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 *

线性方程组的求解方法



解法概述

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

直接法

优点: 稳定可靠, 有限步终止 → 在工程界很受欢迎

缺点: 运算量大 $\mathcal{O}(n^3)$ → 不适合大规模问题

(针对特殊结构的快速算法除外)

1 Gauss 消去法和 LU 分解

1.1 LU 分解

1.2 LU 分解的实现

1.3 三角方程求解

1.4 选主元 LU 分解

1.5 矩阵求逆

1.1 LU 分解

考虑线性方程组

$$Ax = b$$

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:

$$A = LU$$

其中 L 是单位下三角矩阵, U 为非奇异上三角矩阵.

Gauss 消去法

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

算法 1 Gauss 消去法

1: **将** A **进行** LU **分解:** $A = LU$;

2: **向前回代: 求解** $Ly = b$, **即得** $y = L^{-1}b$;

3: **向后回代: 求解** $Ux = y$, **即得** $x = U^{-1}y = (LU)^{-1}b = A^{-1}b$.

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, $k = 1, 2, \ldots, n$.

(板书)



A 非奇异, 则解存在唯一, 但并不一定存在 LU 分解!

推论 (LDU 分解) 设 $A \in \mathbb{R}^{n\times n}$ 的所有顺序主子矩阵都非奇异, 则存在单位下三角矩阵 L 和单位上三角矩阵 U, 以及非奇异的对角矩阵 D, 使得 $A = LDU$, 其中 L, U, D 都是唯一的. 反之, 若 A 存在 LDU 分解, 则 A 的所有顺序主子矩阵都非奇异.

对角占优矩阵的 LU 分解

一般的非奇异矩阵不一定存在 LU 分解, 但若 A 对角占优, 则一定存在 LU 分解.

定理 设 $A \in \mathbb{R}^{n\times n}$ 非奇异且列对角占优, 则 A 存在 LU 分解且 L 中的元素的绝对值都不超过 1. (留作练习, 数学归纳法)

1.2 LU 分解的实现 — 矩阵初等变换

给定一个矩阵 $A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \\ a_{n1} & \cdots & a_{nn} \end{bmatrix} \in \mathbb{R}^{n\times n}$.

第一步: 假定 $a_{11} \neq 0$, 构造矩阵 $(l_{i1} = \dfrac{a_{i1}}{a_{11}}, i = 2, \ldots, n.)$

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

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

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

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

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

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 $a_{kk}^{(k-1)} \neq 0\ (k = 3, 4, \ldots, n-1)$,

我们可以构造一系列的矩阵 $L_3, L_4, \ldots, L_{n-1}$, 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

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

LU 分解算法

算法 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**

算法评价

时间复杂度: 运算执行次数

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

计算优化

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. 比如计算 αABx, 其中 α 是数, A, B 是矩阵, x 是向量, 如果按照从左往右计算的话, 则运算量为 $\mathcal{O}(n^3)$, 但是如果先计算 αx, 然后计算 $B(\alpha x)$, 最后再计算 $A(B(\alpha x))$ 的话, 运算量则为 $\mathcal{O}(n^2)$, 相差一个量级.

Gauss 消去法的运算量

由算法 2 可知, LU 分解的运算量 (加减乘除) 为

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

加上回代过程的运算量 $\mathcal{O}(n^2)$, 总运算量为 $\frac{2}{3}n^3+\mathcal{O}(n^2)$

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, 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.

用 A 存储 L 和 U

算法 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**

```
% Matlab code : LU 分解
function A = mylu(A)
n = size(A,1);
for k = 1 : n-1
   if A(k,k) == 0, return; end
   for i = k+1 : n
      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
   end
end
```

为了充分利用 Matlab 向量运算优势, 提高效率, 可改写为

```
function A = mylu(A)
n = size(A,1);
for k = 1 : n-1
    if A(k,k) == 0, return; end
    A(k+1:n,k) = A(k+1:n,k)/A(k,k);
    A(k+1:n,k+1:n) = A(k+1:n,k+1:n) - A(k+1:n,k)*A(k,k+1:n);
end
```

哪种循环方式更高效: 数据更新顺序与计算效率



- 根据下标的循环次序, 算法 3 也称为 KIJ 型 LU 分解.
  在计算过程中, 算法对下标 k 的每次循环, 都需要更新 A 的第 $k+1$ 至第 n 行, 这种 **反复读写数据** 的做法会降低计算效率, 因此一般不建议使用.

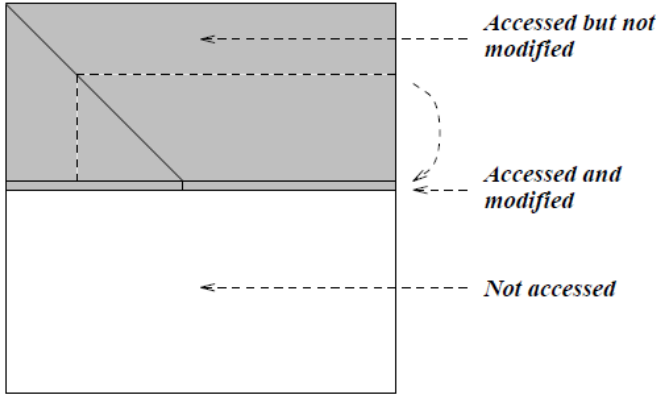
IKJ 型 LU 分解

如果数据是按行存储, 如 C/C++, 可采用下面的 IKJ 型 LU 分解.

