-1,k);
5     if (A(k,k)==0)
6         fprintf('Error: A(%d,%d)=0!\n', i,i);
7         return;
8     end
9     A(k,k+1:n)=A(k,k+1:n)-A(k,1:k-1)*A(1:k-1,k+1:n);
10    A(k+1:n,k)=A(k+1:n,k)-A(k+1:n,1:k-1)*A(1:k-1,k);
11    A(k+1:n,k)=A(k+1:n,k)/A(k,k);
12 end
```

## 1.5 三角方程求解

得到  $A$  的 LU 分解后, 我们最后需要用回代法求解两个三角方程组

$$Ly = b, \quad Ux = y.$$

---

**算法 1.6** 向前回代求解  $Ly = b$

---

```
1:  $y_1 = b_1$ 
2: for  $i = 2 : n$  do
3:    $s = b_i$ 
4:   for  $j = 1 : i - 1$  do
5:      $s = s - u_{ij}x_j$ 
6:   end for
7:    $x_i = s$ 
8: end for
```

---



如果数据是按列存储的, 则采用列存储方式效率会高一些. 下面是按列存储方式求解上三角方程组.

---

**算法 1.7** 向后回代求解  $Ux = y$

---

```
1:  $x_n = y_n / u_{nn}$ 
2: for  $i = n - 1 : -1 : 1$  do
3:    $s = y_i$ 
4:   for  $j = i + 1 : n$  do
5:      $s = s - u_{ij}x_j$ 
6:   end for
7:    $x_i = s / u_{ii}$ 
8: end for
```

---

这两个算法的运算量均为  $n^2 + O(n)$ .

## 1.6 选主元 LU 分解

- 在 LU 分解算法 1.2 中, 我们称  $a_{kk}^{(k-1)}$  为主元. 如果  $a_{kk}^{(k-1)} = 0$ , 则算法就无法进行下去.
- 即使  $a_{kk}^{(k-1)}$  不为零, 但如果  $|a_{kk}^{(k-1)}|$  的值很小, 由于舍入误差的原因, 也可能会给计算结果带来很大的误差.
- 此时我们就需要通过 **选主元** 来解决这个问题.

例 用 LU 分解求解线性方程组  $Ax = b$ , 其中

$$A = \begin{bmatrix} 0.02 & 61.3 \\ 3.43 & -8.5 \end{bmatrix}, \quad b = \begin{bmatrix} 61.5 \\ 25.8 \end{bmatrix},$$

要求在运算过程中保留 3 位有效数字.

$$(x_1 \approx -20.7, x_2 \approx 1.01)$$

(板书)

易知, 方程的精确解为  $x_1 = 10.0$  和  $x_2 = 1.00$ . 我们发现  $x_1$  的误差非常大. 导致这个问题的原因就是  $|a_{11}|$  太小, 用它做主元时会放大舍入误差. 所以我们需要选主元.

## 选主元 LU 分解

**定理** 设  $A \in \mathbb{R}^{n \times n}$  非奇异, 则存在置换矩阵  $P_1, P_2$ , 以及单位下三角矩阵  $L$  和非奇异上三角矩阵  $U$ , 使得  $P_1 A P_2 = LU$ . 其中  $P_1$  和  $P_2$  中只有一个必需的. (板书)

### 第 $k$ 步时, 如何选取置换矩阵 $P_1$ 和 $P_2$ ?

1. 选取  $P_1$  和  $P_2$  使得主元为剩下的矩阵中绝对值最大, 这种选取方法称为“全主元 Gauss 消去法”, 简称 GECP
2. 选取  $P_1$  和  $P_2$  使得主元为第  $k$  列中第  $k$  到第  $n$  个元素中, 绝对值最大, 这种选取方法称为“部分选主元 Gauss 消去法”, 简称 GEPP 此时  $P_2 = I$ , 因此也称为列主元 Gauss 消去法.
  - GECP 比 GEPP 更稳定, 但工作量大, 实际很少使用.
  - GEPP 算法能保证  $L$  所有的元素的绝对值都不超过 1.

---

## 算法 1.8 部分选主元 LU 分解

---

```
1:  $p = 1 : n$     % 用于记录置换矩阵
2: for  $i = 1$  to  $n - 1$  do
3:    $a_{ki} = \max_{i \leq j \leq n} |a_{ji}|$     % 选列主元
4:   if  $k \neq i$  then
5:     for  $j = 1$  to  $n$  do
6:        $tmp = a_{ij}, a_{ij} = a_{kj}, a_{kj} = tmp$     % 交换第  $i$  行与第  $k$  行
7:     end for
8:      $tmp = p(k), p(k) = p(i), p(i) = tmp$     % 更新置换矩阵
9:   end if
10:  for  $j = i + 1$  to  $n$  do
11:     $a_{ji} = a_{ji}/a_{ii}$     % 计算  $L$  的第  $i$  列
12:  end for
13:  for  $j = i + 1$  to  $n$  do
14:    for  $k = i + 1$  to  $n$  do
```

```
15:          $a_{jk} = a_{jk} - a_{ji} * a_{ik}$     % 更新  $A(i + 1 : n, i + 1 : n)$ 
16:     end for
17: end for
18: end for
```

---

## MATLAB 源代码 4 部分选主元 LU 分解

```
1 function [A,p] = myplu(A)
2 [n,n]=size(A); p=1:n;
3 for i=1:n-1
4     [a,k]=max(abs(A(i:n,i)));
5     if a==0
6         error('Error: 第 %d 步的列主元为 0!\n', i);
7     end
8     k=k+i-1;
9     if k~=i
10         tmp=A(i,:); A(i,:)=A(k,:); A(k,:)=tmp;
11         tmp=p(i); p(i)=p(k); p(k)=tmp;
12     end
13     A(i+1:n,i)=A(i+1:n,i)/A(i,i);
14     A(i+1:n,i+1:n)=A(i+1:n,i+1:n)-A(i+1:n,i)*A(i,i+1:n);
15 end
```

例 用部分选主元 LU 分解求解线性方程组  $Ax = b$ , 其中

$$A = \begin{bmatrix} 0.02 & 61.3 \\ 3.43 & -8.5 \end{bmatrix}, \quad b = \begin{bmatrix} 61.5 \\ 25.8 \end{bmatrix},$$

要求在运算过程中保留 3 位有效数字.

$$(x_1 \approx 10.0, x_2 \approx 0.998)$$

(板书)



## 1.7 矩阵求逆

我们可以通过部分选主元 LU 分解来计算矩阵的逆. 设  $PA = LU$ , 则

$$A^{-1} = P^T U^{-1} L^{-1},$$

等价于求解下面  $2n$  个三角线性方程组

$$Ly_i = Pe_i, \quad Ux_i = y_i, \quad i = 1, 2, \dots, n.$$



也可以分别计算  $L^{-1}$  和  $U^{-1}$ , 然后相乘. 哪种方法划算?

## 2 特殊方程组的求解

2.1 对称正定线性方程组

2.2 对称不定线性方程组

2.3 三对角线性方程组

2.4 带状线性方程组

2.5 Toeplitz 线性方程组

## 2.1 对称正定性方程组

我们首先给出对称正定矩阵的几个基本性质.

