

# 数值代数基础

文再文

北京大学北京国际数学研究中心

教材《最优化：建模、算法与理论》配套电子教案

<http://bicmr.pku.edu.cn/~wenzw/optbook.html>

致谢：本教案由张轩熙协助准备

- 1 矩阵的结构与算法复杂度
- 2 数值代数软件
- 3 通过分解矩阵来求解线性方程
- 4 分块消去法
- 5 稀疏数值线性代数
- 6 特征值分解与奇异值分解

# 矩阵结构与算法复杂度

对于  $A \in \mathbb{R}^{n \times n}$ ，求解方程组  $Ax = b$  的时间代价为：

- 对于大多数方法，计算所需时间会以  $n^3$  的量级增长。
- 当  $A$  具有某种结构时（如带状矩阵、稀疏矩阵、常对角矩阵），计算量可以减少。

## flop 计数

- flop (floating-point operation): 两个浮点数之间的一次加法、减法、乘法或除法
- 估计算法复杂度: 将计算所需的 flops 数量表示为一个关于问题规模的（多项式）函数，并简化成只保留最高项。
- 不能准确预测算法在现代计算机上的计算时间。
- 可作为复杂度的粗略估计

## 向量-向量运算 ( $x, y \in \mathbb{R}^n$ )

- 内积  $x^T y$ :  $2n - 1$  flops (当  $n$  较大时, 可记为  $2n$ )
- 求和  $x + y$ , 标量乘法  $\alpha x$ :  $n$  flops

## 矩阵-向量乘法 $y = Ax$ , $A \in \mathbb{R}^{m \times n}$

- $m(2n - 1)$  flops (当  $n$  较大时, 可记为  $2mn$ )
- 当  $A$  为稀疏矩阵时, 若有  $N$  个非零元素, 则计算量为  $2N$
- 当  $A = UV^T$ ,  $U \in \mathbb{R}^{m \times p}$ ,  $V \in \mathbb{R}^{n \times p}$  时, 计算量为  $2p(n + m)$

## 矩阵-矩阵乘法 $C = AB$ , $A \in \mathbb{R}^{m \times n}$ , $B \in \mathbb{R}^{n \times p}$

- $mp(2n - 1)$  flops (当  $n$  较大时, 可记为  $2mnp$ )
- 当  $A$  和 (或)  $B$  是稀疏矩阵时, 运算量会少一些
- 当  $m = p$  且  $C$  是对称矩阵时, flops 数量为  $(1/2)m(m + 1)(2n - 1) \approx m^2 n$

# 提纲

- 1 矩阵的结构与算法复杂度
- 2 数值代数软件
- 3 通过分解矩阵来求解线性方程
- 4 分块消去法
- 5 稀疏数值线性代数
- 6 特征值分解与奇异值分解

# Basic Linear Algebra Subroutines (BLAS)

BLAS实现了数值代数基本操作，这些操作按照运算量可划分为3个层次 (*level*)：

- *level 1*, 1973-1977:  $O(n)$  向量操作: 向量加法、数乘、点乘、范数运算
- *level 2*, 1984-1986:  $O(n^2)$  矩阵-向量操作: 矩阵-向量乘法, 三角矩阵-向量求解, 矩阵的秩1更新与对称矩阵的秩2更新
- *level 3*, 1987-1990:  $O(n^3)$  矩阵-矩阵操作: 矩阵-矩阵乘法, 三角矩阵-矩阵求解, 低秩矩阵更新

# BLAS的运算

Level 1	加法/数乘 点乘, 范数	$\alpha x, \quad \alpha x + y$ $x^T y, \quad \ x\ _2, \quad \ x\ _1$
Level 2	矩阵/向量乘积 秩1更新 秩2更新 三角矩阵-向量求解	$\alpha Ax + \beta y, \quad \alpha A^T x + \beta y$ $A + \alpha xy^T, \quad A + \alpha xx^T$ $A + \alpha xy^T + \alpha yx^T$ $\alpha T^{-1}x, \quad \alpha T^{-T}x$
Level 3	矩阵/矩阵乘积 秩 $k$ 更新 秩 $k$ 更新 三角矩阵求解	$\alpha AB + \beta C, \quad \alpha AB^T + \beta C$ $\alpha A^T B + \beta C, \quad \alpha A^T B^T + \beta C$ $\alpha AA^T + \beta C, \quad \alpha A^T A + \beta C$ $\alpha A^T B + \alpha B^T A + \beta C$ $\alpha T^{-1}C, \quad \alpha T^{-T}C$

