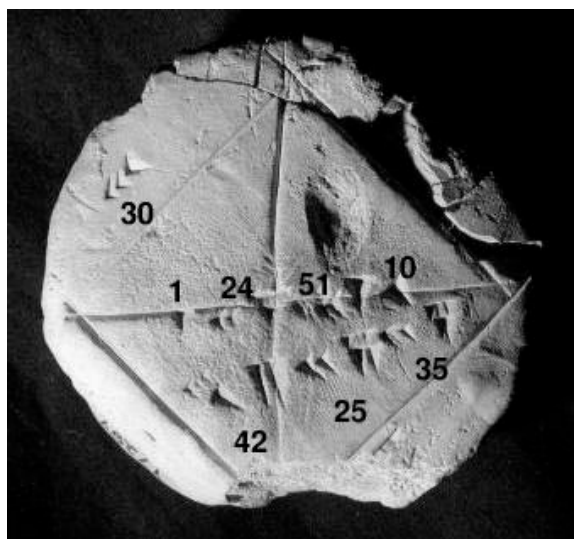


2017-2018年度第一学期 00151102 00151103

计算方法 (B)



童伟华 管理科研楼1205室

E-mail: tongwh@ustc.edu.cn

中国科学技术大学 数学科学学院

<http://math.ustc.edu.cn/>





第五章 解线性方程组的直接法

线性方程组的求解

- 在科学研究和工程应用中，求解线性方程组是非常基础的问题

- 一般的线性方程组

$$\begin{cases} a_{11}x_1 + \cdots + a_{1n}x_n = b_1 \\ \vdots \\ a_{n1}x_1 + \cdots + a_{nn}x_n = b_n \end{cases} \Leftrightarrow \mathbf{Ax} = \mathbf{b}$$

- (Gram公式) 当且仅当 $\det(\mathbf{A}) \neq 0$ 时，方程组有唯一解

$$x_i = \frac{D_i}{D}, \quad i = 1, 2, \dots, n$$

$$D = \det(\mathbf{A}), \quad D_i = \det \begin{pmatrix} a_{11} & \cdots & a_{1i-1} & b_1 & a_{1i+1} & \cdots & a_{1n} \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ a_{n1} & \cdots & a_{ni-1} & b_n & a_{ni+1} & \cdots & a_{nn} \end{pmatrix}$$

线性方程组的求解



■ 求解线性方程组的方法可分为

- 直接解法：采用消元（初等变换）、矩阵分解等技巧；从理论上来说，直接法经过有限次四则运算（假设运算无舍入误差），可以得到线性方程组的精确解
- 迭代解法：采用迭代、分块、预条件处理等技巧；将线性方程的解视为某种极限过程的向量序列，近似解

■ 直接法与迭代法的使用建议

- 一般来说，对同等规模的线性方程组，直接法对计算机的要求高于迭代法
- 对于中等规模的线性方程组，考虑到直接法的准确性与可靠性，建议使用直接法求解
- 对于高阶稀疏线性方程组（非零元素较少），建议使用迭代法求解
- 要充分考虑线性方程的特性，譬如对称性、正定性、稀疏性等，来选用合适的算法

线性方程组的求解



■ 求解线性方程组的常用软件

- Matlab
- LAPACK/EISPACK/LINPACK (一般的线性方程组求解)
- TACUS (大型稀疏线性方程组)
- SuperLU (大型稀疏线性方程组)
- Eigen (开源, C++, 线性代数模板库)
- MKL (商业软件)
- ...