**定理** 设  $A \in \mathbb{R}^{n \times n}$ .

- $A$  对称正定当且仅当  $A$  对称且所有特征值都是正的;
- $A$  对称正定当且仅当  $X^T A X$  对称正定, 其中  $X \in \mathbb{R}^{n \times n}$  是一个任意的非奇异矩阵;
- 若  $A$  对称正定, 则  $A$  的任意主子矩阵都对称正定;
- 若  $A$  对称正定, 则  $A$  的所有对角线元素都是正的, 且

$$\max_{i \neq j} \{|a_{ij}|\} < \max_i \{a_{ii}\},$$

即绝对值最大的元素出现在对角线上.

## Cholesky 分解

**定理 (Cholesky 分解)** 设  $A \in \mathbb{R}^{n \times n}$  对称正定, 则存在唯一的对角线元素为正的下三角矩阵  $L$ , 使得

$$A = LL^T.$$

该分解称为 Cholesky 分解.

(板书)

## Cholesky 分解的实现

设  $A = LL^T$ , 即

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} l_{11} & & & \\ l_{21} & l_{22} & & \\ \vdots & \vdots & \ddots & \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} & \cdots & l_{n1} \\ & l_{22} & \cdots & l_{n2} \\ & & \ddots & \vdots \\ & & & l_{nn} \end{bmatrix}.$$

直接比较等式两边的元素可得

$$a_{ij} = \sum_{k=1}^n l_{ik}l_{jk} = l_{jj}l_{ij} + \sum_{k=1}^{j-1} l_{ik}l_{jk}, \quad i, j = 1, 2, \dots, n.$$

根据上面的计算公式, 可得下面的算法:

---

### 算法 2.1 Cholesky 分解算法

---

1: **for**  $j = 1$  to  $n$  **do**

$$2: \quad l_{jj} = \left( a_{jj} - \sum_{k=1}^{j-1} l_{jk}^2 \right)^{1/2}$$

3:     **for**  $i = j + 1$  to  $n$  **do**

$$4: \quad \quad l_{ij} = \left( a_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk} \right) / l_{jj}$$

5:     **end for**

6: **end for**

---

## 几点说明

- 与 LU 分解一样, 可以利用  $A$  的下三角部分来存储  $L$ ;
- Cholesky 分解算法的运算量为  $\frac{1}{3}n^3 + O(n^2)$ , 大约为 LU 分解的一半;
- Cholesky 分解算法是稳定的 (稳定性与全主元 Gauss 消去法相当), 故不需要选主元.

## 改进的 Cholesky 分解算法

为了避免开方运算, 我们可以将  $A$  分解为:  $A = LDL^T$ , 即

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} 1 & & & \\ l_{21} & 1 & & \\ \vdots & & \ddots & \\ l_{n1} & \cdots & l_{n,n-1} & 1 \end{bmatrix} \begin{bmatrix} d_1 & & & \\ & d_2 & & \\ & & \ddots & \\ & & & d_n \end{bmatrix} \begin{bmatrix} 1 & l_{21} & \cdots & l_{n1} \\ & 1 & \cdots & l_{n2} \\ & & \ddots & \vdots \\ & & & 1 \end{bmatrix}.$$

通过待定系数法可得

$$a_{ij} = \sum_{k=1}^n l_{ik} d_k l_{jk} = d_j l_{ij} + \sum_{k=1}^{j-1} l_{ik} d_k l_{jk}, \quad i, j = 1, 2, \dots, n.$$

基于以上分解来求解对称正定线性方程组的算法称为**改进的平方根法**:



---

## 算法 2.2 改进的平方根法

---

```
1: for  $j = 1$  to  $n$  do    % 先计算分解
2:    $d_j = a_{jj} - \sum_{k=1}^{j-1} l_{jk}^2 d_k$ 
3:   for  $i = j + 1$  to  $n$  do
4:      $l_{ij} = (a_{ij} - \sum_{k=1}^{j-1} l_{ik} d_k l_{jk}) / d_j$ 
5:   end for
6: end for
7:  $y_1 = b_1$     % 解方程组:  $Ly = b$  和  $DL^T x = y$ 
8: for  $i = 2$  to  $n$  do
9:    $y_i = b_i - \sum_{k=1}^{i-1} l_{ik} y_k$ 
10: end for
11:  $x_n = y_n / d_n$ 
12: for  $i = n - 1$  to  $1$  do
13:    $x_i = y_i / d_i - \sum_{k=i+1}^n l_{ki} x_k$ 
14: end for
```

---

## 2.2 对称不定线性方程组

$A \rightarrow$  非奇异, 对称不定

若  $A$  存在 LU 分解, 即  $A = LU$ , 则可写成  $A = LDL^T$

然而, 当  $A$  不定时, 其 LU 分解不一定存在.

若采用选主元 LU 分解, 则其对称性将被破坏. 为了保持对称性, 在选主元时必须对行列进行同样的置换, 即选取置换矩阵  $P$ , 使得

$$PAP^T = LDL^T. \quad (2.4)$$

通常称 (2.4) 为对称矩阵的  $LDL^T$  分解.

不幸的是, 这样的置换矩阵可能不一定存在, 即分解 (2.4) 不一定存在.

例 设对称矩阵

$$A = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}.$$

由于  $A$  的对角线元素都是 0, 对任意置换矩阵  $P$ , 矩阵  $PAP^T$  的对角线元素仍然都是 0. 因此, 矩阵  $A$  不存在  $LDL^T$  分解.

## Aasen 算法

1971 年, Aasen 提出了下面的分解

$$PAP^T = LTL^T, \quad (2.5)$$

其中  $P$  为置换矩阵,  $L$  为单位下三角矩阵,  $T$  为对称三对角矩阵.  
分解 (2.5) 本质上与部分选主元 LU 分解是一样的,

## 块 LDL<sup>T</sup> 分解

设  $A$  对称非奇异, 则存在置换矩阵  $P$  使得

$$PAP^T = \begin{bmatrix} B & E^T \\ E & C \end{bmatrix},$$

其中  $B \in \mathbb{R}$  或  $B \in \mathbb{R}^{2 \times 2}$ , 且非奇异. 因此可以对  $PAP^T$  进行块对角化, 即

$$PAP^T = \begin{bmatrix} I & 0 \\ EB^{-1} & I \end{bmatrix} \begin{bmatrix} B & 0 \\ 0 & C - EA^{-1}E^T \end{bmatrix} \begin{bmatrix} I & B^{-1}E^T \\ 0 & I \end{bmatrix},$$

其中  $C - EA^{-1}E^T$  是 Schur 补.

不断重复以上过程, 就可以得到  $A$  的块 LDL<sup>T</sup> 分解:

$$PAP^T = L\tilde{D}L^T,$$

其中  $\tilde{D}$  是拟对角矩阵, 即块对角矩阵且对角块的大小为 1 或 2.