# Level 1 BLAS 命名惯例

BLAS 的命名惯例受Fortran启发

cblas_	X	XXXX
前缀	数据类型	操作

数据类型:

s	单精度实数	d	双精度实数
c	单精度复数	z	双精度复数

操作:

axpy	$y \leftarrow \alpha x + y$	dot	$r \leftarrow x^T y$
nrm2	$r \leftarrow \ x\ _2 = \sqrt{x^T x}$	asum	$r \leftarrow \ x\ _1 = \sum_i  x_i $

示例:

cblas\_ddot 双精度实值内积运算

# BLAS 命名惯例: Level 2/3

cblas_	X	XX	XXX
前缀	数据类型	结构	操作

矩阵结构:

tr	三角矩阵	tp	压缩三角矩阵	tb	带状三角矩阵
sy	对称矩阵	sp	压缩对称矩阵	sb	带状对称矩阵
hy	厄米特矩阵	hp	压缩厄米特矩阵	hn	带状厄米特矩阵
ge	一般矩阵			gb	带状一般矩阵

操作:

mv	$y \leftarrow \alpha Ax + \beta y$	sv	$x \leftarrow A^{-1}x$ (triangular only)
r	$A \leftarrow A + xx^T$	r2	$A \leftarrow A + xy^T + yx^T$
mm	$C \leftarrow \alpha AB + \beta C$	r2k	$C \leftarrow \alpha AB^T + \alpha BA^T + \beta C$

示例:

cblas_dtrmv	双精度实值三角矩阵与向量乘积
cblas_dsyr2k	双精度实值对称、秩2k矩阵更新

# 高效运用BLAS

用一个高层次的BLAS程序来代替多次调用低层次程序达到同样效果

$$A \leftarrow A + \sum_{i=1}^k x_i y_i^T, \quad A \in \mathbb{R}^{m \times n}, x_i \in \mathbb{R}^m, y_i \in \mathbb{R}^n$$

两种选择: 调用 $k$ 次层次2的程序 `cblas_dger`

$$A \leftarrow A + x_1 y_1^T, \quad \dots \quad A \leftarrow A + x_k y_k^T$$

或调用一次层次3的程序 `cblas_dgemm`

$$A \leftarrow A + XY^T, \quad X = [x_1 \dots x_k], \quad Y = [y_1 \dots y_k]$$

后者表现会更好

## BLAS是必要的吗?

为什么用BLAS可以简化你的程序?

$$A \leftarrow A + XY^T, \quad A \in \mathbb{R}^{m \times n}, x_i \in \mathbb{R}^{m \times p}, y_i \in \mathbb{R}^{n \times p}$$

$$A_{ij} \leftarrow A_{ij} + \sum_{k=1}^p X_{ik} Y_{jk}$$

```
void matmutadd( int m, int n, int p, double* A,
               const double* X, const double* Y ) {
    int i, j, k;
    for ( i = 0 ; i < m ; ++i )
        for ( j = 0 ; j < n ; ++j )
            for ( k = 0 ; k < p ; ++k )
                A[i + j * n] += X[i + k * p] * Y[j + k * p];
}
```

# BLAS是必要的吗?

- 优化调试好的BLAS程序会比你自己写的版本速度更快——通常会快10倍或更多。
- BLAS会根据你的中央处理器和高速缓存大小来选择合适的区块大小，以达到最优的变现。
- ATLAS (automatically tuned linear algebra software)

`http://math-atlas.sourceforge.net`

使用自动代码生成和测试方法，为特定的计算机优化BLAS库。

# Linear Algebra PACKage (LAPACK)

LAPACK包含了求解线性系统和进行常见矩阵分解的程序。

- 首次发行: February 1992; 最新版本(3.0): May 2000
- 取代了原有的EISPACK和LINPACK
- 支持与BLAS相同的数据类型（单/双精度实/复值）和矩阵结构（对称矩阵、带状矩阵等）
- 用BLAS作为程序的内核
- 其程序可以划分为三类: 辅助子程序（auxiliary）、计算子程序（computational）和驱动子程序（driver）

# LAPACK 辅助子程序

辅助子程序主要完成的是一些底层的操作

- 获取当前机器的机器精度和寄存器大小.
- 生成均匀分布或高斯分布的随机数.
- 使用低层次BLAS 运算完成矩阵分解.