算法 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?

1.3 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 5 向前回代求解 $Ly = b$ (假定 L 是一般的非奇异下三角矩阵)

1: $y_1 = b_1/l_{11}$
2: **for** $i = 2 : n$ **do**
3: **for** $j = 1 : i - 1$ **do**
4: $b_i = b_i - l_{ij}y_j$
5: **end for**
6: $y_i = b_i/l_{ii}$
7: **end for**

列存储方式

如果数据是按列存储的, 则采用列存储方式效率会高一些.

算法 6 向后回代求解 $Ux = y$

1: **for** $i = n : -1 : 1$ **do**
2: $\quad x_i = y_i / u_{ii}$
3: $\quad$ **for** $j = i - 1 : -1 : 1$ **do**
4: $\quad\quad y_j = y_j - x_i u_{ji}$
5: $\quad$ **end for**
6: **end for**

这两个算法的运算量均为 $n^2 + \mathcal{O}(n)$.

1.4 选主元 LU 分解

为什么要选主元

- 在 LU 分解算法 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.

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.

$(x_1 \approx -20.7, x_2 \approx 1.01)$ (精确解 $x_1 = 10.0$ 和 $x_2 = 1.00$)

(板书)

解. 根据 LU 分解算法 2, 我们可得

$$l_{11} = 1.00,\ l_{21} = a_{21}/a_{11} = 1.72 \times 10^2,\ l_{22} = 1.00,$$
$$u_{11} = a_{11} = 2.00 \times 10^{-2},\ u_{12} = a_{12} = 6.13 \times 10,$$
$$u_{22} = a_{22} - l_{21}u_{12} \approx -8.5 - 1.05 \times 10^4 \approx -1.05 \times 10^4,$$

即

$$A \approx \begin{bmatrix} 1.00 & 0 \\ 1.72 \times 10^2 & 1.00 \end{bmatrix} \begin{bmatrix} 2.00 \times 10^{-2} & 6.12 \times 10 \\ 0 & -1.05 \times 10^4 \end{bmatrix}.$$

解方程组 $Ly = b$ 可得

$$y_1 = 6.15 \times 10, \quad y_2 = b_2 - l_{21}y_1 \approx -1.06 \times 10^4.$$

解方程组 $Ux = y$ 可得

$$x_2 = y_2/u_{22} \approx 1.01, \quad x_1 = (y_1 - u_{12} * x_2)/u_{11} \approx -0.413/u_{11} \approx -20.7$$

与精确解 $x_1 = 10.0$ 和 $x_2 = 1.00$ 差别很大. □

选主元 LU 分解

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

列主元 LU 分解 / 部分选主元 LU 分解 / PLU

设 $A \in \mathbb{R}^{n\times n}$ 非奇异, 则存在置换矩阵 P, 以及单位下三角矩阵 L 和非奇异上三角矩阵 U, 使得

$$PA = LU.$$

(板书)

第 k 步时, 如何选取置换矩阵?

部分选主元 Gauss 消去法 / GEPP

选取第 k 列的第 k 到第 n 个元素中绝对值最大的作为主元, 这种选取方法称为 “部分选主元 Gauss 消去法”, 简称 GEPP, 也称为列主元 Gauss 消去法.

全主元 Gauss 消去法 / GECP

选取剩下矩阵中绝对值最大的作为主元, 这种选取方法称为 “全主元 Gauss 消去法”, 简称 GECP



- 在绝大多数情况下, GEPP 是向后稳定的, 但理论上也存在失败的例子.
- GECP 比 GEPP 更稳定, 但工作量大.
- 事实上, 在大部分实际应用中, GEPP 与 GECP 具有同样的数值稳定性.

算法 7 部分选主元 LU 分解

1: $p = 1 : n$ % 用于记录置换矩阵
2: **for** $k = 1$ to $n - 1$ **do**
3: $[a_{\max}, l] = \max_{k \le i \le n} |a_{ik}|$ % 选列主元, l 表示主元所在的行
4: **if** $l \neq k$ **then**
5: **for** $j = 1$ to n **do**
6: $tmp = a_{kj}, a_{kj} = a_{lj}, a_{lj} = tmp$ % 交换两行
7: **end for**
8: $tmp = p(k), p(k) = p(l), p(l) = tmp$ % 更新置换矩阵
9: **end if**
10: **for** $i = k + 1$ to n **do**
11: $a_{ik} = a_{ik}/a_{kk}$ % 计算 L 的第 k 列
12: **end for**
13: **for** $i = k + 1$ to n **do**
14: **for** $j = k + 1$ to n **do**
15: $a_{ij} = a_{ij} - a_{ik} * a_{kj}$ % 更新 $A(k + 1 : n, k + 1 : n)$
16: **end for**
17: **end for**
18: **end for**

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.