■ 推荐参考书

- G. H. Golub, C. F. V. Loan. Matrix Computations. 4th Ed. The Johns Hopkins University Press. (有影印本与中译本, 人民邮电出版社)
- L. N. Trefethen, D. B. III. Numerical Linear Algebra. SIAM Press. (有中译本, 人民邮电出版社)

消元法



- 基本思想：通过初等变换，将一般的线性方程组等价变换为特殊形式的线性方程组，如上/下三角方程组、对角方程组，进行求解

- 对角形线性方程组

$$\left\{ \begin{array}{rcl} a_{11}x_1 & & = b_1 \\ & a_{22}x_2 & = b_2 \\ & & \ddots \\ & & a_{nn}x_n = b_n \end{array} \right. \Rightarrow x_i = \frac{b_i}{a_{ii}}, i = 1, 2, \dots, n$$

时间复杂度： $O(n)$

消元法



■ 上三角方程组

$$\begin{cases} u_{11}x_1 + u_{12}x_2 + \cdots + u_{1n}x_n = b_1 \\ u_{22}x_2 + \cdots + u_{2n}x_n = b_2 \\ \cdots = \vdots \\ u_{nn}x_n = b_n \end{cases} \Rightarrow x_i = \frac{b_i - \sum_{j=i+1}^n u_{ij}x_j}{u_{ii}}, i = n, n-1, \dots, 1$$

时间复杂度: $O(n^2)$

■ 下三角方程组

$$\begin{cases} l_{11}x_1 = b_1 \\ l_{21}x_1 + l_{22}x_2 = b_2 \\ \cdots = \vdots \\ l_{n1}x_1 + l_{n2}x_2 + \cdots + l_{nn}x_n = b_n \end{cases} \Rightarrow x_i = \frac{b_i - \sum_{j=1}^{i-1} l_{ij}x_j}{l_{ii}}, i = 1, 2, \dots, n$$

时间复杂度: $O(n^2)$

■ 初等变换:

- 交换矩阵的两行
- 某一行乘以一个非零数
- 某一个乘以一个非零数, 加到另一行

■ Gauss消元法: 先将矩阵化为上/下三角矩阵, 再回代求解

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & b_n \end{pmatrix} \xrightarrow{\text{初等变换}} \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} & b_1 \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2n}^{(2)} & b_2^{(2)} \\ 0 & 0 & a_{33}^{(3)} & \cdots & a_{3n}^{(3)} & b_3^{(3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & a_{nn}^{(n)} & b_n^{(n)} \end{pmatrix}$$

消元法



■ 第一步：第 1 行 $\times \frac{-a_{i1}}{a_{11}} +$ 第 i 行, $i = 2, 3, \dots, n$

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & b_n \end{pmatrix} \xrightarrow{\text{初等变换}} \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} & b_2^{(2)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} & b_n^{(2)} \end{pmatrix}$$

运算量: $(n-1) * (1+n)$

消元法



■ 第二步：第 2 行 $\times \frac{-a_{i2}^{(2)}}{a_{22}^{(2)}} +$ 第 i 行, $i = 3, 4, \dots, n$

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} & b_2^{(2)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} & b_n^{(2)} \end{pmatrix} \xrightarrow{\text{初等变换}} \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} & b_1 \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2n}^{(2)} & b_2^{(2)} \\ 0 & 0 & a_{33}^{(3)} & \cdots & a_{3n}^{(3)} & b_3^{(3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & a_{n3}^{(3)} & \cdots & a_{nn}^{(3)} & b_n^{(3)} \end{pmatrix}$$

运算量: $(n-2) * (1+n-1) = (n-2)n$

消元法



- 类似的做下去，第 k 步：

$$\text{第 } k \text{ 行} \times \frac{-a_{ik}^{(k)}}{a_{kk}^{(k)}} + \text{第 } i \text{ 行}, i = k + 1, \dots, n$$

- 运算量：

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

- $n-1$ 步运算之后

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

- 总运算量：

$$\sum_{k=1}^{n-1} (n-k)(n-k+2) + (n+1)n/2 = \frac{n^3}{3} + n^2 - \frac{n}{3} = O(n^3)$$

消元法



■ Gauss消元算法

Algorithm 9 Gaussian Elimination Algorithm

Input:

$n, (a_{ij}), (b_i)$

```
1: for  $k = 1$  to  $n - 1$  do
2:   for  $i = k + 1$  to  $n$  do
3:      $z \leftarrow a_{ik}/a_{kk}$ ;
4:      $a_{ik} \leftarrow 0$ ;
5:     for  $j = k + 1$  to  $n$  do
6:        $a_{ij} \leftarrow a_{ij} - za_{kj}$ ;
7:     end for
8:      $b_i \leftarrow b_i - zb_k$ ;
9:   end for
10: end for
11: for  $i = n$  to 1 step -1 do
12:    $x_i \leftarrow (b_i - \sum_{j=i+1}^n a_{ij}x_j)/a_{ii}$ ;
13: end for
```

Output:

(x_i)

消元法



- Gauss消元法的可行条件为： $a_{kk}^{(k)} \neq 0$ ，即要求矩阵A的所有顺序主子式均不为零
- 例：求解线性方程组

$$\begin{cases} 10^{-9}x_1 + x_2 = 1 \\ x_1 + x_2 = 2 \end{cases} \Rightarrow \begin{cases} x_1 = 10^9 / (10^9 - 1) \\ x_2 = (10^9 - 2) / (10^9 - 1) \end{cases}$$

高斯消元法：

$$m_{21} = a_{21} / a_{11} = 10^9 \quad \text{8个}$$

$$a_{22} = 1 - m_{21} \times 1 = \underbrace{0.0 \dots 01}_{\text{8个}} \times 10^9 - 10^9 \doteq -10^9$$

$$b_2 = 2 - m_{21} \times 1 \doteq -10^9$$

$$\Rightarrow \begin{bmatrix} 10^{-9} & 1 & 1 \\ 0 & -10^9 & -10^9 \end{bmatrix}$$

$$\Rightarrow x_2 = 1, \quad x_1 = 0$$

■ 局限性：

- 某些有解的问题不能通过Gauss消去法求解
- 如果消元过程中， $a_{kk}^{(k)}$ 很小会带来很大的舍入误差

■ 改进方法：在消元过程中，如果在一系列中选择按模最大的元素，与主方程位置互换，再进行消元，即使使用绝对值尽可能大的系数来消元

■ Gauss列主元消元法

■ Gauss全主元消元法：如何实现？需要记录列互换

■ 延伸阅读：Gauss消元法的舍入误差分析

■ 理论分析与实践说明：Gauss列主元消元法是稳定的

■ Gauss列主元消元法

Algorithm 10 Gaussian Elimination Algorithm with Scaled Row Pivoting

Input:

$n, (a_{ij}), (b_i)$
1: **for** $i = 1$ to n **do**
2: $p_i \leftarrow i$;
3: $s_i \leftarrow \max_{1 \leq j \leq n} |a_{ij}|$;
4: **end for**
5: **for** $k = 1$ to $n - 1$ **do**
6: select $j \geq k$ so that
7: $|a_{p_j k}|/s_{p_j} \geq |a_{p_i k}|/s_{p_i}$ for $i = k, k + 1, \dots, n$;
8: $p_k \leftrightarrow p_j$;
9: **for** $i = k + 1$ to n **do**
10: $z \leftarrow a_{p_i k}/a_{p_k k}$;
11: $a_{p_i k} \leftarrow z$;
12: **for** $j = k + 1$ to n **do**
13: $a_{p_i j} \leftarrow a_{p_i j} - za_{p_k j}$;
14: **end for**
15: $b_{p_i} \leftarrow b_{p_i} - zb_{p_k}$;
16: **end for**
17: **end for**
18: **for** $i = n$ to 1 step -1 **do**
19: $x_i \leftarrow (b_{p_i} - \sum_{j=i+1}^n a_{p_i j} x_j)/a_{p_i i}$;
20: **end for**

Output:

(x_i)

直接分解法



- Gauss消元法的第 k 步:

第 k 行 $\times \frac{-a_{ik}^{(k)}}{a_{kk}^{(k)}} +$ 第 i 行, $i = k + 1, \dots, n$

- 从矩阵理论来看, 相当于左乘矩阵

$$\begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & l_{k+1k}^{(k)} & 1 & \\ & & \vdots & & \ddots \\ & & l_{nk}^{(k)} & & & 1 \end{pmatrix}, \quad l_{ik}^{(k)} = \frac{-a_{ik}^{(k)}}{a_{kk}^{(k)}}, \quad i = k + 1, \dots, n$$