## 选主元块 $LDL^T$ 分解

与选主元 LU 分解类似, 我们需要考虑块  $LDL^T$  分解的选主元策略, 即如何选取置换矩阵. 目前常用的策略有

- 全主元策略: 由 Bunch 和 Parlett 于 1971 年提出, 并证明了其稳定性. 但需要进行  $n^3/6$  次比较运算, 代价比较昂贵.
- 部分选主元策略: 由 Bunch 和 Kaufman 于 1977 年提出, 将比较运算复杂度降低到  $O(n^2)$  量级, 而且具有较满意的向后稳定性. 因此被广泛使用.
- Rook 策略: 由 Ashcraft, Grimes 和 Lewis 于 1998 年提出, 整体上与部分选主元类似, 但在选主元时加了一层迭代, 从而能提供更高的精度.

目前大部分软件都采用部分选主元块  $LDL^T$  分解算法.

## 2.3 三对角线性方程组

$$A = \begin{bmatrix} b_1 & c_1 & & \\ a_1 & \ddots & \ddots & \\ & \ddots & \ddots & c_{n-1} \\ & & a_{n-1} & b_n \end{bmatrix}.$$

我们假定

$$|b_1| > |c_1| > 0, \quad |b_n| > |a_{n-1}| > 0, \quad (2.6)$$

$$|b_i| \geq |a_{i-1}| + |c_i|, \quad a_i c_i \neq 0, \quad i = 1, \dots, n-1. \quad (2.7)$$

即  $A$  是 **不可约弱对角占优**

**思考**

如果  $A$  可约, 怎么处理?

此时, 我们可以得到下面的三角分解

$$A = \begin{bmatrix} b_1 & c_1 & & & \\ a_1 & \ddots & \ddots & & \\ & \ddots & \ddots & c_{n-1} & \\ & & a_{n-1} & b_n & \end{bmatrix} = \begin{bmatrix} \alpha_1 & & & & \\ a_1 & \alpha_2 & & & \\ & \ddots & \ddots & & \\ & & a_{n-1} & \alpha_n & \end{bmatrix} \begin{bmatrix} 1 & \beta_1 & & & \\ & 1 & \ddots & & \\ & & \ddots & \beta_{n-1} & \\ & & & 1 & \end{bmatrix} \triangleq LU.$$

递推公式:

$$\alpha_1 = b_1,$$

$$\beta_1 = c_1/\alpha_1 = c_1/b_1,$$

$$\begin{cases} \alpha_i = b_i - a_{i-1}\beta_{i-1}, \\ \beta_i = c_i/\alpha_i = c_i/(b_i - a_{i-1}\beta_{i-1}), \quad i = 2, 3, \dots, n-1 \end{cases}$$

$$\alpha_n = b_n - a_{n-1}\beta_{n-1}.$$



为了使得算法能够顺利进行下去, 我们需要证明  $\alpha_i \neq 0$ .

**定理** 设三对角矩阵  $A$  满足条件 (2.6) 和 (2.7). 则  $A$  非奇异, 且

(1)  $|\alpha_1| = |b_1| > 0$ ;

(2)  $0 < |\beta_i| < 1, i = 1, 2, \dots, n-1$ ;

(3)  $0 < |c_i| \leq |b_i| - |a_{i-1}| < |\alpha_i| < |b_i| + |a_{i-1}|, i = 2, 3, \dots, n$ ;

(板书)

---

## 算法 2.3 追赶法

---

```
1:  $\beta_1 = c_1/b_1$ 
2:  $y_1 = f_1/b_1$ 
3: for  $i = 2$  to  $n - 1$  do
4:    $\alpha_i = b_i - a_{i-1}\beta_{i-1}$ 
5:    $\beta_i = c_i/\alpha_i$ 
6:    $y_i = (f_i - a_{i-1}y_{i-1})/\alpha_i$ 
7: end for
8:  $\alpha_n = b_n - a_{n-1}\beta_{n-1}$ 
9:  $y_n = (f_n - a_{n-1}y_{n-1})/\alpha_n$ 
10:  $x_n = y_n$ 
11: for  $i = n - 1$  to  $1$  do
12:    $x_i = y_i - \beta_i x_{i+1}$ 
13: end for
```

---

† 追赶法 (也称为 Thomas 算法) 的运算量大约为  $8n - 6$ .

† 具体计算时, 由于求解  $Ly = f$  与矩阵 LU 分解是同时进行的, 因此,  $\alpha_i$  可以不用存储. 但  $\beta_i$  需要存储.

† 由于  $|\beta_i| < 1$ , 因此在回代求解  $x_i$  时, 误差可以得到有效控制.

需要指出的是, 我们也可以考虑下面的分解

$$A = \begin{bmatrix} b_1 & c_1 & & & \\ a_1 & \ddots & \ddots & & \\ & \ddots & \ddots & c_{n-1} & \\ & & a_{n-1} & b_n & \end{bmatrix} = \begin{bmatrix} 1 & & & & \\ \gamma_1 & 1 & & & \\ & \ddots & \ddots & & \\ & & \ddots & \ddots & \\ & & & \gamma_{n-1} & 1 \end{bmatrix} \begin{bmatrix} \alpha_1 & c_1 & & & \\ & \alpha_2 & \ddots & & \\ & & \ddots & c_{n-1} & \\ & & & \ddots & \\ & & & & \alpha_n \end{bmatrix}. \quad (2.11)$$

但此时  $|\gamma_i|$  可能大于 1. 比如  $\gamma_1 = a_1/b_1$ , 因此当  $|b_1| < |a_1|$  时,  $|\gamma_1| > 1$ . 所以在回代求解时, 误差可能得不到有效控制. 另外一方面, 计算  $\gamma_i$  时也可能会产生较大的舍入误差 (大数除以小数).

但如果  $A$  是列对角占优, 则可以保证  $|\gamma_i| < 1$ .

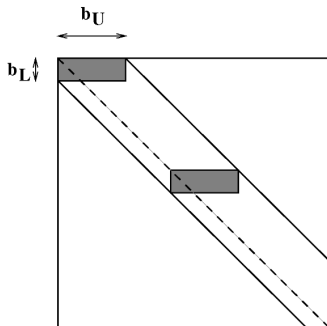
† 如果  $A$  是 (行) 对角占优, 则采用前面的分解;

如果  $A$  是列对角占优, 则采用分解 (2.11).

## 2.4 带状线性方程组

设  $A \in \mathbb{R}^{n \times n}$  是带状矩阵, 其下带宽为  $b_L$ , 上带宽为  $b_U$ , 即

$$a_{ij} = 0 \quad \text{for} \quad i > j + b_L \quad \text{or} \quad i < j - b_U.$$



对于带状矩阵, 其 LU 分解有如下性质:

**定理** 设  $A \in \mathbb{R}^{n \times n}$  是带状矩阵, 其下带宽为  $b_L$ , 上带宽为  $b_U$ . 若  $A = LU$  是**不选主元的 LU 分解**, 则  $L$  为下带宽为  $b_L$  的带状矩阵,  $U$  为上带宽为  $b_U$  的带状矩阵.

**思考**

统计求解带状矩阵  $Ax = b$  的运算量.