$(x_1 \approx 10.0, x_2 \approx 0.998)$ (精确解 $x_1 = 10.0$ 和 $x_2 = 1.00$)

(板书)

解. 由于 $|a_{21}| > |a_{11}|$, 因此我们交换第一行与第二行, 即取 $P = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$, 然后计算 $\tilde{A} = PA$ 的 LU 分解, 可得

$$
\begin{aligned}
&l_{11} = 1.00,\ l_{21} = \tilde{a}_{21}/\tilde{a}_{11} = 5.83 \times 10^{-3},\ l_{22} = 1.00,\\
&u_{11} = \tilde{a}_{11} = 3.43,\ u_{12} = \tilde{a}_{12} = -8.50,\\
&u_{22} = \tilde{a}_{22} - l_{21}u_{12} \approx 6.13 \times 10 + 4.96 \times 10^{-2} \approx 6.13 \times 10,
\end{aligned}
$$

即

$$
PA \approx \begin{bmatrix} 1.00 & 0 \\ 5.83 \times 10^{-3} & 1.00 \end{bmatrix} \begin{bmatrix} 3.43 & -8.50 \\ 0 & 6.13 \times 10 \end{bmatrix}.
$$

解方程组 $Ly = P^T b$ 可得 $y_1 = 2.58 \times 10, \quad y_2 \approx 6.12 \times 10.$

解方程组 $Ux = y$ 可得 $x_2 = y_2/u_{22} \approx 0.998, \quad x_1 = (y_1 - u_{12} * x_2)/u_{11} \approx 34.3/u_{11} \approx 10.0$

所以, 数值解具有 3 位有效数字. □

1.5 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

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

等价于求解下面 $2n$ 个三角线性方程组

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

思考: 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 对称正定线性方程组

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

$$A = LL^{\mathsf{T}}.$$

该分解称为 **Cholesky 分解**.

(板书, 数学归纳法或利用 LU 分解)

Cholesky 分解的实现: 待定系数法

设 $A = LL^{\mathsf{T}}$, 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

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

Cholesky 分解算法

算法 8 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: $\quad$ **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: $\quad$ **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 $\frac{1}{3}n^3 + \mathcal{O}(n^2)$, 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
- 基于 Cholesky 分解的求解方法称为 **平方根法**

例 已知矩阵 $A=\begin{bmatrix}4&2&8&0\\2&10&10&9\\8&10&21&6\\0&9&6&34\end{bmatrix}$, 计算 A 的 Cholesky 分解. (板书)

解. 设 A 的 Cholesky 分解为 $A=LL^{\mathsf{T}}$, 即

$$A=\begin{bmatrix}4&2&8&0\\2&10&10&9\\8&10&21&6\\0&9&6&34\end{bmatrix}=\begin{bmatrix}l_{11}&&&\\l_{21}&l_{22}&&\\l_{31}&l_{32}&l_{33}&\\l_{41}&l_{42}&l_{43}&l_{44}\end{bmatrix}\begin{bmatrix}l_{11}&&&\\l_{21}&l_{22}&&\\l_{31}&l_{32}&l_{33}&\\l_{41}&l_{42}&l_{43}&l_{44}\end{bmatrix}^{\mathsf{T}}.$$

比较等式两边的第一行可得

$$\begin{aligned}&l_{11}^2=4\Longrightarrow l_{11}=2,\quad (l_{11}>0)\\&l_{11}l_{21}=2\Longrightarrow l_{21}=1,\\&l_{11}l_{31}=8\Longrightarrow l_{21}=4,\\&l_{11}l_{41}=0\Longrightarrow l_{21}=0.\end{aligned}$$

比较等式两边的第二行可得

$$l_{21}^2+l_{22}^2=10\Longrightarrow l_{22}^2=9\Longrightarrow l_{22}=3,\quad (l_{22}>0)$$
$$l_{21}l_{31}+l_{22}l_{32}=10\Longrightarrow l_{32}=(10-4)/3=2,$$
$$l_{21}l_{41}+l_{22}l_{42}=9\Longrightarrow l_{42}=(9-0)/3=3.$$

比较等式两边的第三行可得

$$l_{31}^2+l_{32}^2+l_{33}^2=21\Longrightarrow l_{33}^2=21-16-4=1\Longrightarrow l_{33}=1\quad (l_{33}>0)$$
$$l_{31}l_{41}+l_{32}l_{42}+l_{33}l_{43}=6\Longrightarrow l_{43}=(6-0-6)/1=0.$$

比较等式两边的第四行可得

$$l_{41}^2+l_{42}^2+l_{43}^2+l_{44}^2=34\Longrightarrow l_{44}^2=34-0-9-0=25\Longrightarrow l_{44}=5.\quad (l_{44}>0)$$

所以 A 的 Cholesky 分解为

$$A=\begin{bmatrix}2&&&\\1&3&&\\4&2&1&\\0&3&0&5\end{bmatrix}\begin{bmatrix}2&&&\\1&3&&\\4&2&1&\\0&3&0&5\end{bmatrix}^{\mathsf{T}}.$$