# LAPACK 计算子程序

计算子程序完成单一、特定任务

- 矩阵分解:  $LU, LL^T/LL^H, LDL^T/LDL^H, QR, LQ, QRZ$ , 广义  $QR$  和  $RQ$
- 对称/厄米特矩阵和非对称矩阵的特征值分解
- 三对角矩阵的特征值分解和二对角矩阵的奇异值分解.
- 奇异值分解
- 广义特征值和奇异值分解

# LAPACK 驱动子程序

驱动子程序调用一系列计算子程序解决标准的线性代数问题，例如：

- 线性方程:  $AX = B$
- 线性最小二乘:  $\text{minimize}_x \|b - Ax\|_2$
- 线性最小范数:

$$\begin{aligned} &\text{minimize}_x \|c - Ax\|_2 \\ &\text{subject to } Bx = d \end{aligned}$$

$$\begin{aligned} &\text{minimize}_y \|y\|_2 \\ &\text{subject to } d = Ax + By \end{aligned}$$

- 实对称矩阵的特征值分解.

# 其他形式的Lapack

## 并行与分布式计算

- Scalapack

<http://www.netlib.org/scalapack/>

- Elemental

<https://github.com/elemental/Elemental>

## GPU:

- MAGMA

<http://icl.cs.utk.edu/magma/>

- PLASMA

<https://bitbucket.org/icl/plasma>

# 提纲

- 1 矩阵的结构与算法复杂度
- 2 数值代数软件
- 3 通过分解矩阵来求解线性方程
- 4 分块消去法
- 5 稀疏数值线性代数
- 6 特征值分解与奇异值分解

## 易解的线性方程

(块) 对角矩阵 ( $a_{ij} = 0$  if  $i \neq j$ ):  $n$  flops

$$x = A^{-1}b = (b_1/a_{11}, \dots, b_n/a_{nn})$$

下三角矩阵 ( $a_{ij} = 0$  if  $j > i$ ):  $n^2$  flops

$$x_1 := b_1/a_{11}$$

$$x_2 := (b_2 - a_{21}x_1)/a_{22}$$

$$x_3 := (b_3 - a_{31}x_1 - a_{32}x_2)/a_{33}$$

$\vdots$

$$x_n := (b_n - a_{n1}x_1 - a_{n2}x_2 - \dots - a_{n,n-1}x_{n-1})/a_{nn}$$

被称为前代法

上三角矩阵 ( $a_{ij} = 0$  if  $j < i$ ):  $n^2$  flops 回代法

正交矩阵:  $A^{-1} = A^T$

- $2n^2$  flops, 对于一般的 $A$  计算 $x = A^T b$
- 当 $A$  有结构时, 运算量会更少, 如 $A = I - 2uu^T$  (Household矩阵), 其中 $\|u\|_2 = 1$ , 计算 $x = A^T b = b - 2(u^T b)u$ , 只需要 $4n$  flops

置换矩阵:

$$a_{ij} = \begin{cases} 1 & j = \pi_i \\ 0 & \text{otherwise} \end{cases}$$

其中 $\pi = (\pi_1, \pi_2, \dots, \pi_n)$  是 $(1, 2, \dots, n)$ 的一个置换

- 乘法可以表示为:  $Ax = (x_{\pi_1}, \dots, x_{\pi_n})$
- 满足 $A^{-1} = A^T$ , 因此计算 $Ax = b$  的代价是 $0$  flops

例子:

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}, \quad A^{-1} = A^T = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

## 通过因子分解来求解 $Ax = b$

- 将  $A$  分解为几个简单矩阵的乘积（通常为2到3个）

$$A = A_1 A_2 \cdots A_k$$

( $A_i$  是对角矩阵、上三角矩阵、下三角矩阵等)

- 通过求解  $k$  个“简单”的方程来计算  $x = A^{-1}b = A_k^{-1} \cdots A_2^{-1} A_1^{-1} b$

$$A_1 x_1 = b, \quad A_2 x_2 = x_1, \quad \dots, \quad A_k x_k = x_{k-1}$$

分解矩阵的计算代价是求解方程组的主要代价

### 多个系数矩阵相同的方程

$$Ax_1 = b_1, \quad Ax_2 = b_2, \quad \dots, \quad Ax_m = b_m$$

代价：一次分解加  $m$  次求解

# LU 分解

所有的非奇异矩阵 $A$ 均可分解为

$$A = PLU$$

其中 $P$ 是置换矩阵, $L$ 是下三角矩阵且对角元均为1, $U$ 是上三角矩阵

代价:  $(2/3)n^3$  flops

用LU分解来求解线性方程

给定线性方程 $Ax = b$ , 其中 $A$ 非奇异.