直接分解法



- 整个Gauss消元法相当于左乘了一个单位下三角阵：

$$\tilde{\mathbf{L}} = \begin{pmatrix} 1 & & & & & \\ l_{21} & \ddots & & & & \\ & & 1 & & & \\ \vdots & & l_{k+1,k} & 1 & & \\ & & \vdots & & \ddots & \\ l_{n1} & \cdots & l_{nk} & \cdots & l_{nn-1} & 1 \end{pmatrix}$$

- 若Gauss消元法可行，则

$$\exists \mathbf{L}^{-1} \quad s.t. \quad \mathbf{L}^{-1} \mathbf{A} = \mathbf{U} \Rightarrow \exists \mathbf{L} \quad s.t. \quad \mathbf{A} = \mathbf{LU}$$

- 直接分解法

$$\mathbf{Ax} = \mathbf{b} \Rightarrow \mathbf{LUx} = \mathbf{b} \Rightarrow \begin{cases} \mathbf{Ly} = \mathbf{b} \\ \mathbf{Ux} = \mathbf{y} \end{cases}$$

直接分解法



- **Dolittle分解：** L 为单位下三角阵， U 为上三角阵

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

- **计算步骤：**
- **第一行：** $a_{1j} = u_{1j}, j = 1, \cdots, n \Rightarrow u_{1j} = a_{1j}$
- **第一列：** $a_{i1} = l_{i1}u_{11}, i = 2, \cdots, n \Rightarrow l_{i1} = a_{i1} / u_{11}$
- **第二行：** $a_{2j} = l_{21}u_{1j} + u_{2j}, j = 2, \cdots, n \Rightarrow u_{2j} = a_{2j} - l_{21}u_{1j}$
- **第二列：** $a_{i2} = l_{i1}u_{12} + l_{i2}u_{22}, i = 3, \cdots, n \Rightarrow l_{i2} = (a_{i2} - l_{i1}u_{12}) / u_{22}$

直接分解法



■ 第 k 行:

$$a_{kj} = \sum_{r=1}^{k-1} l_{kr} u_{rj} + u_{kj}, \quad j = k, \dots, n \Rightarrow u_{kj} = a_{kj} - \sum_{r=1}^{k-1} l_{kr} u_{rj}$$

■ 第 k 列:

$$a_{ik} = \sum_{r=1}^{k-1} l_{ir} u_{rk} + l_{ik} u_{kk} \quad i = k+1, \dots, n \Rightarrow l_{ik} = (a_{ik} - \sum_{r=1}^{k-1} l_{ir} u_{rk}) / u_{kk}$$

■ 二次回代过程:

$$\begin{cases} y_i = b_i - \sum_{j=1}^{i-1} l_{ij} y_j, & i = 1, \dots, n \\ x_i = (y_i - \sum_{j=i+1}^n u_{ij} x_j) / u_{ii}, & i = n, \dots, 1 \end{cases}$$

■ 总计算量:

$$\sum_{k=1}^n ((n-k+1)(k-1) + (n-k)k) + \frac{n(n-1)}{2} + \frac{n(n+1)}{2} = \frac{n^3}{3} + n^2 - \frac{n}{3} = O(n^3)$$

直接分解法



- 内存分配与管理：空间复杂度 $O(n^2)$

$$\begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ l_{21} & u_{22} & \cdots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \cdots & u_{nn} \end{pmatrix}$$

- 可行性条件：同 Gauss 消元法
- 优点：若需要求解一系列具有相同系数的线性方程组时，即系数矩阵相同，右端向量不同，则直接分解方法可以达到事半功倍的效果
- 思考：如何加入选主元技巧？

直接分解法



■ Doolittle分解算法

Algorithm 11 Doolittle's Factorization Algorithm

Input:

$n, (a_{ij})$

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

2: $l_{kk} = 1$;

3: **for** $j = k$ to n **do**

4: $u_{kj} \leftarrow a_{kj} - \sum_{s=1}^{k-1} l_{ks}u_{sj}$;

5: **end for**

6: **for** $i = k + 1$ to n **do**

7: $l_{ik} \leftarrow (a_{ik} - \sum_{s=1}^{k-1} l_{is}u_{sk})/u_{kk}$;

8: **end for**

9: **end for**

Output:

$(l_{ij}), (u_{ij})$

直接分解法



- Courant分解: L 为下三角阵, U 为单位上三角阵

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

- 计算步骤:

- 第 k 行:

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

- 第 k 列:

$$a_{kj} = \sum_{r=1}^{k-1} l_{kr} u_{rj} + l_{kk} u_{kj}, \quad j = k+1, \dots, n \Rightarrow u_{kj} = (a_{kj} - \sum_{r=1}^{k-1} l_{kr} u_{rj}) / l_{kk}$$

直接分解法



■ 二次回代过程:

$$\begin{cases} y_i = (b_i - \sum_{j=1}^{i-1} l_{ij} y_j) / l_{ii}, & i = 1, \dots, n \\ x_i = y_i - \sum_{j=i+1}^n u_{ij} x_j, & i = n, \dots, 1 \end{cases}$$

■ 总计算量:

$$\sum_{k=1}^n ((n-k+1)(k-1) + (n-k)k) + \frac{n(n-1)}{2} + \frac{n(n+1)}{2} = \frac{n^3}{3} + n^2 - \frac{n}{3} = O(n^3)$$

直接分解法



■ Courant分解算法

Algorithm 12 Courant's Factorization Algorithm

Input:

$n, (a_{ij})$
1: **for** $k = 1$ to n **do**
2: $u_{kk} = 1$;
3: **for** $i = k$ to n **do**
4: $l_{ik} \leftarrow a_{ik} - \sum_{s=1}^{k-1} l_{is}u_{sk}$;
5: **end for**
6: **for** $j = k + 1$ to n **do**
7: $u_{kj} \leftarrow (a_{kj} - \sum_{s=1}^{k-1} l_{ks}u_{sj})/l_{kk}$;
8: **end for**
9: **end for**

Output:

$(l_{ij}), (u_{ij})$

直接分解法



■ 三对角阵的追赶法:

$$\begin{pmatrix} a_1 & b_1 & & & \\ c_2 & a_2 & b_2 & & \\ & \ddots & \ddots & \ddots & \\ & & c_{n-1} & a_{n-1} & b_{n-1} \\ & & & c_n & a_n \end{pmatrix} = \begin{pmatrix} u_1 & & & & \\ w_2 & u_2 & & & \\ & \ddots & \ddots & & \\ & & w_{n-1} & u_{n-1} & \\ & & & w_n & u_n \end{pmatrix} \begin{pmatrix} 1 & v_1 & & & \\ & 1 & v_2 & & \\ & & \ddots & \ddots & \\ & & & 1 & v_{n-1} \\ & & & & 1 \end{pmatrix}$$

■ 分解步骤:

$$\begin{cases} u_i = a_i - c_i v_{i-1}, & i = 1, \dots, n \\ v_i = b_i / u_i, & i = 1, \dots, n \end{cases}$$

■ 回代步骤:

$$y_i = (f_i - c_i y_{i-1}) / u_i, \quad i = 1, 2, \dots, n$$

$$x_i = y_i - v_i x_{i+1}, \quad i = n, n-1, \dots, 1$$

■ 总计算量:

$$6 * n + 2 * n = 8n = O(n)$$

直接分解法



■ 追赶法求解三对角线性方程组算法

Algorithm 13 Tridigonal System Algorithm

Input:

$n, (a_i), (b_i), (c_i), (f_i)$

```
1:  $u_1 \leftarrow a_1$ ;  
2:  $v_1 \leftarrow b_1/u_1$ ;  
3:  $y_1 \leftarrow f_1/u_1$ ;  
4: for  $i = 2$  to  $n$  do  
5:    $u_i \leftarrow a_i - c_i v_{i-1}$ ;  
6:    $v_i \leftarrow b_i/u_i$ ;  
7:    $y_i \leftarrow (f_i - c_i y_{i-1})/u_i$ ;  
8: end for  
9:  $x_n \leftarrow y_n$ ;  
10: for  $i = n - 1$  to  $1$  step  $-1$  do  
11:    $x_i = y_i - v_i x_{i+1}$ ;  
12: end for
```

Output:

(x_i)

直接分解法



■ 对称正定阵的 LDL^T 分解：若 A 对称正定，则

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

■ 分解步骤：

$$\begin{cases} d_k = a_{kk} - \sum_{r=1}^{k-1} d_r l_{kr}^2 \\ l_{ik} = (a_{ik} - \sum_{r=1}^{k-1} d_r l_{ir} l_{kr}) / d_k, \quad i = k+1, \dots, n \end{cases}$$

时间复杂度、空间复杂度较
Dolittle分解或Courant分解
减半，亦称为Cholesky分解

■ 回代步骤：

$$z_i = b_i - \sum_{j=1}^{i-1} l_{ij} z_j, \quad i = 1, 2, \dots, n$$

$$y_i = z_i / d_i, \quad i = 1, 2, \dots, n$$

$$x_i = y_i - \sum_{j=i+1}^n l_{ji} x_j, \quad i = n, n-1, \dots, 1$$

直接分解法



■ LDL^T 分解算法

Algorithm 14 LDL^T Factorization Algorithm

Input:

$n, (a_{ij})$

- 1: **for** $k = 1$ to n **do**
- 2: $d_k \leftarrow a_{kk} - \sum_{s=1}^{k-1} d_s l_{ks}^2$;
- 3: **for** $i = k + 1$ to n **do**
- 4: $l_{ik} = (a_{ik} - \sum_{s=1}^{k-1} d_s l_{is} l_{ks}) / d_k$;
- 5: **end for**
- 6: **end for**

Output:

$(d_i), (l_{ij})$

- **定义：**对任一向量 $\mathbf{x} \in \mathbb{R}^n$ ，按照一个规则确定一个实数与它对应，记该实数为 $\|\mathbf{x}\|$ ，即映射： $\|\cdot\|: \mathbb{R}^n \rightarrow \mathbb{R}^+ \cup \{0\}$ 。若 $\|\mathbf{x}\|$ 满足下面三个性质：

(1) 非负性 $\forall \mathbf{x} \in \mathbb{R}^n, \|\mathbf{x}\| \geq 0$, 且 $\|\mathbf{x}\| = 0 \Leftrightarrow \mathbf{x} = 0$

(2) 齐次性 $\forall a \in \mathbb{R}, \|a\mathbf{x}\| = |a| \cdot \|\mathbf{x}\|$

(3) 三角不等式 $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n, \|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$

则称 $\|\mathbf{x}\|$ 为向量范数。

- **作用：**

- 向量之间的距离、误差
- 向量序列的收敛性
- 向量的邻域、开集、连续性等
-

■ 定义：设 $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(m)}, \dots$ 为 \mathbb{R}^n 空间内的向量序列，若存在向量 $\alpha \in \mathbb{R}^n$ 使得 $\lim_{m \rightarrow \infty} \|\mathbf{x}^{(m)} - \alpha\| = 0$ ，则称向量序列 $\mathbf{x}^{(m)}$ 是收敛的，向量 α 称为向量序列 $\mathbf{x}^{(m)}$ 的极限

■ 性质：

- 范数 $\|\mathbf{x}\| = \|(x_1, x_2, \dots, x_n)\|$ 是关于坐标 x_1, x_2, \dots, x_n 的连续函数
- 向量序列 $\mathbf{x}^{(m)} = (x_1^{(m)}, x_2^{(m)}, \dots, x_n^{(m)})^T$ 收敛的充分必要条件为序列的每个分量收敛，即 $\alpha_i = \lim_{m \rightarrow \infty} x_i^{(m)}$, $i = 1, 2, \dots, n$