□

LDL^{T} 分解与改进的平方根法

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

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

基于该分解的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 9 改进的平方根法

for $j = 1$ to n **do**　% 先计算分解
　　$d_j = a_{jj} - \sum_{k=1}^{j-1} l_{jk}^2 d_k$
　　for $i = j+1$ to n **do**
　　　　$l_{ij} = (a_{ij} - \sum_{k=1}^{j-1} l_{ik} d_k l_{jk})/d_j$
　　end for
end for
$y_1 = b_1$　% 解方程组: $Ly = b$ 和 $DL^{\mathsf{T}}x = y$
for $i = 2$ to n **do**
　　$y_i = b_i - \sum_{k=1}^{i-1} l_{ik} y_k$
end for
$x_n = y_n/d_n$
for $i = n-1$ to 1 **do**
　　$x_i = y_i/d_i - \sum_{k=i+1}^{n} l_{ki} x_k$
end for

2.2 对称不定线性方程组

设 $A \to$ 非奇异, 对称 但 不定

$\Longrightarrow$ 若 A 存在 LU 分解, 即 $A = LU$, 则可写成 $A = LDL^{\mathsf{T}}$

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.

$\mathrm{LDL}^{\mathsf{T}}$ 分解

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

$$PAP^{\mathsf{T}} = LDL^{\mathsf{T}}. \tag{2.1}$$

通常称 (2.1) 为对称矩阵的 $\mathrm{LDL}^{\mathsf{T}}$ 分解.

☹ 不幸的是, 这样的置换矩阵可能不一定存在, 即分解 (2.1) 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 $\mathrm{LDL}^{\mathsf{T}}$ 分解.

解决方法：块 $\mathrm{LDL}^{\mathsf{T}}$ 分解

设 A 对称非奇异, 则存在置换矩阵 P 使得

$$PAP^{\mathsf{T}} = \begin{bmatrix} B & E^{\mathsf{T}} \\ E & C \end{bmatrix}, \quad B \in \mathbb{R} \text{ 或 } B \in \mathbb{R}^{2\times 2}, \text{ 且非奇异.}$$

思考：如何实现?

块 $\mathrm{LDL}^{\mathsf{T}}$ 分解

于是可以对 PAP^{T} 进行块对角化, 即

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

其中 $C - EB^{-1}E^{\mathsf{T}}$ 是 Schur 补, 对称非奇异.

不断重复以上过程, 就可以得到 A 的块 $\mathrm{LDL}^{\mathsf{T}}$ 分解:

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

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

选主元块 LDL^T 分解

选主元策略

- 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 $\mathcal{O}(n^2)$ 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, 精度更高.



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

$$|b_1| > |c_1| > 0,\ |b_n| > |a_{n-1}| > 0,\ |b_i| \geq |a_{i-1}| + |c_i|,\ i = 2, \ldots, n-1 \tag{2.2}$$

$$a_i c_i \neq 0, \quad i = 1, \ldots, n-1. \tag{2.3}$$

即 A 是 不可约弱对角占优.

思考：如果 A 可约, 怎么处理? (什么情况下，三对角矩阵可约？)

此时, 我们可以得到下面的三角分解

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

得递推公式:

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

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

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

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 (2.2) 和 (2.3). 则 A 非奇异, 且

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

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

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

(板书)

算法 10 追赶法

$\beta_1 = c_1/b_1$
$y_1 = f_1/b_1$
for $i = 2$ to $n-1$ **do**
    $\alpha_i = b_i - a_{i-1}\beta_{i-1}$
    $\beta_i = c_i/\alpha_i$
    $y_i = (f_i - a_{i-1}y_{i-1})/\alpha_i$
end for
$\alpha_n = b_n - a_{n-1}\beta_{n-1}$
$y_n = (f_n - a_{n-1}y_{n-1})/\alpha_n$
$x_n = y_n$
for $i = n-1$ to 1 **do**
    $x_i = y_i - \beta_i x_{i+1}$
end for

几点说明

- **追赶法** (也称为 **Thomas 算法**) 的运算量大约为 $8n$.
- 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, α_i 可以不用存储. 但 β_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.

列对角占优情形

需要指出的是, 我们也可以考虑下面的分解

$$A=\begin{bmatrix} b_1 & c_1 & & \\ a_1 & \ddots & \ddots & \\ & \ddots & \ddots & c_{n-1} \\ & & a_{n-1} & b_n \end{bmatrix} = \begin{bmatrix} 1 & & & \\ \gamma_1 & 1 & & \\ & \ddots & \ddots & \\ & & \gamma_{n-1} & 1 \end{bmatrix} \begin{bmatrix} \alpha_1 & c_1 & & \\ & \alpha_2 & \ddots & \\ & & \ddots & c_{n-1} \\ & & & \alpha_n \end{bmatrix}. \tag{2.4}$$