- 1 LU分解 将 $A$ 分解为 $A = PLU$  ( $(2/3)n^3$  flops).
- 2 置换 求解 $Pz_1 = b$  (0 flops).
- 3 前代法 求解 $Lz_2 = z_1$  ( $n^2$  flops).
- 4 回代法 求解 $Ux = z_2$  ( $n^2$  flops).

代价:  $(2/3)n^3 + 2n^2 \approx (2/3)n^3$  (对于较大的 $n$ )

## 稀疏矩阵LU分解

$$A = P_1 L U P_2$$

- 增加一个置换矩阵 $P_2$ ，来保证 $L$ 和 $U$ 的稀疏性(进而减少分解和求解代价)
- (启发式地) 选择 $P_1$ 和 $P_2$ 来生成稀疏的 $L, U$
- $P_1$ 和 $P_2$ 的选择基于 $A$ 的稀疏性模式和取值
- 代价通常低于 $(2/3)n^3$ ; 确切的数值以复杂的形式取决于 $n$ 、 $A$ 中0元的个数、 $A$ 的稀疏模式。

# Cholesky分解

所有正定矩阵 $A$ 可以被分解为

$$A = LL^T$$

其中 $L$ 是下三角矩阵

代价:  $(1/3)n^3$  flops

用 *Cholesky* 求解线性方程

给定线性方程 $Ax = b$ , 其中 $A \in \mathbb{S}_{++}^n$ .

- 1 *Cholesky* 分解 将 $A$  分解为 $A = LL^T$   
( $(1/3)n^3$  flops).
- 2 前代法 求解 $Lz_1 = b$  ( $n^2$  flops).
- 3 回代法 求解 $L^T x = z_1$  ( $n^2$  flops).

代价:  $(1/3)n^3 + 2n^2 \approx (1/3)n^3$  (对于较大的 $n$ )

## 稀疏Cholesky 分解

$$A = PLL^T P^T$$

- 增加一个置换矩阵 $P$ 来保证 $L$ 的稀疏性
- (启发式地) 选择 $P$ 来生成稀疏的 $L$
- $P$ 的选择仅基于 $A$ 的稀疏模式(与稀疏LU不同)
- 代价通常低于 $(1/3)n^3$ ; 确切的数值以复杂的形式取决于 $n$ 、 $A$ 中0元的个数、 $A$ 的稀疏模式。

# LDL<sup>T</sup> 分解

所有非奇异对称矩阵 $A$ 可被分解为

$$A = PLDL^T P^T$$

其中 $P$ 是置换矩阵, $L$ 是下三角矩阵, $D$ 是分块对角矩阵,其块的大小为 $1 \times 1$ 或 $2 \times 2$

代价:  $(1/3)n^3$

- 用LDL<sup>T</sup>分解求解对称形式的线性方程,其代价为:  
 $(1/3)n^3 + 2n^2 \approx (1/3)n^3$  (当 $n$ 较大时)
- 对于稀疏矩阵 $A$ ,可以选择适当的 $P$ 来生成稀疏的 $L$ ; 代价 $\ll (1/3)n^3$

# QR 分解

QR 分解又称正交分解. 对于“瘦高”形状的矩阵  $A \in \mathbb{R}^{m \times n}$ , 其中  $m \geq n$ , QR 分解可以表示为

$$A = QR,$$

其中  $Q \in \mathbb{R}^{m \times m}$  为正交矩阵,  $R \in \mathbb{R}^{m \times n}$  是上三角矩阵,