■ 常见的范数有：

$$\|\mathbf{x}\|_1 = \sum_{i=1}^n |x_i|, \quad \|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^n x_i^2}, \quad \|\mathbf{x}\|_\infty = \max \{|x_i|\}$$

$$\|\mathbf{x}\|_p = \left(\sum_{i=1}^n |x_i|^p \right)^{1/p}, \quad \mathbf{x} = \{x_1, x_2, \dots, x_n\}$$

■ (范数等价性) 设 $\|\mathbf{x}\|_p$ 和 $\|\mathbf{x}\|_q$ 为任意两向量范数，则存在与 \mathbf{x} 无关的正常数 c_1 与 c_2 使得

$$c_1 \|\mathbf{x}\|_q \leq \|\mathbf{x}\|_p \leq c_2 \|\mathbf{x}\|_q, \quad \forall \mathbf{x} \in \mathbb{R}^n$$

■ 一些常用范数的等价关系：

$$\|\mathbf{x}\|_2 \leq \|\mathbf{x}\|_1 \leq \sqrt{n} \|\mathbf{x}\|_2$$

$$\|\mathbf{x}\|_\infty \leq \|\mathbf{x}\|_2 \leq \sqrt{n} \|\mathbf{x}\|_\infty$$

$$\|\mathbf{x}\|_\infty \leq \|\mathbf{x}\|_1 \leq n \|\mathbf{x}\|_\infty$$

- 定义：诱导矩阵范数（或导出范数，或从属范数）

$$\|A\| = \sup_{\mathbf{x} \in \mathbb{R}^n, \mathbf{x} \neq 0} \frac{\|A\mathbf{x}\|}{\|\mathbf{x}\|} = \sup_{\mathbf{x} \in \mathbb{R}^n, \|\mathbf{x}\|=1} \|A\mathbf{x}\|$$

- 不难验证：诱导矩阵范数满足非负性、齐次性、三角不等式
- 性质：

（相容性） $\forall A \in \mathbb{R}^{n \times n}, \mathbf{x} \in \mathbb{R}^n, \|A\mathbf{x}\| \leq \|A\| \cdot \|\mathbf{x}\|$

（可乘性） $\forall A, B \in \mathbb{R}^{n \times n}, \|AB\| \leq \|A\| \cdot \|B\|$

■ 常用矩阵范数

$$\|\mathbf{A}\|_1 = \max_{1 \leq j \leq n} \sum_{i=1}^n |a_{ij}|$$

列和的最大值

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

行和的最大值

$$\|\mathbf{A}\|_2 = \sqrt{\rho(\mathbf{A}^T \mathbf{A})}$$

谱半径: $\rho(\mathbf{A}) = \max_{1 \leq r \leq n} |\lambda_r|$

$$\|\mathbf{A}\|_F = \left(\sum_{i,j=1}^n |a_{ij}|^2 \right)^{1/2}$$

非诱导矩阵范数, 但与 $\|\mathbf{x}\|_2$ 相容

■ 定理: 若 λ 为矩阵 \mathbf{A} 的特征值, 则 $|\lambda| \leq \|\mathbf{A}\|$

■ 推论: 矩阵 \mathbf{A} 的任一 (相容) 矩阵范数均不小于 \mathbf{A} 的谱半径, 即 $\rho(\mathbf{A}) \leq \|\mathbf{A}\|$

■ 定理: 对任意 $\varepsilon > 0$, 则存在一个矩阵相容范数 $\|\cdot\|$, 使得 $\|\mathbf{A}\| \leq \rho(\mathbf{A}) + \varepsilon$, 且 $\|\mathbf{I}\| = 1$