但此时 $|\gamma_i|$ 可能大于 1. 比如当 $|b_1|<|a_1|$ 时, $|\gamma_1|=|a_1/b_1|>1$. 所以在回代求解时, 误差可能得不到有效控制. 同时, 计算 γ_i 时也可能会产生较大误差 (大数除小数). **如果 A 列对角占优, 则可以保证 $|\gamma_i|<1$.**

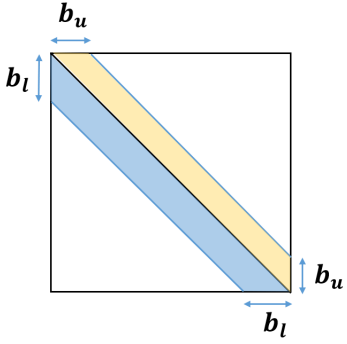


如果 A 是 (行) 对角占优, 则采用前面的分解;
如果 A 是**列对角占优**, 则采用分解 (2.4).

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

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



带状矩阵的 LU 分解

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.

练习：统计求解带状矩阵 $Ax = b$ 的运算量.

带状矩阵的部分选主元 LU 分解

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

Toeplitz 矩阵

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



- 易知, T_n 是反向对称 (persymmetric) 矩阵.
- 通过观察可知, T_n 反转后即为对称矩阵.

反向对称矩阵

记 J_n 为 n 阶 反向单位矩阵:

$$J_n = \begin{bmatrix} & & & 1 \\ & & 1 & \\ & \iddots & & \\ 1 & & & \end{bmatrix}$$

易知 $J_n^{\mathsf{T}} = J_n^{-1} = J_n, \quad J_n^2 = I_n.$

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

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

反向对称矩阵的逆

若 A 可逆, 则可得

$$A^{-1} = J_n^{-1} \left(A^{\mathsf{T}}\right)^{-1} J_n^{-1} = J_n \left(A^{-1}\right)^{\mathsf{T}} J_n,$$

即 **反向对称矩阵的逆也是反向对称矩阵**.

Toeplitz 矩阵的逆

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

$$T_n y = -t^{(n)}$$

其中 $t^{(n)} = [t_1, t_2, \ldots, t_{n-1}, t_n]^{\mathsf{T}}$. 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.

对角线元素的缩放

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/递归** 技巧来求解.

求解 Yule-Walker 方程组的递推方法

记 $T_k y = -t^{(k)}$ 的解为 $y^{(k)}$ $\Longrightarrow$ 假定已经求出

考虑 $T_{k+1} y = -t^{(k+1)}$, 设解为 $y^{(k+1)} = \begin{bmatrix} u^{(k)} \\ \alpha_k \end{bmatrix}$.

代入后可得 递推公式:

$$\alpha_k = \frac{-t_{k+1} - \left(t^{(k)}\right)^{\mathsf{T}} J_k y^{(k)}}{1 + \left(t^{(k)}\right)^{\mathsf{T}} y^{(k)}}, \quad u^{(k)} = y^{(k)} + \alpha_k J_k y^{(k)} \qquad k = 1, 2, \ldots.$$

因此, 可以从一阶 Yule-Walker 方程出发, 利用递推公式计算 $T_n y = -t^{(n)}$ 的解.

思考: 总的运算量 (乘法与加减运算) 大约是多少?

减少运算量 — 递推方法计算分母

引入辅助变量 $\tau_k \triangleq 1 + \left(t^{(k)}\right)^{\mathsf{T}} y^{(k)}$, 则

$$\begin{aligned}
\tau_{k+1} &= 1 + \left(t^{(k+1)}\right)^{\mathsf{T}} y^{(k+1)} \\
&= 1 + \begin{bmatrix} \left(t^{(k)}\right)^{\mathsf{T}} & t_{k+1} \end{bmatrix} \begin{bmatrix} y^{(k)} + \alpha_k J_k y^{(k)} \\ \alpha_k \end{bmatrix} \\
&= 1 + \left(t^{(k)}\right)^{\mathsf{T}} y^{(k)} + \alpha_k \left(t_{k+1} + \left(t^{(k)}\right)^{\mathsf{T}} J_k y^{(k)} \right) \\
&= (1 - \alpha_k^2)\tau_k.
\end{aligned}$$

☞ 总运算量降为 $2n^2$.

Durbin 算法

算法 11 求解 Yule-Walker 方程组的 Durbin 算法

输入数据: $t = [t_1, t_2, \ldots, t_n]$ % 注: 这里假定 $t_0 = 1$

$y(1) = -t_1, \tau = 1, \alpha = -t_1$

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

$\tau = (1 - \alpha^2)\tau$

$$\alpha = -\frac{1}{\tau}\left(t_{k+1} + \sum_{i=1}^{k} t_i\, y(k+1-i)\right)$$

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

$y(k+1) = \alpha$

end for

注记: 算法不会中断 ($\tau_k \neq 0$)

$$\begin{bmatrix} I & J_k y^{(k)} \\ 0 & 1 \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} T_k & J_k t^{(k)} \\ \left(t^{(k)}\right)^{\mathsf{T}} J_k & 1 \end{bmatrix} \begin{bmatrix} I & J_k y^{(k)} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} T_k & 0 \\ 0 & 1 + \left(t^{(k)}\right)^{\mathsf{T}} y^{(k)} \end{bmatrix},$$

由 T_{k+1} 的对称正定性可知 $1 + \left(t^{(k)}\right)^{\mathsf{T}} y^{(k)} > 0$.