若采用部分选主元的 LU 分解, 则有

**定理** 设  $A \in \mathbb{R}^{n \times n}$  是带状矩阵, 其下带宽为  $b_L$ , 上带宽为  $b_U$ . 若  $PA = LU$  是**部分选主元的 LU 分解**, 则  $U$  为上带宽不超过  $b_L + b_U$  的带状矩阵,  $L$  为下带宽为  $b_L$  的“基本带状矩阵”, 即  $L$  每列的非零元素不超过  $b_L + 1$  个.

## 2.5 Toeplitz 线性方程组

$$T_n = \begin{bmatrix} t_0 & t_{-1} & \cdots & t_{-n+1} \\ t_1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & t_{-1} \\ t_{n-1} & \cdots & t_1 & t_0 \end{bmatrix}$$

$T_n$  是反方向对称 (persymmetric) 矩阵. 记  $J_n$  为  $n$  阶反向单位矩阵:

$$J_n = \begin{bmatrix} & & & 1 \\ & & 1 & \\ & & \ddots & \\ & 1 & & \\ 1 & & & \end{bmatrix}.$$

易知  $J_n^T = J_n^{-1} = J_n$ .

**引理** 矩阵  $A \in \mathbb{R}^{n \times n}$  是反向对称矩阵当且仅当

$$A = J_n A^T J_n \quad \text{或} \quad J_n A = A^T J_n.$$

若  $A$  可逆, 则可得

$$A^{-1} = J_n^{-1} (A^T)^{-1} J_n^{-1} = J_n (A^{-1})^T J_n,$$

即反向对称矩阵的逆也是反向对称矩阵.

† Toeplitz 矩阵的逆是反向对称矩阵, 但不一定是 Toeplitz 矩阵.



## Yule-Walker 方程组

假定  $T_n$  对称正定, 考虑线性方程组

$$T_n x = -r_n$$

其中  $r_n = [t_1, t_2, \dots, t_{n-1}, t_n]^T$ . 这类线性方程组称为 **Yule-Walker 方程组**, 其中  $t_n$  为任意给定的实数.

由于  $T_n$  对称正定, 所以  $t_0 > 0$ . 因此我们可以对  $T_n$  的对角线元素进行单位化. 不失一般性, 我们假定  $T_n$  的对角线元素为 1, 即

$$T_n = \begin{bmatrix} 1 & t_1 & \cdots & t_{n-1} \\ t_1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & t_1 \\ t_{n-1} & \cdots & t_1 & 1 \end{bmatrix}.$$

由于方程组右端项的特殊性, 我们可以通过递推来求解.

记  $T_k x = -r_k$  的解为  $x^{(k)}$ . 设  $T_{k+1} x = -r_{k+1}$  的解  $x^{(k+1)} = \begin{bmatrix} z^{(k)} \\ \alpha_k \end{bmatrix}$ .

代入后可得 递推公式:

$$\alpha_k = \frac{-t_{k+1} - r_k^T J_k x^{(k)}}{1 + r_k^T x^{(k)}}, \quad z^{(k)} = x^{(k)} + \alpha_k J_k x^{(k)} \quad k = 1, 2, \dots$$

(2.13)

因此, 我们就可以从一阶 Yule-Walker 方程出发, 利用递推公式 (2.13) 计算  $T_n x = -r_n$  的解. 总的运算量 (乘法与加减运算) 大约为  $3n^2$ .

为了减少运算量, 我们引入一个变量  $\beta_k \triangleq 1 + r_k^T x^{(k)}$ , 则

$$\begin{aligned}\beta_{k+1} &= 1 + r_{k+1}^T x^{(k+1)} \\ &= 1 + [r_k^T, t_{k+1}] \begin{bmatrix} x^{(k)} + \alpha_k J_k x^{(k)} \\ \alpha_k \end{bmatrix} \\ &= 1 + r_k^T x^{(k)} + \alpha_k (t_{k+1} + r_k^T J_k x^{(k)}) \\ &= (1 - \alpha_k^2) \beta_k.\end{aligned}$$

总运算量降为  $2n^2$ . 这就是求解 Yule-Walker 方程组的 Durbin 算法.

---

## 算法 2.4 求解 Yule-Walker 方程组的 Durbin 算法

---

- 1: 输入数据:  $t = [t_1, t_2, \dots, t_n]$     % 注: 这里假定  $t_0 = 1$
  - 2:  $x(1) = -t_1, \beta = 1, \alpha = -t_1$
  - 3: **for**  $k = 1$  **to**  $n - 1$  **do**
  - 4:      $\beta = (1 - \alpha^2)\beta$
  - 5:      $\alpha = - \left( t_{k+1} - \sum_{i=1}^k t_{k+1-i} x(i) \right) / \beta$
  - 6:      $x(1 : k) = x(1 : k) + \alpha x(k : -1 : 1)$
  - 7:      $x(k + 1) = \alpha$
  - 8: **end for**
-

## 一般右端项的对称正定 Toeplitz 线性方程组

考虑一般右端项的方程组  $T_n x = b$ , 其中  $T_n$  对称正定.

我们利用递推方法来求解: 假定  $x^{(k)}$  和  $y^{(k)}$  分别是下面两个方程组的解:

$$T_k x = [b_1, b_2, \dots, b_k]^T, \quad T_k y = -[t_1, t_2, \dots, t_k]^T$$

设  $x^{(k+1)} = \begin{bmatrix} z^{(k)} \\ \alpha_k \end{bmatrix}$  是  $T_{k+1} x = b^{(k+1)}$  的解, 则可得

$$z^{(k)} = x^{(k)} + \mu_k J_k y^{(k)}, \quad \mu_k = \frac{b_{k+1} - r_k^T J_k x^{(k)}}{1 + r_k^T y^{(k)}}.$$

所以, 我们可以先计算  $T_k x = b^{(k)}$  和  $T_k x = -r_k$  的解, 然后利用上述公式得到  $T_{k+1} x = b^{(k+1)}$  的解, 这就是 Levinson 算法, 总运算量大约为  $4n^2$ .

---

## 算法 2.5 求解对称正定 Toeplitz 线性方程组的 Levinson 算法

---

1: 输入数据:  $t = [t_1, t_2, \dots, t_{n-1}]$  和  $b = [b_1, b_2, \dots, b_n]$     % 假定  $t_0 = 1$

2:  $y(1) = -t_1, x(1) = b_1, \beta = 1, \alpha = -t_1$

3: **for**  $k = 1$  to  $n - 1$  **do**

4:      $\beta = (1 - \alpha^2)\beta$

5:      $\mu = \left( b_{k+1} - \sum_{i=1}^k t_{k+1-i}x(i) \right) / \beta$

6:      $x(1:k) = x(1:k) + \mu y(k:-1:1), \quad x(k+1) = \mu$

7:     **if**  $k < n - 1$  **then**

8:          $\alpha = - \left( t_{k+1} + \sum_{i=1}^k t_{k+1-i}y(i) \right) / \beta$

9:          $y(1:k) = y(1:k) + \alpha y(k:-1:1)$

10:          $y(k+1) = \alpha$

11:     **end if**

12: **end for**

---

在数学与工程的许多应用中都会出现 Toeplitz 线性方程组. Levinson 算法是较早的关于对称正定 Toeplitz 线性方程组的快速算法, 但并不稳定 (只具有弱稳定性). 后来人们提出了各种各样的快速和超快速算法:

方法	运算量	存储量
Fast stable	$\geq 20n^2$	$\geq n^2/2$
Fast but unstable	$\geq 3n^2$	$\geq 4n$
Superfast and “unstable”	$O(n \log^2 n)$	$O(n)$
Superfast preconditioner	$O(n \log n)$	$O(n)$

- Fast : Levinson-Durbin (1946), Trench (1964), ...