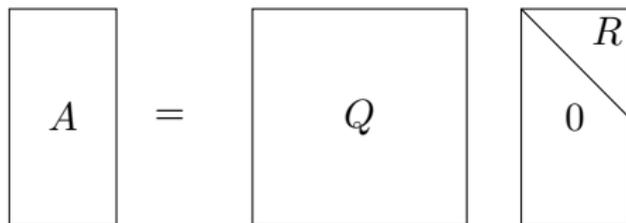


Figure: QR 分解

# QR 分解

- $R$ 的下半部分都是零, 当  $m \gg n$  时, 这种分解方式极大地浪费存储空间
- $Q$ 的后  $(m - n)$ 列是多余的, 约化的QR 分解为

$$A = QR,$$

其中  $Q \in \mathbb{R}^{m \times n}$  满足  $Q^T Q = I$ ,  $R \in \mathbb{R}^{n \times n}$  为上三角矩阵,

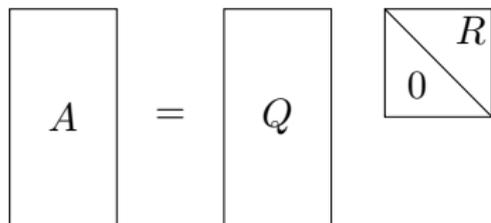


Figure: 约化的QR 分解

# QR 分解

列满秩矩阵 $A$ ，满足 $R$ 对角元为正数的约化的QR分解是唯一的，有如下定理：

## 定理 (约化的QR分解的唯一性)

对于矩阵 $A \in \mathbb{R}^{m \times n}$  ( $m \geq n$ )，假设其列满秩，即 $\text{rank}(A) = n$ ，那么 $A$ 有唯一的使得上三角矩阵 $R$ 的对角元为正数（即 $r_{ii} > 0$ ）的约化的QR分解，

- QR分解可以通过Gram-Schmidt 正交化或Householder 三角化完成
- QR分解数值稳定性好，还可用于估计矩阵的秩和求解线性最小二乘问题。
- 计算代价大于LU分解，至少为 $\mathcal{O}\left(\frac{4mn^2}{3}\right)$

# 提纲

- 1 矩阵的结构与算法复杂度
- 2 数值代数软件
- 3 通过分解矩阵来求解线性方程
- 4 分块消去法**
- 5 稀疏数值线性代数
- 6 特征值分解与奇异值分解

## 有结构子块的矩阵

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \quad (1)$$

- 变量  $x_1 \in \mathbb{R}^{n_1}$ ,  $x_2 \in \mathbb{R}^{n_2}$ ; 子矩阵  $A_{ij} \in \mathbb{R}^{n_i \times n_j}$
- 若  $A_{11}$  非奇异, 可以消去  $x_1$ :  $x_1 = A_{11}^{-1}(b_1 - A_{12}x_2)$ ; 计算  $x_2$ , 可以求解

$$(A_{22} - A_{21}A_{11}^{-1}A_{12})x_2 = b_2 - A_{21}A_{11}^{-1}b_1$$

用分块消去法求解线性方程

给定一个非奇异的线性方程, 并且  $A_{11}$  非奇异

- ① 计算  $A_{11}^{-1}A_{12}$  和  $A_{11}^{-1}b_1$ .
- ② 计算  $S = A_{22} - A_{21}A_{11}^{-1}A_{12}$  和  $\tilde{b} = b_2 - A_{21}A_{11}^{-1}b_1$ .
- ③ 求解  $Sx_2 = \tilde{b}$  来确定  $x_2$
- ④ 求解  $A_{11}x_1 = b_1 - A_{12}x_2$  来确定  $x_1$

## flop计数中的主要部分

- 第一步:  $f + n_2s$  ( $f$  是分解 $A_{11}$ 的代价;  $s$  是求解的代价)
- 第二步:  $2n_2^2n_1$  (主要代价是 $A_{21}$  和 $A_{11}^{-1}A_{12}$ 的乘积)
- 第三步:  $(2/3)n_2^3$

总代价:  $f + n_2s + 2n_2^2n_1 + (2/3)n_2^3$

## 例子

- 对于一般的矩阵 $A_{11}$  ( $f = (2/3)n_1^3$ ,  $s = 2n_1^2$ ): 相比于标准方法没有任何改进

$$\#\text{flops} = (2/3)n_1^3 + 2n_1^2n_2 + 2n_2^2n_1 + (2/3)n_2^3 = (2/3)(n_1 + n_2)^3$$