✍ 注: 事实上, $1 + \left(t^{(k)}\right)^{\mathsf{T}} y^{(k)}$ 是 T_{k+1} 的 Schur 补.

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.

我们仍然利用**递推**方法来求解: 假定 $x^{(k)}$ 和 $y^{(k)}$ 是下面两个方程组的解

$$T_k x = [b_1, b_2, \ldots, b_k]^{\mathsf{T}} \triangleq b^{(k)}, \qquad T_k y = -b^{(k)}$$

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

$$v^{(k)} = x^{(k)} + \beta_k J_k y^{(k)}, \quad \beta_k = \frac{b_{k+1} - \left(t^{(k)}\right)^{\mathsf{T}} J_k x^{(k)}}{1 + \left(t^{(k)}\right)^{\mathsf{T}} y^{(k)}}$$

Levinson 算法

先计算 $T_k x = b^{(k)}$ 和 $T_k x = -t^{(k)}$ 的解, 再得到 $T_{k+1} x = b^{(k+1)}$ 的解.

算法 12 Levinson 算法 (总运算量约 $4n^2$)

输入数据: $t = [t_1, t_2, \ldots, t_{n-1}]$ **和** $b = [b_1, b_2, \ldots, b_n]$ % 这里假定 $t_0 = 1$
$y(1) = -t_1$, $x(1) = b_1$, $\tau = 1$, $\alpha = -t_1$
for $k = 1$ to $n-1$ **do**
  $\tau = (1-\alpha^2)\tau, \quad \beta = \dfrac{1}{\tau}\left(b_{k+1} - \sum_{i=1}^{k} t_i x(k+1-i)\right)$
  $x(1:k) = x(1:k) + \beta y(k:-1:1), \quad x(k+1) = \beta$
  if $k < n-1$ **then**
    $\alpha = -\dfrac{1}{\tau}\left(t_{k+1} + \sum_{i=1}^{k} t_i y(k+1-i)\right)$
    $y(1:k) = y(1:k) + \alpha y(k:-1:1), \quad y(k+1) = \alpha$
  end if
end for

非对称 Toeplitz 线性方程组

考虑非对称 Toeplitz 线性方程组 $T_n x = b$, 其中

$$T_n = \begin{bmatrix} 1 & t_1 & \cdots & t_{n-1} \\ s_1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & t_1 \\ s_{n-1} & \cdots & s_1 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}.$$

假定 T_n 的所有顺序主子矩阵都非奇异.

同样通过递归方式来计算 $T_n x = b$ 的解, 但此时需要用到两个辅助线性方程组:

$$T_k^{\mathsf{T}} y = -t^{(k)} \quad \text{和} \quad T_k z = -[s_1, s_2, \ldots, s_k]^{\mathsf{T}} \triangleq -s^{(k)}.$$

具体推导过程可参见讲义.

注记

在数学与工程的许多应用中都会出现 Toeplitz 线性方程组.

Levinson 算法是较早出现的快速算法, 但只具有弱稳定性.

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"	$\mathcal{O}(n\log^2 n)$	$\mathcal{O}(n)$
Superfast preconditioner	$\mathcal{O}(n\log n)$	$\mathcal{O}(n)$

- Fast : Levinson-Durbin (1946), Trench (1964), ...
- Fast stable: Bareiss (1969), 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 矩阵条件数

3.2 扰动方程

3.3 δx 与 $\hat{x}$ 的关系

3.4 δx 与 x_* 的关系

3.5 δx 与残量的关系

3.6 相对扰动分析 *

3.1 矩阵条件数

例 考虑线性方程组 $Ax = b$, 其中 $A = \begin{bmatrix} 1 & 1 \\ 1 & 1.0001 \end{bmatrix}$, $b = [2, 2.0001]^{\mathsf{T}}$, 可求得其解为

$$x = [1, 1]^{\mathsf{T}}$$

如果 b 的第二个元素出现细微的偏差, 变为 $b = [2, 2]^{\mathsf{T}}$, 则解就变为

$$x = [2, 0]^{\mathsf{T}}$$

由此可见, 当右端项出现细微变化时, 解会出现很大的变化, 因此该线性方程组是**病态**的.

病态与条件数

定义 (病态问题与良态问题) 考虑线性方程组 $Ax = b$, 如果 A 或 b 的微小变化会导致解的巨大变化, 则称此线性方程组是 **病态** 的, 反之则是 **良态** 的.

定义 (矩阵条件数) 设 A 非奇异, $\|\cdot\|$ 是任一算子范数, 则称

$$\kappa(A) \triangleq \|A^{-1}\| \cdot \|A\|$$

为 A 的 **条件数**.