- Fast stable: Bareiss (1969), Gohberg, Kailath and Olshevsky (1995), Chandrasekaran and Sayed (1998), Gu (1998), ...
- Superfast: Brent, Gustavson and Yun (1980), Bitmead and Anderson (1980), Morf (1980), de Hoog (1987), Ammar and Gragg (1988), ...
- Superfast Preconditioners: Strang, Chan, Chan, Tyrtyshnikov, ...

## 3 扰动分析

3.1  $\delta x$  与  $\hat{x}$  的关系

3.2  $\delta x$  与  $x_*$  的关系

3.3  $\delta x$  与残量的关系

3.4 相对扰动分析



考虑线性方程组

$$Ax = b.$$

设  $x_*$  是精确解,  $\hat{x}$  是通过数值计算得到的近似解. 假定  $\hat{x}$  满足线性方程组

$$(A + \delta A)\hat{x} = b + \delta b.$$

下面讨论  $\delta x \triangleq \hat{x} - x_*$  的大小, 即向后误差分析.

### 3.1 $\delta x$ 与 $\hat{x}$ 的关系

**定理** 设  $\|\cdot\|$  是任一向量范数（当该范数作用在矩阵上时就是相应的导出范数），则  $\delta x$  与  $\hat{x}$  满足下面的关系式

$$\frac{\|\delta x\|}{\|\hat{x}\|} \leq \|A^{-1}\| \cdot \|A\| \left( \frac{\|\delta A\|}{\|A\|} + \frac{\|\delta b\|}{\|A\| \cdot \|\hat{x}\|} \right).$$

当  $\delta b = 0$  时, 有

$$\frac{\|\delta x\|}{\|\hat{x}\|} \leq \kappa(A) \frac{\|\delta A\|}{\|A\|} \quad (2.16)$$

(板书)

## 3.2 $\delta x$ 与 $x_*$ 的关系

**引理** 设  $\|\cdot\|$  是任一算子范数,  $X \in \mathbb{R}^{n \times n}$ . 若  $\|X\| < 1$ , 则  $I - X$  可逆, 且有

$$(I - X)^{-1} = \sum_{k=0}^{\infty} X^k \quad \text{和} \quad \|(I - X)^{-1}\| \leq \frac{1}{1 - \|X\|}.$$

(板书)

**定理** 设  $A \in \mathbb{R}^{n \times n}$  非奇异且  $\|A^{-1}\| \cdot \|\delta A\| < 1$ , 则

$$\frac{\|\delta x\|}{\|x_*\|} \leq \frac{\kappa(A)}{1 - \kappa(A) \cdot \frac{\|\delta A\|}{\|A\|}} \left( \frac{\|\delta A\|}{\|A\|} + \frac{\|\delta b\|}{\|b\|} \right). \quad (2.18)$$

如果  $\|\delta A\| = 0$ , 则

$$\frac{1}{\kappa(A)} \frac{\|\delta b\|}{\|b\|} \leq \frac{\|\delta x\|}{\|x_*\|} \leq \kappa(A) \frac{\|\delta b\|}{\|b\|}.$$

(板书)

**定理** 设  $A \in \mathbb{R}^{n \times n}$  非奇异, 则有

$$\min \left\{ \frac{\|\delta A\|_2}{\|A\|_2} : A + \delta A \text{ 奇异} \right\} = \frac{1}{\kappa_2(A)}$$

(板书)

† 上述定理中的结论对所有  $p$ -范数都成立, 参见 [24, 40].

† 度量

$$\text{dist}_p(A) \triangleq \min \left\{ \frac{\|\delta A\|_p}{\|A\|_p} : A + \delta A \text{ 奇异} \right\} = \frac{1}{\kappa_p(A)},$$

表示  $A$  距离奇异矩阵集合的相对距离.

### 3.3 $\delta x$ 与残量的关系

记残量 (残差) 为  $r = b - A\hat{x}$ , 则有

$$\delta x = \hat{x} - x_* = \hat{x} - A^{-1}b = A^{-1}(A\hat{x} - b) = -A^{-1}r,$$

所以可得

$$\|\delta x\| \leq \|A^{-1}\| \cdot \|r\|$$

实际计算中  $r$  是可以计算的, 因此比较实用.

**定理** 设  $A \in \mathbb{R}^{n \times n}$  非奇异,  $\|\cdot\|$  为任一算子范数. 记  $r = b - A\hat{x}$ , 则

- (1) 若存在  $\hat{A}$  满足  $\hat{A}\hat{x} = b$ , 则  $\|\hat{A} - A\| \geq \frac{\|r\|}{\|\hat{x}\|}$ ;
- (2) 存在  $\delta A$  满足  $\|\delta A\| = \frac{\|r\|}{\|\hat{x}\|}$ , 使得  $(A + \delta A)\hat{x} = b$ .

### 3.4 相对扰动分析

前面给出的误差  $\delta x$  与条件数  $\kappa(A)$ ,  $\delta A$  和  $\delta b$  成比例. 许多情况下, 这个界是令人满意的. 但有时相差很大, 不能很好的反映实际计算中解的误差.

**例** 设  $A = \begin{bmatrix} \gamma & 0 \\ 0 & 1 \end{bmatrix}$ ,  $b = \begin{bmatrix} \gamma \\ 1 \end{bmatrix}$ , 其中  $\gamma > 1$ . 则  $Ax = b$  的精确解为  $x_* = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ , 任何合理的直接法求得的解的误差都很小. 但系数矩阵的谱条件数为  $\kappa_2(A) = \gamma$ , 当  $\gamma$  很大时,  $\kappa_2(A)$  也很大, 因此误差界 (2.16) 和 (2.18) 可以是很大.