- 当 $A_{11}$ 具有结构时, 分块消去法非常有用( $f \ll n^3$ )  
例如, 对角矩阵( $f = 0$ ,  $s = n_1$ ):  $\#\text{flops} \approx 2n_2^2n_1 + (2/3)n_2^3$

## 结构矩阵加低秩项

$$(A + BC)x = b$$

- $A \in \mathbb{R}^{n \times n}$ ,  $B \in \mathbb{R}^{n \times p}$ ,  $C \in \mathbb{R}^{p \times n}$
- 假设  $A$  有某种结构( $Ax = b$  解起来很方便)

首先将方程改写为

$$\begin{bmatrix} A & B \\ C & -I \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}$$

然后使用分块消去法：求解

$$(I + CA^{-1}B)y = CA^{-1}b,$$

然后求解  $Ax = b - By$

该方法也证明了 **SMW** 公式：若  $A$  和  $A + BC$  均非奇异，则

$$(A + BC)^{-1} = A^{-1} - A^{-1}B(I + CA^{-1}B)^{-1}CA^{-1}$$

例子:  $A$  是对角矩阵,  $B, C$  是稠密的

- 方法1: 计算  $D = A + BC$ , 然后求解  $Dx = b$

代价:  $(2/3)n^3 + 2pn^2$

- 方法2 (借助SMW公式): 求解

$$(I + CA^{-1}B)y = CA^{-1}b, \quad (2)$$

然后计算  $x = A^{-1}b - A^{-1}By$

总代价由(2)决定:  $2p^2n + (2/3)p^3$  (i.e., 关于  $n$  是线性的)

## 欠定线性方程

若  $A \in \mathbb{R}^{p \times n}$  , 其中  $p < n$ ,  $\text{rank } A = p$ ,

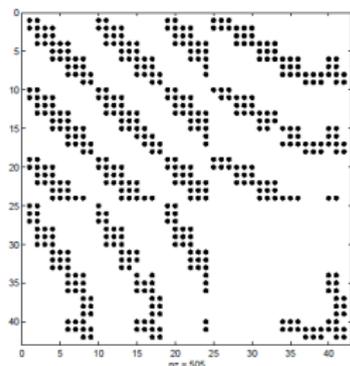
$$\{x | Ax = b\} = \{Fz + \hat{x} | z \in \mathbb{R}^{n-p}\}$$

- $\hat{x}$  是 (任意) 一个特解
- $F \in \mathbb{R}^{n \times (n-p)}$  的列向量张成了  $A$  的零空间
- 有很多计算  $F$  的数值方法  
(QR 分解, rectangular LU 分解, ... )

# 提纲

- 1 矩阵的结构与算法复杂度
- 2 数值代数软件
- 3 通过分解矩阵来求解线性方程
- 4 分块消去法
- 5 稀疏数值线性代数
- 6 特征值分解与奇异值分解

# 稀疏矩阵



- $A \in \mathbb{R}^{m \times n}$  是稀疏的，如果它有“enough zeros that it pays to take advantage of them” (J. Wilkinson)
- 通常意味着非零元个数  $n_{NZ}$  非常小:  $n_{NZ} \ll mn$

# 稀疏矩阵

稀疏矩阵可以节省时间和空间代价

- 使用双精度数存储  $A \in \mathbb{R}^{m \times n}$ 
  - 稠密矩阵:  $8mn$  字节
  - 稀疏矩阵:  $\approx 16n_{\text{NZ}}$  字节或更少, 取决于存储格式
- 计算  $y \leftarrow y + Ax$ 
  - 稠密矩阵:  $mn$  flops
  - 稀疏矩阵:  $n_{\text{NZ}}$  flops
- 计算  $x \leftarrow T^{-1}x, T \in \mathbb{R}^{n \times n}$  是非奇异的三角矩阵:
  - 稠密矩阵:  $n^2/2$  flops
  - 稀疏矩阵:  $n_{\text{NZ}}$  flops

# 稀疏矩阵的表示法

- 有很多种方法在使用
- 最简单(但是并不经常使用)的方法是将数据存储在一个列表中，列表的元素是 $(i, j, A_{ij})$ 三元组。
- 列压缩格式 (column compressed format)：将所有 $(A_{ij}, i)$ 对按顺序放在一个数组中，再用指针记录每一列开始的位置，并将这些位置指针放在一个数组中
- 在高层次的工作中，也会使用其他的数据结构
- 可惜的是，目前没有一个统一的标准