常用的矩阵条件数有

$$\kappa_2(A) \triangleq \|A^{-1}\|_2 \cdot \|A\|_2,\ \kappa_1(A) \triangleq \|A^{-1}\|_1 \cdot \|A\|_1,\ \kappa_\infty(A) \triangleq \|A^{-1}\|_\infty \cdot \|A\|_\infty$$

$\kappa_2(A)$ 也称为 **谱条件数**, 当 A 对称时, 有

$$\kappa_2(A) = \frac{\max\limits_{1\le i\le n} |\lambda_i|}{\min\limits_{1\le i\le n} |\lambda_i|}.$$

一般情况下, 如果没有特别指出, 则 $\kappa(A)$ 指的是矩阵 A 的谱条件数.

条件数是衡量一个矩阵是否病态的主要指标. 当矩阵条件数比较大时, 我们就称这个矩阵是 **病态** 的. 如果矩阵是病态的, 则近似解的误差受数据扰动的影响就可能会很大.

例 $A = \begin{bmatrix} 1 & 1 \\ 1 & 1.0001 \end{bmatrix} \implies \kappa_2(A) = 4.0002 \times 10^4$

例 **Hilbert 矩阵** 是一个典型的病态矩阵, 其定义如下:

$$H_n = [h_{ij}]_{i,j=1}^n, \quad \text{其中} \quad h_{ij} = \frac{1}{i+j-1}.$$

可以验证 H_n 是对称正定的, 但随着 n 的增长, 其条件数会快速增长:

n	2	4	6	8	10
$\kappa_2(H_n)$	1.9e+01	1.6e+04	1.5e+07	1.5e+10	1.6e+13

3.2 扰动方程

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.

向后误差分析

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.

3.3 δx 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 δx 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

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

(板书)

3.4 δx 与 x_* 的关系

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

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

(板书)

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

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

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

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

(留作课外自习)

✍ 上述定理中的结论对所有 p-范数都成立.



度量

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

表示 A 距离奇异矩阵集合的相对距离.

3.5 δx 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.

3.6 相对扰动分析 *

前面给出的误差 δx 与条件数 $\kappa(A)$, δA 和 δb 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 γ 很大时, $\kappa_2(A)$ 也很大, 因此前面定理中的上界可以是很大.

按分量分析: 记 $\delta A=\begin{bmatrix}\delta a_{11} & \\ & \delta a_{22}\end{bmatrix}, \delta b=\begin{bmatrix}\delta b_1\\ \delta b_2\end{bmatrix}$, 并设 $|\delta a_{ij}|\leq\varepsilon|a_{ij}|, |\delta b_i|\leq\varepsilon|b_i|$. 则

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

故 $\|\delta x\|_\infty\leq\dfrac{2\varepsilon}{1-\varepsilon}$. 如果 $\delta b=0$, 则 $\|\delta x\|_\infty\leq\dfrac{\varepsilon}{1-\varepsilon}$.

这个界前面定理中的上界相差约 γ 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

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

有时也称为 **Bauer 条件数**或 **Skeel 条件数**.

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

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

若 $\delta b = 0$, 则有

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

即

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

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 $|\delta A| \le \varepsilon |A|$, $|\delta b| \le \varepsilon |b|$ 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

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

其中 $r = b - A\hat{x}$.

更多数值计算稳定性和矩阵扰动分析方面的知识, 可参考相关资料.

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

4.2 Gauss 消去法的舍入误差分析

引理 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

$$(L+\delta L)\hat{y}=b, \quad |\delta L|\le \gamma_n|L|$$
$$(U+\delta U)\hat{x}=\hat{y}, \quad |\delta U|\le \gamma_n|U|.$$

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 *

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

在实际应用中, 一般会根据实际情况采用 **合适的精度** 进行运算.

如科学计算通常采用双精度计算, 神经网络采用单精度或者半精度即可.

- 为了减小舍入误差的影响, 也可以采用 **更高精度** 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.

$$Ax = b \quad \Longleftrightarrow \quad D_r^{-1} A D_c^{-1} y = D_r^{-1} b, \quad x = D_c^{-1} y$$

✍ 缩放技术在工程计算时经常会用到.

5.3 迭代改进法

设近似解 $\hat{x}$, 如果 $\hat{x}$ 没达到精度要求, 可以考虑对其进行修正

$$\tilde{x} = \hat{x} + z,$$

即加上一个向量 z, 得到新的近似解 $\tilde{x}$.

怎么选取 z

为了使得 $\tilde{x}$ 尽可能地接近真解，我们希望 $\tilde{x}$ 能满足 $A\tilde{x} = b$，即

$$A(\hat{x} + z) = b \Longleftrightarrow Az = r \quad \rightarrow \text{ 残量方程}$$

其中 $r = b - A\hat{x}$ 是残量.

在实际计算中, 我们只能得到近似解 $\hat{z}$, 但 $\|r - A\hat{z}\|$ 会比较小, 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 $\tilde{x} \triangleq \hat{x} + \hat{z}$ 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 13 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 $\mathcal{O}(n^2)$. 所以相对来讲还是比较经济的.



为提高计算精度, 在计算残量 r 时最好使用原始数据 A, 因此对 A 做 LU 分解时需要保留矩阵 A, 不能被 L 和 U 覆盖.