针对这个问题, 我们按分量进行分析. 记

$$\delta A = \begin{bmatrix} \delta a_{11} & \\ & \delta a_{22} \end{bmatrix}, \quad \delta b = \begin{bmatrix} \delta b_1 \\ \delta b_2 \end{bmatrix},$$

并设  $|\delta a_{ij}| \leq \varepsilon |a_{ij}|$ ,  $|\delta b_i| \leq \varepsilon |b_i|$ . 则

$$\delta x = \begin{bmatrix} \hat{x}_1 - x_1 \\ \hat{x}_2 - x_2 \end{bmatrix} = \begin{bmatrix} \frac{\delta b_1 + b_1}{\delta a_{11} + a_{11}} - 1 \\ \frac{\delta b_2 + b_2}{\delta a_{22} + a_{22}} - 1 \end{bmatrix} = \begin{bmatrix} \frac{\delta b_1 + \gamma}{\delta a_{11} + \gamma} - 1 \\ \frac{\delta b_2 + 1}{\delta a_{22} + 1} - 1 \end{bmatrix} = \begin{bmatrix} \frac{\delta b_1 - \delta a_{11}}{\delta a_{11} + \gamma} \\ \frac{\delta b_2 - \delta a_{22}}{\delta a_{22} + 1} \end{bmatrix}.$$

故

$$\|\delta x\|_{\infty} \leq \frac{2\varepsilon}{1 - \varepsilon}.$$

如果  $\delta b = 0$ , 则

$$\|\delta x\|_{\infty} \leq \frac{\varepsilon}{1 - \varepsilon}.$$

这个界与 (2.16) 或 (2.18) 相差约  $\gamma$  倍.



## 相对条件数

为了得到更好误差界, 我们引入**相对条件数**  $\kappa_{cr}(A)$ , 即

$$\kappa_{cr}(A) \triangleq \| |A^{-1}| \cdot |A| \|,$$

有时也称为 Bauer 条件数或 Skeel 条件数.

假定  $\delta A$  和  $\delta b$  满足  $|\delta A| \leq \varepsilon |A|$  和  $|\delta b| \leq \varepsilon |b|$ . 由  $(A + \delta A)\hat{x} = b + \delta b$  可得

$$\begin{aligned} |\delta x| &= |A^{-1}(-\delta A \hat{x} + \delta b)| \\ &\leq |A^{-1}| \cdot (|\delta A| \cdot |\hat{x}| + |\delta b|) \\ &\leq |A^{-1}| \cdot (\varepsilon |A| \cdot |\hat{x}| + \varepsilon |b|) \\ &= \varepsilon |A^{-1}| \cdot (|A| \cdot |\hat{x}| + |b|). \end{aligned}$$

若  $\delta b = 0$ , 则有

$$\|\delta x\| = \| |\delta x| \| \leq \varepsilon \| |A^{-1}| \cdot |A| \cdot |\hat{x}| \| \leq \varepsilon \| |A^{-1}| \cdot |A| \| \cdot \|\hat{x}\|,$$

即

$$\frac{\|\delta x\|}{\|\hat{x}\|} \leq \| |A^{-1}| \cdot |A| \| \cdot \varepsilon = \kappa_{cr}(A) \cdot \varepsilon$$

相对条件数有下面的性质

**引理** 设  $A \in \mathbb{R}^{n \times n}$  非奇异,  $D \in \mathbb{R}^{n \times n}$  为非奇异对角矩阵, 则

$$\kappa_{cr}(DA) = \kappa_{cr}(A).$$

**定理** 设  $A \in \mathbb{R}^{n \times n}$  非奇异, 使得  $|\delta A| \leq \varepsilon |A|$ ,  $|\delta b| \leq \varepsilon |b|$  成立, 且满足

$$(A + \delta A)\hat{x} = b + \delta b$$

的最小的  $\varepsilon > 0$  称为按分量的相对向后误差, 其表达式为

$$\varepsilon = \max_{1 \leq i \leq n} \frac{|r_i|}{(|A| \cdot |\hat{x}| + |b|)_i},$$

其中  $r = b - A\hat{x}$ .

更多关于数值计算的稳定性和矩阵扰动分析方面的知识, 可以参考 [35, 59, 75].

## 4 误差分析

### 4.1 LU 分解的舍入误差分析

### 4.2 Gauss 消去法的舍入误差分析

## 4.1 LU 分解的舍入误差分析

关于 LU 分解的舍入误差分析, 我们有下面的结果.

**定理** 假定  $A \in \mathbb{R}^{n \times n}$  的所有顺序主子式都不为 0, 则带舍入误差的 LU 分解可表示为

$$A = LU + E,$$

其中误差  $E$  满足

$$|E| \leq \gamma_n |L| \cdot |U|.$$

这里  $\gamma_n = \frac{n\varepsilon_u}{1 - n\varepsilon_u}$ ,  $\varepsilon_u$  表示机器精度.

**证明.** 见 [35, page 164].



## 4.2 Gauss 消去法的舍入误差分析

**引理** [35] 设  $\hat{y}$  和  $\hat{x}$  分别是由向前回代和向后回代得到的数值解, 则

$$\begin{aligned}(L + \delta L)\hat{y} &= b, & |\delta L| &\leq \gamma_n |L| \\ (U + \delta U)\hat{x} &= \hat{y}, & |\delta U| &\leq \gamma_n |U|.\end{aligned}$$

该引理表明, 向前回代算法和向后回代算法都是稳定的.

† 在绝大多数情况下, 部分选主元 Gauss 消去法是向后稳定的, 但理论上也存在失败的例子.

† 全主元 Gauss 消去法是数值稳定的. 在大部分实际应用中, 部分选主元 Gauss 消去法与全主元 Gauss 消去法具有同样的数值稳定性.

## 5 解的改进和条件数估计

5.1 高精度运算

5.2 矩阵元素缩放 (Scaling)

5.3 迭代改进法

## 5.1 高精度运算

在计算中, 尽可能采用高精度的运算.

比如, 原始数据是单精度的, 但在计算时都采用双精度运算, 或者更高精度的运算. 但更高精度的运算会带来更大的开销.



## 5.2 矩阵元素缩放 (Scaling)

如果  $A$  的元素在数量级上相差很大, 则在计算过程中很可能会出现大数与小数的加减运算, 这样就可能会引入更多的舍入误差. 为了避免由于这种情况而导致的舍入误差, 我们可以在求解之前先对矩阵元素进行缩放 (Scaling), 即在矩阵两边同时乘以两个适当的对角矩阵.

例 考虑线性方程组

$$\begin{bmatrix} -4000 & 2000 & 2000 \\ 2000 & 0.78125 & 0 \\ 2000 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 400 \\ 1.3816 \\ 1.9273 \end{bmatrix}.$$

用部分选主元 Gauss 法求解, 计算过程保留 8 位有效数字, 求得数值解

$$\tilde{x} = [0.00096365, -0.698496, 0.90042329]^T.$$

与精确解  $x = [1.9273 \cdots, -0.698496 \cdots, 0.9004233 \cdots]^T$  误差较大.

考虑矩阵元素缩放, 即同乘对角阵  $D = \text{diag}(0.00005, 1, 1)$ , 得新方程组

$$DADy = Db.$$

最后令  $\tilde{x} = Dy$ , 即可求得比较精确的数值解.

## 5.3 迭代改进法

设近似解  $\hat{x}$ , 残量  $r = b - A\hat{x}$ . 当  $\hat{x}$  没达到精度要求时, 可以考虑方程  $Az = r$ . 如果  $z$  该方程的精确解, 则

$$A(\hat{x} + z) = A\hat{x} + Az = (b - r) + r = b,$$

因此  $\hat{x} + z$  就是原方程的精确解. 在实际计算中, 我们只能得到近似解  $\hat{z}$ , 但  $\|r - A\hat{z}\|$  会很小, 特别地, 应该比  $\|r\|$  更小. 因此  $\hat{x} + \hat{z}$  应该比  $\hat{x}$  更接近精确解.