# 稀疏BLAS?

然而现在并没有一个标准的稀疏矩阵BLAS库

- “官方”稀疏BLAS

`http://www.netlib.org/blas/blast-forum`

`http://math.nist.gov/spblas`

- C++: Boost uBlas, Matrix Template Library, SparseLib++
- 英特尔的数学核心函数库: MKL, 支持串行和并行计算.
- Python: SciPy, PySparse, CVXOPT

# 稀疏矩阵的分解

求解或分解稀疏矩阵系统的函数库

- 最容易理解的: SuiteSparse (Tim Davis)

<http://www.cise.ufl.edu/research/sparse/SuiteSparse>

- 其他还有 SuperLU, TAUCS, SPOOLES

- 通常包括

-  $A = PLL^T P^T$  Cholesky

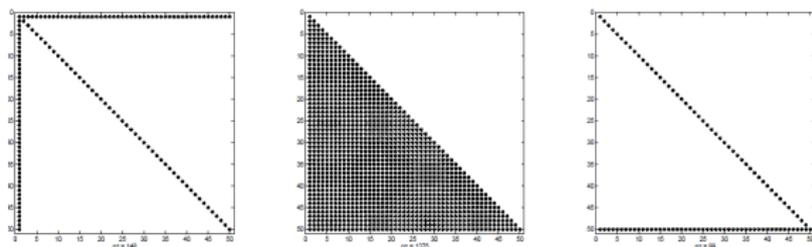
-  $A = PLDL^T P^T$  对于对称不定系统

-  $A = P_1 L U P_2^T$  对于一般的 (非对称) 矩阵

$P, P_1, P_2$  是置换矩阵 (permutations) 或排列矩阵 (orderings)

# 稀疏矩阵的排列

重新排列稀疏矩阵，会对因子矩阵的稀疏性有极其重要的影响



- 左图: 原版NW箭形矩阵的spy图像
- 中图: 不重排, 进行Cholesky分解, 因子矩阵的spy图像( $P = I$ )
- 右图: 选择最优的排列方式, 进行Cholesky分解, 因子矩阵的spy图像(交换 $1 \rightarrow n$ )

# 稀疏矩阵的排列

- 选择产生最稀疏分解的排序的一般问题是很难的。
- 但是，一些简单的启发式方法非常有效
- 还有更多的排列方法，例如nested dissection 也非常有效

# 符号分解

- 对于Cholesky分解，重排的选择仅基于 $A$ 的稀疏格式，与具体值无关
- 分解可以被划分为两步：符号分解和数值分解
  - 在求解多个线性系统时，如果他们有相同的稀疏格式，那么符号分解可以只做一次
  - 可以在选择重排方式时投入更多，因为在多次数值分解中，符号分解的代价会被摊销
- $LDL^T$ 分解的重排通常只能听天由命/按部就班/特事特办/根据数据 (has to be done on the fly) (待修改)

# 提纲

- 1 矩阵的结构与算法复杂度
- 2 数值代数软件
- 3 通过分解矩阵来求解线性方程
- 4 分块消去法
- 5 稀疏数值线性代数
- 6 特征值分解与奇异值分解

# 特征值分解

- 矩阵的特征值与特征向量求解是线性代数的另一个基本问题，在数据降维、聚类分析等各种实际问题中有重要的应用
- 对于矩阵  $A \in \mathbb{R}^{n \times n}$ ，若存在非奇异矩阵  $X \in \mathbb{R}^{n \times n}$  和对角矩阵  $\Sigma \in \mathbb{R}^{n \times n}$ ，使得

$$A = X\Sigma X^{-1}.$$

则称  $A$  可对角化，并称上式为  $A$  的特征值分解。

- 矩阵  $\Sigma$  的对角元为  $A$  的**特征值**，矩阵  $X$  的每一列为其对应的**特征向量**。
- 计算一个任意方阵的特征值分解通常来说是比较难的

# 实对称矩阵的特征值分解

在很多优化问题中，我们研究的往往是实对称矩阵。实对称矩阵  $A \in \mathcal{S}^n$  的特征值分解有非常好的性质：

- 1 实对称矩阵的所有特征值均为实数；
- 2 实对称矩阵对应于不同特征值的特征向量是相互正交的；
- 3 实对称矩阵可以正交对角化，即存在正交矩阵  $U$ ，使得  $U^T A U$  为对角矩阵。

# 实对称矩阵的特征值分解

- 对于  $A \in \mathcal{S}^n$ , 有分解

$$A = U\Lambda U^T,$$

其中  $U \in \mathbb{R}^{n \times n}$  为正交矩阵,  $\Lambda = \text{Diag}(\lambda_1, \lambda_2, \dots, \lambda_n)$  为对角矩阵并且  $\lambda_i, i = 1, 2, \dots, n$  对应于  $A$  的特征值.