矩阵的条件数



- 定义：若矩阵 A 非奇异，称

$$\text{Cond}_p(A) = \|A\|_p \cdot \|A^{-1}\|_p$$

为 A 的条件数，其中 $\|\cdot\|_p$ 表示矩阵的某种范数

- 条件数反应了矩阵对误差的放大率

- 矩阵系数扰动分析：

$$\begin{aligned} (A + \delta A)(x + \delta x) &= b \Rightarrow \delta A(x + \delta x) = b - Ax - A\delta x = -A\delta x \\ \Rightarrow \delta x &= (A + \delta A)^{-1}(-\delta Ax) = [A(I + A^{-1}\delta A)]^{-1}(-\delta Ax) \\ &= (I + A^{-1}\delta A)^{-1}A^{-1}(-\delta Ax) \\ \Rightarrow \|\delta x\| &\leq \|(I + A^{-1}\delta A)^{-1}\| \cdot \|A^{-1}\| \cdot \|\delta A\| \cdot \|x\| \\ \Rightarrow \frac{\|\delta x\|}{\|x\|} &\leq \frac{\text{Cond}(A) \frac{\|\delta A\|}{\|A\|}}{1 - \text{Cond}(A) \frac{\|\delta A\|}{\|A\|}} \end{aligned}$$

矩阵的条件数



■ 右端项系数扰动分析：

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

$$\Rightarrow \mathbf{A}\delta\mathbf{x} = \delta\mathbf{b}$$

$$\Rightarrow \delta\mathbf{x} = \mathbf{A}^{-1}\delta\mathbf{b}$$

$$\Rightarrow \frac{\|\delta\mathbf{x}\|}{\|\mathbf{x}\|} \leq \|\mathbf{A}^{-1}\| \cdot \|\mathbf{A}\| \cdot \frac{\|\delta\mathbf{b}\|}{\|\mathbf{b}\|}$$

■ 整体系数扰动分析：

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

$$\Rightarrow \frac{\|\delta\mathbf{x}\|}{\|\mathbf{x}\|} \leq \frac{\text{Cond}(\mathbf{A})}{1 - \text{Cond}(\mathbf{A}) \frac{\|\delta\mathbf{A}\|}{\|\mathbf{A}\|}} \left(\frac{\|\delta\mathbf{A}\|}{\|\mathbf{A}\|} + \frac{\|\delta\mathbf{b}\|}{\|\mathbf{b}\|} \right)$$

矩阵的条件数



■ 一般判断矩阵是否病态，并不计算 A^{-1} ，而由经验得出，譬如：

- 行列式很大或很小（如某些行、列近似相关）；
- 元素间相差大数量级，且无规则；
- 主元消去过程中出现小主元；
- 特征值相差大数量级。

■ 定理：设矩阵 A 非奇异，则

$$\min \left\{ \frac{\|\delta A\|_2}{\|A\|_2} : A + \delta A \text{ is not singular} \right\} = \frac{1}{\|A^{-1}\|_2 \cdot \|A\|_2} = \frac{1}{\text{Cond}(A)}$$

■ 病态问题：增加计算精度会改善解的精度，但是不能消除（除非进行符号计算）。目前，即使最出色的数值线代数软件对病态问题也束手无策！

矩阵的条件数



■ 例：Hilbert矩阵

$$\mathbf{H} = (h_{ij})_{n \times n}, h_{ij} = \frac{1}{i+j-1} = \int_0^1 x^{i+j-2} dx$$

$$\mathbf{H}^{-1} = (g_{ij})_{n \times n}, g_{ij} = (-1)^{i+j} (i+j-1) \binom{n+i-1}{n-j} \binom{n+j-1}{n-i} \binom{i+j-2}{i-1}^2$$

$$\text{Cond}(\mathbf{H}) = O\left((1 + \sqrt{2})^{4n} / \sqrt{n}\right)$$

$$\mathbf{H} = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} \\ \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} & \frac{1}{8} \\ \frac{1}{5} & \frac{1}{6} & \frac{1}{7} & \frac{1}{8} & \frac{1}{9} \end{bmatrix}$$