如果新的近似解  $\hat{x} + \hat{z}$  还不满足精度要求, 则可重复以上过程.

这就是通过迭代来提高解的精度.

---

### 算法 5.1 通过迭代改进解的精度

---

- 1: 设  $PA = LU$ ,  $\hat{x}$  是  $Ax = b$  的近似解
  - 2: **while** 近似解  $\hat{x}$  不满足精度要求, **do**
  - 3:     计算  $r = b - A\hat{x}$
  - 4:     求解  $Ly = Pr$ , 即  $y = L^{-1}Pr$
  - 5:     求解  $Uz = y$ , 即  $z = U^{-1}y$
  - 6:     令  $\hat{x} = \hat{x} + z$
  - 7: **end while**
-

由于每次迭代只需计算一次残量和求解两个三角线性方程组, 因此运算量为  $O(n^2)$ . 所以相对来讲还是比较经济的.

† 为了提高计算精度, 在计算残量  $r$  时最好使用原始数据  $A$ , 而不是  $P^T LU$ , 因此对  $A$  做 LU 分解时需要保留矩阵  $A$ , 不能被  $L$  和  $U$  覆盖.

† 实际计算经验表明, 当  $A$  病态不是很严重时, 即  $\varepsilon_u \kappa_\infty(A) < 1$ , 迭代法可以有效改进解的精度, 最后达到机器精度. 但  $\varepsilon_u \kappa_\infty(A) \geq 1$  时, 一般没什么效果. 这里  $\varepsilon_u$  表示机器精度.

## 课后习题

见讲义

## 参考文献

- [1] J. O. Aasen, On the reduction of a symmetric matrix to tridiagonal form, *BIT*, 11 (1971), 233–242.
- [2] M. Arioli, V. Pták, and Z. Strakoš, Krylov sequences of maximal length and convergence of GMRES, *BIT*, 38 (1998), 636–643.
- [3] C. Ashcraft, R. G. Grimes and J. G. Lewis, Accurate symmetric indefinite linear equation solvers, *SIAM Journal on Matrix Analysis and Applications*, 20 (1998), 513–561.
- [4] O. Axelsson, *Iterative Solution Methods*, Cambridge University Press, Cambridge, 1994.
- [5] Z.-Z. Bai, G. H. Golub and M. K. Ng, Hermitian and skew-Hermitian splitting methods for non-Hermitian positive definite linear systems, *SIAM Journal on Matrix Analysis and Applications*, 24 (2003), 603–626.
- [6] M. Van Barel, G. Heinig and P. Kravanja, A stabilized superfast solver for non-symmetric toeplitz systems, *SIAM Journal on Matrix Analysis and Applications*, 23 (2001), 494–510. FORTRAN code <http://people.cs.kuleuven.be/~marc.vanbarel/software/>

- [7] R. Barrett, et.al, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, 1994. (<http://www.netlib.org/templates/index.html>)
- [8] V. Barwell and A. George, A comparison of algorithms for solving symmetric indefinite systems of linear equations, *ACM Transactions on Mathematical Software*, 2 (1976), 242–251.
- [9] Åke Björck, Solving linear least square problems by Gram-Schmidt orthogonalization, *BIT*, 7 (1967), 1–21.
- [10] Åke Björck, *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia, PA, 1996.
- [11] J. R. Bunch, Analysis of the diagonal pivoting method, *SIAM Journal on Numerical Analysis*, 8 (1971), 656–680.
- [12] J. R. Bunch and L. Kaufman, Some stable methods for calculating inertia and solving symmetric linear systems, *Mathematics of Computation*, 31 (1977), 163–179.
- [13] J. R. Bunch and B. N. Parlett, Direct methods for solving symmetric indefinite systems of linear equations, *SIAM Journal on Numerical Analysis*, 8 (1971), 639–655.
- [14] R. P. Brent, Stability of fast algorithms for structured linear systems, <http://arxiv.org/pdf/1005.0671v1.pdf>, 2010.



- [15] G. Codevico, G. Heinig and M. Van Barel, A superfast solver for real symmetric Toeplitz systems using real trigonometric transformations, *Numer. Linear Algebra Appl.*, 12 (2005), 699–713. MATLAB code <http://people.cs.kuleuven.be/~marc.vanbareel/software/>
- [16] J. J. M. Cuppen, A Divide and Conquer Method for the Symmetric Tridiagonal Eigenproblem, *Numerische Mathematik*, 36 (1981), 177–195.
- [17] J. W. Demmel, *Applied Numerical Linear Algebra*, SIAM, Philadelphia, PA, 1997.
- [18] J. J. Dongarra, I. S. Duff, D. C. Sorensen and H. A. van der Vorst, *Numerical Linear Algebra for High-Performance Computers*, SIAM, Philadelphia, PA, 1998.
- [19] Z. Drmač and K. Veselić, New fast and accurate jacobi SVD algorithm. I SIAM Journal on Matrix Analysis and Applications, 29 (2008), 1322–1342.
- [20] Z. Drmač and K. Veselić, New fast and accurate jacobi SVD algorithm. II SIAM Journal on Matrix Analysis and Applications, 29 (2008), 1343–1362.
- [21] V. Faber, W. Joubert, E. Knill and T. Manteuffel, Minimal residual method stronger than polynomial preconditioning, *SIAM Journal on Matrix Analysis and Applications*, 17 (1996), 707–729.
- [22] K. Fernando and B. Parlett, Accurate singular values and differential qd algorithms, *Numerische Mathematik*, 67 (1994), 191–229.

- [23] B. Fischer, *Polynomial based iteration methods for symmetric linear systems*, Wiley-Teubner Series Advances in Numerical Mathematics, John Wiley & Sons Ltd., Chichester, 1996.
- [24] N. Gastinel, *Linear Numerical Analysis*, Kershaw Publishing, London, 1083.
- [25] G. H. Golub, History of numerical linear algebra: A personal view, Stanford, 2007. Available at <http://forum.stanford.edu/events/2007slides/plenary/history-revised-2007-03-19-golub.pdf>
- [26] G. H. Golub and W. Kahan, Calculating the singular values and pseudo-inverse of a matrix, *SIAM Journal on Numerical Analysis*, Series B, 2 (1965), 205–224.
- [27] G. H. Golub and C. F. Van Loan, *Matrix Computations*, The 4th Editon, The Johns Hopkins University Press, Baltimore, MD, 2013.
- [28] A. Greenbaum and L. Gurvits, Max-min properties of matrix factor norms, *SIAM Journal on Scientific Computing*, 15 (1994), 348–358.
- [29] A. Greenbaum, V. Pták and Z. Strakoš, Any nonincreasing convergence curve is possible for GMRES, *SIAM Journal on Matrix Analysis and Applications*, 17 (1996), 465–469.