- 记  $U = [u_1, u_2, \dots, u_n]$ , 那么  $u_i$  为特征值  $\lambda_i$  对应的特征向量, 我们还可以将  $A$  写成如下秩1矩阵的求和的形式 (又称为外积形式)

$$A = \sum_{i=1}^n \lambda_i u_i u_i^T.$$

- 特征值的计算有幂法、反幂法、QR方法、Jacobi方法、二分法等

# 奇异值分解

当矩阵不是方阵时，考虑它的奇异值分解.

## 定理

设  $A \in \mathbb{R}^{m \times n} (m \geq n)$ ，则存在正交矩阵  $U \in \mathbb{R}^{m \times m}$  和  $V \in \mathbb{R}^{n \times n}$ ，使得

$$A = U \begin{bmatrix} \Sigma \\ 0 \end{bmatrix} V^T, \quad (3)$$

其中  $\Sigma = \text{Diag}(\sigma_1, \sigma_2, \dots, \sigma_n)$ ， $\sigma_i \in \mathbb{R}$  且  $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_n \geq 0$ .

称(3)式为矩阵  $A$  的奇异值分解， $\sigma_1, \sigma_2, \dots, \sigma_n$  为  $A$  的奇异值， $U$  和  $V$  的每一列分别称为左、右奇异向量.

# 奇异值分解

- 从定义式(3)得： $A^T A = V \Sigma^T \Sigma V^T$ ，即 $A^T A$ 与 $\Sigma^T \Sigma$ 相似，求 $A$ 的奇异值，等价于求 $A^T A$ 的特征值。
- 流程如下：
  - ① 计算 $A^T A$ ；
  - ② 计算 $A^T A$ 的特征值分解： $A^T A = V \Lambda V^T$ ；
  - ③ 对 $\Lambda$ 所有对角元素开根号： $\Sigma = \sqrt{\Lambda}$ ；
  - ④ 求解正交矩阵 $U$ ，使得 $U \begin{bmatrix} \Sigma \\ 0 \end{bmatrix} = AV$ （可以利用QR分解）。
- 因为 $A^T A$ 损失了原矩阵 $A$ 的部分信息，该方法可能数值不稳定。
- 优点是，当 $n \ll m$ 时，仅涉及 $n$ 阶小方阵的特征值分解

# 奇异值分解

- 另一种方法将这一问题转化为另一个矩阵的特征值分解.
- 对于矩阵 $A$ 及其奇异值分解 $A = U\Sigma V^T$ , 构造矩阵

$$H = \begin{bmatrix} 0 & A^T \\ A & 0 \end{bmatrix}, \quad X = \begin{bmatrix} V & V \\ U & -U \end{bmatrix},$$

那么

$$HX = X \begin{bmatrix} \Sigma & 0 \\ 0 & -\Sigma \end{bmatrix}.$$

- 将 $A$ 的奇异值分解问题转化为 $H$ 的特征值分解问题.
- 该方法数值稳定, 但增大了问题维数, 需要更大的存储量和迭代次数.

# 计算主要特征（奇异）值和特征（奇异）向量

## 特征值对的计算

- ARPACK (eigs in matlab)
- LOBPCG
- Arrabit
- SLEPc

## 奇异值对的计算

- PROPACK, a good implementation is lansvd in  
<http://www.math.nus.edu.sg/~mattohkc/NNLS.html>
- LMSVD with warm-starting:  
<https://ww2.mathworks.cn/matlabcentral/fileexchange/46875-lmsvd-m>

我们列出了数值线性代数中其他一些备受关注的领域。

- 稀疏与有结构的线性系统的迭代解法
- 并行与分布式计算方法(MPI)
- 快速线性算子: 快速傅立叶变换(FFT), 卷积, 状态空间线性系统模拟

现有相当多有趣的研究, 并有公开(或免费授权)的代码。