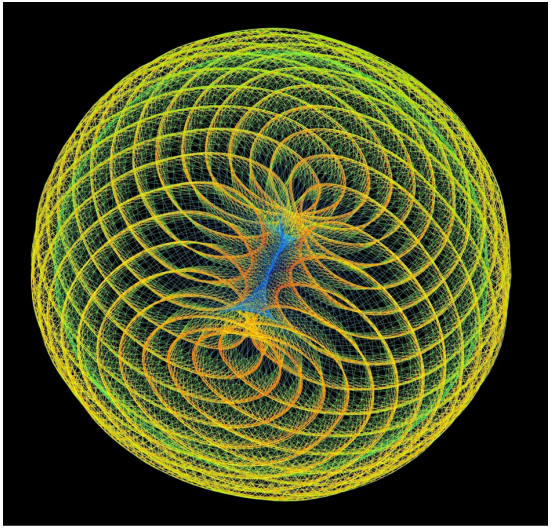
实际计算经验表明, 当 A 病态不是很严重时, 即 $\varepsilon_u \kappa_\infty(A) < 1$, 迭代法可以有效改进解的精度, 最后达到机器精度. 这里 ε_u 表示机器精度.
但 $\varepsilon_u \kappa_\infty(A) \geq 1$ 时, 效果可能不佳，得不到任何改善.

大规模 稀疏 线性方程组的直接解法

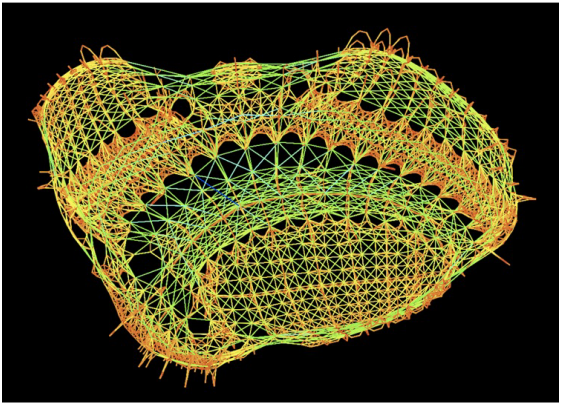
存在的困难

- 规模大, 百万量级以上或以亿为单位.
- LU/Cholesky 分解无法保持充分的稀疏性, 可能出现大量非零元, 导致存储困难, 即存储爆炸, 同时也无法有效降低运算量.

Example sparse matrices from THE SUITESPARSE MATRIX COLLECTION



A Barrier Hessian Matrix from Convex Quadratic Programming



The structural loading of an exhibition hall in Beijing

目前的策略

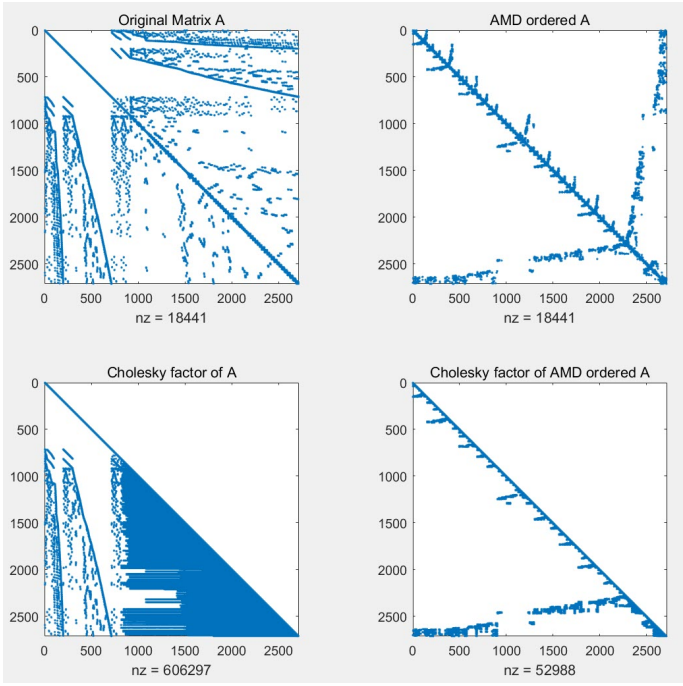
(1) 减少存储量: COO, CSR, CSC, MSR 等稀疏存储方式

(2) 与图相结合, 对矩阵进行重排序, 使得新矩阵的 LU/Cholesky 分解具有更好的稀疏性.

(3) 利用稀疏矩阵的重复结构: Supernodal, Frontal, Multifrontal, Hierarchical

(4) 并行计算, GPU 加速, 低秩逼近, update/downdate methods

Davis, Rajamanickam and Sid-Lakhdar, A survey of direct methods for sparse linear systems, *Acta Numerica*, 2016.


Original Matrix A
nz = 18441
AMD ordered A
nz = 18441
Cholesky factor of A
nz = 606297
Cholesky factor of AMD ordered A
nz = 52988
0
500
1000
1500
2000
2500


谢 谢
THANK YOU