- [30] A. Greenbaum and Z. Strakoš, Matrices that generate the same Krylov residual spaces, in *Recent Advances in Iterative Methods*, vol. 60 of IMA Vol. Math. Appl., Springer, New York, 1994, pp. 95–118.

- [31] M. Gu and S. C. Eisenstat, A stable algorithm for the rank-1 modification of the symmetric eigenproblem, *SIAM Journal on Matrix Analysis and Applications*, 15 (1994), 1266–1276.
- [32] M. Gu and S. C. Eisenstat, A Divide-and-Conquer algorithm for the bidiagonal SVD, *SIAM Journal on Matrix Analysis and Applications*, 16 (1995), 79–92.
- [33] M. Gu and S. C. Eisenstat, A Divide-and-Conquer algorithm for the symmetric tridiagonal eigenproblem, *SIAM Journal on Matrix Analysis and Applications*, 16 (1995), 172–191.
- [34] A. Hadjidimos, Accelerated overrelaxation method, *Mathematics of Computation*, 32 (1978), 149–157.
- [35] Nicholas J. Higham, *Accuracy and Stability of Numerical Algorithms*, Second Edition, SIAM, Philadelphia, 2002.
- [36] L. Hogben, *Handbook of Linear Algebra*, 2nd, CRC Press, 2014.
- [37] R.A. Horn and C.R. Johnson, *Matrix Analysis*, Cambridge University Press, New York, 1985.
- [38] R.A. Horn and C.R. Johnson, *Topics in Matrix Analysis*, Cambridge University Press, New York, 1991.
- [39] W. Joubert, A robust GMRES-based adaptive polynomial preconditioning algorithm for nonsymmetric linear systems, *SIAM Journal on Scientific Computing*, 15 (1994), 427–439.

- [40] W. Kahan Numerical Linear Algebra, Canadian Math. Bull., 9 (1966), 757–801.
- [41] C. T. Kelley, *Iterative Methods for Linear and Nonlinear Equations*, SIAM, Philadelphia, 1995.
- [42] D. Kressner, *Numerical Methods for General and Structured Eigenvalue Problems*, Lecture Notes in Computational Sciences and Engineering 46, Springer-Verlag, 2005.
- [43] R. Lehoucq, *Analysis and Implementation of an Implicitly Restarted Arnoldi Iteration*, Ph.D. thesis, Rice University, Houston, TX, 1995.
- [44] N. Levinson The Wiener RMS (root mean square) error criterion in filter design and prediction, *J. Math. Phys.*, 25 (1946), 261–278.
- [45] J. Liesen, Computable convergence bounds for GMRES, *SIAM Journal on Matrix Analysis and Applications*, 21 (2000), 882–903.
- [46] J. Liesen and P. Tichý, Convergence analysis of Krylov subspace methods, *GAMM-Mitteilungen*, 27 (2004), 153–173.
- [47] E. H. Moore, On the reciprocal of the general algebraic matrix, *Bull. Amer. Math. Soc.*, 26 (1920), 394–395.
- [48] Christopher C. Paige, Miroslav Rozložník and Zdeněk Strakoš, Modified Gram-Schmidt (MGS), least squares, and backward stability of MGS-GMRES, *SIAM Journal on Matrix Analysis and Applications*, (28) 2006, 264–284.

- [49] B. N. Parlett, *The Symmetric Eigenvalue Problem*, The 2nd Edition, SIAM, Philadelphia, PA, 1998.
- [50] D. W. Peaceman and H. H. Rachford, Jr., The numerical solution of parabolic and elliptic differential equations, *Journal of the Society for Industrial and Applied Mathematics*, 3 (1955), 28–41.
- [51] R. Penrose, A generalized inverse for matrices, *Proc. Cambridge Philos. Soc.*, 51 (1955), 406–413.
- [52] V. Britanak, P. Yip and K. Rao, *Discrete Cosine and Sine Transforms: General properties, Fast algorithms and Integer Approximations*, Academic Press, 2007.
- [53] J. Rutter, *A Serial Implementation of Cuppen's Divide and Conquer Algorithm for the Symmetric Eigenvalue Problem*, Master's Thesis, University of California, 1994.
- [54] Y. Saad and M. H. Schultz, GMRES: A generalized minimal residual method for solving nonsymmetric linear systems, *SIAM Journal on Scientific & Statistical Computing*, 7 (1986), 856–869.
- [55] Y. Saad, *Numerical Methods for Large Eigenvalue Problems: Theory and Algorithms*, Manchester University Press, Manchester, UK, 1992.
- [56] D. Sorensen, Implicit application of polynomial filters in a  $k$ -step Arnoldi method, *SIAM Journal on Matrix Analysis and Applications*, 13 (1992), 357–385.
- [57] G. W. Stewart, *Matrix Algorithms, Vol I: Basic Decomposition*, SIAM, Philadelphia, PA, 1998.

- [58] G. W. Stewart, *Matrix Algorithms, Vol II: Eigensystems*, SIAM, Philadelphia, PA, 2001.
- [59] G. W. Stewart and Ji-guang Sun, *Matrix Perturbation Theory*, Academic Press, New York, 1990.
- [60] M. Stewart, Fast algorithms for structured matrix computations, in *Handbook of Linear Algebra*, 2nd, section 62, CRC Press, 2014.
- [61] L. N. Trefethen and D. Bau, *Numerical Linear Algebra*, SIAM, Philadelphia, PA, 1997.
- [62] L. N. Trefethen, Numerical Analysis, in *Princeton Companion to Mathematics*, Edited by T. Gowers, J. Barrow-Green and I. Leader, Princeton University Press, 2008.
- [63] D. S. Watkins, *The Matrix Eigenvalue Problem: GR and Krylov Subspace Methods*, SIAM, Philadelphia, 2007.
- [64] D. S. Watkins and L. Elsner, Convergence of algorithms of decomposition type for the eigenvalue problem, *Linear Algebra and its Applications*, 143 (1991), 19–47.
- [65] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford University, Oxford, 1965.
- [66] R. S. Varga, *Matrix Iterative Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1962. 2nd edition, Springer-Verlag, Berlin, 2000.

- [67] D. M. Young, *Iterative Methods for Solving Partial Difference Equations of Elliptic Type*, PhD thesis, Harvard University, 1950.
- [68] D. M. Young, *Iterative Solution of Large Linear Systems*, Academic Press, New York, 1971.
- [69] 北京大学数学系, 高等代数 (第三版), 高等教育出版社, 2003.
- [70] 陈志明, 科学计算: 科技创新的第三种方法, 中国科学院院刊, 27 (2012), 161-166.
- [71] 戴华, 矩阵论, 科学出版社, 2001.
- [72] 胡家赣, 线性代数方程组的迭代解法, 科学出版社, 1991.
- [73] 蒋尔雄, 矩阵计算, 科学出版社, 2008.
- [74] 李大明, 数值线性代数, 清华大学出版社.
- [75] 孙继广, 矩阵扰动分析, 科学出版社, 北京, 2001.
- [76] 魏木生, 广义最小二乘问题的理论与计算, 科学出版社, 北京, 2006.
- [77] 徐树方, 矩阵计算的理论与方法, 北京大学出版社, 北京, 1995.
- [78] 徐树方, 钱江, 矩阵计算六讲, 高等教育出版社, 北京, 2011.